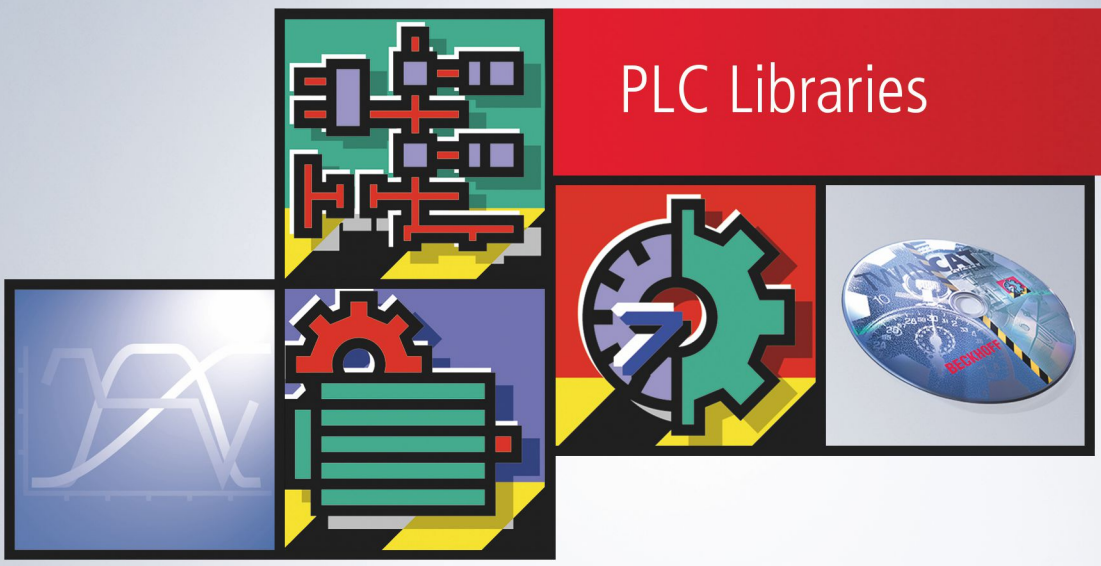
BECKHOFF New Automation Technology

Handbuch | DE

TX1200

TwinCAT 2 | PLC-Bibliothek: TcEtherCAT



24.10.2023 | Version: 1.2

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®, TwinCAT®, TwinCAT/BSD®, TC/BSD®, EtherCAT®, EtherCAT G®, EtherCAT G10®, EtherCAT P®, Safety over EtherCAT®, TwinSAFE®, XFC®, XTS® und XPlanar® 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 Bezeichnungen führen.

Patente

Die EtherCAT-Technologie ist patentrechtlich geschützt, insbesondere durch folgende Anmeldungen und Patente:
EP1590927, EP1789857, EP1456722, EP2137893, DE102015105702
mit den entsprechenden Anmeldungen und Eintragungen in verschiedenen anderen Ländern.

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

Copyright

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 Übersicht

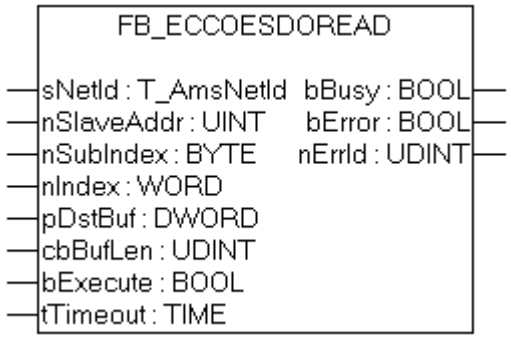
Die SPS-Bibliothek: **TcEtherCAT.Lib** enthält Funktionsbausteine, mit denen Dienste bzw. Funktionen auf einem EtherCAT Master-Gerät und/oder an dessen Slave-Geräten ausgeführt werden können.

Beispielprojekt und Beispielkonfiguration für Diagnose

Siehe https://infosys.beckhoff.com/content/1031/tcplclibethercat/Resources/11934864011.exe

3 CoE

3.1 FB_EcCoeSdoRead



Mit dem Funktionsbaustein FB_EcCoeSdoRead können per SDO (Service Daten Objekt) -Zugriff Daten aus dem Objektverzeichnis eines EtherCAT Slaves ausgelesen werden. Dazu muss der Slave eine Mailbox besitzen und das "CANopen over EtherCAT"(CoE) Protokoll unterstützen. Mit Hilfe der Parameter nSubIndex und nIndex wird ausgewählt, welches Objekt ausgelesen werden soll.

VAR_INPUT

```
VAR_INPUT
    sNetId      : T_AmsNetId;
    nSlaveAddr  : UINT;
    nSubIndex   : BYTE;
    nIndex      : WORD;
    pDstBuf     : DWORD;
    cbBufLen    : UDINT;
    bExecute    : BOOL;
    tTimeout    : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

sNetId: Ist ein String, der die AMS-Netzwerkkennung des EtherCAT Master Gerätes enthält.

nSlaveAddr: Feste Adresse des EtherCAT Slaves an den das SDO-Upload Kommando geschickt werden soll.

nSubIndex: Subindex des Objektes, das gelesen werden soll.

nIndex: Index des Objektes, das gelesen werden soll.

pDstBuf: Die Adresse (Pointer) auf den Empfangspuffer.

cbBufLen: Die maximal verfügbare Puffergröße für die zu lesenden Daten in Bytes.

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

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

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy       : BOOL;
    bError      : BOOL;
    nErrId      : UDINT;
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.

nErrId: Liefert bei einem gesetzten bError-Ausgang den ADS-Fehlercode des zuletzt ausgeführten Befehles

Beispiel für eine Implementierung in ST:

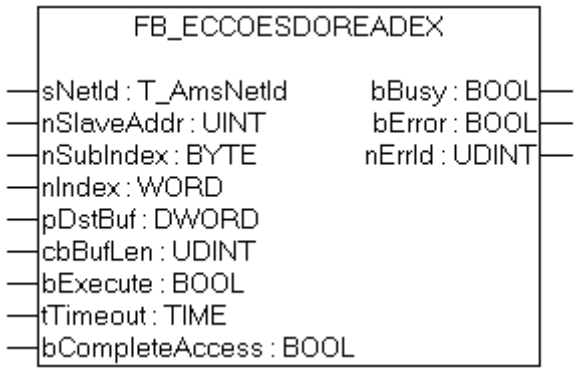
```
PROGRAM TEST_SdoRead
VAR
    fbSdoRead   : FB_EcCoESdoRead;
    sNetId      : T_AmsNetId := '172.16.2.131.2.1';
    bExecute    : BOOL;
    nSlaveAddr  : UINT := 1006;
    nIndex      : WORD := 16#1018;
    nSubIndex   : BYTE :=1;
    vendorId    : UDINT;
    bError      : BOOL;
    nErrId      : UDINT;
END_VAR
```

```
fbSdoRead(sNetId:= sNetId,nSlaveAddr :=nSlaveAddr, nIndex:=nIndex, nSubIndex :=nSubIndex, pDstBuf:=
ADR(vendorId), cbBufLen:=SIZEOF(vendorId),bExecute:=bExecute);
bError:=fbSdoRead.bError;
nErrId:=fbSdoRead.nErrId;
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 oder höher	PC or CX (x86)	TcEtherCAT.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib, TcUtilities.Lib werden automatisch eingebunden)
TwinCAT v2.10.0 Build >= 1301 oder höher	CX (ARM)	

3.2 FB_EcCoeSdoReadEx



Mit dem Funktionsbaustein FB_EcCoeSdoReadEx können per SDO (Service Daten Objekt) -Zugriff Daten aus dem Objektverzeichnis eines EtherCAT Slaves ausgelesen werden. Dazu muss der Slave eine Mailbox besitzen und das "CANopen over EtherCAT"(CoE) Protokoll unterstützen. Mit Hilfe der Parameter nSubIndex und nIndex wird ausgewählt, welches Objekt ausgelesen werden soll. Über bCompleteAccess := TRUE kann der Parameter samt Unterelementen eingelesen werden.

VAR_INPUT

```
VAR_INPUT
    sNetId          : T_AmsNetId; (* AmsNetId of the EtherCAT master device.*)
    nSlaveAddr      : UINT; (* Address of the slave device.*)
    nSubIndex       : BYTE; (* CANopen Sdo subindex.*)
    nIndex          : WORD; (* CANopen Sdo index.*)
    pDstBuf         : DWORD; (* Contains the address of the buffer for the received data. *)
    cbBufLen        : UDINT; (* Contains the max. number of bytes to be received. *)
    bExecute        : BOOL; (* Function block execution is triggered by a rising edge at this input.
*)
    tTimeout        : TIME := DEFAULT_ADS_TIMEOUT;
```

```
(* States the time before the function is cancelled. *)
    bCompleteAccess : BOOL; (* access complete object*)
END_VAR
```

sNetId: Ist ein String, der die AMS-Netzwerkkennung des EtherCAT Master Gerätes enthält.

nSlaveAddr: Feste Adresse des EtherCAT Slaves an den das SDO-Upload Kommando geschickt werden soll.

nSubIndex: Subindex des Objektes, das gelesen werden soll.

nIndex: Index des Objektes, das gelesen werden soll.

pDstBuf: Die Adresse (Pointer) auf den Empfangspuffer.

cbBufLen: Die maximal verfügbare Puffergröße für die zu lesenden Daten in Bytes.

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

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

bCompleteAccess: Bei gesetztem bCompleteAccess kann der gesamte Parameter in einem Zugriff eingelesen werden.

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy       : BOOL;
    bError      : BOOL;
    nErrId      : UDINT;
END_VAR
```

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

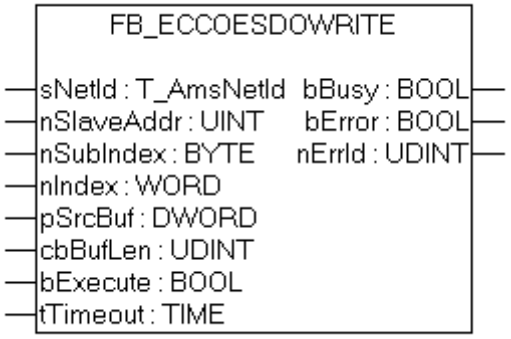
bError: Dieser Ausgang wird, nachdem der bBusy-Ausgang zurückgesetzt wurde, gesetzt, sollte ein Fehler bei der Übertragung des Kommandos erfolgen.

nErrId: Liefert bei einem gesetzten bError-Ausgang den ADS-Fehlercode des zuletzt ausgeführten Befehles

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 Build >= 1319 oder höher	PC or CX (x86)	TcEtherCAT.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib, TcUtilities.Lib werden automatisch eingebunden)
TwinCAT v2.10.0 Build >= 1319 oder höher	CX (ARM)	

3.3 FB_EcCoeSdoWrite



Mit dem Funktionsbaustein FB_EcCoeSdoWrite kann per SDO-Download ein Objekt aus dem Objektverzeichnis eines EtherCAT Slaves beschrieben werden. Dazu muss der Slave eine Mailbox besitzen und das "CANopen over EtherCAT" (CoE) Protokoll unterstützen. Mit Hilfe der Parameter nSubIndex und nIndex wird ausgewählt, welches Objekt beschrieben werden soll.

VAR_INPUT

```
VAR_INPUT
    sNetId      : T_AmsNetId;
    nSlaveAddr  : UINT;
    nSubIndex   : BYTE;
    nIndex      : WORD;
    pSrcBuf     : DWORD;
    cbBufLen    : UDINT;
    bExecute    : BOOL;
    tTimeout    : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

sNetId: Ist ein String, der die AMS-Netzwerkkennung des EtherCAT Master Gerätes enthält.

nSlaveAddr: Feste Adresse des EtherCAT Slaves and den das SDO-Download Kommando geschickt werden soll.

nSubIndex: Subindex des Objektes, das geschrieben werden soll.

nIndex: Index des Objektes, das geschrieben werden soll.

pSrcBuf: Die Adresse (Pointer) auf den Sendepuffer.

cbBufLen: Die Anzahl der zu sendenden Daten in Bytes.

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

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

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy       : BOOL;
    bError      : BOOL;
    nErrId      : UDINT;
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.

nErrId: Liefert bei einem gesetzten bError-Ausgang den ADS-Fehlercode des zuletzt ausgeführten Befehles

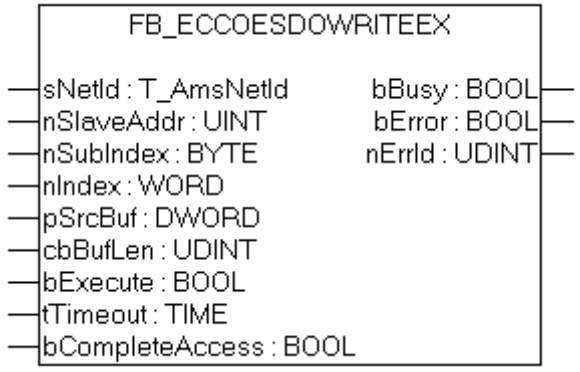
Beispiel für eine Implementierung in ST:

```
PROGRAM TEST_SdoWrite

VAR
    fbSdoWrite      : FB_EcCoESdoWrite;
    sNetId          : T_AmsNetId := '172.16.2.131.2.1'; (* NetId of EtherCAT Master *)
    nSlaveAddr      : UINT := 1005; (* Port Number of EtherCAT Slave *)
    nIndex          : WORD := 16#4062; (* CoE Object Index *)
    nSubIndex       : BYTE := 1; (* Subindex of CoE Object *)
    nValue          : UINT := 2; (* variable to be written to the CoE Object *)
    bExecute        : BOOL; (* rising edge starts writing to the CoE Object *)
    bError          : BOOL;
    nErrId          : UDINT;
END_VAR


fbSdoWrite(
    sNetId      := sNetId,
    nSlaveAddr  := nSlaveAddr,
    nIndex      := nIndex,
    nSubIndex   := nSubIndex,
    pSrcBuf     := ADR(nValue),
    cbBufLen    := SIZEOF(nValue),
    bExecute    := bExecute
);


IF NOT fbSdoWrite.bBusy THEN
    bExecute := FALSE;
    IF NOT bError THEN
        (* write successful *)
        bError := FALSE;
        nErrId := 0;
    ELSE
        (* write failed *)
        bError := fbSdoWrite.bError;
        nErrId := fbSdoWrite.nErrId;
    END_IF

    fbSdoWrite(bExecute := FALSE);
END_IF
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 oder höher	PC or CX (x86)	TcEtherCAT.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib, TcUtilities.Lib werden automatisch eingebunden)
TwinCAT v2.10.0 Build >= 1301 oder höher	CX (ARM)	

3.4 FB_EcCoeSdoWriteEx



Mit dem Funktionsbaustein FB_EcCoeSdoWriteEx kann per SDO-Download ein Objekt aus dem Objektverzeichnis eines EtherCAT Slaves beschrieben werden. Dazu muss der Slave eine Mailbox besitzen und das "CANopen over EtherCAT" (CoE) Protokoll unterstützen. Mit Hilfe der Parameter nSubIndex und nIndex wird ausgewählt, welches Objekt beschrieben werden soll. Über bCompleteAccess := TRUE kann der Parameter samt Unterelementen geschrieben werden.

VAR_INPUT

```
VAR_INPUT
    sNetId          : T_AmsNetId; (* AmsNetId of the EtherCAT master device.*)
    nSlaveAddr      : UINT; (* Address of the slave device.*)
    nSubIndex       : BYTE; (* CANopen Sdo subindex.*)
    nIndex          : WORD; (* CANopen Sdo index.*)
    pSrcBuf         : DWORD; (* Contains the address of the buffer containing the data to be send. *
)
    cbBufLen        : UDINT; (* Contains the max. number of bytes to be received. *)
    bExecute        : BOOL; (* Function block execution is triggered by a rising edge at this input.
*)
    tTimeout        : TIME := DEFAULT_ADS_TIMEOUT;
(* States the time before the function is cancelled. *)
    bCompleteAccess : BOOL; (* access complete object*)
END_VAR
```

sNetId: Ist ein String, der die AMS-Netzwerkkennung des EtherCAT Master Gerätes enthält.

nSlaveAddr: Feste Adresse des EtherCAT Slaves and den das SDO-Download Kommando geschickt werden soll.

nSubIndex: Subindex des Objektes, das geschrieben werden soll.

nIndex: Index des Objektes, das geschrieben werden soll.

pSrcBuf: Die Adresse (Pointer) auf den Sendepuffer.

cbBufLen: Die Anzahl der zu sendenden Daten in Bytes.

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

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

bCompleteAccess: Bei gesetztem bCompleteAccess kann der gesamte Parameter in einem Zugriff geschrieben werden.

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy       : BOOL;
    bError      : BOOL;
    nErrId      : UDINT;
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.

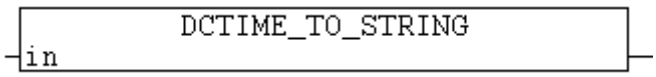
nErrId: Liefert bei einem gesetzten bError-Ausgang den ADS-Fehlercode des zuletzt ausgeführten Befehles

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 Build >= 1319 oder höher	PC or CX (x86)	TcEtherCAT.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib, TcUtilities.Lib werden automatisch eingebunden)
TwinCAT v2.10.0 Build >= 1319 oder höher	CX (ARM)	

4 Conversion Functions

4.1 DCTIME_TO_STRING



Die Funktion konvertiert eine "Distributed Clock System Time"-Variable in einen String.

Der resultierende String hat nach der Konvertierung das folgende Format: **'YYYY-MM-DD-hh:mm:ss.nnnnnnnnn'**

- YYYY: Jahr;
- MM: Monat;
- DD: Tag;
- hh: Stunde;
- mm: Minute;
- ss: Sekunde;
- nnnnnnnnn: Nanosekunden;

FUNCTION DCTIME_TO_STRING: STRING(29)

```
VAR_INPUT
    in : T_DCTIME;
END_VAR
```

in: Die zu konvertierende "Distributed Clock System Time [▶ 87]"-Variable.

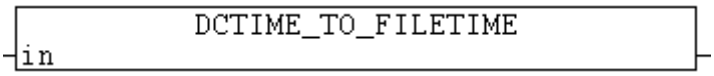
Beispiel:

Siehe in der Beschreibung der Funktion: F_GetCurDcTickTime [▶ 28].

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 Build > 1310 oder höher	PC oder CX (x86, ARM)	TcEtherCAT.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib, TcUtilities.Lib werden automatisch eingebunden)

4.2 DCTIME_TO_FILETIME



Die Funktion konvertiert eine 64 bit "Distributed Clock System Time"-Variable in eine 64 bit "Windows File Time"-Variable

FUNCTION DCTIME_TO_FILETIME: T_FILETIME

T_FILETIME

```
VAR_INPUT
    in : T_DCTIME;
END_VAR
```

in: Die zu konvertierende "Distributed Clock System Time [▶ 87]"-Variable.

Beispiel:

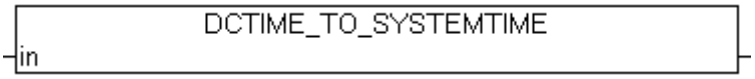
```
PROGRAM P_TEST
VAR
    ft      : T_FILETIME;
    dct     : T_DCTIME;
END_VAR
```

```
dct      := F_GetCurDcTickTime();
ft       := DCTIME_TO_FILETIME(dct);
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 Build >= 1324 oder höher	PC oder CX (x86, ARM)	TcEtherCAT.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib, TcUtilities.Lib werden automatisch eingebunden)

4.3 DCTIME_TO_SYSTEMTIME



Die Funktion konvertiert eine 64 bit "Distributed Clock System Time"-Variable in eine strukturierte "Windows System Time"-Variable.

FUNCTION DCTIME_TO_SYSTEMTIME: TIMESTRUCT

TIMESTRUCT

```
VAR_INPUT
    in : T_DCTIME;
END_VAR
```

in: Die zu konvertierende "Distributed Clock System Time [▶ 87]"-Variable.

Beispiel:

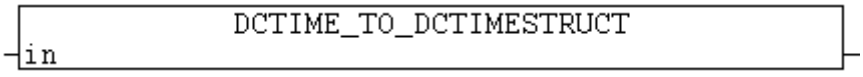
```
PROGRAM P_TEST
VAR
    syst : TIMESTRUCT;
END_VAR
```

```
syst := DCTIME_TO_SYSTEMTIME( F_GetCurDcTickTime() );
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 Build > 1340 TwinCAT v2.11.0 Build > 1536	PC oder CX (x86, ARM)	TcEtherCAT.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib, TcUtilities.Lib werden automatisch eingebunden)

4.4 DCTIME_TO_DCTIMESTRUCT



Die Funktion konvertiert eine 64 bit "Distributed Clock System Time"-Variable in eine strukturierte Variable vom Typ: DCTIMESTRUCT [▶ 78].

FUNCTION DCTIME_TO_DCTIMESTRUCT: DCTIMESTRUCT

DCTIMESTRUCT [▶ 78]

```
VAR_INPUT
    in : T_DCTIME;
END_VAR
```

in: Die zu konvertierende "Distributed Clock System Time [▶ 87]"-Variable.

Beispiel:

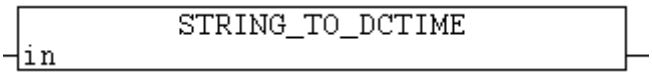
```
PROGRAM P_TEST
VAR
    dcStruct : DCTIMESTRUCT;
    dcTime   : T_DCTIME;
END_VAR
```

```
dcTime := F_GetCurDcTickTime();
dcStruct := DCTIME_TO_DCTIMESTRUCT(dcTime);
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 Build > 1316 oder höher	PC oder CX (x86, ARM)	TcEtherCAT.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib, TcUtilities.Lib werden automatisch eingebunden)

4.5 STRING_TO_DCTIME



Die Funktion konvertiert einen String in die "Distributed Clock System Time"-Variable.

FUNCTION STRING_TO_DCTIME: T_DCTIME

T_DCTIME [▶ 87]

```
VAR_INPUT
    in : STRING(29);
END_VAR
```

in: Der zu konvertierende String muss folgendes Format haben: **'YYYY-MM-DD-hh:mm:ss.nnnnnnnnn'**

- YYYY: Jahr;
- MM: Monat;
- DD: Tag;
- hh: Stunde;
- mm: Minute;
- ss: Sekunde;
- nnnnnnnnn: Nanosekunden;

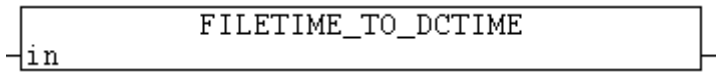
Beispiel:

Siehe in der Beschreibung der Funktion: F_GetCurDcTickTime [▶ 28].

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 Build > 1310 oder höher	PC oder CX (x86, ARM)	TcEtherCAT.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib, TcUtilities.Lib werden automatisch eingebunden)

4.6 FILETIME_TO_DCTIME



Die Funktion konvertiert die 64 bit "Windows File Time"-Variable in eine 64 bit "Distributed Clock System Time"-Variable. Beim Konvertierungsfehler liefert die Funktion den Wert Null zurück.

FUNCTION FILETIME_TO_DCTIME: T_DCTIME

T_DCTIME [▶ 87]

```
VAR_INPUT
    in : T_FILETIME;
END_VAR
```

in: Die zu konvertierende "Windows File Time"-Variable.

Beispiel:

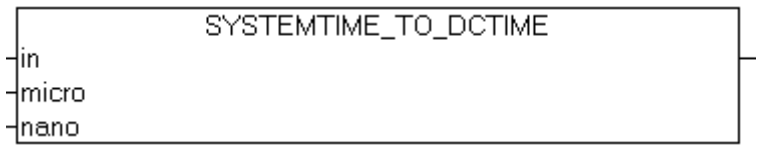
```
PROGRAM P_TEST
VAR
    fbSysFileTime    : GETSYSTEMTIME;
    ft               : T_FILETIME;
    dct              : T_DCTIME;
END_VAR
```

```
fbSysFileTime(timeLoDW=>ft.dwLowDateTime, timeHiDW=>ft.dwHighDateTime);
dct := FILETIME_TO_DCTIME(ft);
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 Build >= 1324 oder höher	PC oder CX (x86, ARM)	TcEtherCAT.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib, TcUtilities.Lib werden automatisch eingebunden)

4.7 SYSTEMTIME_TO_DCTIME



Die Funktion konvertiert die strukturierte "Windows System Time"-Variable in eine 64 bit "Distributed Clock System Time"-Variable. Beim Konvertierungsfehler liefert die Funktion den Wert Null zurück.

FUNCTION SYSTEMTIME_TO_DCTIME: T_DCTIME

T_DCTIME [▶ 87]

```
VAR_INPUT
    in    : TIMESTRUCT;
    micro : WORD(0..999); (* Microseconds: 0..999 *)
    nano  : WORD(0..999); (* Nanoseconds: 0..999 *)
END_VAR
```

in: Die zu konvertierende "Windows System Time"-Variable.

micro: Microsekunden.

nano: Nanosekunden.

Beispiel:

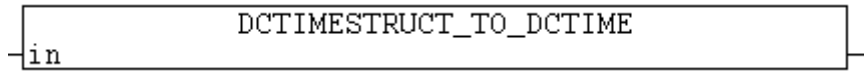
```
PROGRAM P_TEST
VAR
    syst : TIMESTRUCT := ( wYear := 2009, wMonth := 9, wDay := 16,    wHour := 12, wMinute := 22, wS
econd := 44, wMilliseconds := 123 );
END_VAR
```

```
dct := SYSTEMTIME_TO_DCTIME( syst, 456, 789 );
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 Build > 1340 TwinCAT v2.11.0 Build > 1536	PC oder CX (x86, ARM)	TcEtherCAT.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib, TcUtilities.Lib werden automatisch eingebunden)

4.8 DCTIMESTRUCT_TO_DCTIME



Die Funktion konvertiert die strukturierte Variable vom Typ: DCTIMESTRUCT [▶ 78] in eine 64 bit "Distributed Clock System Time"-Variable.
Die Strukturkomponente wDayOfWeek wird bei der Konvertierung ignoriert. Die wYear-Strukturkomponente muss einen Wert grösser oder gleich 2000 oder kleiner 2584 haben. Bei unzulässigen Werten der Strukturkomponenten liefert die Funktion den Wert Null zurück.

FUNCTION DCTIMESTRUCT_TO_DCTIME: T_DCTIME

T_DCTIME [▶ 87]

```
VAR_INPUT
    in : DCTIMESTRUCT;
END_VAR
```

in: Die zu konvertierende strukturierte [▶ 78] Variable.

Beispiel:

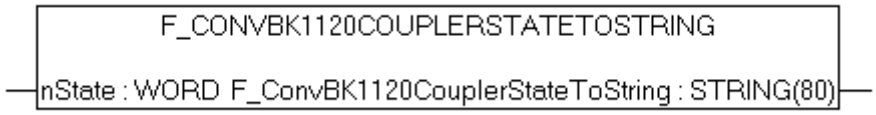
```
PROGRAM P_TEST
VAR
    dcStruct : DCTIMESTRUCT := (    wYear := 2008, wMonth := 3, wDay := 13,
                    wHour := 1, wMinute := 2, wSecond :=3,
                    wMilliseconds := 123, wMicroseconds := 456, wNanoseconds := 789 );
    dc64    : T_DCTIME;
END_VAR
```

```
dc64 := DCTIMESTRUCT_TO_DCTIME( dcStruct );
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 Build >= 1324 oder höher	PC oder CX (x86, ARM)	TcEtherCAT.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib, TcUtilities.Lib werden automatisch eingebunden)

4.9 F_ConvBK1120CouplerStateToString



Die Funktion F_ConvBK1120CouplerStateToString liefert den Kopplerstatus des BK1120/BK1250 als String.

VAR_INPUT

```
VAR_INPUT
    nState : WORD;
END_VAR
```

nState: Kopplerstatus (CouplerState), kann im System Manager Von den Eingängen des BK1120/BK1250 in die SPS gelinkt werden.

```
    0x0000 = Kein Fehler
    0x0001 = K-Bus-Fehler
    0x0002 = Konfigurationsfehler
    0x0010 = Ausgänge deaktiviert
    0x0020 = K-Bus-Überlauf
    0x0040 = Kommunikationsfehler (Eingänge)
    0x0080 = Kommunikationsfehler (Ausgänge)
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 oder höher	PC or CX (x86)	TcEtherCAT.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib, TcUtilities.Lib werden automatisch eingebunden)
TwinCAT v2.10.0 Build >= 1301 oder höher	CX (ARM)	

4.10 F_ConvMasterDevStateToString

Die Funktion F_ConvMasterDevStateToString liefert den Gerätestatus des EtherCAT Masters als String.

VAR_INPUT

```
VAR_INPUT
    nState : WORD;
END_VAR
```

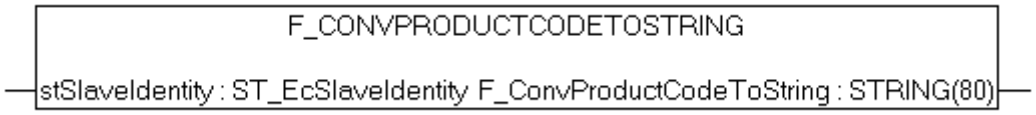
nstate: Gerätestatus des EtherCAT Masters, kann als DevState im System Manager von den Eingängen des EtherCAT Masters in die SPS geinkt werden

```
    0x0001 = Link-Fehler
    0x0002 = I/O geblockt nach Link Fehler (I/O Reset erforderlich)
    0x0004 = Link-Fehler (Redundanzadapter)
    0x0008 = Fehlender Frame (Redundanz-Modus)
    0x0010 = Nicht genügend Sende-Resourcen (I/O Reset erforderlich)
    0x0020 = Zeitüberwachungs(Watchdog)-Fehler
    0x0040 = Ethernet-Treiber (Miniport) nicht
gefunden
    0x0080 = I/O Reset aktiv
    0x0100 = Mindenstens eine Slave in
'INIT'
    0x0200 = Mindenstens eine Slave in
'PRE-OP'
    0x0400 = Mindenstens eine Slave in
'SAFE-OP'
    0x0800 = Mindenstens eine Slave in im
Fehler
    0x1000 = Distributed Clocks nicht
synchronisiert
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 oder höher	PC or CX (x86)	TcEtherCAT.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib, TcUtilities.Lib werden automatisch eingebunden)
TwinCAT v2.10.0 Build >= 1301 oder höher	CX (ARM)	

4.11 F_ConvProductCodeToString



Die Funktion F_ConvProductToString liefert den ProductCode als String, z.B. 'EL6731-0000-0017'.

VAR_INPUT

```
VAR_INPUT
    stSlaveIdentity : ST_EcSlaveIdentity;
END_VAR
```

stSlaveIdentity: Die SlaveIdentity, wie sie mit dem FB_EcGetSlaveIdentity [▶ 47] eingelesen werden kann.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 oder höher	PC or CX (x86)	TcEtherCAT.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib, TcUtilities.Lib werden automatisch eingebunden)
TwinCAT v2.10.0 Build >= 1301 oder höher	CX (ARM)	

4.12 F_ConvSlaveStateToString

Die Funktion F_ConvSlaveStateToString liefert den EtherCAT Slave State als String.

VAR_INPUT

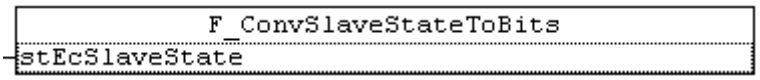
```
VAR_INPUT
    state : ST_EcSlaveState;
END_VAR
```

state: EtherCAT Slave State Struktur (bestehend aus: deviceState : BYTE; linkState : BYTE;)

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 oder höher	PC or CX (x86)	TcEtherCAT.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib, TcUtilities.Lib werden automatisch eingebunden)
TwinCAT v2.10.0 Build >= 1301 oder höher	CX (ARM)	

4.13 F_ConvSlaveStateToBits



Die Funktion F_ConvSlaveStateToBits liefert den EtherCAT Slave State als Struktur TYPE ST_EcSlaveStateBits [▶ 85].

VAR_INPUT

```
VAR_INPUT
    stEcSlaveState : ST_EcSlaveState;
END_VAR
```

stEcSlaveState: EtherCAT Slave State Struktur (bestehend aus: deviceState : BYTE; linkState : BYTE;)

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 oder höher	PC or CX (x86)	TcEtherCAT.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib, TcUtilities.Lib werden automatisch eingebunden)
TwinCAT v2.10.0 Build >= 1301 oder höher	CX (ARM)	

4.14 F_ConvStateToString

Die Funktion F_ConvStateToString liefert den EtherCAT Slave State als String.

VAR_INPUT

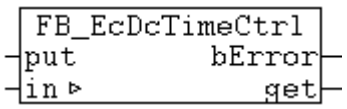
```
VAR_INPUT
    nState : WORD;
END_VAR
```

Name	Typ	Beschreibung
nState	WORD	EtherCAT Slave State als WORD 0x___1 = Slave ist in 'INIT' 0x___2 = Slave ist in 'PREOP' 0x___3 = Slave ist in 'BOOT' 0x___4 = Slave ist in 'SAFEOP' 0x___8 = Slave ist in 'OP' 0x001_ = Slave signalisiert Fehler 0x002_ = Ungültige VendorId, ProductCode...Gefunden 0x004_ = Initialisierungsfehler aufgetreten 0x010_ = Slave nicht vorhanden 0x020_ = Slave signalisiert Link-Fehler 0x040_ = Slave signalisiert fehlenden Link 0x080_ = Slave signalisiert unerwarteten Link 0x100_ = Kommunikations-Port A 0x200_ = Kommunikations-Port B 0x400_ = Kommunikations-Port C 0x800_ = Kommunikations-Port D

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 oder höher	PC or CX (x86)	TcEtherCAT.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib, TcUtilities.Lib werden automatisch eingebunden)
TwinCAT v2.10.0 Build >= 1301 oder höher	CX (ARM)	

4.15 FB_EcDcTimeCtrl



Mit diesem Funktionsbaustein können die einzelnen Komponenten wie Jahr, Monat, Tag usw. einer 64 Bit TwinCAT "Distributed Clock System Time"-Variablen gelesen werden. Der Funktionsbaustein besitzt mehrere A_GetXYZ-Aktionen. Nach dem Aufruf der gewünschten Aktion steht der Wert der XYZ-Komponente in der *get*-Ausgangsvariablen zur Verfügung. Die put-Eingangsvariable wird zurzeit nicht benutzt.

Der Funktionsbaustein besitzt zurzeit folgende Aktionen:

- A_GetYear;
- A_GetMonth;

- A_GetDay;
- A_GetDayOfWeek;
- A_GetHour;
- A_GetMinute;
- A_GetSecond;
- A_GetMilli;
- A_GetMicro;
- A_GetNano;

VAR_IN_OUT

```
VAR_IN_OUT
    in          : T_DCTIME;
END_VAR
```

in: TwinCAT "Distributed Clock System Time [▶ 87]"-Variable;

VAR_INPUT

```
VAR_INPUT
    put         : WORD;
END_VAR
```

put: Eingangsparameter (zur Zeit nicht benutzt);

VAR_OUTPUT

```
VAR_OUTPUT
    bError      : BOOL;
    get         : WORD;
END_VAR
```

bError: Dieser Ausgang wird gesetzt wenn ein Fehler beim Aktionsaufruf aufgetreten ist;

get: Ausgangsparameter (Jahr, Monat, Tag usw.);

Beispiel für eine Implementierung in ST:

```
PROGRAM P_TEST
VAR
    dcStruct    : DCTIMESTRUCT;
    dcTime      : T_DCTIME;
    fbCtrl      : FB_EcDcTimeCtrl;

    wYear       : WORD;
    wMonth      : WORD;
    wDay        : WORD;
    wDayOfWeek  : WORD;
    wHour       : WORD;
    wMinute     : WORD;
    wSecond     : WORD;
    wMilli      : WORD;
    wMicro      : WORD;
    wNano       : WORD;
END_VAR
```

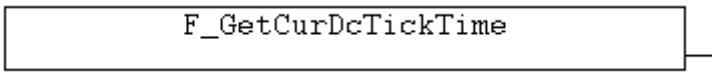
```
dcTime := F_GetCurDcTickTime();

fbCtrl.A_GetYear( in := dcTime, get => wYear );
fbCtrl.A_GetMonth( in := dcTime, get => wMonth );
fbCtrl.A_GetDay( in := dcTime, get => wDay );
fbCtrl.A_GetDayOfWeek( in := dcTime, get => wDayOfWeek );
fbCtrl.A_GetHour( in := dcTime, get => wHour );
fbCtrl.A_GetMinute( in := dcTime, get => wMinute );
fbCtrl.A_GetSecond( in := dcTime, get => wSecond );
fbCtrl.A_GetMilli( in := dcTime, get => wMilli );
fbCtrl.A_GetMicro( in := dcTime, get => wMicro );
fbCtrl.A_GetNano( in := dcTime, get => wNano );
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 Build > 1316 oder höher	PC oder CX (x86, ARM)	TcEtherCAT.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib, TcUtilities.Lib werden automatisch eingebunden)

5 Distributed Clocks

5.1 F_GetCurDcTickTime



Diese Funktion liefert die Zeit des aktuellen (letzten) Ticks im TwinCAT Distributed Clock Systemzeitformat.

FUNCTION F_GetCurDcTickTime : T_DCTIME

T_DCTIME [▶ 87]

```
VAR_INPUT
(*none*)
END_VAR
```

Beispiel in ST:

```
PROGRAM MAIN
VAR
    tDC : T_DCTIME;
    sDC : STRING;
    tDCBack : T_DCTIME;

    sDCZero : STRING;(* DCTIME = zero time starts on 01.01.2000 *)
    tDCBackFromZero : T_DCTIME;

    tDCFromString : T_DCTIME;
    sDCBackFromString : STRING;
END_VAR
```

```
tDC := F_GetCurDcTickTime();
sDC := DCTIME_TO_STRING( tDC );
tDCBack := STRING_TO_DCTIME( sDC );

sDCZero := DCTIME_TO_STRING( ULARGE_INTEGER(0, 0) );
tDCBackFromZero := STRING_TO_DCTIME( sDCZero );

tDCFromString := STRING_TO_DCTIME( '2007-03-09-11:31:09.223456789' );
sDCBackFromString := DCTIME_TO_STRING( tDCFromString );
```

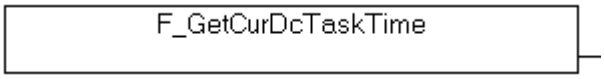
Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 Build > 1310 oder höher	PC oder CX (x86, ARM)	TcEtherCAT.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib, TcUtilities.Lib werden automatisch eingebunden)

Sehen Sie dazu auch

- DCTIME_TO_STRING [▶ 17]
- STRING_TO_DCTIME [▶ 19]

5.2 F_GetCurDcTaskTime



Diese Funktion liefert die Startzeit der Task (Zeitpunkt zu dem die Task starten sollte) im TwinCAT Distributed Clock Systemzeitformat. Die Funktion liefert immer die Startzeit der Task in der sie aufgerufen wurde.

FUNCTION F_GetCurDcTaskTime : T_DCTIME

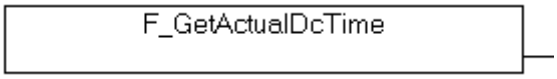
T_DCTIME [▶ 87]

```
VAR_INPUT
(*none*)
END_VAR
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.11.0 Build >= 1524 oder höher	PC oder CX (x86, ARM)	TcEtherCAT.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib, TcUtilities.Lib werden automatisch eingebunden)

5.3 F_GetActualDcTime



Diese Funktion liefert die aktuelle Zeit im TwinCAT Distributed Clock Systemzeitformat.

FUNCTION F_GetActualDcTime: T_DCTIME

T_DCTIME [▶ 87]

```
VAR_INPUT
(*none*)
END_VAR
```

Beispiel in ST:

```
PROGRAM MAIN
VAR
    actDC : T_DCTIME;
    sAct : STRING;
END_VAR
```

```
actDC := F_GetActualDcTime();
sAct := DCTIME_TO_STRING( actDC );
```

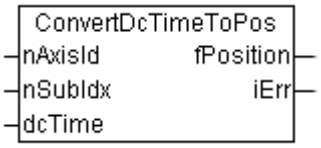
Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.11.0 Build > 1535 oder höher	PC oder CX (x86, ARM) SPS Laufzeitsystem: TwinCAT v2.11 Build > 1535	TcEtherCAT.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib, TcUtilities.Lib werden automatisch eingebunden)

Sehen Sie dazu auch

DCTIME_TO_STRING [▶ 17]

5.4 ConvertDcTimeToPos



Dieser Funktionsblock konvertiert eine 32 Bit *Distributed Clock System Time* in eine zugehörige NC Achsposition (d.h. diejenige NC Achsposition, die genau zu diesem Zeitpunkt vorgelegen hat bzw. vorliegen wird).

VAR_INPUT

```
VAR_INPUT
    nAxisId : UDINT;
    nSubIdx : UDINT;
    dcTime  : T_DCTIME32;(* 32 bit distributed clock time *)
END_VAR
```

nAxisId	ID der NC-Achse.
nSubIdx	Diese 32 Bit Eingangsgröße beinhaltet zwei verschiedene Informationen und unterteilt sich somit in zwei 16 Bit Werte: Das LowWord (die niederwertigsten 16 Bit) beinhaltet den Sub-Index zum relativen Adressieren eines Encoder-Unterelementes an einer Achse. Der Sub-Index wird von Null an gezählt. Für den typischen Fall einer Achse mit genau einem Encoder ist der Sub-Index Null richtig. Das HighWord (die höchstwertigen 16 Bit) beinhaltet ein Steuerwort (Bitmaske), das die Art der Positionsberechnung beeinflusst (z.B. den Inter- bzw. Extrapolationstyp). Die Bitmaske 0x0001 bedeutet, dass die Soll-Beschleunigung der Achse mit in die Berechnung einbezogen werden soll.
dcTime	32 Bit Distributed Clock System Time (T_DCTIME32 [▶ 87]). Diese Eingangsgröße wird in die korrespondierende NC-Achsposition umgerechnet. **Diese 32 Bit Zeit darf nur im zeitlichen Nahbereich von ± 2.147 Sekunden um die aktuelle Systemzeit verwendet wird, da sie nur hier eindeutig ist. Innerhalb des FB's kann diese Voraussetzung nicht überprüft werden.**

VAR_OUTPUT

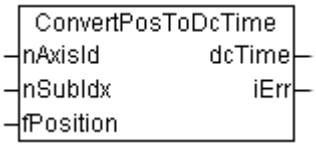
```
VAR_OUTPUT
    fPosition : LREAL;
    iErr      : UDINT;
END_VAR
```

fPosition	Liefert die zur *dcTime* korrespondierende NC Achsposition. Hierbei handelt es sich um eine um die Skalierung und Offset verrechnete NC-Achsposition mit z.B. der physikalischen Einheit Grad oder mm.
iErr	Liefert im Fehlerfall eine Fehlernummer, z.B. Fehler 0x4012 (Achs-ID ist nicht erlaubt bzw. Achse ist im System nicht vorhanden).

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.11.0 Build >= 1524 oder höher	PC oder CX (x86, ARM)	TcEtherCAT.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib, TcUtilities.Lib werden automatisch eingebunden)

5.5 ConvertPosToDcTime



Dieser Funktionsblock konvertiert eine NC-Achsposition in eine zugehörige 32 Bit *Distributed Clock System Time* (d. h. derjenige Zeitpunkt, zu dem genau diese NC Achsposition erreicht wurde bzw. erreicht wird).

VAR_INPUT

```
VAR_INPUT
    nAxisId         : UDINT;
    nSubIdx         : UDINT;
    fPosition       : LREAL;
END_VAR
```

nAxisId	ID der NC-Achse.
nSubIdx	Diese 32 Bit Eingangsgröße setzt sich aus zwei verschiedenen Informationen zusammen und unterteilt sich in zwei 16 Bit Werte: Das LowWord (die niederwertigsten 16 Bit) beinhaltet den Sub-Index zum relativen Adressieren eines Encoder-Unterelementes an einer Achse. Der Sub-Index wird von Null an gezählt. Für den typischen Fall einer Achse mit genau einem Encoder ist der Sub-Index Null richtig. Das HighWord (die höchstwertigen 16 Bit) beinhaltet ein Steuerwort (Bitmaske), das die Art der Positionsberechnung beeinflusst (z. B. den Inter- bzw. Extrapolationstyp). Die Bitmaske 0x0001 bedeutet dass die Soll-Beschleunigung der Achse mit in die Berechnung einbezogen werden soll.
fPosition	NC-Achsposition, die in die korrespondierende 32 Bit Distributed Clock System Time (T_DCTIME32 [▶ 87]) umgerechnet wird. Liegt die zur Position zugehörige DC System Time außerhalb des erwarteten Zeitfensters von ± 2.147 Sekunden wird diese Umrechnung mit einer Fehlernummer abgelehnt.

VAR_OUTPUT

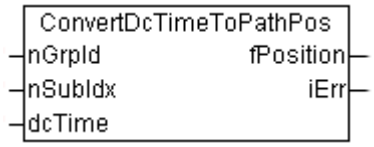
```
VAR_OUTPUT
    dcTime   : T_DCTIME32;(* 32 bit distributed clock time *)
    iErr     : UDINT;
END_VAR
```

dcTime	Liefert die zum Eingang *fPosition* zugehörige 32 Bit Distributed Clock System Time (T_DCTIME32 [▶ 87]).
iErr	Liefert im Fehlerfall eine Fehlernummer, z.B. - Fehler 0x4012: Achs-ID ist nicht erlaubt bzw. Achse ist im System nicht vorhanden, - Fehler 0x4361: Zeitbereichsüberschreitung (Zukunft), - Fehler 0x4362: Zeitbereichsüberschreitung (Vergangenheit), - Fehler 0x4363: Position ist mathematisch nicht zu ermitteln.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.11.0 Build >= 1524 oder höher	PC oder CX (x86, ARM)	TcEtherCAT.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib, TcUtilities.Lib werden automatisch eingebunden)

5.6 ConvertDcTimeToPathPos



Dieser Funktionsblock konvertiert eine 32 Bit *Distributed Clock System Time* in einen relativen Nci-Wegabstand auf der Kontur des momentan aktiven Nci-Programms (d.h. je nach Zeitpunkt, liefert der FB einen positiven oder negativen relativen Abstand zurück).

VAR_INPUT

```
VAR_INPUT
    nGrpId      : UDINT;
    nSubIdx     : UDINT;
    dcTime      : T_DCTIME32;(* 32 bit distributed clock time *)
END_VAR
```

nGrpId	Group ID des zugehörigen Nci-Kanal
nSubIdx	Diese 32 Bit Eingangsgröße beinhaltet zwei verschiedene Informationen und unterteilt sich somit in zwei 16 Bit Werte: Das LowWord (die niederwertigsten 16 Bit) beinhaltet den Sub-Index zum relativen Adressieren eines Encoder-Unterelementes an einer Achse. Der Sub-Index wird von Null an gezählt. Für den typischen Fall einer Achse mit genau einem Encoder ist der Sub-Index Null richtig. Das HighWord (die höchstwertigen 16 Bit) beinhaltet ein Steuerwort (Bitmaske), das die Art der Positionsberechnung beeinflusst (z.B. den Inter- bzw. Extrapolationstyp). Die Bitmaske 0x0001 bedeutet, dass die Soll-Beschleunigung der Achse mit in die Berechnung einbezogen werden soll. **Die Bitmaske 0x0010 bedeutet, dass die Berechnung relativ erfolgt und ist momentan zwingend. Der Aufruf wird anderenfalls mit Fehler abgelehnt.**
dcTime	32 Bit Distributed Clock System Time (T_DCTIME32 [▶ 87]). Diese Eingangsgröße wird in die korrespondierende relative Nci-Wegabstand auf der Kontur umgerechnet. **Diese 32 Bit Zeit darf nur im zeitlichen Nahbereich von±2.147 Sekunden um die aktuelle Systemzeit verwendet werden, da sie nur hier eindeutig ist. Innerhalb des FB kann diese Voraussetzung nicht überprüft werden.**

VAR_OUTPUT

```
VAR_OUTPUT
    fPosition   : LREAL;
    iErr        : UDINT;
END_VAR
```

fPosition	Liefert die zur *dcTime* korrespondierende relative Nci-Wegabstand auf der Kontur.
iErr	Liefert im Fehlerfall eine Fehlernummer

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.11.0 Build > 2214 oder höher	PC oder CX (x86, ARM)	TcEtherCAT.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib, TcUtilities.Lib werden automatisch eingebunden)

5.7 ConvertPathPosToDcTime

Dieser Funktionsblock konvertiert einen relativen Nci-Wegabstand in eine zugehörige 32 Bit *Distributed Clock System Time* (d. h. derjenige zugehörige Zeitpunkt, der dem relativen Nci-Wegabstand entspricht bzw. entsprach).

VAR_INPUT

```
VAR_INPUT
    nGrpId      : UDINT;
    nSubIdx     : UDINT;
    fPosition   : LREAL;
END_VAR
```

nGrpId	Group ID des zugehörigen Nci-Kanal
nSubIdx	Diese 32 Bit Eingangsgröße beinhaltet zwei verschiedene Informationen und unterteilt sich somit in zwei 16 Bit Werte: Das LowWord (die niederwertigsten 16 Bit) beinhaltet den Sub-Index zum relativen Adressieren eines Encoder-Unterelementes an einer Achse. Der Sub-Index wird von Null an gezählt. Für den typischen Fall einer Achse mit genau einem Encoder ist der Sub-Index Null richtig. Das HighWord (die höchstwertigen 16 Bit) beinhaltet ein Steuerwort (Bitmaske), das die Art der Positionsberechnung beeinflusst (z.B. den Inter- bzw. Extrapolationstyp). Die Bitmaske 0x0001 bedeutet, dass die Soll-Beschleunigung der Achse mit in die Berechnung einbezogen werden soll. **Die Bitmaske 0x0010 bedeutet, dass die Berechnung relativ erfolgt und ist momentan zwingend. Der Aufruf wird anderenfalls mit Fehler abgelehnt.**
fPosition	Relativer Nci-Wegabstand, der in die korrespondierende 32 Bit Distributed Clock System Time (T_DCTIME32 [▶ 87]) umgerechnet wird. Liegt die zum relativen Nci-Wegabstand zugehörige DC System Time außerhalb des erwarteten Zeitfensters von ± 2.147 Sekunden wird diese Umrechnung mit einer Fehlernummer abgelehnt.

VAR_OUTPUT

```
VAR_OUTPUT
    dcTime  : T_DCTIME32;(* 32 bit distributed clock time *)
    iErr    : UDINT;
END_VAR
```

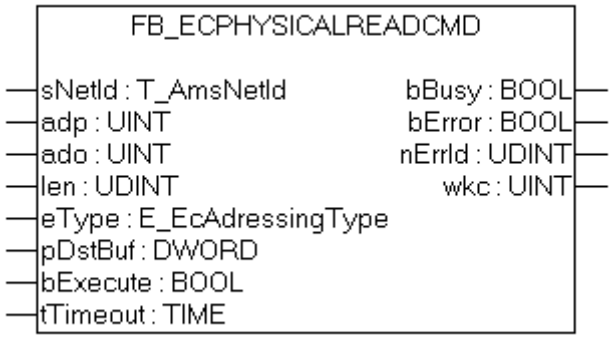
dcTime	Liefert die zum Eingang *fPosition* zugehörige 32 Bit Distributed Clock System Time (T_DCTIME32 [▶ 87]).
iErr	Liefert im Fehlerfall eine Fehlernummer, z.B. - Fehler 0x4361: Zeitbereichsüberschreitung (Zukunft), - Fehler 0x4362: Zeitbereichsüberschreitung (Vergangenheit), - Fehler 0x4363: Position ist mathematisch nicht zu ermitteln.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.11.0 Build > 2214 oder höher	PC oder CX (x86, ARM)	TcEtherCAT.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib, TcUtilities.Lib werden automatisch eingebunden)

6 EtherCAT Commands

6.1 FB_EcPhysicalReadCmd



Mit dem Funktionsbaustein FB_EcPhysicalReadCmd kann ein EtherCAT Lese-Kommando(FPRD, APRD, BRD) an einen bestimmten EtherCAT-Slave oder an alle EtherCAT-Slaves gesendet werden. Hiermit kann die Sps Register oder DPRAM des EtherCAT Slave Controllers auslesen.

VAR_INPUT

```
VAR_INPUT
    sNetId   : T_AmsNetId;
    adp      : UINT;
    ado      : UINT;
    len      : UDINT;
    eType    : E_EcAdressingType := eAdressingType_Fixed;
    pDstBuf  : DWORD;
    bExecute : BOOL;
    tTimeout : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

sNetId: Ist ein String, der die AMS-Netzwerkkennung des EtherCAT Master Gerätes enthält.

adp: Dieser Wert bestimmt, welcher EtherCAT-Slave mit diesem Kommando adressiert werden soll. Die Bedeutung dieses Wertes hängt von der mit eType ausgewählten Adressierungsart ab:

eType	Beschreibung
eAdressingType_Fixed	Der Slave wird an Hand seiner konfigurierten EtherCAT-Adresse adressiert. Diese EtherCAT-Adressen können mit Hilfe des Bausteins FB_EcGetAllSlaveAddr ausgelesen werden.
eAdressingType_AutoInc	Der Slave wird an Hand seiner Position im Ring adressiert. Der erste Teilnehmer hat die Adresse 0(adp=0) und für alle darauf folgenden Slaves wird adp um eins dekrementiert: 1.Slave adp = 0 2.Slave adp = 16#ffff (-1) 3.Slave adp = 16#fffe(-2) 4.Slave adp = 16#fffd(-3) etc.
eAdressingType_BroadCAST	Alle Slaves werden von diesem Kommando adressiert. Adp kann auf 0 gesetzt werden.

ado: Physikalischer Speicher(DPRAM) oder Register das ausgelesen werden soll.

len: Anzahl der zu lesenden Bytes.

eType: Abhängig des Wertes von eType werden verschiedene EhterCAT-Kommandos geschickt:

eType	Kommando
eAdressingType_Fixed	Configured Address Physical Read (FPRD)
eAdressingType_AutoInc	Auto Increment Physical Read (APRD)
eAdressingType_BroadCAST	Broadcast Read (BRD)

Die einzelnen Kommandos unterscheiden sich in der Adressierungsart (siehe adp).

pDstBuf: Die Adresse (Pointer) auf den Empfangspuffer.

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

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

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy    : BOOL;
    bError   : BOOL;
    nErrId   : UDINT;
    wkc      : UINT;
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.

nErrId: Liefert bei einem gesetzten bError-Ausgang den ADS-Fehlercode des zuletzt ausgeführten Befehles

wkc: Der Working Counter wird von jedem EtherCAT-Slave der dieses Kommando erfolgreich bearbeitet hat, um eins inkrementiert. Wenn nur ein EtherCAT-Slave von diesem Kommando adressiert worden ist, sollte dieser Wert somit 1 entsprechen.

Beispiel für eine Implementierung in ST:

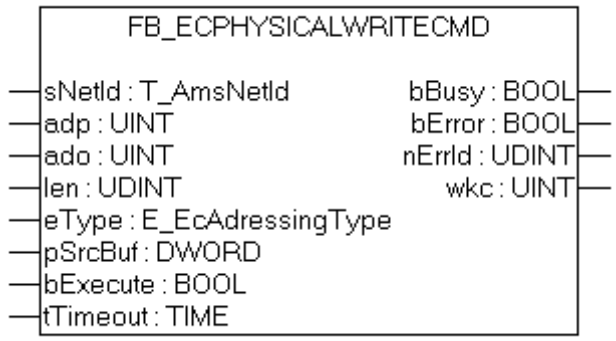
```
PROGRAM TEST_PhysicalReadCmd
VAR
    fbReadCmd    : FB_EcPhysicalReadCmd;
    bExecute     : BOOL;
    value        : UINT;
    adp          : UINT:=16#3E9;
    ado          : UINT:=16#1100;
    eType        : E_EcAdressingType := eAdressingType_Fixed;
    sNetId       : T_AmsNetId:='192.168.1.5.3.1';

    wkc          : UINT;
    bError       : BOOL;
    nErrId       : UDINT;
END_VAR
```

```
fbReadCmd (sNetId:=sNetID, ado:=ado, adp:=adp, eType:=eType, LEN := SIZEOF(value), pDstBuf:=ADR(valu
e), bExecute:=bExecute);
wkc := fbReadCmd.wkc;
bError:=fbReadCmd.bError;
nErrId:=fbReadCmd.nErrId;
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 oder höher	PC or CX (x86)	TcEtherCAT.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib, TcUtilities.Lib werden automatisch eingebunden)
TwinCAT v2.10.0 Build >= 1301 oder höher	CX (ARM)	

6.2 FB_EcPhysicalWriteCmd



Mit dem Funktionsbaustein FB_EcPhysicalWriteCmd kann ein EtherCAT Write-Kommando(FPWR, APWR, BWR) an einen bestimmten EtherCAT-Slave oder an alle EtherCAT-Slaves gesendet werden. Hiermit kann ein Sps Register oder DPRAM des EtherCAT Slave Controllers beschrieben werden.

VAR_INPUT

```
VAR_INPUT
    sNetId   : T_AmsNetId;
    adp      : UINT;
    ado      : UINT;
    len      : UDINT;
    eType    : E_EcAdressingType := eAdressingType_Fixed;
    pSrcBuf  : DWORD;
    bExecute : BOOL;
    tTimeout : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

sNetId: Ist ein String, der die AMS-Netzwerkkennung des EtherCAT Master Gerätes enthält.

adp: Dieser Wert bestimmt, welcher EtherCAT-Slave mit diesem Kommando adressiert werden soll. Die Bedeutung dieses Wertes hängt von der mit eType ausgewählten Adressierungsart ab:

eType	Beschreibung
eAdressingType_Fixed	Der Slave wird an Hand seiner konfigurierten EtherCAT-Adresse adressiert. Diese EtherCAT-Adressen können mit Hilfe des Bausteins FB_EcGetAllSlaveAddr ausgelesen werden.
eAdressingType_AutoInc	Der Slave wird an Hand seiner Position im Ring adressiert. Der erste Teilnehmer hat die Adresse 0(adp=0) und für alle darauf folgenden Slaves wird adp um eins dekrementiert: 1.Slave adp = 0 2.Slave adp = 16#ffff (-1) 3.Slave adp = 16#fffe(-2) 4.Slave adp = 16#fffd(-3) etc.
eAdressingType_BroadCAST	Alle Slaves werden von diesem Kommando adressiert. Adp sollte auf 0 gesetzt werden.

ado: Physikalischer Speicher(DPRAM) oder Register das ausgelesen werden soll.

len: Anzahl der zu schreibenden Bytes.

eType: Abhängig des Wertes von eType werden verschiedene EhterCAT-Kommandos geschickt:

eType	Kommando
eAdressingType_Fixed	Configured Address Physical Write (FPWR)
eAdressingType_AutoInc	Auto Increment Physical Write (APWR)

eType	Kommando
eAdressingType_BroadCAST	Broadcast Write (BWR)

Die einzelnen Kommandos unterscheiden sich in der Adressierungsart (siehe adp).

pSrcBuf: Die Adresse (Pointer) auf den Sendepuffer.

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

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

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy    : BOOL;
    bError   : BOOL;
    nErrId   : UDINT;
    wkc      : UINT;
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.

nErrId: Liefert bei einem gesetzten bError-Ausgang den ADS-Fehlercode des zuletzt ausgeführten Befehles

wkc: Der Working Counter wird von jedem EtherCAT-Slave der dieses Kommando erfolgreich bearbeitet hat, um eins inkrementiert. Wenn nur ein EtherCAT-Slave von diesem Kommando adressiert worden ist, sollte dieser Wert somit 1 entsprechen.

Beispiel für eine Implementierung in ST:

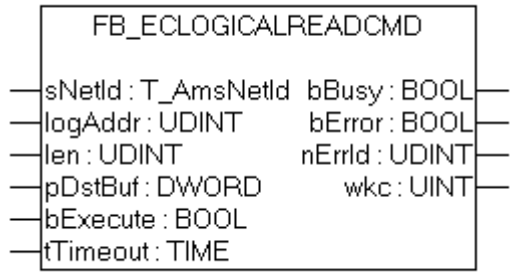
```
PROGRAM Test_PhysicalWriteCmd
VAR
    fbWriteCmd  : FB_EcPhysicalWriteCmd;
    bExecute    : BOOL;
    value       : UINT :=16#5555;
    adp         : UINT:=16#3E9;
    ado         : UINT:=16#1100;
    eType       : E_EcAdressingType := eAdressingType_Fixed;
    sNetId      : T_AmsNetId:='192.168.1.5.3.1';

    wkc         : UINT;
    bError      : BOOL;
    nErrId      : UDINT;
END_VAR
```

```
fbWriteCmd (sNetId:=sNetID, ado:=ado, adp:=adp, eType:=eType, LEN := SIZEOF(value), pSrcBuf:=ADR(val
ue), bExecute:=bExecute);
wkc := fbWriteCmd.wkc;
bError:=fbWriteCmd.bError;
nErrId:=fbWriteCmd.nErrId;
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 oder höher	PC or CX (x86)	TcEtherCAT.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib, TcUtilities.Lib werden automatisch eingebunden)
TwinCAT v2.10.0 Build >= 1301 oder höher	CX (ARM)	

6.3 FB_EcLogicalReadCmd



Mit dem Funktionsbaustein FB_EcLogicalReadCmd wird ein logisches EtherCAT Write-Kommando(LRD) vom Master gesendet. In jedem Slave können lokale Adressbereiche(DPRAM) auf globale logische Adressebereich gemappt werden. Somit adressiert diese Kommandos alle EtherCAT-Slave, die ein Mapping für den ausgewählten logischen Adressbereich konfiguriert haben.

VAR_INPUT

```
VAR_INPUT
    sNetId   : T_AmsNetId;
    logAddr  : UDINT;
    len      : UDINT;
    pDstBuf  : DWORD;
    bExecute : BOOL;
    tTimeout : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

sNetId: Ist ein String, der die AMS-Netzwerkkennung des EtherCAT Master Gerätes enthält.

logAddr: Logische Adresse.

len: Anzahl der zu lesenden Bytes.

pDstBuf: Die Adresse (Pointer) auf den Empfangspuffer.

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

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

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy   : BOOL;
    bError  : BOOL;
    nErrId  : UDINT;
    wkc     : UINT;
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.

nErrId: Liefert bei einem gesetzten bError-Ausgang den ADS-Fehlercode des zuletzt ausgeführten Befehles

wkc: Der Working Counter wird von jedem EtherCAT-Slave der dieses Kommando erfolgreich bearbeitet hat, um eins inkrementiert. Wenn nur ein EtherCAT-Slave von diesem Kommando adressiert worden ist, sollte dieser Wert somit 1 entsprechen.

Beispiel für eine Implementierung in ST:

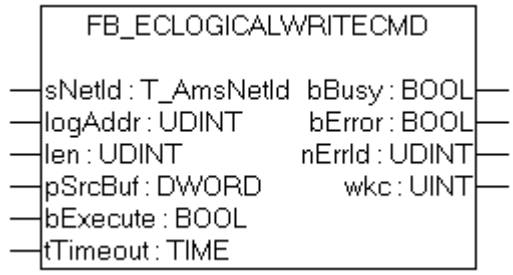
```
PROGRAM Test_LogicalReadCmd
VAR
    fbReadCmd   : FB_EcLogicalReadCmd;
    bExecute    : BOOL;
    value       : USINT;
    logAddr     : UDINT :=16#10000;
    sNetId      : T_AmsNetId:='192.168.1.5.3.1';

    wkc         : UINT;
    bError      : BOOL;
    nErrId      : UDINT;
END_VAR
```

```
fbReadCmd (sNetId:=sNetID, logAddr:=logAddr, LEN := SIZEOF(value), pDstBuf:=ADR(value), bExecute:=bE
xecute);
wkc := fbReadCmd.wkc;
bError:=fbReadCmd.bError;
nErrId:=fbReadCmd.nErrId;
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 oder höher	PC or CX (x86)	TcEtherCAT.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib, TcUtilities.Lib werden automatisch eingebunden)
TwinCAT v2.10.0 Build >= 1301 oder höher	CX (ARM)	

6.4 FB_EcLogicalWriteCmd



Mit dem Funktionsbaustein FB_EcLogicalWriteCmd wird ein logisches EtherCAT Write-Kommando(LWR) vom Master gesendet. In jedem Slave können lokale Adressbereiche(DPRAM) auf globale logische Adressebereich gemappt werden. Somit adressiert diese Kommandos alle EtherCAT-Slave, die ein Mapping für den ausgewählten logischen Adressbereich konfiguriert haben.

VAR_INPUT

```
VAR_INPUT
    sNetId   : T_AmsNetId;
    logAddr  : UDINT;
    len      : UDINT;
    pSrcBuf  : DWORD;
    bExecute : BOOL;
    tTimeout : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

sNetId: Ist ein String, der die AMS-Netzwerkkennung des EtherCAT Master Gerätes enthält.

logAddr: Logische Adresse.

len: Anzahl der zu schreibenden Bytes.

pSrcBuf: Die Adresse (Pointer) auf den Sendepuffer.

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

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

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy    : BOOL;
    bError   : BOOL;
    nErrId   : UDINT;
    wkc      : UINT;
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.

nErrId: Liefert bei einem gesetzten bError-Ausgang den ADS-Fehlercode des zuletzt ausgeführten Befehles

wkc: Der Working Counter wird von jedem EtherCAT-Slave der dieses Kommando erfolgreich bearbeitet hat, um eins inkrementiert. Wenn nur ein EtherCAT-Slave von diesem Kommando adressiert worden ist, sollte dieser Wert somit 1 entsprechen.

Beispiel für eine Implementierung in ST:

```
PROGRAM Test_LogicalWriteCmd
VAR
    fbWriteCmd  : FB_EcLogicalWriteCmd;
    bExecute    : BOOL;
    value       : USINT :=16#55;
    logAddr     : UDINT :=16#10000;
    sNetId      : T_AmsNetId:='192.168.1.5.3.1';

    wkc         : UINT;
    bError      : BOOL;
    nErrId      : UDINT;
END_VAR
```

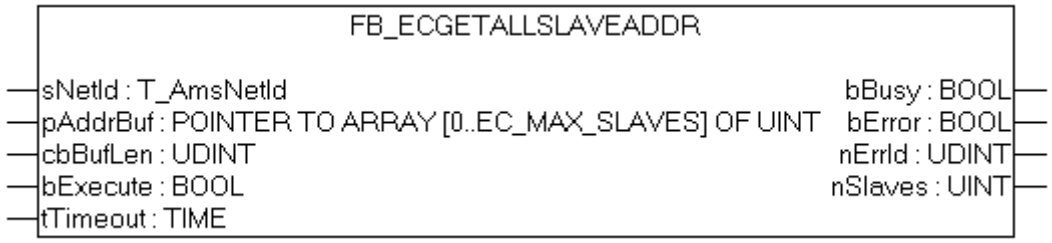
```
fbWriteCmd (sNetId:=sNetID, logAddr:=logAddr, LEN := SIZEOF(value), pSrcBuf:=ADR(value), bExecute:=b
Execute);
wkc := fbWriteCmd.wkc;
bError:=fbWriteCmd.bError;
nErrId:=fbWriteCmd.nErrId;
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 oder höher	PC or CX (x86)	TcEtherCAT.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib, TcUtilities.Lib werden automatisch eingebunden)
TwinCAT v2.10.0 Build >= 1301 oder höher	CX (ARM)	

7 EtherCAT Diagnostic

7.1 FB_EcGetAllSlaveAddr



Mit dem Funktionsbaustein FB_EcGetAllSlaveAddr können die Adressen von allen an den Master angeschlossenen Slaves ausgelesen werden. Bei erfolgreichem Aufruf enthält der im Parameter pAddrBuf übergebene Puffer die Adressen aller Slaves als Array von UINTs.

VAR_INPUT

```
VAR_INPUT
    sNetId      : T_AmsNetId;
    pAddrBuf    : POINTER TO ARRAY[0..EC_MAX_SLAVES] OF UINT;
    cbBufLen    : UDINT
    bExecute    : BOOL;
    tTimeout    : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

sNetId: Ist ein String, der die AMS-Netzwerkkennung des EtherCAT Master Gerätes enthält.

pAddrBuf: Die Adresse eines Arrays von UINTs, in das die Adressen der einzelnen Slaves geschrieben werden sollen.

cbBufLen: Die maximal verfügbare Puffergröße für die zu lesenden Daten in Bytes.

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

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

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy       : BOOL;
    bError      : BOOL;
    nErrId      : UDINT;
    nSlaves     : UINT;
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.

nErrId: Liefert bei einem gesetzten bError-Ausgang den ADS-Fehlercode des zuletzt ausgeführten Befehles

nSlaves: Anzahl der an den Master angeschlossenen Slaves.

Beispiel für eine Implementierung in ST:

```
PROGRAM TEST_GetAllSlaveAddresses
VAR
fbGetAllSlaveAddr : FB_EcGetAllSlaveAddr;
sNetId : T_AmsNetId := '172.16.2.131.2.1';
bExecute : BOOL;
slaveAddresses : ARRAY[0..255] OF UINT;
```

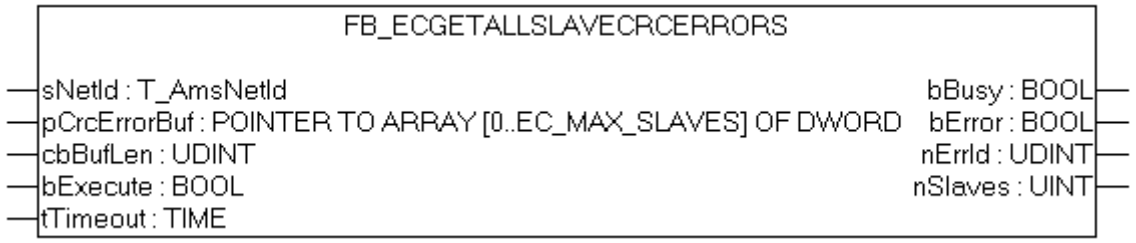
```
nSlaves : UINT := 0;
bError : BOOL;
nErrId : UDINT;
END_VAR
```

```
fbGetAllSlaveAddr(sNetId:= sNetId,pAddrBuf := ADR(slaveAddresses), cbBufLen:= SIZEOF(slaveAddresses)
, bExecute:=bExecute);
nSlaves:=fbGetAllSlaveAddr.nSlaves;
bError:=fbGetAllSlaveAddr.bError;
nErrId:=fbGetAllSlaveAddr.nErrId;
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 oder höher	PC or CX (x86)	TcEtherCAT.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib, TcUtilities.Lib werden automatisch eingebunden)
TwinCAT v2.10.0 Build >= 1301 oder höher	CX (ARM)	

7.2 FB_EcGetAllSlaveCrcErrors



Mit dem Funktionsbaustein FB_EcGetAllSlaveCrcErrors können die CRC-Error Zähler aller am Master angeschlossenen Slaves ausgelesen werden. Die CRC-Error an den einzelnen Ports eines Slaves werden aufaddiert. Um die CRC-Error der einzelnen Ports (A,B und C) eines Slaves auszulesen, muss der Funktionsbock FB_EcGetSlaveCrcError [▶ 44] aufgerufen werden.

VAR_INPUT

```
VAR_INPUT
    sNetId      : T_AmsNetId;
    pCrcErrorBuf: POINTER TO ARRAY[0..EC_MAX_SLAVES] OF DWORD;
    cbBufLen    : UDINT;
    bExecute    : BOOL;
    tTimeout    : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

sNetId: Ist ein String, der die AMS-Netzwerkkennung des EtherCAT Master Gerätes enthält.

pCrcErrorBuf : Die Adresse eines Arrays von DWORDs, in das die CRC-Error Zähler geschrieben werden sollen.

cbBufLen: Die maximal verfügbare Puffergröße für die zu lesenden Daten in Bytes.

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

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

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy       : BOOL;
    bError      : BOOL;
    nErrId      : UDINT;
    nSlaves     : UINT;
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.

nErrId: Liefert bei einem gesetzten bError-Ausgang den ADS-Fehlercode des zuletzt ausgeführten Befehles

nSlaves: Anzahl der an den Master angeschlossenen Slaves.

Beispiel für eine Implementierung in ST:

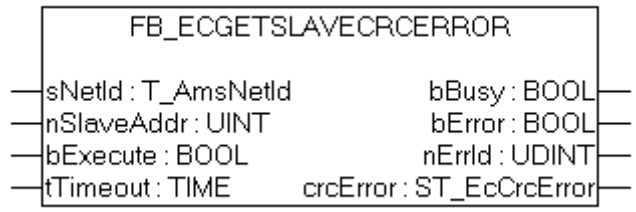
```
PROGRAM TEST_GetAllSlaveCrcErrors
VAR
    fbGetAllSlaveCrcErrors  : FB_EcGetAllSlaveCrcErrors;
    sNetId          : T_AmsNetId := '172.16.2.131.2.1';
    bExecute            : BOOL;
    crcErrors           : ARRAY[0..255] OF DWORD;
    nSlaves             : UINT := 0;
    bError          : BOOL;
    nErrId          : UDINT;
END_VAR
```

```
fbGetAllSlaveCrcErrors(sNetId:= sNetId, pCrcErrorBuf := ADR(crcErrors), cbBufLen:= SIZEOF(crcErrors)
, bExecute:=bExecute);
nSlaves:=fbGetAllSlaveCrcErrors.nSlaves;
bError:=fbGetAllSlaveCrcErrors.bError;
nErrId:=fbGetAllSlaveCrcErrors.nErrId;
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 oder höher	PC or CX (x86)	TcEtherCAT.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib, TcUtilities.Lib werden automatisch eingebunden)
TwinCAT v2.10.0 Build >= 1301 oder höher	CX (ARM)	

7.3 FB_EcGetSlaveCrcError



Mit dem Funktionsbaustein FB_EcGetSlaveCrcError können die CRC-Error Zähler der einzelnen Ports (A, B und C) eines Slave ausgelesen werden. Bei erfolgreichem Aufruf enthält die Ausgangsvariable *crcError* vom Typ ST_EcCrcError die angeforderten CRC-Error Zähler.

Der Funktionsbaustein FB_EcGetSlaveCrcError kann nur mit Slaves mit bis zu 3 Ports (z.B. EK1100) eingesetzt werden, der Funktionsbaustein FB_EcGetSlaveCrcErrorEx kann auch mit Slaves mit bis zu 4 Ports (z.B. EK1122) eingesetzt werden.

VAR_INPUT

```
VAR_INPUT
    sNetId      : T_AmsNetId;
    nSlaveAddr  : UINT;
    bExecute    : BOOL;
    tTimeout    : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

sNetId: Ist ein String, der die AMS-Netzwerkkennung des EtherCAT Master Gerätes enthält.

nSlaveAddr: Feste Adresse des EtherCAT Slaves, dessen CRC-Error Zähler ausgelesen werden soll.

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

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

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy       : BOOL;
    bError      : BOOL;
    nErrId      : UDINT;
    crcError    : ST_EcCrcError;
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.

nErrId: Liefert bei einem gesetzten bError-Ausgang den ADS-Fehlercode des zuletzt ausgeführten Befehles

crcError: CRC-Error [▶ 80] Zähler der einzelnen Ports.

Beispiel für eine Implementierung in ST:

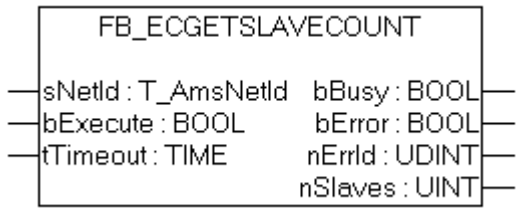
```
PROGRAM TEST_GetSlaveCrcError
VAR
    fbGetSlaveCrcError      : FB_EcGetSlaveCrcError;
    sNetId          : T_AmsNetId := '172.16.2.131.2.1';
    bExecute        : BOOL;
    crcError        : ST_EcCrcError;
    nSlaveAddr          : UINT := 1001;
    bError          : BOOL;
    nErrId          : UDINT;
END_VAR
```

```
fbGetSlaveCrcError(sNetId:= sNetId, nSlaveAddr:= nSlaveAddr, bExecute:=bExecute);
crcError:=fbGetSlaveCrcError.crcError;
bError:=fbGetSlaveCrcError.bError;
nErrId:=fbGetSlaveCrcError.nErrId;
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 oder höher	PC or CX (x86)	TcEtherCAT.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib, TcUtilities.Lib werden automatisch eingebunden)
TwinCAT v2.10.0 Build >= 1301 oder höher	CX (ARM)	

7.4 FB_EcGetSlaveCount



Mit dem Funktionsbaustein FB_EcGetSlaveCount kann die Anzahl der Slaves, die an den Master angeschlossen sind, ermittelt werden.

VAR_INPUT

```
VAR_INPUT
    sNetId      : T_AmsNetId;
    bExecute    : BOOL;
    tTimeout    : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

sNetId: Ist ein String, der die AMS-Netzwerkkennung des EtherCAT Master Gerätes enthält.

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

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

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy       : BOOL;
    bError      : BOOL;
    nErrId      : UDINT;
    nSlaves     : UINT;
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.

nErrId: Liefert bei einem gesetzten bError-Ausgang den ADS-Fehlercode des zuletzt ausgeführten Befehles

nSlaves: Anzahl der Slaves, die an den Master angeschlossen sind.

Beispiel für eine Implementierung in ST:

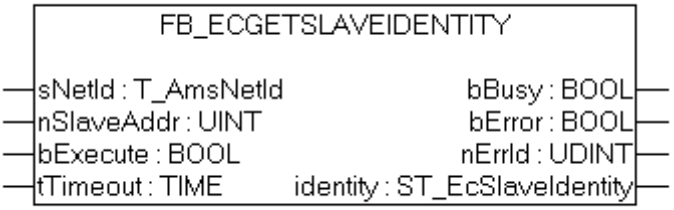
```
PROGRAM TEST_GetSlaveCount
VAR
    fbGetSlaveCount         : FB_EcGetSlaveCount;
    sNetId          : T_AmsNetId := '172.16.2.131.2.1';
    bExecute            : BOOL;
    nSlaves             : UINT;
    bError          : BOOL;
    nErrId          : UDINT;
END_VAR
```

```
fbGetSlaveCount(sNetId:= sNetId, bExecute:=bExecute);
nSlaves:=fbGetSlaveCount.nSlaves;
bError:=fbGetSlaveCount.bError;
nErrId:=fbGetSlaveCount.nErrId;
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 oder höher	PC or CX (x86)	TcEtherCAT.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib, TcUtilities.Lib werden automatisch eingebunden)
TwinCAT v2.10.0 Build >= 1301 oder höher	CX (ARM)	

7.5 FB_EcGetSlaveIdentity



Mit dem Funktionsbaustein FB_EcGetSlaveIdentity kann die CANopen Identity eines einzelnen EtherCAT Slave Gerätes ausgelesen werden. Bei erfolgreichem Aufruf enthält die Ausgangsvariable *identity* vom Typ ST_EcSlaveIdentity die angeforderte Identity-Information.

VAR_INPUT

```
VAR_INPUT
    sNetId      : T_AmsNetId;
    nSlaveAddr  : UINT;
    bExecute    : BOOL;
    tTimeout    : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

sNetId: Ist ein String, der die AMS-Netzwerkkennung des EtherCAT Master Gerätes enthält.

nSlaveAddr: Feste Adresse des EtherCAT Slaves.

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

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

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy       : BOOL;
    bError      : BOOL;
    nErrId      : UDINT;
    identity    : ST_EcSlaveIdentity;
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.

nErrId: Liefert bei einem gesetzten bError-Ausgang den ADS-Fehlercode des zuletzt ausgeführten Befehles

identity: CANopen Identity [▶ 82] des EtherCAT Gerätes.

Beispiel für eine Implementierung in ST:

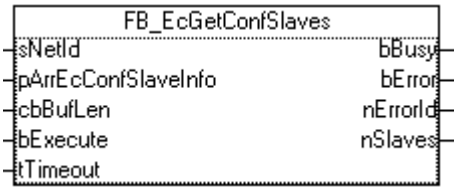
```
PROGRAM TEST_GetSlaveIdentity
VAR
fbGetSlaveIdentity : FB_EcGetSlaveIdentity;
sNetId : T_AmsNetId := '172.16.2.131.2.1';
bExecute : BOOL;
identity : ST_EcSlaveIdentity;
nSlaveAddr : UINT := 1001;
bError : BOOL;
nErrId : UDINT;
END_VAR
```

```
fbGetSlaveIdentity(sNetId:= sNetId, nSlaveAddr:= nSlaveAddr, bExecute:=bExecute);
identity:=fbGetSlaveIdentity.identity;
bError:=fbGetSlaveIdentity.bError;
nErrId:=fbGetSlaveIdentity.nErrId;
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 oder höher	PC or CX (x86)	TcEtherCAT.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib, TcUtilities.Lib werden automatisch eingebunden)
TwinCAT v2.10.0 Build >= 1301 oder höher	CX (ARM)	

7.6 FB_EcGetConfSlaves



Mit dem Funktionsbaustein FB_EcGetConfSlaveskann eine Liste der konfigurierten Slaves aus dem EtherCAT Master Objektverzeichnis ausgelesen werden.

VAR_INPUT

```
VAR_INPUT
    sNetId               : T_AmsNetId;
    pArrEcConfSlaveInfo  : POINTER TO ARRAY[0..EC_MAX_SLAVES] OF ST_EcSlaveConfigData;
    cbBufLen             : UDINT;
    bExecute             : BOOL;
    tTimeout             : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

sNetId: Ist ein String, der die AMS-Netzwerkkennung des EtherCAT Master Gerätes enthält.

pArrEcConfSlaveInfo : Die Adresse eines Arrays von Strukturen des Typs ST_EcSlaveConfigData, [▶ 81] in das Daten jeden konfigurierten Slaves geschrieben werden sollen.

cbBufLen: Die maximal verfügbare Puffergröße für die zu lesenden Daten in Bytes.

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

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

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy       : BOOL;
    bError      : BOOL;
    nErrId      : UDINT;
    nSlaves     : UINT;
ND_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.

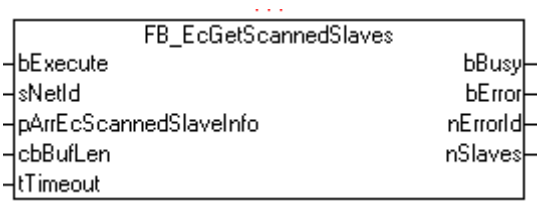
nErrId: Liefert bei einem gesetzten bError-Ausgang den ADS-Fehlercode des zuletzt ausgeführten Befehles

nSlaves: Liefert die Anzahl der konfigurierten Slaves.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 oder höher	PC or CX (x86)	TcEtherCAT.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib, TcUtilities.Lib werden automatisch eingebunden)
TwinCAT v2.10.0 Build >= 1301 oder höher	CX (ARM)	

7.7 FB_EcGetScannedSlaves



Mit dem Funktionsbaustein FB_EcGetScannedSlaves kann eine Liste der aktuell verfügbaren (gescannten) Slaves aus dem EtherCAT Master Objektverzeichnis ausgelesen werden.

VAR_INPUT

```
VAR_INPUT
    bExecute            : BOOL;
    sNetId              : T_AmsNetId;
    pArrEcConfSlaveInfo : POINTER TO ARRAY[0..EC_MAX_SLAVES] OF ST_EcSlaveScannedData;
    cbBufLen            : UDINT;
    tTimeout            : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

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

sNetId: Ist ein String, der die AMS-Netzwerkkennung des EtherCAT Master Gerätes enthält.

pArrEcConfSlaveInfo : Die Adresse eines Arrays von Strukturen des Typs ST_EcSlaveScannedData [▶ 82], in das Daten jeden gescannten Slaves geschrieben werden sollen.

cbBufLen: Die maximal verfügbare Puffergröße für die zu lesenden Daten in Bytes.

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

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy       : BOOL;
    bError      : BOOL;
    nErrId      : UDINT;
    nSlaves     : UINT;
ND_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.

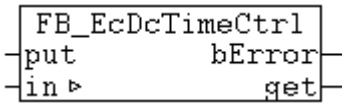
nErrId: Liefert bei einem gesetzten bError-Ausgang den ADS-Fehlercode des zuletzt ausgeführten Befehles

nSlaves: Liefert die Anzahl der gescannten Slaves.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 oder höher	PC or CX (x86)	TcEtherCAT.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib, TcUtilities.Lib werden automatisch eingebunden)
TwinCAT v2.10.0 Build >= 1301 oder höher	CX (ARM)	

7.8 FB_EcDcTimeCtrl



Mit diesem Funktionsbaustein können die einzelnen Komponenten wie Jahr, Monat, Tag usw. einer 64 Bit TwinCAT "Distributed Clock System Time"-Variablen gelesen werden. Der Funktionsbaustein besitzt mehrere A_GetXYZ-Aktionen. Nach dem Aufruf der gewünschten Aktion steht der Wert der XYZ-Komponente in der *get*-Ausgangsvariablen zur Verfügung. Die put-Eingangsvariable wird zur Zeit nicht benutzt.

Der Funktionsbaustein besitzt zur Zeit folgende Aktionen:

- A_GetYear;
- A_GetMonth;
- A_GetDay;
- A_GetDayOfWeek;
- A_GetHour;
- A_GetMinute;
- A_GetSecond;
- A_GetMilli;
- A_GetMicro;
- A_GetNano;

VAR_IN_OUT

```
VAR_IN_OUT
    in          : T_DCTIME;
END_VAR
```

in: TwinCAT "Distributed Clock System Time [▶ 87]"-Variable;

VAR_INPUT

```
VAR_INPUT
    put         : WORD;
END_VAR
```

put: Eingangsparameter (zur Zeit nicht benutzt);

VAR_OUTPUT

```
VAR_OUTPUT
    bError      : BOOL;
    get         : WORD;
END_VAR
```

bError: Dieser Ausgang wird gesetzt wenn ein Fehler beim Aktionsaufruf aufgetreten ist;

get: Ausgangsparameter (Jahr, Monat, Tag usw.);

Beispiel für eine Implementierung in ST:

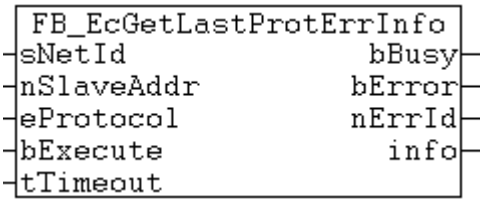
```
PROGRAM P_TEST
VAR
    dcStruct    : DCTIMESTRUCT;
    dcTime      : T_DCTIME;
    fbCtrl      : FB_EcDcTimeCtrl;

    wYear       : WORD;
    wMonth      : WORD;
    wDay        : WORD;
    wDayOfWeek  : WORD;
    wHour       : WORD;
    wMinute     : WORD;
    wSecond     : WORD;
    wMilli      : WORD;
    wMicro      : WORD;
    wNano       : WORD;
END_VAR
```

```
dcTime := F_GetCurDcTickTime();

fbCtrl.A_GetYear( in := dcTime, get => wYear );
fbCtrl.A_GetMonth( in := dcTime, get => wMonth );
fbCtrl.A_GetDay( in := dcTime, get => wDay );
fbCtrl.A_GetDayOfWeek( in := dcTime, get => wDayOfWeek );
fbCtrl.A_GetHour( in := dcTime, get => wHour );
fbCtrl.A_GetMinute( in := dcTime, get => wMinute );
fbCtrl.A_GetSecond( in := dcTime, get => wSecond );
fbCtrl.A_GetMilli( in := dcTime, get => wMilli );
fbCtrl.A_GetMicro( in := dcTime, get => wMicro );
fbCtrl.A_GetNano( in := dcTime, get => wNano );
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 Build > 1316 oder höher	PC oder CX (x86, ARM)	TcEtherCAT.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib, TcUtilities.Lib werden automatisch eingebunden)

7.9 FB_EcGetLastProtErrInfo



Mit dem Funktionsbaustein FB_EcGetLastProtErrInfo können zusätzliche Fehlerinformationen zum zuletzt aufgetretenen Mailboxprotokollfehler ausgelesen werden. Ein fehlerfreies Mailboxkommando setzt den letzten Fehler jedes mal zurück.

VAR_INPUT

```
VAR_INPUT
    sNetId      : T_AmsNetId;
    nSlaveAddr  : UINT;
    eProtocol   : E_EcMbxProtType := eEcMbxProt_FoE;
    bExecute    : BOOL;
    tTimeout    : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

sNetId: Ist ein String, der die AMS-Netzwerkkennung des EtherCAT Master Gerätes enthält.

nSlaveAddr: Feste Adresse des EtherCAT Slaves, dessen Fehlerinformationen ausgelesen werden sollen.

eProtocol: EtherCAT-Mailboxprotokolltyp [▶ 79].

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

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

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy       : BOOL;
    bError      : BOOL;
    nErrId      : UDINT;
    info        : ST_EcLastProtErrInfo;
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.

nErrId: Liefert bei einem gesetzten *bError*-Ausgang den ADS-Fehlercode des zuletzt ausgeführten Befehles

info: Struktur [▶ 80] mit zusätzlichen Fehlerinformationen.

Beispiel in ST:

Bei einer steigenden Flanke am *bGet* werden zusätzliche Fehlerinformationen zum zuletzt aufgetretenen Mailboxprotokollfehler ausgelesen.

```
PROGRAM MAIN
VAR
    fbGetInfo : FB_EcGetLastProtErrInfo := ( sNetID := '172.16.6.195.2.1',
                        nSlaveAddr := 1004,
                        eProtocol := eEcMbxProt_FoE,
                        tTimeout := DEFAULT_ADS_TIMEOUT );
    bGet    : BOOL;
    bBusy   : BOOL;
    bError  : BOOL;
    nErrID  : UDINT;
    sInfo   : T_MaxString;
END_VAR
```

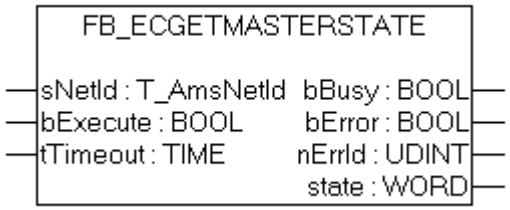
```
fbGetInfo(       bExecute:= bGet,
        bBusy=>bBusy,
        bError=>bError,
        nErrId=>nErrId );

sInfo := BYTEARR_TO_MAXSTRING( fbGetInfo.info.binDesc );
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 Build > 1307 oder höher	PC oder CX (x86, ARM)	TcEtherCAT.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib, TcUtilities.Lib werden automatisch eingebunden)

8 EtherCAT State Machine

8.1 FB_EcGetMasterState



Mit dem Funktionsbaustein FB_EcGetMasterState kann der EtherCAT Zustand des Master ausgelesen werden. Bei erfolgreichem Aufruf enthält die Ausgangsvariable *state* vom Typ WORD die angeforderte Status-Information.

VAR_INPUT

```
VAR_INPUT
    sNetId      : T_AmsNetId;
    bExecute    : BOOL;
    tTimeout    : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

sNetId: Ist ein String, der die AMS-Netzwerkkennung des EtherCAT Master Gerätes enthält.

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

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

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy       : BOOL;
    bError      : BOOL;
    nErrId      : UDINT;
    state       : WORD;
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.

nErrId: Liefert bei einem gesetzten bError-Ausgang den ADS-Fehlercode des zuletzt ausgeführten Befehles

state: Aktueller EtherCAT Zustand des Masters. Die möglichen Werte sind:

Konstante	Wert	Beschreibung
EC_DEVICE_STATE_INIT	0x01	Master ist im Init-Zustand
EC_DEVICE_STATE_PREOP	0x02	Master ist im Pre-Operational Zustand
EC_DEVICE_STATE_SAFEOP	0x04	Master ist im Safe-Operational Zustand
EC_DEVICE_STATE_OP	0x08	Master ist im Operational Zustand

Beispiel für eine Implementierung in ST:

```
PROGRAM TEST_GetMasterState
VAR
    fbGetMasterState  : FB_EcGetMasterState;
    sNetId            : T_AmsNetId := '172.16.2.131.2.1';
    bExecute          : BOOL;
    state             : WORD;
```

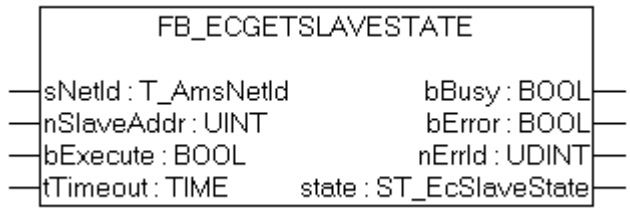
```
    bError            : BOOL;
    nErrId            : UDINT;
END_VAR
```

```
fbGetMasterState(sNetId:= sNetId, bExecute:=bExecute);
state:=fbGetMasterState.state;
bError:=fbGetMasterState.bError;
nErrId:=fbGetMasterState.nErrId;
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 oder höher	PC or CX (x86)	TcEtherCAT.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib, TcUtilities.Lib werden automatisch eingebunden)
TwinCAT v2.10.0 Build >= 1301 oder höher	CX (ARM)	

8.2 FB_EcGetSlaveState



Mit dem Funktionsbaustein FB_EcGetSlaveState kann der EtherCAT Status und der Link Status eines einzelnen EtherCAT Slave Gerätes ausgelesen werden. Bei erfolgreichem Aufruf enthält die Ausgangsvariable *state* vom Typen ST_EcSlaveState die angeforderte Status-Information.

VAR_INPUT

```
VAR_INPUT
    sNetId      : T_AmsNetId;
    nSlaveAddr  : UINT;
    bExecute    : BOOL;
    tTimeout    : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

sNetId: Ist ein String, der die AMS-Netzwerkkennung des EtherCAT Master Gerätes enthält.

nSlaveAddr: Feste Adresse des EtherCAT Slaves, dessen Status ausgelesen werden soll

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

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

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy       : BOOL;
    bError      : BOOL;
    nErrId      : UDINT;
    state       : ST_EcSlaveState;
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.

nErrId: Liefert bei einem gesetzten bError-Ausgang den ADS-Fehlercode des zuletzt ausgeführten Befehles

state: Struktur [▶ 84] die den EtherCAT Status und den Link Status des Slaves enthält.

Beispiel für eine Implementierung in ST:

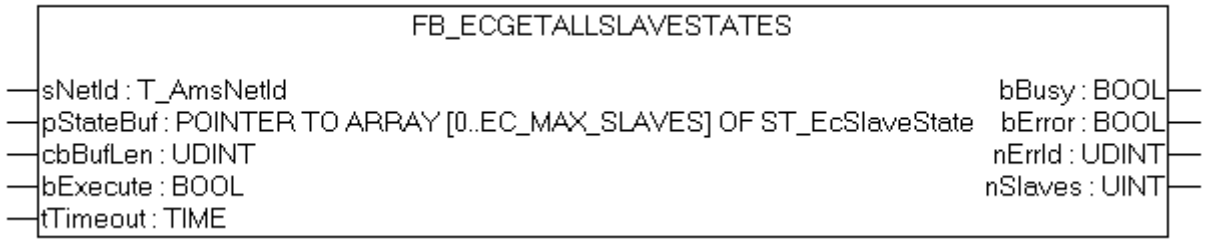
```
PROGRAM TEST_GetSlaveState
VAR
    fbGetSlaveState : FB_EcGetSlaveState;
    sNetId          : T_AmsNetId := '172.16.2.131.2.1';
    bExecute        : BOOL;
    state           : ST_EcSlaveState;
    nSlaveAddr      : UINT := 1001;
    bError          : BOOL;
    nErrId          : UDINT;
END_VAR
```

```
fbGetSlaveState(sNetId:= sNetId, nSlaveAddr:= nSlaveAddr, bExecute:=bExecute);
state:=fbGetSlaveState.state;
bError:=fbGetSlaveState.bError;
nErrId:=fbGetSlaveState.nErrId;
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 oder höher	PC or CX (x86)	TcEtherCAT.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib, TcUtilities.Lib werden automatisch eingebunden)
TwinCAT v2.10.0 Build >= 1301 oder höher	CX (ARM)	

8.3 FB_EcGetAllSlaveStates



Mit dem Funktionsbaustein FB_EcGetAllSlaveStates kann der EtherCAT Status und der Link Status von allen an den Master angeschlossenen Slaves ausgelesen werden. Bei erfolgreichem Aufruf enthält der im Parameter pStateBuf übergebene Puffer die angeforderte Status-Information als Array von ST_EcSlaveState.

VAR_INPUT

```
VAR_INPUT
    sNetId      : T_AmsNetId;
    pStateBuf   : POINTER TO ARRAY[0..EC_MAX_SLAVES] OF ST_EcSlaveState;
    cbBufLen    : UDINT;
    bExecute    : BOOL;
    tTimeout    : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

sNetId: Ist ein String, der die AMS-Netzwerkkennung des EtherCAT Master Gerätes enthält.

pStateBuf : Die Adresse eines ARRAY of ST_EcSlaveStates [▶ 84], in das die Slave-Stati geschrieben werden sollen.

cbBufLen: Die maximal verfügbare Puffergröße für die zu lesenden Daten in Bytes.

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

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

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy       : BOOL;
    bError      : BOOL;
    nErrId      : UDINT;
    nSlaves     : UINT;
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.

nErrId: Liefert bei einem gesetzten bError-Ausgang den ADS-Fehlercode des zuletzt ausgeführten Befehles

nSlaves: Anzahl der an den Master angeschlossenen Slaves.

Beispiel für eine Implementierung in ST:

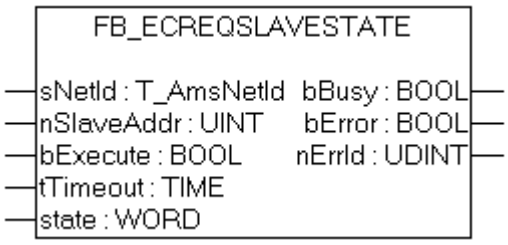
```
PROGRAM TEST_GetAllSlaveStates
VAR
    fbGetAllSlaveStates     : FB_EcGetAllSlaveStates;
    sNetId          : T_AmsNetId := '172.16.2.131.2.1';
    bExecute            : BOOL;
    devStates           : ARRAY[0..255] OF ST_EcSlaveState;
    nSlaves             : UINT := 0;
    bError          : BOOL;
    nErrId          : UDINT;
END_VAR
```

```
fbGetAllSlaveStates(sNetId:= sNetId, pStateBuf := ADR(devStates), cbBufLen:=SIZEOF(devStates), bExec
ute:=bExecute);
nSlaves:=fbGetAllSlaveStates.nSlaves;
bError:=fbGetAllSlaveStates.bError;
nErrId:=fbGetAllSlaveStates.nErrId;
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 oder höher	PC or CX (x86)	TcEtherCAT.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib, TcUtilities.Lib werden automatisch eingebunden)
TwinCAT v2.10.0 Build >= 1301 oder höher	CX (ARM)	

8.4 FB_EcReqSlaveState



Mit dem Funktionsbaustein FB_EcReqSlaveState wird versucht den EtherCAT Slave in den Zustand zu setzen, der mit dem Parameter *state* angegeben ist.

VAR_INPUT

```
VAR_INPUT
    sNetId      : T_AmsNetId;
    nSlaveAddr  : UINT;
    bExecute    : BOOL;
    tTimeout    : TIME := DEFAULT_ADS_TIMEOUT;
    state       : WORD;
END_VAR
```

sNetId: Ist ein String, der die AMS-Netzwerkkennung des EtherCAT Master Gerätes enthält.

nSlaveAddr: Feste Adresse des EtherCAT Slaves, dessen EtherCAT Zustand gesetzt werden soll.

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

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

state: EtherCAT Zustand in den der Slave gesetzt werden soll. Die möglichen Werte für *state* sind:

Konstante	Wert	Beschreibung
EC_DEVICE_STATE_INIT	0x01	Slave in den Init-Zustand setzen
EC_DEVICE_STATE_PREOP	0x02	Slave in den Pre-Operational Zustand setzen
EC_DEVICE_STATE_BOOTSTRAP	0x03	Slave in Bootstrap-Zustand setzen, dieser Zustand wird verwendet um ein Firmware-Download durchzuführen
EC_DEVICE_STATE_SAFEOP	0x04	Slave in den Safe-Operational Zustand setzen
EC_DEVICE_STATE_OP	0x08	Slave in den Operational Zustand setzen
EC_DEVICE_STATE_ERROR	0x10	Wenn bei dem EtherCAT Slave das Error-Bit im Status-Byte gesetzt ist(state.deviceState & EC_DEVICE_STATE_ERROR = TRUE), kann man das Error-Bit durch Setzen von EC_DEVICE_STATE_ERROR wieder zurücksetzen

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy       : BOOL;
    bError      : BOOL;
    nErrId      : UDINT;
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.

nErrId: Liefert bei einem gesetzten bError-Ausgang den ADS-Fehlercode des zuletzt ausgeführten Befehles

Beispiel für eine Implementierung in ST:

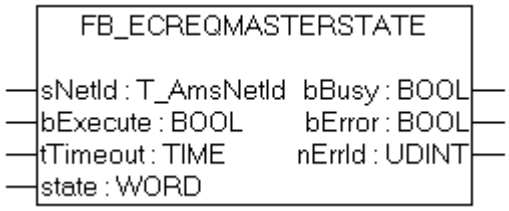
```
PROGRAM TEST_ReqSlaveState
VAR
fbGetSlaveState : FB_EcReqSlaveState;
sNetId : T_AmsNetId := '172.16.2.131.2.1';
bExecute : BOOL;
state : WORD := EC_DEVICE_STATE_INIT;
nSlaveAddr : UINT := 1001;
bError : BOOL;
nErrId : UDINT;
END_VAR
```

```
fbGetSlaveState(sNetId:= sNetId, nSlaveAddr:= nSlaveAddr, bExecute:=bExecute, state:=state);
bError:=fbGetSlaveState.bError;
nErrId:=fbGetSlaveState.nErrId;
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 oder höher	PC or CX (x86)	TcEtherCAT.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib, TcUtilities.Lib werden automatisch eingebunden)
TwinCAT v2.10.0 Build >= 1301 oder höher	CX (ARM)	

8.5 FB_EcReqMasterState



Mit dem Funktionsbaustein FB_EcReqMasterState kann der EtherCAT Zustand, der in der Variablen **state** übergeben wird, vom Master angefordert werden.

VAR_INPUT

```
VAR_INPUT
    sNetId      : T_AmsNetId;
    bExecute    : BOOL;
    tTimeout    : TIME := DEFAULT_ADS_TIMEOUT;
    state       : WORD;
END_VAR
```

sNetId: Ist ein String, der die AMS-Netzwerkkennung des EtherCAT Master Gerätes enthält.

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

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

state: EtherCAT Zustand der vom Master angefordert wird. Die möglichen Werte für *state* sind:

Konstante	Wert	Beschreibung
EC_DEVICE_STATE_INIT	0x01	Init-Zustand vom Master anfordern
EC_DEVICE_STATE_PREOP	0x02	Pre-Operational Zustand vom Master anfordern
EC_DEVICE_STATE_SAFEOP	0x04	Safe-Operational Zustand vom Master anfordern
EC_DEVICE_STATE_OP	0x08	Operational-Zustand vom Master anfordern

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy       : BOOL;
    bError      : BOOL;
    nErrId      : UDINT;
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.

nErrId: Liefert bei einem gesetzten bError-Ausgang den ADS-Fehlercode des zuletzt ausgeführten Befehles

Beispiel für eine Implementierung in ST:

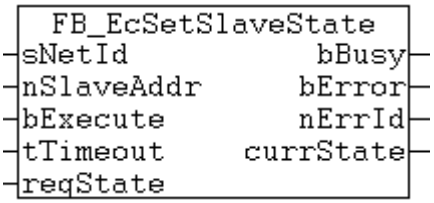
```
PROGRAM TEST_ReqMasterState
VAR
fbReqMasterState : FB_EcReqMasterState;
sNetId : T_AmsNetId := '172.16.2.131.2.1';
bExecute : BOOL;
state : WORD :=EC_DEVICE_STATE_INIT;
bError : BOOL;
nErrId : UDINT;
END_VAR
```

```
fbReqMasterState(sNetId:= sNetId, bExecute:=bExecute, state:=state);
bError:=fbGetMasterState.bError;
nErrId:=fbGetMasterState.nErrId;
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 oder höher	PC or CX (x86)	TcEtherCAT.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib, TcUtilities.Lib werden automatisch eingebunden)
TwinCAT v2.10.0 Build >= 1301 oder höher	CX (ARM)	

8.6 FB_EcSetSlaveState



Mit dem Funktionsbaustein FB_EcSetSlaveState wird versucht den EtherCAT Slave in den Zustand zu setzen, der mit dem Parameter *reqState* angegeben ist. Der Funktionsbaustein wartet für die max. *tTimeout*-Zeit bis der neue Zustand gesetzt wurde.

VAR_INPUT

```
VAR_INPUT
    sNetId      : T_AmsNetId;
    nSlaveAddr  : UINT;
    bExecute    : BOOL;
    tTimeout    : TIME := T#10s;
    reqState    : WORD;
END_VAR
```

sNetId: Ist ein String, der die AMS-Netzwerkkennung des EtherCAT Master Gerätes enthält.

nSlaveAddr: Feste Adresse des EtherCAT Slaves, dessen EtherCAT Zustand gesetzt werden soll.

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

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

reqState: EtherCAT Zustand in den der Slave gesetzt werden soll. Die möglichen Werte für *reqState* sind:

Konstante	Wert	Beschreibung
EC_DEVICE_STATE_INIT	0x01	Slave in den Init-Zustand setzen
EC_DEVICE_STATE_PREOP	0x02	Slave in den Pre-Operational Zustand setzen
EC_DEVICE_STATE_BOOTSTRAP	0x03	Slave in Bootstrap-Zustand setzen, dieser Zustand wird verwendet um ein Firmware-Download durchzuführen
EC_DEVICE_STATE_SAFEOP	0x04	Slave in den Safe-Operational Zustand setzen

Konstante	Wert	Beschreibung
EC_DEVICE_STATE_OP	0x08	Slave in den Operational Zustand setzen
EC_DEVICE_STATE_ERROR	0x10	Wenn bei dem EtherCAT Slave das Error-Bit im Status-Byte gesetzt ist(currState.deviceState AND EC_DEVICE_STATE_ERROR = TRUE), kann man das Error-Bit durch Setzen von EC_DEVICE_STATE_ERROR wieder zurücksetzen

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy       : BOOL;
    bError      : BOOL;
    nErrId      : UDINT;
    currState   : ST_EcSlaveState;
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.

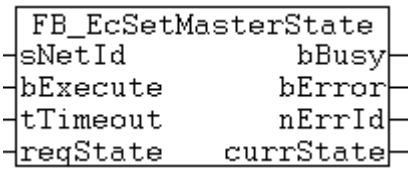
nErrId: Liefert bei einem gesetzten bError-Ausgang den ADS-Fehlercode des zuletzt ausgeführten Befehles.

currState: Aktueller EtherCAT Zustand [▶ 84] des Slaves.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 Build > 1307 oder höher	PC or CX (x86) CX (ARM)	TcEtherCAT.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib, TcUtilities.Lib werden automatisch eingebunden)

8.7 FB_EcSetMasterState



Mit dem Funktionsbaustein FB_EcSetMasterState kann der EtherCAT Master Zustand, der in der Variablen **reqState** übergeben wird, vom Master angefordert werden. Der Funktionsbaustein wartet für die maximale *tTimeout*-Zeit bis der neue Zustand gesetzt wurde.

VAR_INPUT

```
VAR_INPUT
    sNetId      : T_AmsNetId;
    bExecute    : BOOL;
    tTimeout    : TIME := T#10s;
    reqState    : WORD;
END_VAR
```

sNetId: Ist ein String, der die AMS-Netzwerkkennung des EtherCAT Master Gerätes enthält.

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

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

reqState: EtherCAT Zustand der vom Master angefordert wird. Die möglichen Werte für *reqState* sind:

Konstante	Wert	Beschreibung
EC_DEVICE_STATE_INIT	0x01	Init-Zustand vom Master anfordern
EC_DEVICE_STATE_PREOP	0x02	Pre-Operational Zustand vom Master anfordern
EC_DEVICE_STATE_SAFEOP	0x04	Safe-Operational Zustand vom Master anfordern
EC_DEVICE_STATE_OP	0x08	Operational-Zustand vom Master anfordern

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy       : BOOL;
    bError      : BOOL;
    nErrId      : UDINT;
    currState   : WORD;
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.

nErrId: Liefert bei einem gesetzten bError-Ausgang den ADS-Fehlercode des zuletzt ausgeführten Befehles.

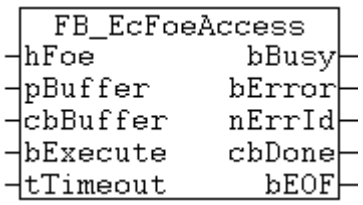
currState: Aktueller EtherCAT Zustand des Masters.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 Build > 1307 oder höher	PC or CX (x86) CX (ARM)	TcEtherCAT.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib, TcUtilities.Lib werden automatisch eingebunden)

9 FoE (File over EtherCAT)

9.1 FB_EcFoeAccess



Dieser Funktionsbaustein schreibt oder liest Daten über den Kommunikationsport des "File access over EtherCAT"-Mailboxprotokolls.

VAR_INPUT

```
VAR_INPUT
    hFoe     : T_HFoe;
    pBuffer  : DWORD;
    cbBuffer : UDINT;
    bExecute : BOOL;
    tTimeout : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

hFoe: "File access over EtherCAT"-Handle [▶ 86].

pBuffer: Enthält die Adresse des Puffers, in den die Daten gelesen werden sollen (Lesezugriff) oder die Adresse des Puffers der die zu schreibenden Daten enthält (Schreibzugriff). Der Puffer kann eine Einzelvariable, ein Array oder eine Struktur sein, dessen Adresse man mit dem ADR - Operator ermitteln kann.

cbBuffer: Enthält die Anzahl der zu schreibenden oder zu lesenden Datenbytes.

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

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

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy   : BOOL;
    bError  : BOOL;
    nErrId  : UDINT;
    cbDone  : UDINT;
    bEOF    : BOOL;
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.

nErrId: Liefert bei einem gesetzten bError-Ausgang den ADS-Fehlercode des zuletzt ausgeführten Befehles

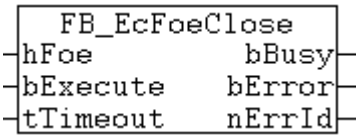
cbDone: Anzahl der zuletzt erfolgreich geschriebenen oder gelesenen Datenbytes.

bEOF: End of File. Diese Variable wird TRUE wenn beim Lesezugriff das Ende der Datei erreicht wurde. Beim Schreibzugriff hat diese Variable keine Bedeutung.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 oder höher	PC or CX (x86) CX (ARM)	TcEtherCAT.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib werden automatisch eingebunden)

9.2 FB_EcFoeClose



Dieser Funktionsbaustein schließt den Kommunikationsport für das "File access over EtherCAT"-Mailboxprotokoll.

VAR_INPUT

```
VAR_INPUT
    hFoe        : T_HFoe ;
    bExecute    : BOOL;
    tTimeout    : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

hFoe: "File access over EtherCAT"-Handle [▶ 86].

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

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

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy       : BOOL;
    bError      : BOOL;
    nErrId      : UDINT;
END_VAR
```

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

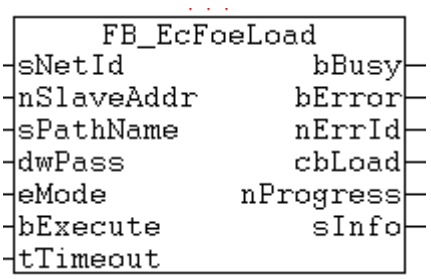
bError: Dieser Ausgang wird, nachdem der bBusy-Ausgang zurückgesetzt wurde, gesetzt, sollte ein Fehler bei der Übertragung des Kommandos erfolgen.

nErrId: Liefert bei einem gesetzten bError-Ausgang den ADS-Fehlercode des zuletzt ausgeführten Befehles.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 oder höher	PC or CX (x86) CX (ARM)	TcEtherCAT.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib, TcUtilities.Lib werden automatisch eingebunden)

9.3 FB_EcFoeLoad



Mit dem Funktionsbaustein FB_EcFoeLoad können Dateien über das "File access over EtherCAT"-Mailboxprotokoll (FoE) zu einem EtherCAT-Gerät herunter geladen oder von einem EtherCAT-Gerät hochgeladen werden.

Der Dateipfad kann nur auf das lokale Dateisystem des Rechners zeigen! Das bedeutet, Netzwerkpfade können hier nicht angegeben werden! Um Dateien über das FoE-Protokoll hoch- oder runterladen zu können, setzt der Funktionsbaustein das EtherCAT-Gerät automatisch in den BOOTSTRAP-Mode zurück. Zum Schluss versucht der Funktionsbaustein, das Gerät wieder in den ursprünglichen Zustand zu versetzen.

VAR_INPUT

```
VAR_INPUT
    sNetId      : T_AmsNetId;
    nSlaveAddr  : UINT;
    sPathName   : T_MaxString;
    dwPass      : DWORD := 0;
    eMode       : E_EcFoeMode := eFoeMode_Write;
    bExecute    : BOOL;
    tTimeout    : TIME := T#200s;
END_VAR
```

sNetId: Ist ein String, der die AMS-Netzwerkkennung des EtherCAT Master Gerätes enthält.

nSlaveAddr: Feste Adresse des EtherCAT Slaves, dessen Datei hoch- oder herunter geladen werden soll.

sPathName: Enthält den Pfad- und Dateinamen der zu schreibenden oder zu lesenden Datei (z.B.: 'C:\FOE_Test\EL6751\ECATFW__EL6751_C6_V0030.efw').

dwPass: Passwort (Default: 0).

eMode: "File access over EtherCAT"-Zugriffsmode [▶ 79] (Default: Schreibzugriff).

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

tTimeout: Maximale Zeit, die bei der Ausführung des Funktionsbausteins nicht überschritten werden darf (Default: 200s.).

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy       : BOOL;
    bError      : BOOL;
    nErrId      : UDINT;
    cbLoad      : UDINT;
    nProgress   : UDINT;
    sInfo       : T_MaxString;
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.

nErrId: Liefert bei einem gesetzten bError-Ausgang den ADS-Fehlercode des zuletzt ausgeführten Befehles

cbLoad: Anzahl der erfolgreich geschriebenen oder gelesenen Datenbytes.

nProgress: Prozentualer Fortschritt beim Schreibzugriff (Bereich: 0 - 100%). Beim Lesezugriff wird diese Variable zur Zeit nicht benutzt und ist immer 0.

sInfo: Zusätzliche Befehlsinformation als String (reserviert).

Beispiel in ST:

Bei einer steigenden Flanke an der bLoad-Variablen wird der Firmware-Download über das "File access over EtherCAT"-Mailboxprotokoll gestartet.

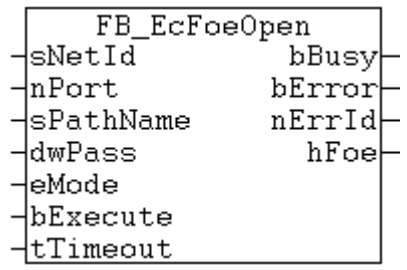
```
PROGRAM MAIN
VAR
    fbDownload : FB_EcFoeLoad := (
        sNetID := '5.0.34.38.3.1',
        nSlaveAddr := 1004,
        sPathName := 'C:\FOE_Test\EL6751\ECATFW__EL6751_C6_V0030.efw',
        dwPass := 0,
        eMode := eFoeMode_Write );

    bLoad : BOOL;
    bBusy : BOOL;
    bError : BOOL;
    nErrID : UDINT;
    nBytesWritten : UDINT;
    nPercent : UDINT;
END_VAR
```

```
fbDownload(      bExecute := bLoad,
        bBusy => bBusy,
        bError => bError,
        nErrId => nErrID,
        cbLoad => nBytesWritten,
        nProgress => nPercent );
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 Build > 1307	PC or CX (x86) CX (ARM)	TcEtherCAT.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib, TcUtilities.Lib werden automatisch eingebunden)

9.4 FB_EcFoeOpen



Mit diesem Funktionsbaustein wird der Kommunikationsport für das "File access over EtherCAT"-Mailboxprotokoll geöffnet.

VAR_INPUT

```
VAR_INPUT
    sNetId      : T_AmsNetId;
    nPort       : UINT;
    sPathName   : T_MaxString;
    dwPass      : DWORD;
    eMode       : E_EcFoeMode;
    bExecute    : BOOL;
    tTimeout    : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

sNetId: Ist ein String, der die AMS-Netzwerkkennung des EtherCAT-Gerätes enthält.

nPort: Feste Adresse des EtherCAT-Gerätes.

sPathName: Der Dateipfadname (z.B.: 'c:\TwinCAT\FOE\Data.fwp').

dwPass: Passwort.

eMode: Zugriffsmode [▶ 79] (Schreib-/Lesezugriff).

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

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

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy       : BOOL;
    bError      : BOOL;
    nErrId      : UDINT;
    hFoe        : T_HFoe;
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.

nErrId: Liefert bei einem gesetzten bError-Ausgang den ADS-Fehlercode des zuletzt ausgeführten Befehles.

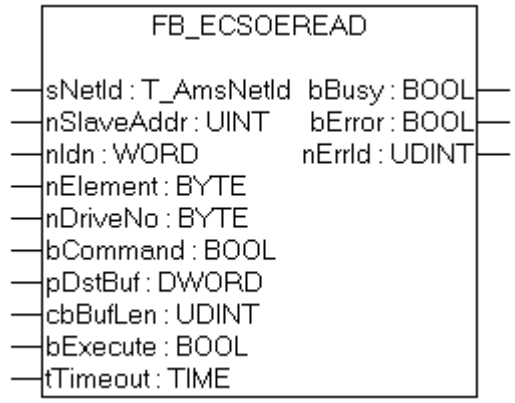
hFoe: "File access over EtherCAT"-Handle [▶ 86].

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 oder höher	PC or CX (x86) CX (ARM)	TcEtherCAT.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib, TcUtilities.Lib werden automatisch eingebunden)

10 SoE

10.1 FB_EcSoeRead



Mit dem Funktionsbaustein FB_EcSoeRead können Antriebs-Parameter mit Hilfe des "Servo Drive Profile over EtherCAT"(SoE) Protokolls ausgelesen werden. Dazu muss der Slave eine Mailbox besitzen und das SoE Protokoll unterstützen. Der auszulesende Antriebs-Parameter wird mit den Parametern nIdn (Identification number), nElement und nDriveNo spezifiziert.

VAR_INPUT

```
VAR_INPUT
    sNetId      : T_AmsNetId;
    nSlaveAddr  : UINT;
    nSubIndex   : BYTE;
    nIdn        : WORD;
    nElement    : BYTE;
    nDriveNo    : BYTE;
    bCommand    : BOOL
    pDstBuf     : DWORD;
    cbBufLen    : UDINT;
    bExecute    : BOOL;
    tTimeout    : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

sNetId: Ist ein String, der die AMS-Netzwerkkennung des EtherCAT Master Gerätes enthält.

nSlaveAddr: Feste Adresse des EtherCAT Slaves an den das SoE Read Kommando geschickt werden soll.

nIdn: Identifikations-Nummer des zu lesenden Parameters.

nElement: Element-Nummer des zu lesenden Parameters. Folgende Werte sind zulässig:

Wert	Beschreibung
0x01	Data Status
0x02	Name (read only)
0x04	Attribut
0x08	Einheit
0x10	Minimum
0x20	Maximum
0x40	Wert
0x80	Default

nDriveNo: Nummer des Antriebs.

bCommand: Dieser Parameter sollte gesetzt werden, wenn interne Kommando-Ausführung verwendet werden soll.

pDstBuf: Die Adresse (Pointer) auf den Empfangspuffer.

cbBufLen: Die maximal verfügbare Puffergröße für die zu lesenden Daten in Bytes.

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

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

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy       : BOOL;
    bError      : BOOL;
    nErrId      : UDINT;
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.

nErrId: Liefert bei einem gesetzten bError-Ausgang den ADS-Fehlercode des zuletzt ausgeführten Befehles

Beispiel für eine Implementierung in ST:

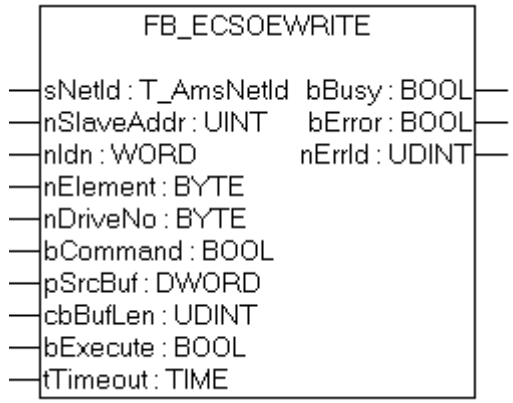
```
PROGRAM TEST_SoERead
VAR
    fbSoERead : FB_EcSoERead;
    sNetId : T_AmsNetId := '172.16.2.131.2.1';
    bExecute : BOOL;
    nSlaveAddr : UINT := 1006;
    nIdn : WORD := 15;
    nElement : BYTE := 0;
    nDriveNo : BYTE := 0;
    bCommand : BOOL := FALSE;
    val : UINT;
    bError : BOOL;
    nErrId : UDINT;
END_VAR
```

```
fbSoERead(sNetId:= sNetId,nSlaveAddr :=nSlaveAddr, nIdn := nIdn, nElement:=nElement, nDriveNo := nDr
iveNo, bCommand:=bCommand, pDstBuf:= ADR(val), cbBufLen:=SIZEOF(val),bExecute:=bExecute);
bError:=fbSoERead.bError;
nErrId:=fbSoERead.nErrId;
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 oder höher	PC or CX (x86)	TcEtherCAT.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib, TcUtilities.Lib werden automatisch eingebunden)
TwinCAT v2.10.0 Build >= 1301 oder höher	CX (ARM)	

10.2 FB_EcSoeWrite



Mit dem Funktionsbaustein FB_EcSoeWrite können Antriebs-Parameter mit Hilfe des "Servo Drive Profile over EtherCAT"(SoE) Protokolls beschrieben werden. Dazu muss der Slave eine Mailbox besitzen und das SoE Protokoll unterstützen. Der zu schreibende Antriebs-Parameter wird mit den Parametern nIdn (Identification number), nElement und nDriveNo spezifiziert.

VAR_INPUT

```
VAR_INPUT
    sNetId      : T_AmsNetId;
    nSlaveAddr  : UINT;
    nIdn        : WORD;
    nElement    : BYTE;
    nDriveNo    : BYTE;
    bCommand    : BOOL;
    pSrcBuf     : DWORD;
    cbBufLen    : UDINT;
    bExecute    : BOOL;
    tTimeout    : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

sNetId: Ist ein String, der die AMS-Netzwerkkennung des EtherCAT Master Gerätes enthält.

nSlaveAddr: Feste Adresse des EtherCAT Slaves and den das SoE Write Kommando geschickt werden soll.

nIdn: Identifikations-Nummer des zu schreibenden Parameters.

nElement: Element-Nummer des zu schreibenden Parameters. Folgende Werte sind zulässig:

Wert	Beschreibung
0x01	Data Status
0x02	Name (read only)
0x04	Attribut
0x08	Einheit
0x10	Minimum
0x20	Maximum
0x40	Wert
0x80	Default

nDriveNo: Nummer des Antriebs.

bCommand: Dieser Parameter sollte gesetzt werden, wenn interne Kommando-Ausführung verwendet werden soll.

pSrcBuf: Die Adresse (Pointer) auf den Sendepuffer.

cbBufLen: Die Anzahl der zu sendenden Daten in Bytes.

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

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

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy       : BOOL;
    bError      : BOOL;
    nErrId      : UDINT;
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.

nErrId: Liefert bei einem gesetzten bError-Ausgang den ADS-Fehlercode des zuletzt ausgeführten Befehles

Beispiel für eine Implementierung in ST:

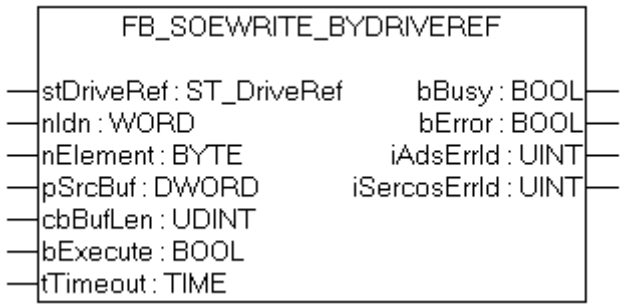
```
PROGRAM TEST_SoEWrite
VAR
    fbSoeWrite : FB_EcSoEWrite;
    sNetId : T_AmsNetId := '172.16.2.131.2.1';
    bExecute : BOOL;
    nSlaveAddr : UINT := 1006;
    nIdn : WORD := 15;
    nElement : BYTE := 0;
    nDriveNo : BYTE := 0;
    bCommand : BOOL := FALSE;
    val : UINT;
    bError : BOOL;
    nErrId : UDINT;
END_VAR
```

```
fbSoEWrite(sNetId:= sNetId,nSlaveAddr :=nSlaveAddr, nIdn := nIdn, nElement:=nElement, nDriveNo := nD
riveNo,bCommand:=bCommand, pSrcBuf:= ADR(val), cbBufLen:=SIZEOF(val),bExecute:=bExecute);
bError:=fbSoEWrite.bError;
nErrId:=fbSoEWrite.nErrId;
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 oder höher	PC or CX (x86)	TcEtherCAT.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib, TcUtilities.Lib werden automatisch eingebunden)
TwinCAT v2.10.0 Build >= 1301 oder höher	CX (ARM)	

10.3 FB_SoEWrite_ByDriveRef



Mit dem Funktionsbaustein FB_SoeWrite_ByRef können Antriebs-Parameter mit Hilfe des "Servo Drive Profile over EtherCAT"(SoE) Protokolls beschrieben werden. Dazu muss der Slave eine Mailbox besitzen und das SoE Protokoll unterstützen. Der zu schreibende Antriebs-Parameter wird mit den Parametern nIdn (Identification number), nElement und stDriveRef spezifiziert.

VAR_INPUT

```
VAR_INPUT
    stDriveRef      : ST_DriveRef; (* contains sNetID of EcMaster, nSlaveAddr of EcDrive, nDriveNo o
f EcDrive, either preset or read from NC *)
    nIdn            : WORD; (* SoE IDN: e.g. "S_0_IDN + 1" for S-0-0001 or "P_0_IDN + 23" for P-0-00
23*)
    nElement        : BYTE; (* SoE element.*)
    pSrcBuf         : DWORD; (* Contains the address of the buffer containing the data to be send. *
)
    cbBufLen        : UDINT; (* Contains the max. number of bytes to be received. *)
    bExecute        : BOOL; (* Function block execution is triggered by a rising edge at this input.
*)
    tTimeout        : TIME := DEFAULT_ADS_TIMEOUT;
(* States the time before the function is cancelled. *)
END_VAR
```

stDriveRef: Die Referenz auf den Antrieb kann im System Manager direkt in die SPS gelinkt werden. Hierzu muss eine Instanz der ST_PlcDriveRef verwendet werden und die NetID vom Bytearray in einen String konvertiert werden. Siehe ST_DriveRef.

nIdn: Identifikations-Nummer des zu lesenden Parameters.

nElement: Element-Nummer des zu lesenden Parameters. Folgende Werte sind zulässig:

Wert	Beschreibung
0x01	Data Status
0x02	Name (read only)
0x04	Attribut
0x08	Einheit
0x10	Minimum
0x20	Maximum
0x40	Wert
0x80	Default

pSrcBuf: Die Adresse (Pointer) auf den sendenden Puffer.

cbBufLen: Die maximal verfügbare Puffergröße für die zu lesenden Daten in Bytes.

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

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

VAR_OUTPUT

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

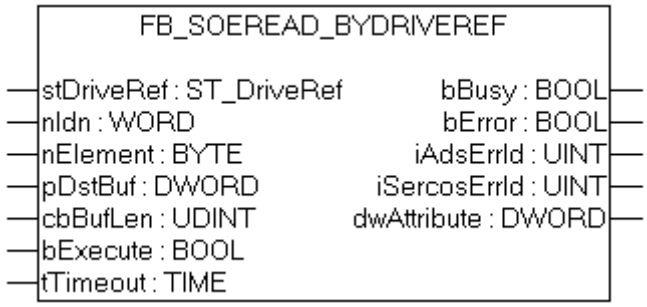
iAdsErrId: Liefert bei gesetztem bError-Ausgang den ADS-Fehlercode des zuletzt ausgeführten Befehles

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

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 oder höher	PC or CX (x86)	TcEtherCAT.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib, TcUtilities.Lib werden automatisch eingebunden)
TwinCAT v2.10.0 Build >= 1301 oder höher	CX (ARM)	

10.4 FB_SoERead_ByDriveRef



Mit dem Funktionsbaustein FB_SoeRead_ByRef können Antriebs-Parameter mit Hilfe des "Servo Drive Profile over EtherCAT"(SoE) Protokolls gelesen werden. Dazu muss der Slave eine Mailbox besitzen und das SoE Protokoll unterstützen. Der zu lesende Antriebs-Parameter wird mit den Parametern nIdn (Identification number), nElement und stDriveRef spezifiziert.

VAR_INPUT

```
VAR_INPUT
    stDriveRef      : ST_DriveRef; (* contains sNetID of EcMaster, nSlaveAddr of EcDrive, nDriveNo o
f EcDrive, either preset or read from NC *)
    nIdn            : WORD; (* SoE IDN: e.g. "S_0_IDN + 1" for S-0-0001 or "P_0_IDN + 23" for P-0-00
23*)
    nElement        : BYTE; (* SoE element.*)
    pDstBuf         : DWORD; (* Contains the address of the buffer for the received data. *)
    cbBufLen        : UDINT; (* Contains the max. number of bytes to be received. *)
    bExecute        : BOOL; (* Function block execution is triggered by a rising edge at this input.
*)
    tTimeout        : TIME := DEFAULT_ADS_TIMEOUT;
(* States the time before the function is cancelled. *)
END_VAR
```

stDriveRef: Die Referenz auf den Antrieb kann im System Manager direkt in die SPS gelinkt werden. Hierzu muss eine Instanz der ST_PlcDriveRef verwendet werden und die NetID vom Bytearray in einen String konvertiert werden. Siehe ST_DriveRef.

nIdn: Identifikations-Nummer des zu lesenden Parameters.

nElement: Element-Nummer des zu lesenden Parameters. Folgende Werte sind zulässig:

Wert	Beschreibung
0x01	Data Status
0x02	Name (read only)
0x04	Attribut
0x08	Einheit
0x10	Minimum
0x20	Maximum

Wert	Beschreibung
0x40	Wert
0x80	Default

pDstBuf: Die Adresse (Pointer) auf den lesenden Puffer.

cbBufLen: Die maximal verfügbare Puffergröße für die zu lesenden Daten in Bytes.

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

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

VAR_OUTPUT

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

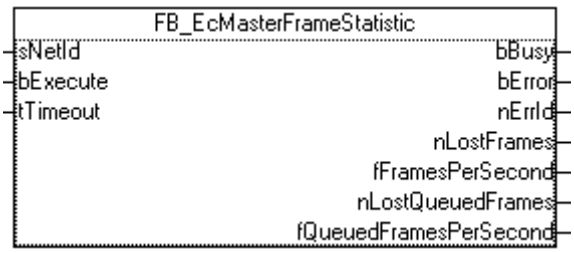
iSercosErrId: Liefert bei gesetztem bError-Ausgang den Sercos-Fehler des zuletzt ausgeführten Befehles

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 oder höher	PC or CX (x86)	TcEtherCAT.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib, TcUtilities.Lib werden automatisch eingebunden)
TwinCAT v2.10.0 Build >= 1301 oder höher	CX (ARM)	

11 Frame Statistic

11.1 FB_EcMasterFrameStatistic



Mit dem Funktionsbaustein FB_EcMasterFrameStatistic kann die Framestatistic des EtherCAT Master ausgelesen werden. Die Anzahl der 'lost frames', die Frames pro Sekunde, die Anzahl der verloren gegangenen Queued Frames und die Anzahl der Queued Frames pro Sekunde wird am Ausgang des Bausteins zur Verfügung gestellt.

VAR_INPUT

```
VAR_INPUT
    sNetId      : T_AmsNetId;
    bExecute    : BOOL;
    tTimeout    : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

sNetId: Ist ein String, der die AMS-Netzwerkkennung des EtherCAT Master Gerätes enthält.

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

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

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy                 : BOOL;
    bError                : BOOL;
    nErrId                : UDINT;
    nLostFrames           : UDINT;
    fFramesPerSecond      : LREAL;
    nLostQueuedFrames     : UDINT;
    fQueuedFramesPerSecond : LREAL;
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.

nErrId: Liefert bei einem gesetzten bError-Ausgang den ADS-Fehlercode des zuletzt ausgeführten Befehles

nLostFrames: Liefert die momentane Anzahl der verloren gegangenen Frames.

fFramesPerSecond: Liefert die momentane Anzahl der Frames pro Sekunde.

nLostQueuedFrames: Liefert die momentane Anzahl der verloren gegangenen Queued Frames.

fQueuedFramesPerSecond: Liefert die momentane Anzahl der Queued Frames pro Sekunde.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 oder höher	PC or CX (x86)	TcEtherCAT.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib, TcUtilities.Lib werden automatisch eingebunden)
TwinCAT v2.10.0 Build >= 1301 oder höher	CX (ARM)	

11.2 FB_EcMasterFrameStatisticClearCRC

Mit dem Funktionsbaustein FB_EcMasterFrameStatisticClearCRC können die CRC Fehlerzähler aller EtherCAT Slaves gelöscht werden. .

VAR_INPUT

```
VAR_INPUT
    sNetId      : T_AmsNetId;
    bExecute    : BOOL;
    tTimeout    : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

sNetId: Ist ein String, der die AMS-Netzwerkkennung des EtherCAT Master Gerätes enthält.

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

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

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy       : BOOL;
    bError      : BOOL;
    nErrId      : UDINT;
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.

nErrId: Liefert bei einem gesetzten bError-Ausgang den ADS-Fehlercode des zuletzt ausgeführten Befehles

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 oder höher	PC or CX (x86)	TcEtherCAT.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib, TcUtilities.Lib werden automatisch eingebunden)
TwinCAT v2.10.0 Build >= 1301 oder höher	CX (ARM)	

11.3 FB_EcMasterFrameStatisticClearFrames

Mit dem Funktionsbaustein FB_EcMasterFrameStatisticClearFrames kann der Zähler der 'lost frames' gelöscht werden.

VAR_INPUT

```
VAR_INPUT
    sNetId      : T_AmsNetId;
    bExecute    : BOOL;
    tTimeout    : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

sNetId: Ist ein String, der die AMS-Netzwerkkennung des EtherCAT Master Gerätes enthält.

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

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

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy       : BOOL;
    bError      : BOOL;
    nErrId      : UDINT;
END_VAR
```

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

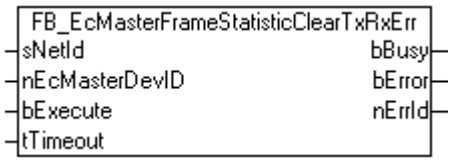
bError: Dieser Ausgang wird, nachdem der bBusy-Ausgang zurückgesetzt wurde, gesetzt, sollte ein Fehler bei der Übertragung des Kommandos erfolgen.

nErrId: Liefert bei einem gesetzten bError-Ausgang den ADS-Fehlercode des zuletzt ausgeführten Befehles

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 oder höher	PC or CX (x86)	TcEtherCAT.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib, TcUtilities.Lib werden automatisch eingebunden)
TwinCAT v2.10.0 Build >= 1301 oder höher	CX (ARM)	

11.4 FB_EcMasterFrameStatisticClearTxRxErr



Mit dem Funktionsbaustein FB_EcMasterFrameStatisticClearTxRxErr können die Fehlerzähler des Miniport-Treiber der Netzwerkkarte gelöscht werden.

VAR_INPUT

```
VAR_INPUT
    sNetId          : T_AmsNetId;
    nEcMasterDevID  : INT;
    bExecute        : BOOL;
    tTimeout        : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

sNetId: Ist ein String, der die AMS-Netzwerkkennung der CPU (PC) enthält.

nEcMasterDevID: Device ID des EtherCAT Masters.

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

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

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy       : BOOL;
    bError      : BOOL;
    nErrId      : UDINT;
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.

nErrId: Liefert bei einem gesetzten bError-Ausgang den ADS-Fehlercode des zuletzt ausgeführten Befehles

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 oder höher	PC or CX (x86)	TcEtherCAT.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib, TcUtilities.Lib werden automatisch eingebunden)
TwinCAT v2.10.0 Build >= 1301 oder höher	CX (ARM)	

12 Datentypen

12.1 DCTIMESTRUCT

Strukturierter TwinCAT "Distributed Clock System Time"-Zeitformat. Die kleinste Einheit ist eine Nanosekunde. Dieser Datentyp repräsentiert die **Anzahl der Nanosekunden seit dem 01.01.2000 (GMT)**.

```
TYPE DCTIMESTRUCT :
STRUCT
    wYear           : WORD;
    wMonth          : WORD;
    wDayOfWeek      : WORD;
    wDay            : WORD;
    wHour           : WORD;
    wMinute         : WORD;
    wSecond         : WORD;
    wMilliseconds   : WORD;
    wMicroseconds   : WORD;
    wNanoseconds    : WORD;
END_STRUCT
END_TYPE
```

wYear : Das Jahr: 2000 ~ 2584;

wMonth : Der Monat: 1 ~ 12 (Januar = 1, Februar = 2 usw.);

wDayOfWeek : Der Wochentag: 0 ~ 6 (Sonntag = 0, Montag = 1 usw.);

wDay : Tag des Monats: 1 ~ 31;

wHour : Stunde: 0 ~ 23;

wMinute : Minute: 0 ~ 59;

wSecond : Sekunde: 0 ~ 59;

wMilliseconds : Millisekunde: 0 ~ 999;

wMicroseconds : Microsekunde: 0 ~ 999;

wNanoseconds : Nanosekunde: 0 ~ 999;

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 Build > 1316 oder höher	PC oder CX (x86, ARM)	TcEtherCAT.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib, TcUtilities.Lib werden automatisch eingebunden)

12.2 E_EcAdressingType

```
TYPE E_EcAdressingType :
(
    eAdressingType_AutoInc:=1, (* Adress slave by it's position. (adp = 1-
position, 1.Slave = 0, 2.Slave = 0xffff(-1) etc) *)
                (* EtherCAT commands: APRD, APWR, APRW *)
    eAdressingType_Fixed,   (* Adress slave by configured ethercat slave address (adp = configured a
ddress) *)
                (* EtherCAT commands: FPRD, FPWR, FPRW *)
    eAdressingType_Broadcast    (* Adress all slaves. *)
                (* EtherCAT commands: BRD, BWR, BRW *)
);
END_TYPE
```

12.3 E_EcFoeMode

Zugriffsmode für das "File access over EtherCAT"-Mailboxprotokoll.

```
TYPE E_EcFoeMode :
(
    eFoeMode_Write := 1,
    eFoeMode_Read
);
END_TYPE
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 Build > 1307	PC or CX (x86) CX (ARM)	TcEtherCAT.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib, TcUtilities.lib werden automatisch eingebunden)

12.4 E_EcMbxProtType

EtherCAT-Mailboxprotokoll.

```
TYPE E_EcMbxProtType:
(
    eEcMbxProt_CoE := 3,(* CANopen over EtherCAT *)
    eEcMbxProt_FoE := 4,(* File over EtherCAT *)
    eEcMbxProt_SoE := 5 (* Servo Drive Profile over EtherCAT *)
);
END_TYPE
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 Build > 1307	PC or CX (x86) CX (ARM)	TcEtherCAT.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib, TcUtilities.lib werden automatisch eingebunden)

12.5 E_EcScanSlavesCommandStatus

```
TYPE E_EcScanSlavesCommandStatus :
    (
    eEcScanSlavesCommandState_Completed_NoErrors_NoReply := 0,(* completed, no errors, no reply *)
    eEcScanSlavesCommandState_Completed_NoErrors_Reply := 1, (* completed, no errors, reply *)
    eEcScanSlavesCommandState_Completed_Error_NoReply := 2, (* completed, errors, no reply *)
    eEcScanSlavesCommandState_Completed_Error_Reply := 3, (* completed, errors, reply *)
    eEcScanSlavesCommandState_Completed_Reserved := 4 (* reserved *)
    );
END_TYPE
```

12.6 ProductCode

```
TYPE ProductCode :
    (
    PCTYPE_XXDDDD       := 0,
    PCTYPE_XXDDDD_DDDD      := 1,
    PCTYPE_XXDDDD_DDDD_DDDD     := 2,
    PCTYPE_XXDDDD_XDDD      := 3,
    PCTYPE_XXDDDD_XDDD_DDDD     := 4,
    PCTYPE_XXDDDD_DDDD_XDDD     := 5,
    PCTYPE_XXDDDD_XDDD_XDDD     := 6
    );
END_TYPE
```

12.7 ST_EcCrcError

Struktur mit den CRC-Error Zählern der einzelnen Ports (A,B und C) eines EtherCAT Slave Gerätes.

```
TYPE ST_EcCrcError :
STRUCT
    portA : UDINT;
    portB : UDINT;
    portC : UDINT;
END_STRUCT
END_TYPE
```

portA: CRC-Error Zähler des PortA

portB: CRC-Error Zähler des PortB

portC: CRC-Error Zähler des PortC

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 oder höher	PC or CX (x86)	TcEtherCAT.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib, TcUtilities.Lib werden automatisch eingebunden)
TwinCAT v2.10.0 Build >= 1301 oder höher	CX (ARM)	

12.8 ST_EcCrcErrorEx

Struktur mit den CRC-Error Zählern der einzelnen Ports (A,B,C und D) eines EtherCAT Slave Gerätes.

```
TYPE ST_EcCrcErrorEx :
STRUCT
    portA : UDINT;
    portB : UDINT;
    portC : UDINT;
    portD : UDINT;
END_STRUCT
END_TYPE
```

portA: CRC-Error Zähler des PortA

portB: CRC-Error Zähler des PortB

portC: CRC-Error Zähler des PortC

portD: CRC-Error Zähler des PortD

12.9 ST_EcLastProtErrInfo

Die Struktur ST_EcLastProtErrInfo enthält zusätzliche Fehlerinformationen zum zuletzt aufgetretenen "EtherCAT-Mailboxprotokollfehler".

```
TYPE ST_EcSlaveState:
STRUCT
    ownAddr : ST_AmsAddr;
    orgAddr : ST_AmsAddr;
    errCode : UDINT;
    binDesc : ARRAY[0..MAX_STRING_LENGTH] OF BYTE;
END_STRUCT
END_TYPE
```

ownAddr: Eigene AMS-Adresse (Adresse des Kommunikationsteilnehmers der die Fehlerinformationen abfragt).

orgAddr: AMS-Adresse des Fehlerverursachers (Adresse des Kommunikationsteilnehmers der den Protokollfehler ausgelöst oder verursacht hat).

errCode: Die Mailboxprotokollfehlernummer [▶ 90] (SoE, CoE, FoE error code).

binDesc: Zusätzliche Fehlerinformationen als Binärdaten. Die zusätzliche Fehlerinformation ist gerätespezifisch und kann z.B. einen String oder Binärdaten enthalten.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 Build > 1307 oder höher	PC or CX (x86, ARM)	TcEtherCAT.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib, TcUtilities.Lib werden automatisch eingebunden)

12.1 ST_EcMasterStatistic 0

```
TYPE ST_EcMasterStatistic :
STRUCT
    nSysTime                : UDINT;
    nCycFrameCnt            : UDINT;
    nCycFrameMissedCnt      : UDINT;
    nQueuedFrameCnt         : UDINT;
    nQueuedFrameMissedCnt   : UDINT;
END_STRUCT
END_TYPE
```

nSysTime: Systemzeit in µs

nCycFrameCnt: Anzahl der zyklischen EtherCAT-Frames

nCycFrameMissedCnt: Anzahl der verlorenen zyklischen EtherCAT-Frames

nQueuedFrameCnt: Anzahl der azyklischen EtherCAT-Frames

nQueuedFrameMissedCnt: Anzahl der verlorenen azyklischen EtherCAT-Frames

12.1 ST_EcSlaveConfigData 1

Die Struktur ST_EcSlaveConfigData enthält die EtherCAT Konfigurationsdaten eines EtherCAT Slave Gerätes.

```
TYPE ST_EcSlaveConfigData:
STRUCT
    nEntries         : WORD;
    nAddr            : WORD;
    sType            : STRING[15];
    sName            : STRING[31];
    nDevType         : DWORD;
    stSlaveIdentity  : ST_EcSlaveIdentity;
    nMailboxOutSize  : WORD;
    nMailboxInSize   : WORD;
    nLinkStatus      : BYTE;
END_STRUCT
END_TYPE
```

nEntries: intern verwendet!

nAddr: Adresse eines EtherCAT Slaves.

sType: EtherCAT Typ eines Slaves.

sName: Name eines EtherCAT Slaves.

nDevType: EtherCAT Device Typ eines Slaves.

stSlaveIdentity: Identity eines EtherCAT Slaves (s. ST_EcSlaveIdentity [▶ 82]).

nMailboxOutSize: OutSize der Mailbox eines EtherCAT Slaves.

nMailboxInSize: InSize der Mailbox eines EtherCAT Slaves.

nLinkStatus: Link Status eines EtherCAT Slaves.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 oder höher	PC or CX (x86)	TcEtherCAT.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib, TcUtilities.Lib werden automatisch eingebunden)
TwinCAT v2.10.0 Build >= 1307 oder höher	CX (ARM)	

12.1 ST_EcSlaveIdentity 2

```
TYPE ST_EcSlaveIdentity :
STRUCT
    vendorId        : UDINT;
    productCode     : UDINT;
    revisionNo      : UDINT;
    serialNo        : UDINT;
END_STRUCT
END_TYPE
```

vendorId: Vendor-ID des Slave-Gerätes.

productCode: Produkt-Code des Slave-Gerätes.

revisionNo: Zeigt die Revision-Nummer des Slave-Gerätes an.

serialNo: Zeigt die Serien-Nummer des Slave-Gerät an.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 oder höher	PC or CX (x86)	TcEtherCAT.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib, TcUtilities.Lib werden automatisch eingebunden)
TwinCAT v2.10.0 Build >= 1301 oder höher	CX (ARM)	

12.1 ST_EcSlaveScannedData 3

Die Struktur ST_EcSlaveScannedData enthält die EtherCAT Konfigurationsdaten eines gescannten EtherCAT Slave Gerätes.

```
TYPE ST_EcSlaveConfigData:
STRUCT
    nEntries        : WORD;
    nAddr           : WORD;
    stSlaveIdentity : ST_EcSlaveIdentity;
```

```
    ndlStatusReg    : WORD;
END_STRUCT
END_TYPE
```

nEntries: intern verwendet!

nAddr: Adresse eines EtherCAT Slaves.

stSlaveIdentity: Identity eines EtherCAT Slaves (s. ST_EcSlaveIdentity).

ndlStatusReg: Link Status eines EtherCAT Slaves aus dem ESC Register $0110/0111_{hex}$ bzw. $272/273_{dec}$. Ist der Slave nicht erreichbar/offline, wird der Status 0 angezeigt. Die Zuordnung „PortNummer <=> Buchse/ Ebus Kontakt“ ist der jeweiligen Gerätedokumentation zu entnehmen. Wenn nicht anders beschrieben, ist Port0 der linke Ebus-Kontakt einer EL/ES-Klemme bzw. RJ45-Buchse einer EP-Box, Port1 der rechte abgehende Ebus-Kontakt/RJ45-Buchse.

Die Bitbedeutungen lauten

Bit	Bedeutung
1	internal use
2	internal use
3	internal use
4	physical link on Port 0 0: no link, 1: Link detected
5	physical link on Port 1 0: no link, 1: Link detected
6	physical link on Port 2 0: no link, 1: Link detected
7	physical link on Port 3 0: no link, 1: Link detected
8	Loop Port 0 0: Open, 1:Closed
9	Communication on Port 0 0:no stable communication, 1:Communication established
10	Loop Port 1 0: Open, 1:Closed
11	Communication on Port 1 0:no stable communication, 1:Communication established
12	Loop Port 2 0: Open, 1:Closed
13	Communication on Port 2 0:no stable communication, 1:Communication established
14	Loop Port 3 0: Open, 1:Closed
15	Communication on Port 3 0:no stable communication, 1:Communication established

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 oder höher	PC or CX (x86)	TcEtherCAT.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib, TcUtilities.Lib werden automatisch eingebunden)
TwinCAT v2.10.0 Build >= 1307 oder höher	CX (ARM)	

12.1 ST_EcSlaveState

Die Struktur ST_EcSlaveState enthält den EtherCAT Status und den Link Status eines EtherCAT Slave Gerätes.

```
TYPE ST_EcSlaveState:
STRUCT
    deviceState     : BYTE;
    linkState       : BYTE;
END_STRUCT
END_TYPE
```

deviceState: EtherCAT Status eines Slaves. Der Status kann einen der folgenden Werte annehmen:

Konstante	Wert	Beschreibung
EC_DEVICE_STATE_INIT	0x01	Init-Zustand
EC_DEVICE_STATE_PREOP	0x02	Pre-Operational Zustand
EC_DEVICE_STATE_BOOTSTRAP	0x03	Bootstrap Zustand
EC_DEVICE_STATE_SAFEOP	0x04	Safe-Operational Zustand
EC_DEVICE_STATE_OP	0x08	Operational-Zustand

Zusätzlich können noch folgende Bits gesetzt sein:

Konstante	Wert	Beschreibung
EC_DEVICE_STATE_ERROR	0x10	Statemachine-Fehler im EtherCAT-Slave
EC_DEVICE_STATE_INVALID_VPRS	0x20	Ungültige VendorId, Product-Code, Revisionsnummer oder Seriennummer
EC_DEVICE_STATE_INITCMD_ERROR	0x40	Fehler beim Senden von Initialsierungs-Kommandos aufgetreten.

linkState: Link Status eines EtherCAT Slaves. Der Link Status kann eine Oder-Verknüpfung folgender Bits sein.

Konstante	Wert	Beschreibung
EC_LINK_STATE_OK	0x00	
EC_LINK_STATE_NOT_PRESENT	0x01	Keine EtherCAT-Kommunikation mit dem EtherCAT-Slave
EC_LINK_STATE_LINK_WITHOUT_COMM	0x02	Fehler an Port X(festgelegt durch EC_LINK_STATE_PORT_A/B/C/D). Der Port hat einen Link, aber keine Kommunikation über diesen Port ist möglich.
EC_LINK_STATE_MISSING_LINK	0x04	Fehlender Link an Port X(festgelegt durch EC_LINK_STATE_PORT_A/B/C/D).
EC_LINK_STATE_ADDITIONAL_LINK	0x08	Zusätzlicher Link an Port X(festgelegt durch EC_LINK_STATE_PORT_A/B/C/D).
EC_LINK_STATE_PORT_A	0x10	Port 0
EC_LINK_STATE_PORT_B	0x20	Port 1
EC_LINK_STATE_PORT_C	0x40	Port 2
EC_LINK_STATE_PORT_D	0x80	Port 3

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 oder höher	PC or CX (x86)	TcEtherCAT.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib, TcUtilities.Lib werden automatisch eingebunden)
TwinCAT v2.10.0 Build >= 1301 oder höher	CX (ARM)	

12.1 ST_EcSlaveStateBits 5

Die Struktur ST_EcSlaveStateBits enthält den EtherCAT Status und den Link Status eines EtherCAT Slave Gerätes.

```
TYPE ST_EcSlaveStateBits:
STRUCT
    bInit             : BOOL;
    bPreop            : BOOL;
    bBootStrap        : BOOL;
    bSafeOp           : BOOL;
    bOp               : BOOL;
    bError            : BOOL;
    bInvVPRS          : BOOL;
    bInitCmdError     : BOOL;

    bLinkNotPresent   : BOOL;
    bLinkWithoutComm  : BOOL;
    bLinkMissing      : BOOL;
    bAdditionalLink   : BOOL;
    bPortA            : BOOL;
    bPortB            : BOOL;
    bPortC            : BOOL;
    bPortD            : BOOL;
END_STRUCT
END_TYPE
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 oder höher	PC or CX (x86)	TcEtherCAT.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib, TcUtilities.Lib werden automatisch eingebunden)
TwinCAT v2.10.0 Build >= 1301 oder höher	CX (ARM)	

12.1 ST_TPCTYPE_CODE_XXDDD 6

```
TYPE ST_TPCTYPE_CODE_XXDDD :
STRUCT
    ty : UINT;
    c1 : UINT;
    c2 : UINT;
    d1 : UINT;
    d2 : UINT;
    d3 : UINT;
END_STRUCT
END_TYPE
```

12.1 ST_TPCTYPE_CODE_XXDDXD 7

```
TYPE ST_TPCTYPE_CODE_XXDDXD :
STRUCT
    ty : UINT;
    c1 : UINT;
    c2 : UINT;
    d1 : UINT;
    c3 : UINT;
    d2 : UINT;
    d3 : UINT;
END_STRUCT
END_TYPE
```

12.1 ST_TPCTYPE_CODE_XXDXDD 8

```
TYPE ST_TPCTYPE_CODE_XXDXDD :
STRUCT
    ty : UINT;
    c1 : UINT;
    c2 : UINT;
    d1 : UINT;
    d2 : UINT;
    c3 : UINT;
    d3 : UINT;
END_STRUCT
END_TYPE
```

12.1 ST_TPCTYPE_CODE_XXDXDXD 9

```
TYPE ST_TPCTYPE_CODE_XXDXDXD :
STRUCT
    ty : UINT;
    c1 : UINT;
    c2 : UINT;
    d1 : UINT;
    c3 : UINT;
    d2 : UINT;
    c4 : UINT;
    d3 : UINT;
    END_STRUCT
END_TYPE
```

12.2 T_HFoe 0

"File access over EtherCAT"-Handle. Das Handle muss vor der Benutzung einmalig mit dem Funktionsbaustein FB_EcFoeOpen [▶ 65] initialisiert werden. Die Variablen dieses strukturierten Typs dürfen nicht direkt beschrieben werden.

```
TYPE T_HFoe :
STRUCT
    sNetID  : T_AmsNetId := '';
    nPort   : T_AmsPort := 0;
    handle  : UDINT := 0;
    eMode   : E_EcFoeMode := eFoeMode_Write;
END_STRUCT
END_TYPE
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 Build > 1307	PC or CX (x86) CX (ARM)	TcEtherCAT.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib, TcUtilities.Lib werden automatisch eingebunden)

Sehen Sie dazu auch

E_EcFoeMode [▶ 79]

12.2 T_DCTIME 1

Der Datentyp T_DCTIME repräsentiert die *Distributed Clock System Time* (kurz *DC Time* genannt) als linearen 64 Bit Integer Wert. Die Zeit wird in Nanosekunden seit dem 1.1.2000 UTC dargestellt.
Der Datentyp wird als zwei 32 Bit DWORD Variablen dargestellt, so dass er in der SPS einfach verarbeitet werden kann. Operationen (Addition und Subtraktion von Zeiten) können mit ui64 Funktionen aus der Bibliothek *TcUtilities.lib* ausgeführt werden.

```
TYPE T_DCTIME : T_ULARGE_INTEGER;
END_TYPE
```

Nützliche "Distributed Clock System Time"-Konstanten	Beschreibung
EC_DCTIME_DELTA_OFFSET	Anzahl der 100-Nanosekunden-Ticks zwischen dem 01.01.1601 und 01.01.2000. Es ist die Differenz zwischen der "Windows File Time" und der "Distributed Clock System Time".
EC_DCTIME_DATEDELTA_OFFSET	Anzahl der vergangenen Tage seit dem Jahr Null bis zum 1 Januar 2000
EC_DCTIME_TICKSPERMSEC	Anzahl der "Distributed Clock System Time"-Nanosekunden pro Millisekunde
EC_DCTIME_TICKSPERSEC	Anzahl der "Distributed Clock System Time"-Nanosekunden pro Sekunde
EC_DCTIME_TICKSPERDAY	Anzahl der "Distributed Clock System Time"-Nanosekunden pro Tag

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 Build > 1310 oder höher	PC oder CX (x86, ARM)	TcEtherCAT.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib, TcUtilities.Lib werden automatisch eingebunden)

12.2 T_DCTIME32 2

32 Bit TwinCAT *Distributed Clock System Time* Zeitformat. Die kleinste Einheit ist eine Nanosekunde.

Anmerkung:

Diese 32 Bit *DC System Time* wird aus der vollständigen absoluten 64 Bit *DC System Time* (T_DCTIME [▶ 87]) gebildet, indem nur die niederwertigsten 32 Bit verwendet werden. Hierdurch verliert man die Eigenschaft einer absoluten eindeutigen Zeit und geht deshalb von der Annahme aus, dass diese 32 Bit Zeit nur im zeitlichen Nahbereich von ± 2.147 Sekunden um die aktuelle Systemzeit verwendet wird, da sie nur

hier eindeutig ist. Hiervon kann in vielen Anwendungsfällen ausgegangen werden.
Verletzt man diese Annahme, dann kann es zu Fehlern bei der Interpretation und der Weiterverarbeitung dieser Zeit kommen.

```
TYPE T_DCTIME32 : UDINT;
END_TYPE
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.11.0 Build >=1524 oder höher	PC oder CX (x86, ARM)	TcEtherCAT.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib, TcUtilities.Lib werden automatisch eingebunden)

13 Konstanten

13.1 Globale Konstanten

VAR_GLOBAL CONSTANT

```
EC_AMSPORT_MASTER               :UINT   :=16#FFFF;
EC_MAX_SLAVES           :UINT   :=16#FFFF;

(* Device states *)
EC_DEVICE_STATE_MASK         :BYTE   :=16#0F;
EC_DEVICE_STATE_INIT        :BYTE   :=16#01;
EC_DEVICE_STATE_PREOP       :BYTE   :=16#02;
EC_DEVICE_STATE_BOOTSTRAP           :BYTE   :=16#03;
EC_DEVICE_STATE_SAFEOP      :BYTE   :=16#04;
EC_DEVICE_STATE_OP              :BYTE   :=16#08;
EC_DEVICE_STATE_ERROR       :BYTE   :=16#10;
EC_DEVICE_STATE_INVALID_VPRS    :BYTE   :=16#20;
EC_DEVICE_STATE_INITCMD_ERROR   :BYTE   :=16#40;

(* Link states *)
EC_LINK_STATE_OK            :BYTE   :=16#00;
EC_LINK_STATE_NOT_PRESENT           :BYTE   :=16#01;
EC_LINK_STATE_LINK_WITHOUT_COMM :BYTE   :=16#02;
EC_LINK_STATE_MISSING_LINK          :BYTE   :=16#04;
EC_LINK_STATE_ADDITIONAL_LINK   :BYTE   :=16#08;
EC_LINK_STATE_PORT_A        :BYTE   :=16#10;
EC_LINK_STATE_PORT_B        :BYTE   :=16#20;
EC_LINK_STATE_PORT_C        :BYTE   :=16#40;
EC_LINK_STATE_PORT_D        :BYTE   :=16#80;


(* Device/Link state IG/IO *)
EC_ADS_IGRP_MASTER_STATEMACHINE  :UDINT :=16#00000003;
EC_ADS_IOFFS_MASTER_CURSTATE    :UDINT :=16#00000100;
EC_ADS_IOFFS_MASTER_REQSTATE    :UDINT :=16#00000101;
EC_ADS_IOFFS_MASTER_INTERNALSTATE       :UDINT :=16#00000102;

EC_ADS_IGRP_MASTER_COUNT_SLAVE  :UDINT :=16#00000006;
EC_ADS_IOFFS_MASTER_COUNT_SLAVE     :UDINT :=16#00000000;
EC_ADS_IOFFS_MASTER_COUNT_PORT  :UDINT :=16#00000001;
EC_ADS_IOFFS_MASTER_COUNT_ROUTER    :UDINT :=16#00000002;

EC_ADS_IGRP_MASTER_SLAVE_ADDRESSES      :UDINT :=16#00000007;
EC_ADS_IGRP_SLAVE_STATEMACHINE  :UDINT :=16#00000009;
EC_ADS_IGRP_MASTER_SLAVE_IDENTITY       :UDINT :=16#00000011;
EC_ADS_IGRP_MASTER_SLAVE_CRC    :UDINT :=16#00000012;

(* SoE IG/IO *)
EC_ADS_IGRP_ECAT_SOE         :UDINT :=16#0000F420;
EC_ADS_IGRP_ECAT_SOE_LASTERROR  :UDINT :=16#0000F421;

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;

(* FoE IG/IO *)
EC_ADS_IGRP_FOE_FOPENREAD           :UDINT :=16#0000F401;
EC_ADS_IGRP_FOE_FOPENWRITE          :UDINT :=16#0000F402;
EC_ADS_IGRP_FOE_FCLOSE      :UDINT :=16#0000F403;
EC_ADS_IGRP_FOE_FREAD       :UDINT :=16#0000F404;
EC_ADS_IGRP_FOE_FWRITE      :UDINT :=16#0000F405;
EC_ADS_IGRP_FOE_PROGRESSINFO    :UDINT :=16#0000F406;
EC_ADS_IGRP_FOE_LASTERROR           :UDINT :=16#0000F407;

(* CoE IG/IO *)
EC_ADS_IGRP_CANOPEN_SDO         :UDINT :=16#0000F302;
EC_ADS_IGRP_CANOPEN_SDO_LASTERROR       :UDINT :=16#0000F303;
```

```
EC_DCTIME_DELTA_OFFSET  : T_ULARGE_INTEGER := ( dwHighPart := 16#01BF53EB, dwLowPart := 16#256D4000
); (* Number of 100ns ticks between 1.1.1601 and 1.1.2000 *)
EC_DCTIME_DATEDELTA_OFFSET       : DWORD := 730120; (* Number of past days since year zero until 1 J
anuary 2000 *)

EC_DCTIME_TICKSPERMSEC  : T_ULARGE_INTEGER := ( dwHighPart := 16#00000000, dwLowPart := 16#000F4240)
;(* Number of nanosecond ticks per millisecond *)
EC_DCTIME_TICKSPERSEC    : T_ULARGE_INTEGER := ( dwHighPart := 16#00000000, dwLowPart := 16#3B9ACA00
);(* Number of nanosecond ticks per second *)
EC_DCTIME_TICKSPERDAY    : T_ULARGE_INTEGER := ( dwHighPart := 16#00004E94, dwLowPart := 16#914F0000
);(* Number of nanosecond ticks per day *)
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 oder höher	PC or CX (x86)	TcEtherCAT.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib, TcUtilities.Lib werden automatisch eingebunden)
TwinCAT v2.10.0 Build >= 1301 oder höher	CX (ARM)	

13.2 EhterCAT mailbox protocol error codes

VAR_GLOBAL CONSTANT

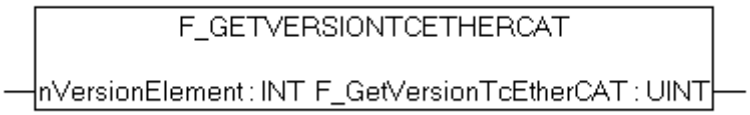
```
(* FoE mailbox protocol error codes *)
EC_FOE_PROTERR_NOTDEFINED        : UDINT := 0;
EC_FOE_PROTERR_NOTFOUND          : UDINT := 1;
EC_FOE_PROTERR_ACCESS       : UDINT := 2;
EC_FOE_PROTERR_DISKFULL          : UDINT := 3;
EC_FOE_PROTERR_ILLEAGAL          : UDINT := 4;
EC_FOE_PROTERR_PACKENO      : UDINT := 5;
EC_FOE_PROTERR_EXISTS       : UDINT := 6;
EC_FOE_PROTERR_NOUSER       : UDINT := 7;
EC_FOE_PROTERR_BOOTSTRAPONLY    : UDINT := 8;
EC_FOE_PROTERR_NOTINBOOTSTRAP   : UDINT := 9;
EC_FOE_PROTERR_INVALIDPASSWORD  : UDINT := 10;

(* CoE mailbox protocol error codes *)
EC_COE_PROTERR_TOGGLE        : UDINT := 16#05030000; (* Toggle bit not alternated. *)
EC_COE_PROTERR_TIMEOUT      : UDINT := 16#05040000; (* SDO protocol timed out. *)
EC_COE_PROTERR_CCS_SCS      : UDINT := 16#05040001; (* Client/
server command specifier not valid or unknown. *)
EC_COE_PROTERR_BLK_SIZE          : UDINT := 16#05040002; (* Invalid block size (block mode only). *)
EC_COE_PROTERR_SEQNO        : UDINT := 16#05040003; (* Invalid sequence number (block mode only). *)
EC_COE_PROTERR_CRC          : UDINT := 16#05040004; (* CRC error (block mode only). *)
EC_COE_PROTERR_MEMORY       : UDINT := 16#05040005; (* Out of memory. *)
EC_COE_PROTERR_ACCESS       : UDINT := 16#06010000; (* Unsupported access to an object. *)
EC_COE_PROTERR_WRITEONLY         : UDINT := 16#06010001; (* Attempt to read a write only object. *)
EC_COE_PROTERR_READONLY          : UDINT := 16#06010002; (* Attempt to write a read only object. *)
EC_COE_PROTERR_INDEX        : UDINT := 16#06020000; (* Object does not exist in the object dictionar
y. *)
EC_COE_PROTERR_PDO_MAP      : UDINT := 16#06040041; (* Object cannot be mapped to the PDO. *)
EC_COE_PROTERR_PDO_LEN      : UDINT := 16#06040042; (* The number and length of the objects to be ma
pped would exceed PDO length. *)
EC_COE_PROTERR_P_INCOMP          : UDINT := 16#06040043; (* General parameter incompatibility reason.
 *)
EC_COE_PROTERR_I_INCOMP          : UDINT := 16#06040047; (* General internal incompatibility in the d
evice. *)
EC_COE_PROTERR_HARDWARE          : UDINT := 16#06060000; (* Access failed due to an hardware error. *
)
EC_COE_PROTERR_DATA_SIZE         : UDINT := 16#06070010; (* Data type does not match, length of servi
ce parameter does not match *)
EC_COE_PROTERR_DATA_SIZE1           : UDINT := 16#06070012; (* Data type does not match, length of s
ervice parameter too high *)
EC_COE_PROTERR_DATA_SIZE2           : UDINT := 16#06070013; (* Data type does not match, length of s
ervice parameter too low *)
EC_COE_PROTERR_OFFSET       : UDINT := 16#06090011; (* Sub-index does not exist. *)
EC_COE_PROTERR_DATA_RANGE           : UDINT := 16#06090030; (* Value range of parameter exceeded (on
ly for write access). *)
EC_COE_PROTERR_DATA_RANGE1      : UDINT := 16#06090031; (* Value of parameter written too high. *)
EC_COE_PROTERR_DATA_RANGE2      : UDINT := 16#06090032; (* Value of parameter written too low. *)
EC_COE_PROTERR_MINMAX       : UDINT := 16#06090036; (* Maximum value is less than minimum value. *)
EC_COE_PROTERR_GENERAL      : UDINT := 16#08000000; (* general error *)
EC_COE_PROTERR_TRANSFER          : UDINT := 16#08000020; (* Data cannot be transferred or stored to t
he application. *)
```

```
EC_COE_PROTERR_TRANSFER1        : UDINT := 16#08000021; (* Data cannot be transferred or stored to t
he application because of local control. *)
EC_COE_PROTERR_TRANSFER2        : UDINT := 16#08000022; (* Data cannot be transferred or stored to t
he application because of the present device state. *)
EC_COE_PROTERR_DICTIONARY           : UDINT := 16#08000023; (* Object dictionary dynamic generation
fails or no object dictionary is present (e.g. object dictionary is generated from file and generati
on fails because of an file error). *)
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 Build > 1307 oder höher	PC or CX (x86, ARM)	TcEtherCAT.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib, TcUtilities.Lib werden automatisch eingebunden)

14 F_GetVersionTcEtherCAT



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

FUNCTION F_GetVersionTcEtherCAT : 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 oder höher	PC or CX (x86)	TcEtherCAT.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib, TcUtilities.Lib werden automatisch eingebunden)
TwinCAT v2.10.0 Build >= 1301 oder höher	CX (ARM)	

15 F_CheckVendorId

Die Funktion F_CheckVendorId liefert ein TRUE, falls die VendorID Beckhoff ist, ansonsten ein FALSE.

VAR_INPUT

```
VAR_INPUT
    stSlaveIdentity : ST_EcSlaveIdentity;
END_VAR
```

stSlaveIdentity: Die SlaveIdentity, wie sie mit dem FB_EcGetSlaveIdentity [▶ 47] eingelesen werden kann.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 oder höher	PC or CX (x86)	TcEtherCAT.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib, TcUtilities.Lib werden automatisch eingebunden)
TwinCAT v2.10.0 Build >= 1301 oder höher	CX (ARM)	

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