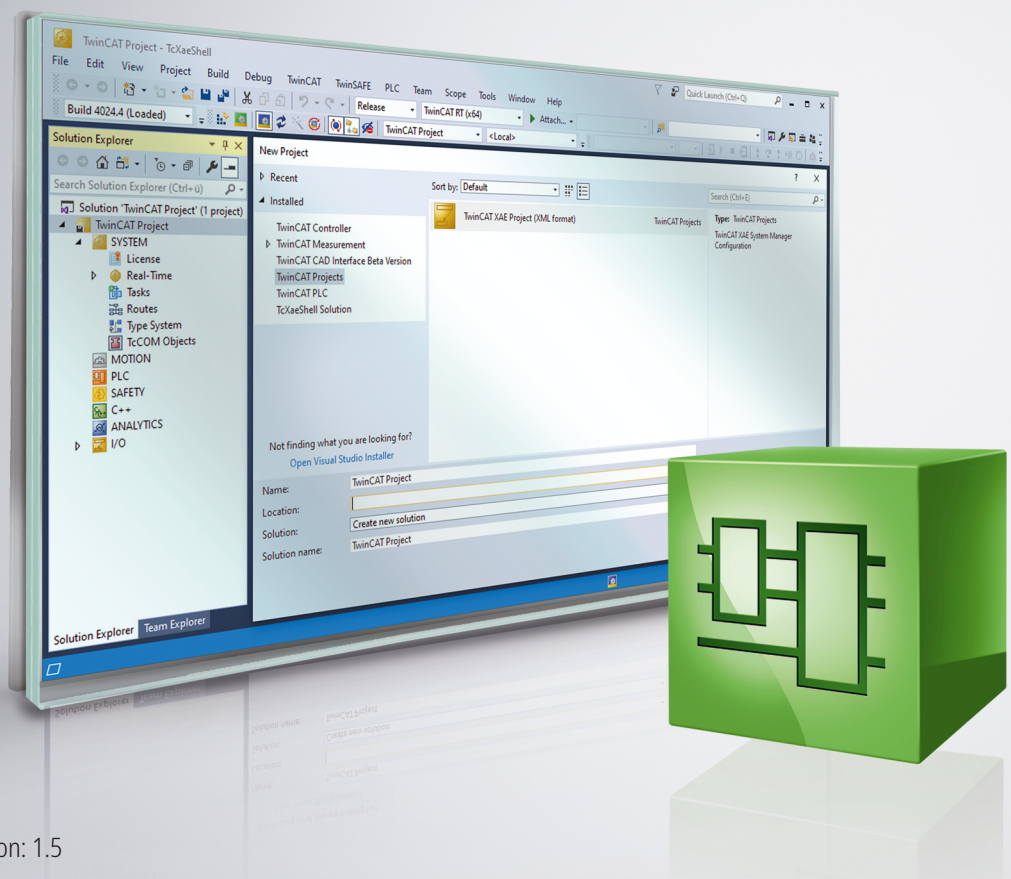
BECKHOFF New Automation Technology

Handbuch | DE

TE1000

TwinCAT 3 | PLC-Bibliothek: Tc2_IoFunctions



26.04.2021 | Version: 1.5

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, für jede Installation und Inbetriebnahme die zu dem betreffenden Zeitpunkt veröffentliche 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 weiter entwickelt.
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®, 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®

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

Copyright

© Beckhoff Automation GmbH & Co. KG, Deutschland.
Weitergabe sowie Vervielfältigung dieses Dokuments, Verwertung und Mitteilung seines Inhalts sind verboten, soweit nicht ausdrücklich gestattet.
Zuwiderhandlungen verpflichten zu Schadenersatz. Alle Rechte für den Fall der Patent-, Gebrauchsmuster- oder Geschmacksmustereintragung vorbehalten.

1.2 Sicherheitshinweise

Sicherheitsbestimmungen

Beachten Sie die folgenden Sicherheitshinweise und Erklärungen!
Produktspezifische Sicherheitshinweise finden Sie auf den folgenden Seiten oder in den Bereichen Montage, Verdrahtung, Inbetriebnahme usw.

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.

Erklärung der Symbole

In der vorliegenden Dokumentation werden die folgenden Symbole mit einem nebenstehenden Sicherheitshinweis oder Hinweistext verwendet. Die Sicherheitshinweise sind aufmerksam zu lesen und unbedingt zu befolgen!

⚠ GEFAHR

Akute Verletzungsgefahr!

Wenn der Sicherheitshinweis neben diesem Symbol nicht beachtet wird, besteht unmittelbare Gefahr für Leben und Gesundheit von Personen!

⚠ WARNUNG

Verletzungsgefahr!

Wenn der Sicherheitshinweis neben diesem Symbol nicht beachtet wird, besteht Gefahr für Leben und Gesundheit von Personen!

⚠ VORSICHT

Schädigung von Personen!

Wenn der Sicherheitshinweis neben diesem Symbol nicht beachtet wird, können Personen geschädigt werden!

HINWEIS

Schädigung von Umwelt oder Geräten

Wenn der Hinweis neben diesem Symbol nicht beachtet wird, können Umwelt oder Geräte geschädigt werden.

Tipp oder Fingerzeig

Dieses Symbol kennzeichnet Informationen, die zum besseren Verständnis beitragen.

2 Übersicht

Die IO-Functions-Bibliothek beinhaltet Funktionsbausteine, mit denen Dienste/Funktionen auf den IO-Geräten (Feldbus-Master oder Slaves) ausgeführt werden können.

Allgemeine Gerätefunktionen

Name	Beschreibung
IOF_DeviceReset [▶ 13]	Reset eines IO-Gerätes
IOF_GetBoxAddrByName [▶ 14]	Die Feldbusadresse der Box über die Geräte-Id und die Box-Bezeichnung ermitteln
IOF_GetBoxAddrByNameEx [▶ 15]	Die Feldbusadresse der Box über die Geräte-Bezeichnung und die Box-Bezeichnung ermitteln
IOF_GetBoxCount [▶ 16]	Die Anzahl der Boxen lesen
IOF_GetBoxNameByAddr [▶ 17]	Die Box-Bezeichnung anhand der Feldbusadresse der Box und der Geräte-Id lesen
IOF_GetBoxNetId [▶ 18]	Die AmsNetId einer Box anhand der Feldbusadresse der Box und der Geräte-Id lesen
IOF_GetDeviceCount [▶ 19]	Die Anzahl der IO-Geräte lesen
IOF_GetDeviceIDByName [▶ 19]	Die Geräte-Id anhand der Geräte-Bezeichnung ermitteln
IOF_GetDeviceIDs [▶ 20]	Alle Geräte-Ids lesen
IOF_GetDeviceName [▶ 23]	Die Geräte-Bezeichnung anhand der Geräte-Id lesen
IOF_GetDeviceNetId [▶ 24]	Die AmsNetId anhand der Geräte-Id lesen
IOF_GetDeviceType [▶ 25]	Den Geräte-Typ anhand der Geräte-Id lesen
IOF_GetDeviceInfoByName [▶ 22]	Die Geräte-Id und die AmsNetId anhand der Geräte-Bezeichnung ermitteln

Feldbusspezifische und gerätespezifische Funktionen

CANopen

Name	Beschreibung
IOF_CAN_Layer2Command [▶ 62]	Ein Layer 2 Kommando ausführen

Beckhoff Lightbus

Name	Beschreibung
IOF_LB_BreakLocationTest [▶ 44]	Bruchstellen-Test des Lichtwellenleiterrings
IOF_LB_ParityCheck [▶ 45]	Parity-Zähler lesen
IOF_LB_ParityCheckWithReset [▶ 47]	Parity-Zähler lesen und zurücksetzen

SERCOS

Name	Beschreibung
IOF_SER_GetPhase [▶ 91]	Die aktuelle Phase lesen
IOF_SER_ResetErr [▶ 93]	Reset des Fehlerpuffers
IOF_SER_SaveFlash [▶ 92]	Parameter im Flash speichern
IOF_SER_SetPhase [▶ 94]	Die aktuelle Phase setzen
IOF_SER_IDN_Read [▶ 95]	Sercos-Drive-Parameter lesen
IOF_SER_IDN_Write [▶ 97]	Sercos-Drive-Parameter schreiben
IOF_SER_DRIVE_Backup [▶ 98]	Backup und Restore der Sercos-Drive-Parameter in/aus einer Datei
IOF_SER_DRIVE_BackupEx [▶ 100]	Backup und Restore der Sercos-Drive-Parameter in/aus einer Datei (erweiterte Funktionalität)
IOF_SER_DRIVE_Reset [▶ 102]	Drive-Reset eines Sercos-Drives per Kommando auf Parameter S-0-0099 (IDN99)

Profibus DPV1 (Sinamics

Name	Beschreibung
F_CreateDpv1ReadReqPkg [▶ 72]	DPV1 Telegramm für Parameterlesen erzeugen
FB_Dpv1Read [▶ 75]	DPV1 Telegramm für Parameterlesen senden
F_SplitDpv1ReadResPkg [▶ 73]	DPV1 Antwort-Telegramm für Parameterlesen auswerten
F_CreateDpv1WriteReqPkg [▶ 72]	DPV1 Telegramm für Parameterschreiben erzeugen
FB_Dpv1Write [▶ 78]	DPV1 Telegramm für Parameterschreiben senden
F_SplitDpv1WriteResPkg [▶ 74]	DPV1 Antwort-Telegramm für Parameterschreiben auswerten

Profinet DPV1 (Sinamics)

Name	Beschreibung
F_CreateDpv1ReadReqPkgPNET [▶ 80]	DPV1 Telegramm für Parameterlesen erzeugen
FB_Dpv1ReadPNET [▶ 83]	DPV1 Telegramm für Parameterlesen senden
F_SplitDpv1ReadResPkgPNET [▶ 82]	DPV1 Antwort-Telegramm für Parameterlesen auswerten
F_CreateDpv1WriteReqPkgPNET [▶ 81]	DPV1 Telegramm für Parameterschreiben erzeugen
FB_Dpv1WritePNET [▶ 85]	DPV1 Telegramm für Parameterschreiben senden
F_SplitDpv1WriteResPkgPNET [▶ 82]	DPV1 Antwort-Telegramm für Parameterschreiben auswerten

NOV/DP-RAM

Name	Beschreibung
FB_NovRamReadWrite [▶ 63]	Daten in das NOV-RAM schreiben oder aus dem NOV-RAM lesen
FB_NovRamReadWriteEx [▶ 66]	Daten in das NOV-RAM schreiben oder aus dem NOV-RAM lesen. Überprüft ob ein spezielle Zugriffsart auf den Speicher notwendig ist und kopiert dementsprechend die Daten auf die korrekte Art (z-B. beim Zugriff auf das CX_9000 NOV-RAM).
FB_GetDPRAMInfo [▶ 68]	Den Adresspointer und die konfigurierte Größe vom NOV/DP-RAM lesen
FB_GetDPRAMInfoEx [▶ 70]	Den Adresspointer und die konfigurierte Größe vom NOV/DP-RAM lesen (Erweiterung)

AX200x Profibus

Funktionsbausteine für den Zugriff auf den AX200X über den Profibus: Übersicht [▶ 36].

ASI Master Terminal

Funktionsbausteine für den Zugriff auf eine ASI-Masterklemme: Übersicht [▶ 26].

Beckhoff USV (unter Windows USV Service)

Name	Beschreibung
FB_GetUPSStatus [▶ 48]	Den Status der USV aus der SPS lesen.

Drittherstellergeräte

INTERBUS Phoenix IBS SC/I-T Funktionen

Phoenix IBS SC/I-T Funktionen: Übersicht [▶ 106].

Name	Beschreibung
SCIT_ActivateConfiguration [▶ 107]	Führt den Befehl **Activate_Configuration** aus
SCIT_DeactivateConfiguration [▶ 108]	Führt den Befehl **Deactivate_Configuration** aus
SCIT_StartDataTransfer [▶ 109]	Führt den Befehl **Start_Data_Transfer** aus
SCIT_StopDataTransfer [▶ 110]	Führt den Befehl **Stop_Data_Transfer** aus
SCIT_AlarmStop [▶ 111]	Führt den Befehl **Alarm_Stop** aus
SCIT_ControlActiveConfiguration [▶ 112]	Dient zur Beeinflussung der aktiven Konfiguration der Interbus-Teilnehmer. Dieses Kommando kann sowohl im Zustand *PAR_READY* als auch im Zustand *ACTIVE* und *RUN* ausgeführt werden. Hierüber können einzelne, abhängige und gruppierte Teilnehmer aktiviert und deaktiviert werden.
SCIT_GetErrorInfo [▶ 113]	Liefert Fehlerart und Fehlerort eines Interbus-Teilnehmers nach einem Busfehler
SCIT_ConfDevErrAll [▶ 114]	Peripheriestörungen aller Geräte quittieren

ads-tec

Name	Beschreibung
FB_ReadAdsTecSysData [▶ 115]	Liest die Systemdaten/Diagnosedaten

RAID_Controller

Following function blocks are available for RAID controller services.

Name	Description
FB_RAIDFindCntlr [▶ 88]	liefert die RAID-Controller Anzahl und die entsprechenden RAID-Controller IDs zurück
FB_RAIDGetInfo [▶ 89]	liefert ein RAID Info, das die Anzahl der RAID-Controller-Sets enthält und die maximale Anzahl der RAID Antriebe pro Set.
FB_RAIDGetStatus [▶ 90]	liefert den RAID-Set-Index, den RAID Typ, den RAID Status, die Anzahl der RAID Antriebe und den Status der RAID Antriebe zurück.

3 Funktionsbausteine

3.1 Allgemeine IO FBs

3.1.1 IOF_DeviceReset

Der Funktionsbaustein IOF_DeviceReset führt ein Reset eines IO-Gerätes (z.B. einer Feldbuskarte) durch. Die Funktion entspricht der Online-Reset-Funktion aus dem TwinCAT->I/O->Geräte->Gerät xyz-Kontextmenü.

VAR_INPUT

```
VAR_INPUT
    NETID     : T_AmsNetId;
    DEVICEID  : UDINT;
    RESET     : BOOL;
    TMOUT     : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

NETID: Hier kann die Netzwerkadresse des TwinCAT-Rechners angegeben werden, auf dem die Funktion ausgeführt werden soll (Typ: T_AmsNetId). Für den lokalen Rechner kann auch ein Leerstring angegeben werden.

DEVICEID: Über die DeviceId (Geräte-ID) wird das IO-Gerät spezifiziert, auf dem die Funktion ausgeführt werden soll. Die Geräte-IDs werden während der Hardware-Konfiguration von TwinCAT-System festgelegt.

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

TMOUT: Gibt die Timeout-Zeit an, die bei der Ausführung des Kommandos nicht überschritten werden darf.

VAR_OUTPUT

```
VAR_OUTPUT
    BUSY    : BOOL;
    ERR     : BOOL;
    ERRID   : UDINT;
END_VAR
```

BUSY: Bei der Aktivierung des Funktionsbausteins wird dieser Ausgang gesetzt und bleibt gesetzt, bis eine Rückmeldung erfolgt.

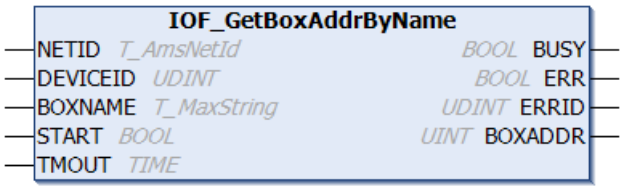
ERR: Sollte ein Fehler bei der Übertragung des Kommandos erfolgen, dann wird dieser Ausgang gesetzt, nachdem der BUSY-Ausgang zurückgesetzt wurde.

ERRID: Liefert bei einem gesetzten ERR-Ausgang die ADS-Fehlernummer [▶ 154].

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO Hardware	Einzubindende SPS-Bibliotheken (Kategoriegruppe)
TwinCAT v3.1.0	PC oder CX (x86, x64, ARM)	Alle IO-Geräte	Tc2_IoFunctions (IO)

3.1.2 IOF_GetBoxAddrByName



Der Funktionsbaustein IOF_GetBoxAddrByName ermittelt die Feldbusadresse einer Box (Box = Slave, Modul, Station) anhand der Box-Bezeichnung und der Geräte-ID. Ist eine Feldbusadresse nicht vorhanden, dann liefert der Funktionsbaustein eine logische oder physikalische Adresse zurück (bei Beckhoff Lightbus ist es z.B. die physikalische Boxnummer im Lichtwellenleiter-Ring und bei Profibus die Stationsadresse). Die Box-Bezeichnung wird als ein String an den Funktionsbaustein übergeben und kann während der Konfiguration in TwinCAT System vom Benutzer festgelegt werden.

VAR_INPUT

```
VAR_INPUT
    NETID     : T_AmsNetId;
    DEVICEID  : UDINT;
    BOXNAME   : T_MaxString;
    START     : BOOL;
    TMOUT     : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

NETID: Hier kann die Netzwerkadresse des TwinCAT-Rechners angegeben werden, auf dem die Funktion ausgeführt werden soll (Typ: T_AmsNetID). Für den lokalen Rechner kann auch ein Leerstring angegeben werden.

DEVICEID: Über die DeviceId (Geräte-ID) wird das Gerät spezifiziert, auf dem die Funktion ausgeführt werden soll. Die Geräte-Ids werden während der Hardware-Konfiguration von TwinCAT-System festgelegt.

BOXNAME: Die Box-Bezeichnung als String (Typ: T_MaxString).

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

TMOUT: Gibt die Timeout-Zeit an, die bei der Ausführung des Kommandos nicht überschritten werden darf.

VAR_OUTPUT

```
VAR_OUTPUT
    BUSY    : BOOL;
    ERR     : BOOL;
    ERRID   : UDINT;
    BOXADDR : UINT;
END_VAR
```

BUSY: Bei der Aktivierung des Funktionsbausteins wird dieser Ausgang gesetzt und bleibt gesetzt, bis eine Rückmeldung erfolgt.

ERR: Sollte ein Fehler bei der Übertragung des Kommandos erfolgen, dann wird dieser Ausgang gesetzt, nachdem der BUSY-Ausgang zurückgesetzt wurde.

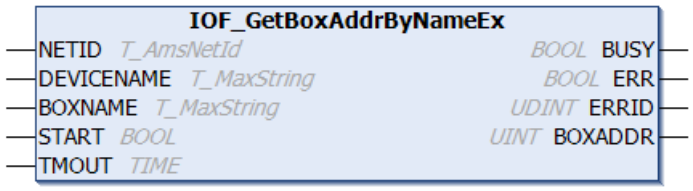
ERRID: Liefert bei einem gesetzten ERR-Ausgang die ADS-Fehlernummer [▶ 154].

BOXADDR: Die Feldbusadresse der Box.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO Hardware	Einzubindende SPS-Bibliotheken (Kategoriegruppe)
TwinCAT v3.1.0	PC oder CX (x86, x64, ARM)	Alle IO-Geräte	Tc2_IoFunctions (IO)

3.1.3 IOF_GetBoxAddrByNameEx



Der Funktionsbaustein IOF_GetBoxAddrByNameEx ermittelt die Feldbusadresse einer Box (Box = Slave, Modul, Station) anhand der Box-Bezeichnung und der Geräte-Bezeichnung. Ist eine Feldbusadresse nicht vorhanden, dann liefert der Funktionsbaustein eine logische oder physikalische Adresse zurück (bei Beckhoff Lightbus ist es z.B. die physikalische Boxnummer im Lichtwellenleiter-Ring und bei Profibus die Stationsadresse). Die Box-Bezeichnung und Geräte-Bezeichnung werden als Strings an den Funktionsbaustein übergeben und können während der Konfiguration in TwinCAT System vom Benutzer festgelegt werden.

VAR_INPUT

```
VAR_INPUT
    NETID      : T_AmsNetId;
    DEVICENAME : T_MaxString;
    BOXNAME    : T_MaxString;
    START      : BOOL;
    TMOUT      : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

NETID: Hier kann die Netzwerkadresse des TwinCAT-Rechners angegeben werden, auf dem die Funktion ausgeführt werden soll (Typ: T_AmsNetID). Für den lokalen Rechner kann auch ein Leerstring angegeben werden.

DEVICENAME: Die Geräte-Bezeichnung eines IO-Gerätes als String (Typ: T_MaxString).

BOXNAME: Die Box-Bezeichnung als String (Typ: T_MaxString).

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

TMOUT: Gibt die Timeout-Zeit an, die bei der Ausführung des Kommandos nicht überschritten werden darf.

VAR_OUTPUT

```
VAR_OUTPUT
    BUSY    : BOOL;
    ERR     : BOOL;
    ERRID   : UDINT;
    BOXADDR : UINT;
END_VAR
```

BUSY: Bei der Aktivierung des Funktionsbausteins wird dieser Ausgang gesetzt und bleibt gesetzt, bis eine Rückmeldung erfolgt.

ERR: Sollte ein Fehler bei der Übertragung des Kommandos erfolgen, dann wird dieser Ausgang gesetzt, nachdem der BUSY-Ausgang zurückgesetzt wurde.

ERRID: Liefert bei einem gesetzten ERR-Ausgang die ADS-Fehlernummer [▶ 154].

BOXADDR: Die Feldbusadresse oder logische Adresse der Box.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO Hardware	Einzubindende SPS-Bibliotheken (Kategoriegruppe)
TwinCAT v3.1.0	PC oder CX (x86, x64, ARM)	Alle IO-Geräte	Tc2_IoFunctions (IO)

3.1.4 IOF_GetBoxCount

Der Funktionsbaustein IOF_GetBoxCount liest die Anzahl der konfigurierten und aktiven Boxen (Box = Slave, Modul, Station) eines IO-Gerätes.

VAR_INPUT

```
VAR_INPUT
    NETID     : T_AmsNetId;
    DEVICEID  : UDINT;
    START     : BOOL;
    TMOUT     : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

NETID: Hier kann die Netzwerkadresse des TwinCAT-Rechners angegeben werden, auf dem die Funktion ausgeführt werden soll (Typ: T_AmsNetID). Für den lokalen Rechner kann auch ein Leerstring angegeben werden.

DEVICEID: Über die DeviceId (Geräte-ID) wird das Gerät spezifiziert, auf dem die Funktion ausgeführt werden soll. Die Geräte-Ids werden während der Hardware-Konfiguration von TwinCAT-System festgelegt.

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

TMOUT: Gibt die Timeout-Zeit an, die bei der Ausführung des Kommandos nicht überschritten werden darf.

VAR_OUTPUT

```
VAR_OUTPUT
    BUSY     : BOOL;
    ERR      : BOOL;
    ERRID    : UDINT;
    BOXCOUNT : UDINT;
END_VAR
```

BUSY: Bei der Aktivierung des Funktionsbausteins wird dieser Ausgang gesetzt und bleibt gesetzt, bis eine Rückmeldung erfolgt.

ERR: Sollte ein Fehler bei der Übertragung des Kommandos erfolgen, dann wird dieser Ausgang gesetzt, nachdem der BUSY-Ausgang zurückgesetzt wurde.

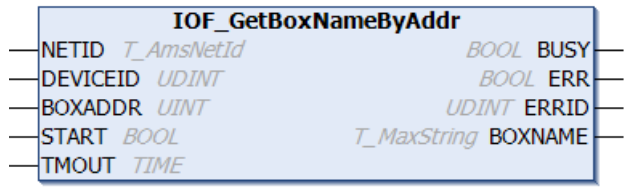
ERRID: Liefert bei einem gesetzten ERR-Ausgang die ADS-Fehlernummer [▶ 154].

BOXCOUNT: Anzahl der Boxen.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO Hardware	Einzubindende SPS-Bibliotheken (Kategoriegruppe)
TwinCAT v3.1.0	PC oder CX (x86, x64, ARM)	Alle IO-Geräte	Tc2_IoFunctions (IO)

3.1.5 IOF_GetBoxNameByAddr



Der Funktionsbaustein IOF_GetBoxNameByAddr ermittelt die Box-Bezeichnung anhand der Geräte-Id und der Feldbusadresse einer Box (Box = Slave, Modul, Station) . Ist eine Feldbusadresse nicht vorhanden, dann kann als Feldbusadresse an den Funktionsbaustein eine logische oder physikalische Adresse übergeben werden (bei Beckhoff Lightbus ist es z.B. die physikalische Boxnummer im Lichtwellenleiter-Ring). Beim Erfolg liefert der Funktionsbaustein die im TwinCAT konfigurierte Box-Bezeichnung als String zurück.

VAR_INPUT

```
VAR_INPUT
    NETID     : T_AmsNetId;
    DEVICEID  : UDINT;
    BOXADDR   : UINT;
    START     : BOOL;
    TMOUT     : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

NETID: Hier kann die Netzwerkadresse des TwinCAT-Rechners angegeben werden, auf dem die Funktion ausgeführt werden soll (Typ: T_AmsNetID). Für den lokalen Rechner kann auch ein Leerstring angegeben werden.

DEVICEID: Über die DeviceId (Geräte-ID) wird das Gerät spezifiziert, auf dem die Funktion ausgeführt werden soll. Die Geräte-Ids werden während der Hardware-Konfiguration von TwinCAT-System festgelegt.

BOXADDR: Die Feldbusadresse der Box.

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

TMOUT: Gibt die Timeout-Zeit an, die bei der Ausführung des Kommandos nicht überschritten werden darf.

VAR_OUTPUT

```
VAR_OUTPUT
    BUSY    : BOOL;
    ERR     : BOOL;
    ERRID   : UDINT;
    BOXNAME : T_MaxString;
END_VAR
```

BUSY: Bei der Aktivierung des Funktionsbausteins wird dieser Ausgang gesetzt und bleibt gesetzt, bis eine Rückmeldung erfolgt.

ERR: Sollte ein Fehler bei der Übertragung des Kommandos erfolgen, dann wird dieser Ausgang gesetzt, nachdem der BUSY-Ausgang zurückgesetzt wurde.

ERRID: Liefert bei einem gesetzten ERR-Ausgang die ADS-Fehlernummer [▶ 154].

BOXNAME: Die Box-Bezeichnung als String (Typ: T_MaxString).

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO Hardware	Einzubindende SPS-Bibliotheken (Kategoriegruppe)
TwinCAT v3.1.0	PC oder CX (x86, x64, ARM)	Alle IO-Geräte	Tc2_IoFunctions (IO)

3.1.6 IOF_GetBoxNetId

Einigen Boxen (Slave-Module) kann eine AmsNetId während der Konfiguration im TwinCAT zugewiesen werden. Die AmsNetId kann dann benutzt werden, um auf der Box Firmware-Funktionen ausführen zu können. Der Funktionsbaustein IOF_GetBoxNetId ermittelt die TwinCAT Netzwerkadresse anhand der Geräte-ID des Masters und der Feldbusadresse oder logischen Adresse im Feldbus. Die Geräte-IDs werden während der Konfiguration vom TwinCAT-System festgelegt und können nicht vom Benutzer konfiguriert werden.

VAR_INPUT

```
VAR_INPUT
    NETID     : T_AmsNetId;
    DEVICEID  : UDINT;
    BOXARDDR  : WORD;
    START     : BOOL;
    TMOUT     : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

NETID: Hier kann die Netzwerkadresse des TwinCAT-Rechners angegeben werden, auf dem die Funktion ausgeführt werden soll (Typ: T_AmsNetID). Für den lokalen Rechner kann auch ein Leerstring angegeben werden.

DEVICEID: Die Geräte-ID des Masters.

BOXADDR: Die Feldbusadresse oder logische Adresse der Box (Slave-Modul) deren AmsNetId gelesen werden soll.

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

TMOUT: Gibt die Timeout-Zeit an, die bei der Ausführung des Kommandos nicht überschritten werden darf.

VAR_OUTPUT

```
VAR_OUTPUT
    BUSY     : BOOL;
    ERR      : BOOL;
    ERRID    : UDINT;
    BoxNetId : T_AmsNetId;
END_VAR
```

BUSY: Bei der Aktivierung des Funktionsbausteins wird dieser Ausgang gesetzt und bleibt gesetzt, bis eine Rückmeldung erfolgt.

ERR: Sollte ein Fehler bei der Übertragung des Kommandos erfolgen, dann wird dieser Ausgang gesetzt, nachdem der BUSY-Ausgang zurückgesetzt wurde.

ERRID: Liefert bei einem gesetzten ERR-Ausgang die ADS-Fehlernummer [▶ 154].

BoxNetId: Die TwinCAT Netzwerkadresse der Box als String (Typ: T_AmsNetID).

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO Hardware	Einzubindende SPS-Bibliotheken (Kategoriegruppe)
TwinCAT v3.1.0	PC oder CX (x86, x64, ARM)	Alle IO-Geräte	Tc2_IoFunctions (IO)

3.1.7 IOF_GetDeviceCount

Der Funktionsbaustein IOF_GetDeviceCount liest die Anzahl der konfigurierten und aktiven IO-Geräte.

VAR_INPUT

```
VAR_INPUT
    NETID     : T_AmsNetId;
    START     : BOOL;
    TMOUT     : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

NETID: Hier kann die Netzwerkadresse des TwinCAT-Rechners angegeben werden, auf dem die Funktion ausgeführt werden soll (Typ: T_AmsNetID). Für den lokalen Rechner kann auch ein Leerstring angegeben werden.

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

TMOUT: Gibt die Timeout-Zeit an, die bei der Ausführung des Kommandos nicht überschritten werden darf.

VAR_OUTPUT

```
VAR_OUTPUT
    BUSY        : BOOL;
    ERR         : BOOL;
    ERRID       : UDINT;
    DEVICECOUNT : UDINT;
END_VAR
```

BUSY: Bei der Aktivierung des Funktionsbausteins wird dieser Ausgang gesetzt und bleibt gesetzt, bis eine Rückmeldung erfolgt.

ERR: Sollte ein Fehler bei der Übertragung des Kommandos erfolgen, dann wird dieser Ausgang gesetzt, nachdem der BUSY-Ausgang zurückgesetzt wurde.

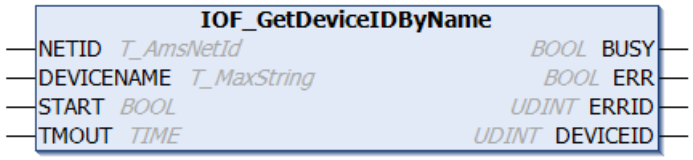
ERRID: Liefert bei einem gesetzten ERR-Ausgang die ADS-Fehlernummer [▶ 154].

DEVICECOUNT: Anzahl der IO-Geräte.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO Hardware	Einzubindende SPS-Bibliotheken (Kategoriegruppe)
TwinCAT v3.1.0	PC oder CX (x86, x64, ARM)	Alle IO-Geräte	Tc2_IoFunctions (IO)

3.1.8 IOF_GetDeviceIDByName



Der Funktionsbaustein IOF_GetDeviceIDByName ermittelt die Geräte-Id eines IO-Gerätes anhand der Geräte-Bezeichnung. Beim Erfolg liefert der Funktionsbaustein die vom TwinCAT-System während der Konfiguration festgelegte Geräte-ID. Die Geräte-Ids können vom Benutzer nicht konfiguriert werden.

VAR_INPUT

```
VAR_INPUT
    NETID      : T_AmsNetId;
    DEVICENAME : T_MaxString;
    START      : BOOL;
    TMOUT      : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

NETID: Hier kann die Netzwerkadresse des TwinCAT-Rechners angegeben werden, auf dem die Funktion ausgeführt werden soll (Typ: T_AmsNetID). Für den lokalen Rechner kann auch ein Leerstring angegeben werden.

DEVICENAME: Die Geräte-Bezeichnung eines IO-Gerätes (Typ: T_MaxString).

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

TMOUT: Gibt die Timeout-Zeit an, die bei der Ausführung des Kommandos nicht überschritten werden darf.

VAR_OUTPUT

```
VAR_OUTPUT
    BUSY     : BOOL;
    ERR      : BOOL;
    ERRID    : UDINT;
    DEVICEID : UDINT;
END_VAR
```

BUSY: Bei der Aktivierung des Funktionsbausteins wird dieser Ausgang gesetzt und bleibt gesetzt, bis eine Rückmeldung erfolgt.

ERR: Sollte ein Fehler bei der Übertragung des Kommandos erfolgen, dann wird dieser Ausgang gesetzt, nachdem der BUSY-Ausgang zurückgesetzt wurde.

ERRID: Liefert bei einem gesetzten ERR-Ausgang die ADS-Fehlernummer [▶ 154].

DEVICEID: Die Geräte-ID eines IO-Gerätes.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO Hardware	Einzubindende SPS-Bibliotheken (Kategoriegruppe)
TwinCAT v3.1.0	PC oder CX (x86, x64, ARM)	Alle IO-Geräte	Tc2_IoFunctions (IO)

3.1.9 IOF_GetDeviceIDs

Der Funktionsbaustein IOF_GetDeviceIDs liest die Geräte-IDs aller konfigurierten und aktiven IO-Geräte in einen Datenpuffer ein. Der Datenpuffer kann als ein Array von Word-Variablen definiert werden. Beim Erfolg liefert der Funktionsbaustein im ersten Daten-Word die gesamte Anzahl der vorhandenen Geräte-IDs und in den weiteren Daten-Worten die entsprechenden Geräte-IDs der einzelnen IO-Geräte. Die Geräte-IDs werden während der Konfiguration vom TwinCAT-System festgelegt und können nicht vom Benutzer konfiguriert werden.

VAR_INPUT

```
VAR_INPUT
    NETID    : T_AmsNetId;
    LEN      : UDINT;
    DESTADDR : PVOID;
    START    : BOOL;
    TMOUT    : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

NETID: Hier kann die Netzwerkadresse des TwinCAT-Rechners angegeben werden, auf dem die Funktion ausgeführt werden soll (Typ: T_AmsNetID). Für den lokalen Rechner kann auch ein Leerstring angegeben werden.

LEN: Länge des Datenpuffers in Bytes in den die Geräte-IDs eingelesen werden sollen.

DESTADDR: Adresse des Datenpuffers in den die Geräte-IDs eingelesen werden sollen.

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

TMOUT: Gibt die Timeout-Zeit an, die bei der Ausführung des Kommandos nicht überschritten werden darf.

VAR_OUTPUT

```
VAR_OUTPUT
    BUSY    : BOOL;
    ERR     : BOOL;
    ERRID   : UDINT;
END_VAR
```

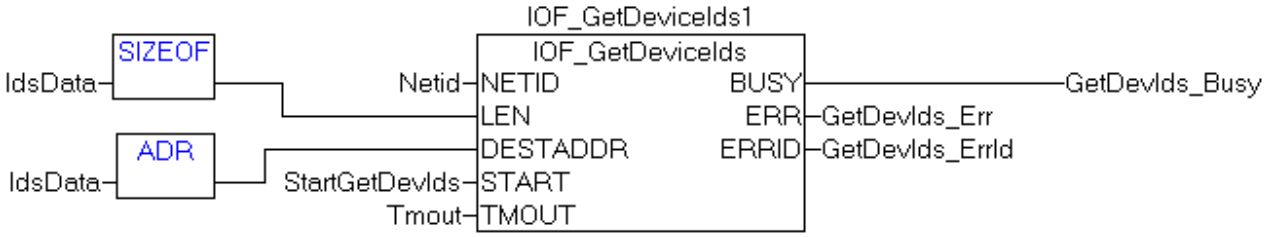
BUSY: Bei der Aktivierung des Funktionsbausteins wird dieser Ausgang gesetzt und bleibt gesetzt, bis eine Rückmeldung erfolgt.

ERR: Sollte ein Fehler bei der Übertragung des Kommandos erfolgen, dann wird dieser Ausgang gesetzt, nachdem der BUSY-Ausgang zurückgesetzt wurde.

ERRID: Liefert bei einem gesetzten ERR-Ausgang die ADS-Fehlernummer [▶ 154].

Beispiel:

```
PROGRAM MAIN
VAR
    IOF_GetDeviceIds1 : IOF_GetDeviceIDs;
    IdsData           : ARRAY[1..201] OF WORD;
    StartGetDevIds    : BOOL;
    GetDevIds_Busy    : BOOL;
    GetDevIds_Err     : BOOL;
    GetDevIds_ErrId   : UDINT;
END_VAR
```



Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO Hardware	Einzubindende SPS-Bibliotheken (Kategoriegruppe)
TwinCAT v3.1.0	PC oder CX (x86, x64, ARM)	Alle IO-Geräte	Tc2_IoFunctions (IO)

3.1.10 IOF_GetDeviceInfoByName

Der Funktionsbaustein IOF_GetDeviceInfoByName ermittelt die Geräte-ID eines IO-Gerätes und dessen TwinCAT Netzwerkadresse anhand der Geräte-Bezeichnung. Die Geräte-Ids können vom Benutzer nicht konfiguriert werden.

VAR_INPUT

```
VAR_INPUT
    NETID      : T_AmsNetId;
    DEVICENAME : T_MaxString;
    START      : BOOL;
    TMOUT      : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

NETID: Hier kann die AmsNetId des TwinCAT-Rechners angegeben werden, auf dem die Funktion ausgeführt werden soll (Typ: T_AmsNetID). Für den lokalen Rechner kann auch ein Leerstring angegeben werden.

DEVICENAME: Die Geräte-Bezeichnung eines IO-Gerätes (Typ: T_MaxString).

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

TMOUT: Gibt die Timeout-Zeit an, die bei der Ausführung des Kommandos nicht überschritten werden darf.

VAR_OUTPUT

```
VAR_OUTPUT
    BUSY        : BOOL;
    ERR         : BOOL;
    ERRID       : UDINT;
    DEVICEID    : UDINT;
    DEVICENETID : T_AmsNetId;
END_VAR
```

BUSY: Bei der Aktivierung des Funktionsbausteins wird dieser Ausgang gesetzt und bleibt gesetzt, bis eine Rückmeldung erfolgt.

ERR: Sollte ein Fehler bei der Übertragung des Kommandos erfolgen, dann wird dieser Ausgang gesetzt, nachdem der BUSY-Ausgang zurückgesetzt wurde.

ERRID: Liefert bei einem gesetzten ERR-Ausgang die ADS-Fehlernummer [▶ 154].

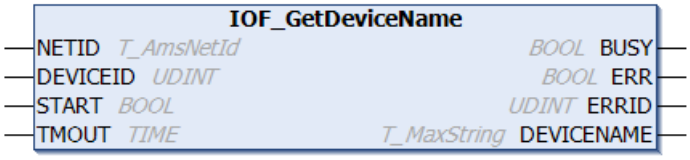
DEVICEID: Die Geräte-ID eines IO-Gerätes.

DEVICENETID: Die Netzwerkadresse eines IO-Gerätes (Typ: T_AmsNetID).

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO Hardware	Einzubindende SPS-Bibliotheken (Kategoriegruppe)
TwinCAT v3.1.0	PC oder CX (x86, x64, ARM)	Alle IO-Geräte	Tc2_IoFunctions (IO)

3.1.11 IOF_GetDeviceName



Der Funktionsbaustein IOF_GetDeviceName liest die Gerätebezeichnung eines IO-Gerätes. Die Gerätebezeichnung kann während der Konfiguration in TwinCAT-System vom Benutzer festgelegt werden. Beim Systemstart wird diese dann als String in den IO-Treiber gesendet und kann über die ADS-Kommandos gelesen werden. Über die Eingangsvariable DEVICEID wird das IO-Gerät spezifiziert, dessen Gerätebezeichnung gelesen werden soll.

VAR_INPUT

```
VAR_INPUT
    NETID     : T_AmsNetId;
    DEVICEID  : UDINT;
    START     : BOOL;
    TMOUT     : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

NETID: Hier kann die Netzwerkadresse des TwinCAT-Rechners angegeben werden, auf dem die Funktion ausgeführt werden soll (Typ: T_AmsNetID). Für den lokalen Rechner kann auch ein Leerstring angegeben werden.

DEVICEID: Über die Geräte-ID wird das Gerät spezifiziert, auf dem die Funktion ausgeführt werden soll. Die Geräte-IDs werden während der Hardware-Konfiguration von TwinCAT-System festgelegt.

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

TMOUT: Gibt die Timeout-Zeit an, die bei der Ausführung des Kommandos nicht überschritten werden darf.

VAR_OUTPUT

```
VAR_OUTPUT
    BUSY       : BOOL;
    ERR        : BOOL;
    ERRID      : UDINT;
    DEVICENAME : T_MaxString;
END_VAR
```

BUSY: Bei der Aktivierung des Funktionsbausteins wird dieser Ausgang gesetzt und bleibt gesetzt, bis eine Rückmeldung erfolgt.

ERR: Sollte ein Fehler bei der Übertragung des Kommandos erfolgen, dann wird dieser Ausgang gesetzt, nachdem der BUSY-Ausgang zurückgesetzt wurde.

ERRID: Liefert bei einem gesetzten ERR-Ausgang die ADS-Fehlernummer [▶ 154].

DEVICENAME: Die Gerätebezeichnung des IO-Gerätes (Typ: T_MaxString).

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO Hardware	Einzubindende SPS-Bibliotheken (Kategoriegruppe)
TwinCAT v3.1.0	PC oder CX (x86, x64, ARM)	Alle IO-Geräte	Tc2_IoFunctions (IO)

3.1.12 IOF_GetDeviceNetId

Einigen IO-Geräten kann eine TwinCAT Netzwerkadresse während der Konfiguration im TwinCAT-System zugewiesen werden (z.B. FC310x Profibuskarte oder CP9030-Karte). Die Netzwerkadresse kann dann benutzt werden, um auf dem Gerät Firmware-Funktionen ausführen zu können. Der Funktionsbaustein IOF_GetDeviceNetId ermittelt die Netzwerkadresse anhand der Geräte-ID. Die Geräte-IDs werden während der Konfiguration vom TwinCAT-System festgelegt und können nicht vom Benutzer konfiguriert werden.

VAR_INPUT

```
VAR_INPUT
    NETID     : T_AmsNetId;
    DEVICEID  : UDINT;
    START     : BOOL;
    TMOUT     : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

NETID: Hier kann die Netzwerkadresse des TwinCAT-Rechners angegeben werden, auf dem die Funktion ausgeführt werden soll (Typ: T_AmsNetID). Für den lokalen Rechner kann auch ein Leerstring angegeben werden.

DEVICEID: Die ID des Gerätes, dessen TwinCAT Netzwerkadresse gelesen werden soll

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

TMOUT: Gibt die Timeout-Zeit an, die bei der Ausführung des Kommandos nicht überschritten werden darf.

VAR_OUTPUT

```
VAR_OUTPUT
    BUSY        : BOOL;
    ERR         : BOOL;
    ERRID       : UDINT;
    DeviceNetId : T_AmsNetId;
END_VAR
```

BUSY: Bei der Aktivierung des Funktionsbausteins wird dieser Ausgang gesetzt und bleibt gesetzt, bis eine Rückmeldung erfolgt.

ERR: Sollte ein Fehler bei der Übertragung des Kommandos erfolgen, dann wird dieser Ausgang gesetzt, nachdem der BUSY-Ausgang zurückgesetzt wurde.

ERRID: Liefert bei einem gesetzten ERR-Ausgang die ADS-Fehlernummer [▶ 154].

DeviceNetId: Die AmsNetId des Gerätes als String (Typ: T_AmsNetID).

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO Hardware	Einzubindende SPS-Bibliotheken (Kategoriegruppe)
TwinCAT v3.1.0	PC oder CX (x86, x64, ARM)	Alle IO-Geräte	Tc2_IoFunctions (IO)

3.1.13 IOF_GetDeviceType

Der Funktionsbaustein "IOF_GetDeviceType" ermittelt den Geräte-Typ anhand der Geräte-ID. Die Geräte-IDs werden während der Konfiguration vom TwinCAT-System festgelegt und können nicht vom Benutzer konfiguriert werden.

VAR_INPUT

```
VAR_INPUT
    NETID     : T_AmsNetId;
    DEVICEID  : UDINT;
    START     : BOOL;
    TMOUT     : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

NETID: Hier kann die Netzwerkadresse des TwinCAT-Rechners angegeben werden, auf dem die Funktion ausgeführt werden soll (Typ: T_AmsNetID). Für den lokalen Rechner kann auch ein Leerstring angegeben werden.

DEVICEID: Die ID des Gerätes, dessen Geräte-Typ gelesen werden soll

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

TMOUT: Gibt die Timeout-Zeit an, die bei der Ausführung des Kommandos nicht überschritten werden darf.

VAR_OUTPUT

```
VAR_OUTPUT
    BUSY         : BOOL;
    ERR          : BOOL;
    ERRID        : UDINT;
    IODeviceType : IODEVICETYPES;
END_VAR
```

BUSY: Bei der Aktivierung des Funktionsbausteins wird dieser Ausgang gesetzt und bleibt gesetzt, bis eine Rückmeldung erfolgt.

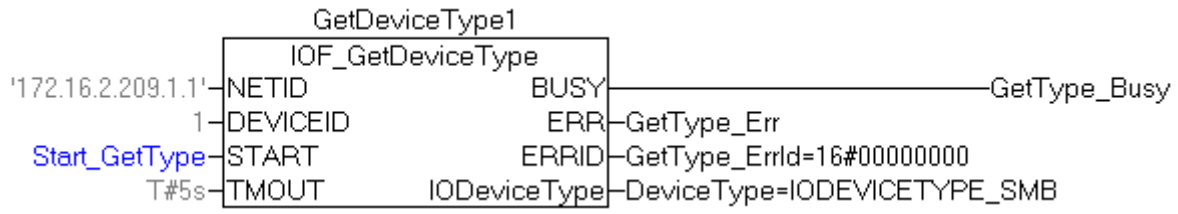
ERR: Sollte ein Fehler bei der Übertragung des Kommandos erfolgen, dann wird dieser Ausgang gesetzt, nachdem der BUSY-Ausgang zurückgesetzt wurde.

ERRID: Liefert bei einem gesetzten ERR-Ausgang die ADS-Fehlernummer [▶ 154].

IODeviceType: Die Geräte-Typ-Konstante (Typ: IODEVICETYPES [▶ 126]).

Beispiel:

```
PROGRAM MAIN
VAR
    GetDeviceType1 : IOF_GetDeviceType;
    Start_GetType  : BOOL;
    GetType_Busy   : BOOL;
    GetType_Err    : BOOL;
    GetType_ErrId  : UDINT;
    DeviceType     : IODEVICETYPES;
END_VAR
```



Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO Hardware	Einzubindende SPS-Bibliotheken (Kategoriegruppe)
TwinCAT v3.1.0	PC oder CX (x86, x64, ARM)	Alle IO-Geräte	Tc2_IoFunctions (IO)

3.2 ASI-Masterklemme

3.2.1 Übersicht

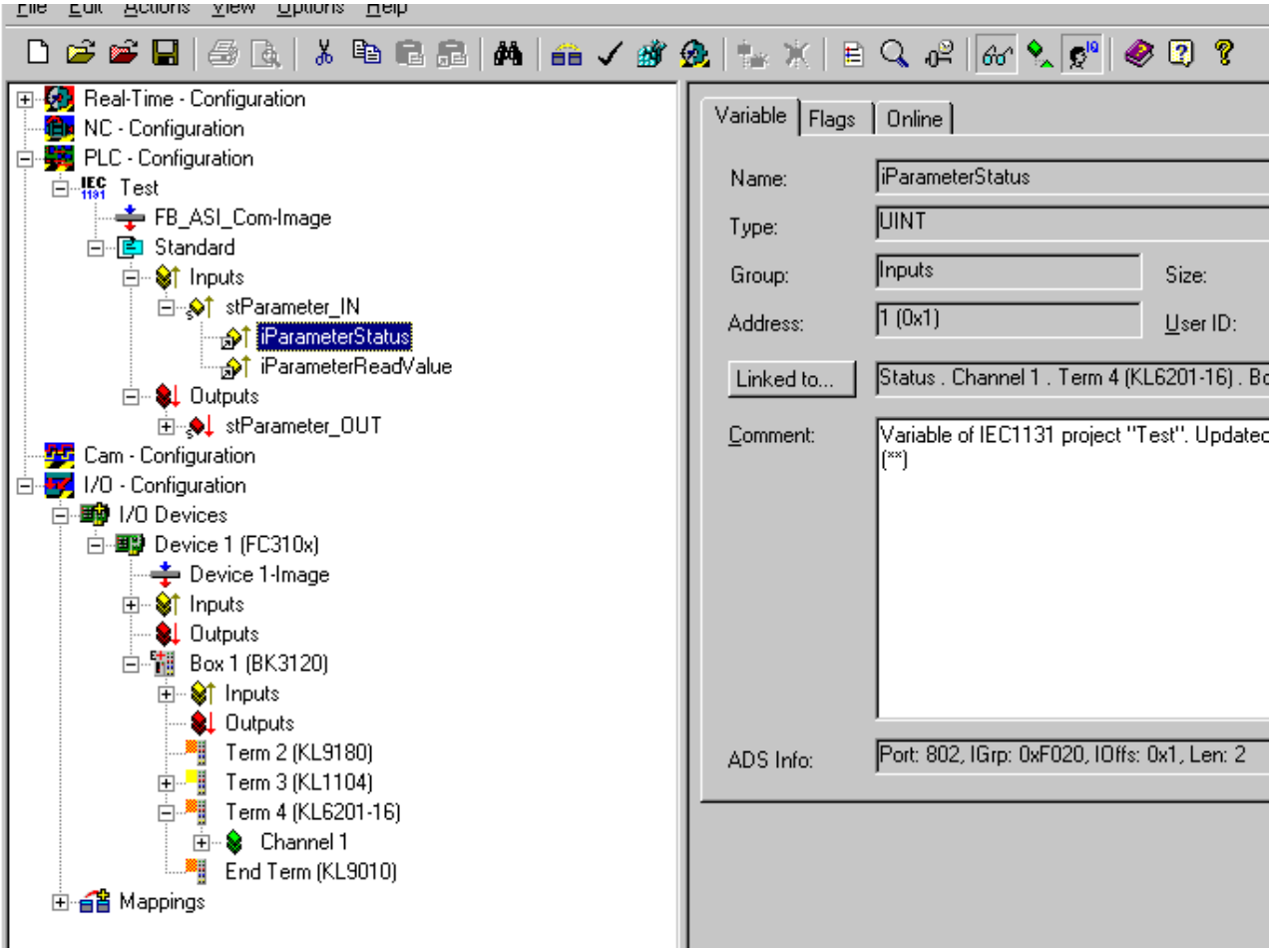
Funktionsbausteine für den Zugriff auf die ASI-Masterklemme.

Funktionsbausteine:

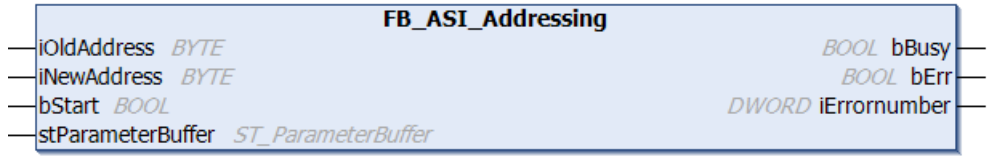
Name	Beschreibung
FB_ASI_Addressing [▶ 27]	Festlegen oder Verändern von Adressen der ASI-Slaves
FB_ASI_SlaveDiag [▶ 28]	Zyklische Slave-Diagnose (z.B. Zählerstände)
FB_ASI_ReadParameter [▶ 29]	Universeller FB zum Auslesen aller Parameter eines ASI-Slaves
FB_ASI_WriteParameter [▶ 31]	Universeller FB zum Setzen aller Parameter eines ASI-Slaves
FB_ReadInput_analog [▶ 34]	Lesen von analogen Werten
FB_WriteOutput_analog [▶ 35]	Schreiben von analogen Werten
FB_ASI_Processdata_digital [▶ 32]	Lesen/Schreiben von digitalen Werten
FB_ASI_ParameterControl [▶ 33]	Hintergrundkommunikation **Dieser Baustein muss immer zyklisch aufgerufen werden!!!**

Einbinden in den System Manager

Die Bibliothek besitzt eine Eingangsstruktur: ST_Parameter_IN und eine Ausgangsstruktur: ST_Parameter_OUT. Diese müssen instanziert und adressiert werden, um sie einerseits dem FB_ParameterControl als VAR_IN_OUT übergeben zu können und andererseits im System Manager verknüpft zu werden. Die Prozessdaten der Klemme beinhalten 6Byte und 16Byte, je nachdem, welches ASI-Modul im System Manager eingebunden wurde. Diese können direkt verknüpft werden.



3.2.2 FB_ASI_Addressing



VAR_IN_OUT

```
VAR_IN_OUT
    stParameterBuffer : ST_ParameterBuffer;
END_VAR
```

stParameterBuffer: Datenpuffer für die Hintergrundkommunikation (Typ: ST_ParameterBuffer [▶ 133]).

VAR_INPUT

```
VAR_INPUT
    iOldAddress  : BYTE;(*old address*)
    iNewAddress  : BYTE;(*new address*)
    bStart       : BOOL;(*START*)
END_VAR
```

iOldAddress: alte Adresse des zu adressierenden Slaves (neue Slaves haben die Adresse 0).

iNewAddress: neue Adresse des zu adressierenden Slaves.

bStart: Mit einer positiven Flanke an diesem boolschen Eingang wird die Adressierung vorgenommen.

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy        : BOOL;
    bErr         : BOOL;
    iErrornumber : DWORD; (* Error code of ASI-Master *)
END_VAR
```

bBusy: Dieser Ausgang bleibt solange auf TRUE, bis der Baustein eine Befehlsanforderung ausführt. Während Busy = TRUE wird an den Eingängen kein neuer Befehl angenommen. Bitte beachten Sie, dass nicht die Ausführung des Dienstes, sondern nur dessen Annahme zeitlich überwacht wird.

bErr: Dieser Ausgang zeigt den Fehlerstatus an.

iErrornumber: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Wird durch das Ausführen eines Befehls an den Eingängen auf 0 zurückgesetzt.

Befehlsspezifischer Fehlercode (dezimal)	Beschreibung
1	Kommunikationstimeout
2	ASI-Slaveadresse nicht vorhanden
3 - 10	Reserviert
11	ASI-Slave ist nicht aktiviert (Slave ist nicht in LAS)
12	Bei der Kommunikation ist ein Fehler aufgetreten
13	Datenaustauschbit (CN.4) nicht gesetzt

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO Hardware	Einzubindende SPS-Bibliotheken (Kategoriegruppe)
TwinCAT v3.1.0	PC oder CX (x86)	ASI Masterklemme	Tc2_IoFunctions (IO)

3.2.3 FB_ASI_SlaveDiag

VAR_IN_OUT

```
VAR_IN_OUT
    stParameterBuffer : ST_ParameterBuffer;
END_VAR
```

stParameterBuffer : Datenpuffer für die Hintergrundkommunikation (Typ: ST_ParameterBuffer [▶ 133]).

VAR_INPUT

```
VAR_INPUT
    iSlaveaddress : BYTE;
    iCounter      : INT;(*1:PhysicalFaultCounter, 2:TimeoutCounter, 3:ResponseCounter, 4:Leave-
DataExchCounter, 5:DataExch-FailedCounter *)
    bCounterReset : BOOL;
    bReadLES      : BOOL;(*Read List of all detected Slaves*)
    bReadLAS      : BOOL;(*Read List of all activated Slaves*)
    bStart        : BOOL;
END_VAR
```

iSlaveaddress: Slaveadresse

iCounter: 1:PhysicalFaultCounter, 2:TimeoutCounter, 3:ResponseCounter, 4:Leave-DataExchCounter, 5:DataExch-FailedCounter.

bCounterReset: Rücksetzen des aktuellen Zählers.

bReadLES: Liste der erkannten ASI-Slaves(LES).

bReadWrite: 0=READ, 1=WRITE.

bReadLAS: Liste der aktivierten ASI-Slaves(LAS).

bStart: Mit einer positiven Flanke an diesem boolschen Eingang wird der entsprechende Auftrag ausgeführt.

bCycleMode: 0=continuous reading 1= reading once.

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy         : BOOL;
    bErr          : BOOL;
    iErrornumber  : DWORD;(* Error code of ASI-Master *)
    iCounterValue : WORD;(*Counter of a slave*)
    iSlaveList    : DWORD;(*LES or LAS of all Slaves*)
END_VAR
```

bBusy: Dieser Ausgang bleibt solange auf TRUE, bis der Baustein eine Befehlsanforderung ausführt. Während Busy = TRUE wird an den Eingängen kein neuer Befehl angenommen. Bitte beachten Sie, dass nicht die Ausführung des Dienstes, sondern nur dessen Annahme zeitlich überwacht wird.

bErr: Dieser Ausgang zeigt den Fehlerstatus an.

iErrornumber: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Wird durch das Ausführen eines Befehls an den Eingängen auf 0 zurückgesetzt.

Befehlsspezifischer Fehlercode (dezimal)	Beschreibung
1	Kommunikationstimeout
2	ASI-Slaveadresse nicht vorhanden
3 - 10	Reserviert
11	ASI-Slave ist nicht aktiviert (Slave ist nicht in LAS)
12	Bei der Kommunikation ist ein Fehler aufgetreten
13	Datenaustauschbit (CN.4) nicht gesetzt

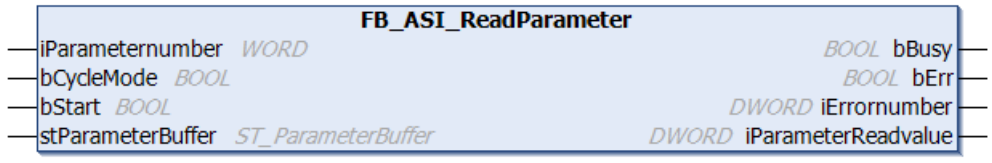
iCountervalue: Zählerstand.

iSlaveList: LES bzw. LAS.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO Hardware	Einzubindende SPS-Bibliotheken (Kategoriegruppe)
TwinCAT v3.1.0	PC oder CX (x86)	ASI Masterklemme	Tc2_IoFunctions (IO)

3.2.4 FB_ASI_ReadParameter



VAR_IN_OUT

```
VAR_IN_OUT
    stParameterBuffer : ST_ParameterBuffer;
END_VAR
```

stParameterBuffer: Datenpuffer für die Hintergrundkommunikation (Typ: ST_ParameterBuffer [▶ 133]).

VAR_INPUT

```
VAR_INPUT
    iParameternumber : WORD;
    bCycleMode       : BOOL;(*0: Acyclic , 1:Cyclic (permanent Read/Write) *)
    bStart           : BOOL;
END_VAR
```

iParameterNumber: Parameternummer.

bCycleMode: 0: Acyclic , 1:Cyclic (permanent Read/Write) Ist dieses Bit gesetzt, wird der Ausgang bBusy erst zurückgenommen, wenn der Eingang bStart auf FALSE gezogen wird. Wird der Eingang bStart zu früh auf FALSE gezogen, steht noch kein aktueller Wert am Ausgang an.

bStart: Mit einer positiven Flanke an diesem boolschen Eingang wird der entsprechende Auftrag ausgeführt.

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy               : BOOL;
    bErr                : BOOL;
    iErrornumber        : DWORD;(* Error code of ASI-Master *)
    iParameterReadvalue : BYTE;
END_VAR
```

bBusy: Dieser Ausgang bleibt solange auf TRUE, bis der Baustein eine Befehlsanforderung ausführt. Während Busy = TRUE wird an den Eingängen kein neuer Befehl angenommen. Bitte beachten Sie, dass nicht die Ausführung des Dienstes, sondern nur dessen Annahme zeitlich überwacht wird.

bErr: Dieser Ausgang zeigt den Fehlerstatus an.

iErrornumber: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Wird durch das Ausführen eines Befehls an den Eingängen auf 0 zurückgesetzt.

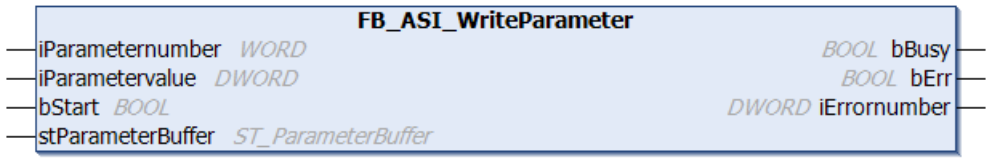
Befehlsspezifischer Fehlercode (dezimal)	Beschreibung
1	Kommunikationstimeout
2	ASI-Slaveadresse nicht vorhanden
3 - 10	Reserviert
11	ASI-Slave ist nicht aktiviert (Slave ist nicht in LAS)
12	Bei der Kommunikation ist ein Fehler aufgetreten
13	Datenaustauschbit (CN.4) nicht gesetzt

iParameterReadvalue: E/A-Kennung bzw. ID-Code des angesprochenen Slaves.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO Hardware	Einzubindende SPS-Bibliotheken (Kategoriegruppe)
TwinCAT v3.1.0	PC oder CX (x86)	ASI Masterklemme	Tc2_IoFunctions (IO)

3.2.5 FB_ASI_WriteParameter



VAR_IN_OUT

```
VAR_IN_OUT
    stParameterBuffer : ST_ParameterBuffer;
END_VAR
```

stParameterBuffer: Datenpuffer für die Hintergrundkommunikation (Typ: ST_ParameterBuffer [▶ 133]).

VAR_INPUT

```
VAR_INPUT
    iParameternumber : WORD;
    iParametervalue  : DWORD;
    bStart           : BOOL;
END_VAR
```

iParameterNumber: Parameternummer.

iParametervalue: Parameterwert.

bStart: Mit einer positiven Flanke an diesem boolschen Eingang wird der entsprechende Auftrag ausgeführt.

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy        : BOOL;
    bErr         : BOOL;
    iErrornumber : DWORD;(* Error code of ASI-Master *)
END_VAR
```

bBusy: Dieser Ausgang bleibt solange auf TRUE, bis der Baustein eine Befehlsanforderung ausführt. Während Busy = TRUE wird an den Eingängen kein neuer Befehl angenommen. Bitte beachten Sie, dass nicht die Ausführung des Dienstes, sondern nur dessen Annahme zeitlich überwacht wird.

bErr: Dieser Ausgang zeigt den Fehlerstatus an.

iErrornumber: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Wird durch das Ausführen eines Befehls an den Eingängen auf 0 zurückgesetzt.

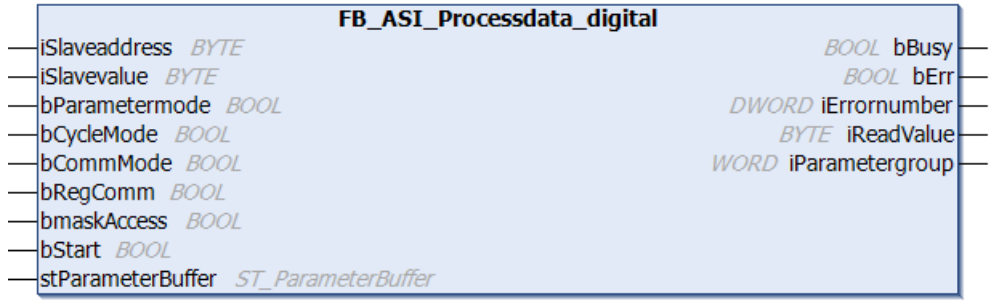
Befehlsspezifischer Fehlercode (dezimal)	Beschreibung
1	Kommunikationstimeout
2	ASI-Slaveadresse nicht vorhanden
3 - 10	Reserviert
11	ASI-Slave ist nicht aktiviert (Slave ist nicht in LAS)
12	Bei der Kommunikation ist ein Fehler aufgetreten
13	Datenaustauschbit (CN.4) nicht gesetzt

iParameterReadvalue: E/A-Kennung bzw. ID-Code des angesprochenen Slaves.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO Hardware	Einzubindende SPS-Bibliotheken (Kategoriegruppe)
TwinCAT v3.1.0	PC oder CX (x86)	ASI Masterklemme	Tc2_IoFunctions (IO)

3.2.6 FB_ASI_Processdata_digital



VAR_IN_OUT

```
VAR_IN_OUT
    stParameterBuffer : ST_ParameterBuffer;
END_VAR
```

stParameterBuffer: Datenpuffer für die Hintergrundkommunikation (Typ: ST_ParameterBuffer [▶ 133]).

VAR_INPUT

```
VAR_INPUT
    iSlaveaddress  : BYTE;
    iSlavevalue    : WORD;
    bParametermode : BOOL;(*0: Read, 1: Write *)
    bCycleMode     : BOOL;(*0: Acyclic , 1:Cyclic (permanent Read/Write) *)
    bCommMode      : BOOL;(*0: Parameterzugriff, 1: ADS*)
    bRegComm       : BOOL;(*Registerkommunikation: 0: Parameterzugriff, 1: Registerkommunikation *)
    bmaskAccess    : BOOL;(*0:usual access, 1:mask access*)
    bStart         : BOOL;
END_VAR
```

iSlaveaddress: Slaveadresse.

iSlavevalue: Prozesswert.

bParametermode: 0: Read, 1: Write.

bCycleMode: 0: Acyclic , 1:Cyclic (permanent Read/Write) Ist dieses Bit gesetzt, wird der Ausgang bBusy erst zurückgenommen, wenn der Eingang bStart auf FALSE gezogen wird. Wird der Eingang bStart zu früh auf FALSE gezogen, steht noch kein aktueller Wert am Ausgang an.

bCommMode: 0: Parameterzugriff, 1: ADS (z.Z. immer 0).

bRegComm: Registerkommunikation: 0: Parameterzugriff, 1: Registerkommunikation (z.Z. immer 0).

bmaskAccess: 0:normaler Zugriff, 1:maskierter Zugriff.

bStart: Mit einer positiven Flanke an diesem boolschen Eingang wird der entsprechende Auftrag ausgeführt.

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy          : BOOL;
    bErr           : BOOL;
    iErrornumber   : DWORD;(* Error code of ASI-Master *)
    iReadValue     : WORD;
    iParametergroup : WORD;
END_VAR
```

bBusy: Dieser Ausgang bleibt solange auf TRUE, bis der Baustein eine Befehlsanforderung ausführt. Während Busy = TRUE wird an den Eingängen kein neuer Befehl angenommen. Bitte beachten Sie, dass nicht die Ausführung des Dienstes, sondern nur dessen Annahme zeitlich überwacht wird.

bErr: Dieser Ausgang zeigt den Fehlerstatus an.

iErrornumber: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Wird durch das Ausführen eines Befehls an den Eingängen auf 0 zurückgesetzt.

Befehlsspezifischer Fehlercode (dezimal)	Beschreibung
1	Kommunikationstimeout
2	ASI-Slaveadresse nicht vorhanden
3 - 10	Reserviert
11	ASI-Slave ist nicht aktiviert (Slave ist nicht in LAS)
12	Bei der Kommunikation ist ein Fehler aufgetreten
13	Datenaustauschbit (CN.4) nicht gesetzt

iReadvalue: Prozesswert.

iParametergroup: Parametergruppe.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO Hardware	Einzubindende SPS-Bibliotheken (Kategoriegruppe)
TwinCAT v3.1.0	PC oder CX (x86)	ASI Masterklemme	Tc2_IoFunctions (IO)

3.2.7 FB_ASI_ParameterControl

Der FB_ASI_ParameterControl realisiert die Hintergrundkommunikation zwischen der ASI-Masterklemme und den einzelnen Bausteinen der Lib.

Aufruf des Bausteins

Dieser Baustein muss immer zyklisch aufgerufen werden

```
VAR_IN_OUT
    stParameterBuffer : ST_ParameterBuffer;
    stParameter_IN    : ST_Parameter_IN;
    stParameter_OUT   : ST_Parameter_OUT;
END_VAR
```

stParameterBuffer: Datenpuffer für die Hintergrundkommunikation (Typ: ST_ParameterBuffer [▶ 133]).

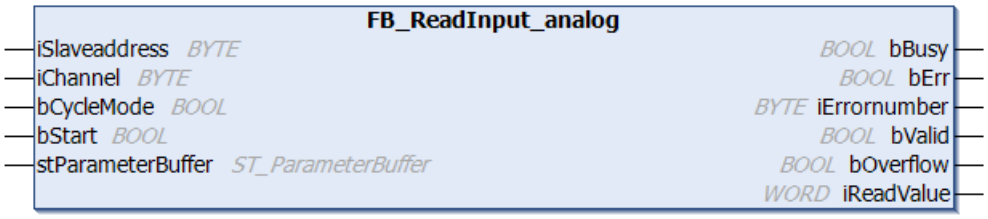
stParameter_IN: Eingangsdaten von der ASI-Klemme (Typ: ST_Parameter_IN [▶ 131]).

stParameter_OUT: Eingangsdaten von der ASI-Klemme (Typ: ST_Parameter_OUT [▶ 132]).

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO Hardware	Einzubindende SPS-Bibliotheken (Kategoriegruppe)
TwinCAT v3.1.0	PC oder CX (x86)	ASI Masterklemme	Tc2_IoFunctions (IO)

3.2.8 FB_ReadInput_analog



VAR_IN_OUT

```
VAR_IN_OUT
    stParameterBuffer : ST_ParameterBuffer;
END_VAR
```

stParameterBuffer: Datenpuffer für die Hintergrundkommunikation (Typ: ST_ParameterBuffer [▶ 133]).

VAR_INPUT

```
VAR_INPUT
    iSlaveaddress : BYTE;
    iChannel      : BYTE;
    bCycleMode    : BOOL;(*0: Acyclic , 1:Cyclic (permanent Read/Write) *)
    bStart        : BOOL;
END_VAR
```

iSlaveaddress: Slaveadresse.

iChannel: Kanal des Slaves.

bCycleMode: 0: Acyclic , 1:Cyclic (permanent Read/Write) Ist dieses Bit gesetzt, wird der Ausgang bBusy erst zurückgenommen, wenn der Eingang bStart auf FALSE gezogen wird. Wird der Eingang bStart zu früh auf FALSE gezogen, steht noch kein aktueller Wert am Ausgang an.

bStart: Mit einer positiven Flanke an diesem boolschen Eingang wird der entsprechende Auftrag ausgeführt.

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy        : BOOL;
    bErr         : BOOL;
    iErrornumber : DWORD;(* Error code of ASI-Master *)
    bValid       : BOOL;
    bOverflow    : BOOL;
    iReadValue   : WORD;
END_VAR
```

bBusy: Dieser Ausgang bleibt solange auf TRUE, bis der Baustein eine Befehlsanforderung ausführt. Während Busy = TRUE wird an den Eingängen kein neuer Befehl angenommen. Bitte beachten Sie, dass nicht die Ausführung des Dienstes, sondern nur dessen Annahme zeitlich überwacht wird.

bErr: Dieser Ausgang zeigt den Fehlerstatus an.

iErrornumber: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Wird durch das Ausführen eines Befehls an den Eingängen auf 0 zurückgesetzt.

Befehlsspezifischer Fehlercode (dezimal)	Beschreibung
1	Kommunikationstimeout
2	ASI-Slaveadresse nicht vorhanden
3 - 10	Reserviert
11	ASI-Slave ist nicht aktiviert (Slave ist nicht in LAS)
12	Bei der Kommunikation ist ein Fehler aufgetreten
13	Datenaustauschbit (CN.4) nicht gesetzt

bValid: Gültigkeit der gelesenen Werte.

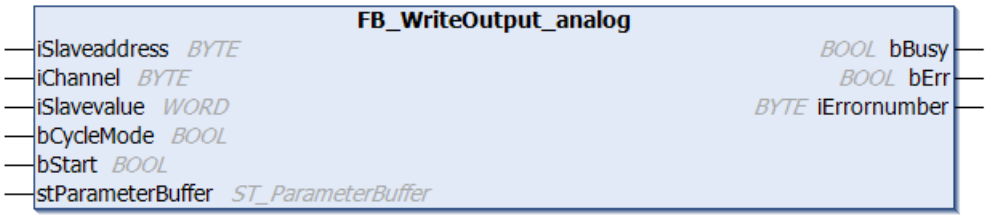
bOverflow: Slave hat einen Wert außerhalb seines Wertebereiches.

iReadvalue: Prozesswert.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO Hardware	Einzubindende SPS-Bibliotheken (Kategoriegruppe)
TwinCAT v3.1.0	PC oder CX (x86)	ASI Masterklemme	Tc2_IoFunctions (IO)

3.2.9 FB_WriteOutput_analog



VAR_IN_OUT

```
VAR_IN_OUT
    stParameterBuffer : ST_ParameterBuffer;
END_VAR
```

stParameterBuffer: Datenpuffer für die Hintergrundkommunikation (Typ: ST_ParameterBuffer [▶ 133]).

VAR_INPUT

```
VAR_INPUT
    iSlaveaddress : BYTE;
    iChannel      : BYTE;
    iSlavevalue   : WORD;
    bCycleMode    : BOOL;(*0: Acyclic , 1:Cyclic (permanent Read/Write) *)
    bStart        : BOOL;
END_VAR
```

iSlaveaddress: Slaveadresse.

iChannel: Kanal des Slaves.

iSlavevalue: zu schreibende Daten.

bCycleMode: 0: Acyclic , 1:Cyclic (permanent Read/Write) Ist dieses Bit gesetzt, wird der Ausgang bBusy erst zurückgenommen, wenn der Eingang bStart auf FALSE gezogen wird. Wird der Eingang bStart zu früh auf FALSE gezogen, steht noch kein aktueller Wert am Ausgang an.

bStart: Mit einer positiven Flanke an diesem boolschen Eingang wird der entsprechende Auftrag ausgeführt.

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy        : BOOL;
    bErr         : BOOL;
    iErrornumber : DWORD;(* Error code of ASI-Master *)
    iReadValue   : WORD;
END_VAR
```

bBusy: Dieser Ausgang bleibt solange auf TRUE, bis der Baustein eine Befehlsanforderung ausführt. Während Busy = TRUE wird an den Eingängen kein neuer Befehl angenommen. Bitte beachten Sie, dass nicht die Ausführung des Dienstes, sondern nur dessen Annahme zeitlich überwacht wird.

bErr: Dieser Ausgang zeigt den Fehlerstatus an.

iErrornumber: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Wird durch das Ausführen eines Befehls an den Eingängen auf 0 zurückgesetzt.

Befehlsspezifischer Fehlercode (dezimal)	Beschreibung
1	Kommunikationstimeout
2	ASI-Slaveadresse nicht vorhanden
3 - 10	Reserviert
11	ASI-Slave ist nicht aktiviert (Slave ist nicht in LAS)
12	Bei der Kommunikation ist ein Fehler aufgetreten
13	Datenaustauschbit (CN.4) nicht gesetzt

bValid: Gültigkeit der gelesenen Werte.

bOverflow: Slave hat einen Wert außerhalb seines Wertebereiches.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO Hardware	Einzubindende SPS-Bibliotheken (Kategoriegruppe)
TwinCAT v3.1.0	PC oder CX (x86)	ASI Masterklemme	Tc2_IoFunctions (IO)

3.3 AX200x Profibus

3.3.1 Übersicht

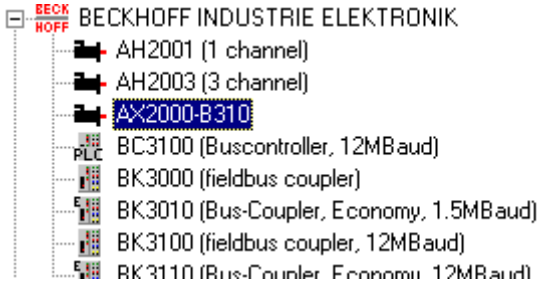
Funktionsbausteine für den Zugriff auf den AX20XX über den Profibus. Voraussetzung für den Betrieb am Profibus ist die Verwendung einer FC310x mit einer Firmwareversion größer 1.20.

Funktionsbausteine

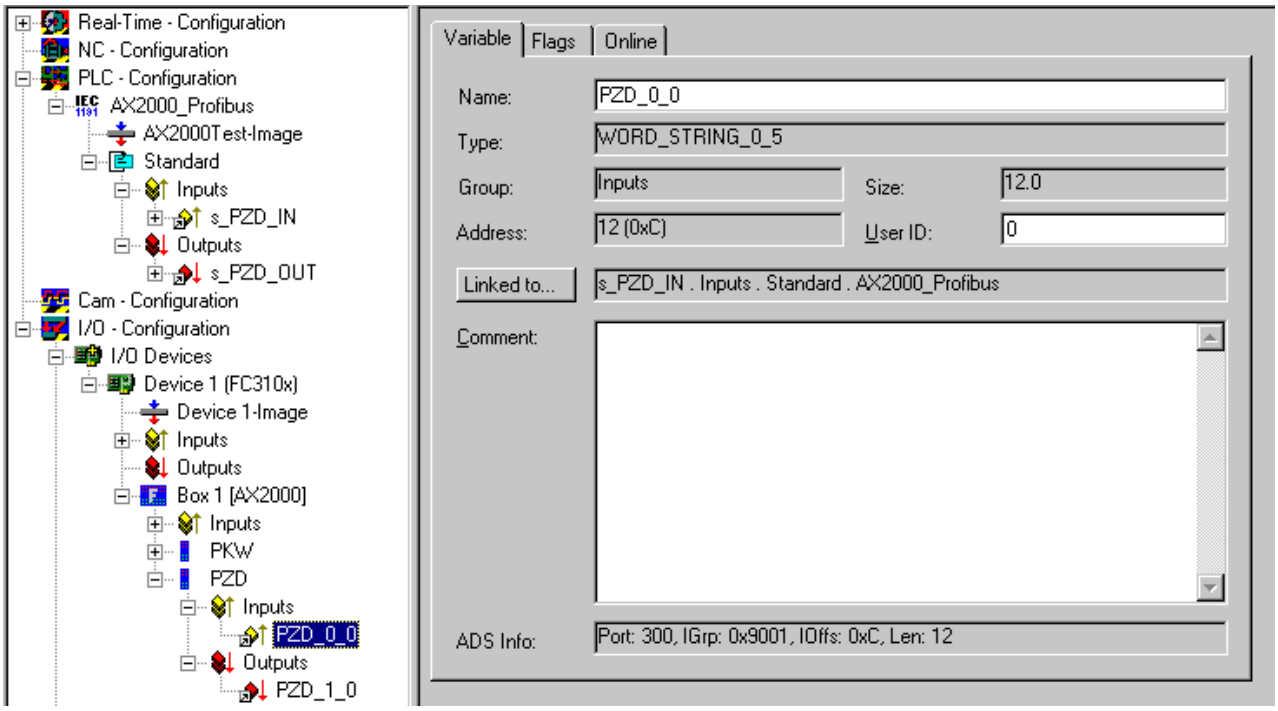
Name	Beschreibung
FB_AX2000_Parameter [▶ 40]	Schreiben/Lesen der Daten zur Parametrierung des Antriebs. **Es muss beachtet werden, dass während des Schreibens eines Parameters zum Wechsel der Betriebsart der Eingang "STOP" des Bausteins AX2000AXACT auf TRUE gehalten werden muss.**
FB_AX2000_AXACT [▶ 38]	Starten von Achsaktionen (muss immer zyklisch aufgerufen werden)
FB_AX2000_Jogmode [▶ 39]	Tippbetrieb
FB_AX2000_Reference [▶ 41]	Setzen des Referenzpunktes bzw. starten einer Referenzfahrt
FB_AX200X_Profibus [▶ 42]	Dieser Baustein fasst die drei vorangegangenen Bausteine zusammen. Er bietet die komplette Schnittstelle zum AX2000 mit Zugriff auf sämtliche Funktionen (ausg. Parameter).

Einbindung in den System Manager

Im TwinCAT System Manager wird in der I/O-Konfiguration unter der entsprechenden ProfibusKarte direkt die Box "AX2000" angefügt.



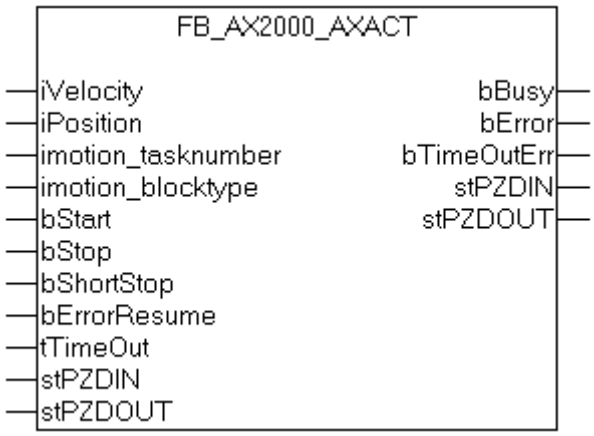
Im Modul "PZD" (Prozessdaten) der AX2000-Box können nun die I/O-Variablen direkt mit den entsprechenden I/O-Variablen der SPS-Applikation verknüpft werden. Das Modul "PKW" wird nicht verknüpft, da die PKW-Daten per ADS übertragen werden.



Hinweise zur Benutzung der Funktionsbausteine

- Die E/A-Strukturen stPZD_IN und stPZD_OUT müssen instanziiert und adressiert werden, um sie im System Manager mit der Achse verknüpfen zu können.
- Der Antrieb befindet sich nach dem Einschalten in einem Sicherheits-Betriebsmodus, d.h. vor der ersten Achsaktion muss der Betriebsmodus "Positionierung" oder "Drehzahl digital" eingestellt werden. Dies geschieht durch setzen des Eingangs "bInit" und "bMode_DigitalSpeed" (bei Drehzahlmodus) am Baustein AX200X_Profibus.
- Die Fahrtrichtung im Tippbetrieb wird durch das Vorzeichen der "JogModeBasicVelo" festgelegt.
- Jede Referenzfahrt und jedes Setzen des Referenzpunktes **muss** mit einem bStop =TRUE beendet werden.

3.3.2 FB_AX2000_AXACT



VAR_IN_OUT

```
VAR_IN_OUT
    stPZDIN    : ST_PZD_IN;
    stPZDOUT   : ST_PZD_OUT;
END_VAR
```

stPZDIN: Datenwörter vom Antrieb zur PLC (Typ: ST_PZD_IN [▶ 135]).

stPZDOUT: Datenwörter von der PLC zum Antrieb (Typ: ST_PZD_OUT [▶ 136]).

VAR_INPUT

```
VAR_INPUT
    bMode_DigitalSpeed : BOOL;(*OP-Mode digital speed instead of Positioning*)
    iDigitalSpeed      : DWORD;(*digital speed if OP-Mode = digital speed*)
    iVelocity          : DWORD;(*Velocity*)
    iPosition          : DINT;(*Position*)
    imotion_tasknumber : WORD;(*number of EEPROM-saved motion-task*)
    imotion_blocktype  : WORD;(*optional Parameters of motion tasks*)
    bStart             : BOOL;(*START*)
    bStop              : BOOL;(*STOP*)
    bShortStop         : BOOL;(*1: break of motion task, 0: continue same motion task*)
    bErrorResume       : BOOL;(*Error resume*)
    tTimeOut           : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

bMode_DigitalSpeed: wird gesetzt, wenn der Antrieb bei der Initialisierung in die Betriebsart 'Drehzahl digital' versetzt werden soll.

iDigitalSpeed: Drehzahl in der Betriebsart 'Drehzahl digital'.

iVelocity: Der Parameter enthält die geforderte Fahrgeschwindigkeit für einen nachfolgenden Fahrauftrag z.B. µm/s.

iPosition: Zielposition in physikalischen Größen z.B. µm, Grad.

imotion_tasknumber: Fahrsatznummer. Mit diesem Eingang kann ein vorher im Speicher des Antriebes abgelegter Fahrsatz ausgewählt werden.

imotion_blocktype: Fahrsatzart (optional) Mit diesem Eingang können Eigenschaften eines Direktfahrauftrages verändert werden.

bStart: Mit einer positiven Flanke an diesem boolschen Eingang wird ein Startbefehl an die Achse gesendet.

bStop: Mit einer positiven Flanke an diesem boolschen Eingang wird ein Stoppbefehl an die Achse gesendet. Die Achse hält und geht in den Zustand "disabled".

bShortStop: Mit einer positiven Flanke an diesem boolschen Eingang wird ein Stoppbefehl an die Achse gesendet. Die Achse hält, bleibt aber im Zustand "enabled".

bErrorResume: Mit einer positiven Flanke an diesem boolschen Eingang wird ein "AX200X-Fehler" zurückgesetzt (kein TimeOut-Fehler).

tTimeOut: Maximale Zeit die bei der Ausführung des Befehls nicht überschritten werden soll.

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy       : BOOL;
    bError      : BOOL;(*Errorstatus of Servo*)
    bTimeOutErr : BOOL;
END_VAR
```

bBusy: Dieser Ausgang bleibt solange auf TRUE, bis der Baustein eine Befehlsanforderung ausführt. Während Busy = TRUE wird an den Eingängen kein neuer Befehl angenommen. Bitte beachten Sie, dass nicht die Ausführung des Dienstes, sondern nur dessen Annahme zeitlich überwacht wird.

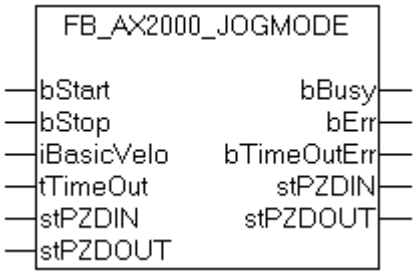
bError: Dieser Ausgang zeigt den Fehlerstatus an.

bTimeOutErr: TimeOut-Fehler.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO Hardware	Einzubindende SPS-Bibliotheken (Kategoriegruppe)
TwinCAT v3.1.0	PC oder CX (x86)	AX2000 Profibus box	Tc2_IoFunctions (IO)

3.3.3 FB_AX2000_JogMode



Tippbetrieb.

VAR_IN_OUT

```
VAR_IN_OUT
    stPZDIN    : ST_PZD_IN;
    stPZDOUT   : ST_PZD_OUT;
END_VAR
```

stPZDIN: Datenwörter vom Antrieb zur PLC (Typ: ST_PZD_IN [▶ 135]).

stPZDOUT: Datenwörter von der PLC zum Antrieb (Typ: ST_PZD_OUT [▶ 136]).

VAR_INPUT

```
VAR_INPUT
    bStart     : BOOL;
    bStop      : BOOL;
    iBasicVelo : INT;(*BasicVelocity*)
    tTimeOut   : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

bStart: Starten des Tippbetriebes.

bStop: Stoppen des Tippbetriebes.

iBasicVelo: Basisgeschwindigkeit, die tatsächliche Geschwindigkeit ergibt sich aus der Basisgeschwindigkeit und dem Faktor "v-Tippbetrieb" des Antriebes.

tTimeOut: Maximale Zeit die bei der Ausführung des Befehls nicht überschritten werden soll.

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy       : BOOL;
    bErr        : BOOL;
    bTimeOutErr : BOOL;
END_VAR
```

bBusy: Dieser Ausgang bleibt solange auf TRUE, bis der Baustein eine Befehlsanforderung ausführt. Während Busy = TRUE wird an den Eingängen kein neuer Befehl angenommen. Bitte beachten Sie, dass nicht die Ausführung des Dienstes, sondern nur dessen Annahme zeitlich überwacht wird.

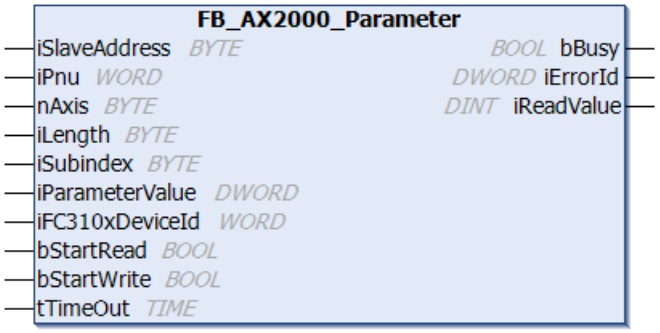
bErr: Dieser Ausgang zeigt den Fehlerstatus an.

bTimeOutErr: TimeOut-Fehler.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO Hardware	Einzubindende SPS-Bibliotheken (Kategoriegruppe)
TwinCAT v3.1.0	PC oder CX (x86)	AX2000 Profibus box	Tc2_IoFunctions (IO)

3.3.4 FB_AX2000_Parameter



Lesen/Schreiben der Parameter über den Parameterkanal.

VAR_INPUT

```
VAR_INPUT
    iSlaveAddress   : BYTE := 0;(* Station Address of the Slave *)
    iPnu            : WORD := 16#03A2;(* Parameter-Number *)
    nAxis           : BYTE := 1;(* Number of Axis *)
    iLength         : BYTE := 4;(* Length of the parameter (2 or 4) *)
    iParameterValue : DWORD := 2;(* Parameter value *)
    iFC310xDeviceId : WORD := 1;(* Device-ID of the FCxxxx *)
    bStartRead      : BOOL;(* StartFlag to start the PKW-Read *)
    bStartWrite     : BOOL;(* StartFlag to start the PKW-Write *)
    tTimeOut        : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

iSlaveAddress: Stationsadresse.

iPnu: Auswahl des zu schreibenden / zu lesenden Parameters. Liste mit den verfügbaren Parameternummern [▶ 149].

nAxis: Achsen-Id.

iLength: Länge der Parameter (2 oder 4).

iParameterValue: Wert des zu schreibenden / zu lesenden Parameters.

iFC310xDeviceId: Device-Id

bStartRead: Mit einer positiven Flanke an diesem boolschen Eingang wird ein Startbefehl zum Lesen des mit ‚Pnu' gewählten Parameters an die Achse gesendet.

bStartWrite: Mit einer positiven Flanke an diesem boolschen Eingang wird ein Startbefehl zum Schreiben des mit ‚Pnu' gewählten Parameters an die Achse gesendet. Bei Betriebsartenwechsel ist der Schreibbefehl nur bei Stop=TRUE an dem Baustein FB_AX2000_AXACT [▶ 38] wirksam.

tTimeOut: Maximale Zeit die bei der Ausführung des Befehls nicht überschritten werden darf.

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy      :BOOL;
    iErrorId   :DWORD;
    iReadValue :DINT;
END_VAR
```

bBusy: Dieser Ausgang bleibt solange auf TRUE, bis der Baustein eine Befehlsanforderung ausführt. Während Busy = TRUE wird an den Eingängen kein neuer Befehl angenommen. Bitte beachten Sie, dass nicht die Ausführung des Dienstes, sondern nur dessen Annahme zeitlich überwacht wird.

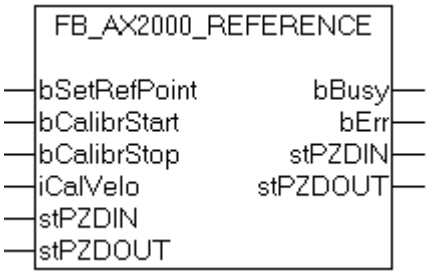
iErrorId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Wird durch das Ausführen eines Befehls an den Eingängen auf 0 zurückgesetzt.

iReadValue: Parameterwert als Antwort auf den Befehl 'StartRead'.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO Hardware	Einzubindende SPS-Bibliotheken (Kategoriegruppe)
TwinCAT v3.1.0	PC oder CX (x86)	AX2000 Profibus box	Tc2_IoFunctions (IO)

3.3.5 FB_AX2000_Reference



Referenzfahrt.

VAR_IN_OUT

```
VAR_IN_OUT
    stPZDIN  : ST_PZD_IN;
    stPZDOUT : ST_PZD_OUT;
END_VAR
```

stPZDIN: Datenwörter vom Antrieb zur PLC (Typ: ST_PZD_IN [▶ 135]).

stPZDOUT: Datenwörter von der PLC zum Antrieb (Typ: ST_PZD_OUT [▶ 136]).

VAR_INPUT

```
VAR_INPUT
    bSetRefPoint : BOOL;(* set Reference Point*)
    bCalibrStart : BOOL;(* start home running*)
    bCalibrStop  : BOOL;(* stop home running*)
    iCalVelo     : WORD;(* basic velocity of Calibration*)
END_VAR
```

bSetRefPoint: Setzen des Referenzpunktes.

bCalibrStart: Starten der Referenzfahrt.

bCalibrStop: Stoppen der Referenzfahrt.

iCalVelo: Basisgeschwindigkeit der Referenzfahrt. Die Endgeschwindigkeit setzt sich aus der Basisgeschwindigkeit und dem Faktor "v-Tippbetrieb" des Antriebes zusammen.

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy : BOOL;
    bErr  : BOOL;
END_VAR
```

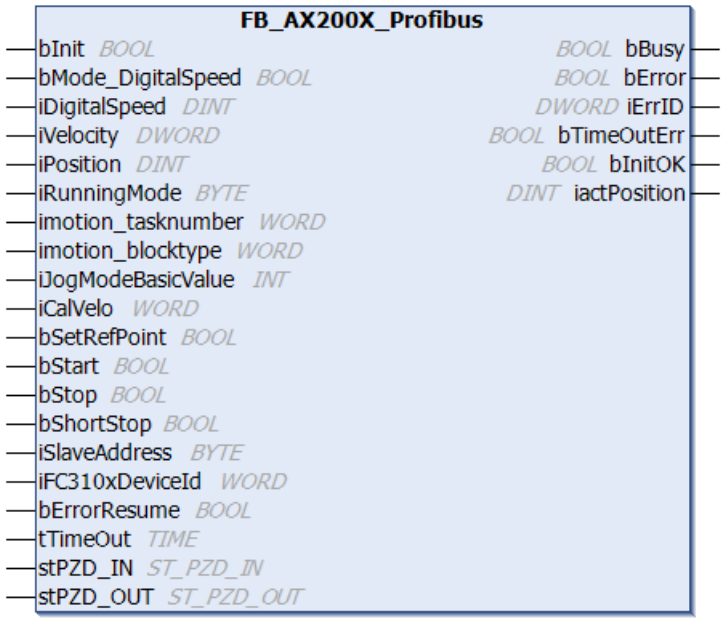
bBusy: Dieser Ausgang bleibt solange auf TRUE, bis der Baustein eine Befehlsanforderung ausführt. Während Busy = TRUE wird an den Eingängen kein neuer Befehl angenommen. Bitte beachten Sie, dass nicht die Ausführung des Dienstes, sondern nur dessen Annahme zeitlich überwacht wird.

bErr: Dieser Ausgang zeigt den Fehlerstatus an.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO Hardware	Einzubindende SPS-Bibliotheken (Kategoriegruppe)
TwinCAT v3.1.0	PC oder CX (x86)	AX2000 Profibus box	Tc2_IoFunctions (IO)

3.3.6 FB_AX200X_Profibus



VAR_IN_OUT

```
VAR_IN_OUT
    stPZD_IN  : ST_PZD_IN;
    stPZD_OUT : ST_PZD_OUT;
END_VAR
```

stPZD_IN: Datenwörter vom Antrieb zur PLC (Typ: ST_PZD_IN [▶ 135]).

stPZD_OUT: Datenwörter von der PLC zum Antrieb (Typ: ST_PZD_OUT [▶ 136]).

VAR_INPUT

```
VAR_INPUT
    bInit              : BOOL;(*Initialization*)
    bMode_DigitalSpeed : BOOL;(*OP-Mode digital speed instead of Positioning*)
    iDigitalSpeed      : DWORD;(*digital speed if OP-Mode = digital speed*)
    iVelocity          : DWORD;(*Velocity*)
    iPosition          : DINT;(*Position*)
    iRunningMode       : BYTE;(*0:digital speed, 1: motiontask, 2: JogMode, 3: Calibration*)
    imotion_tasknumber : WORD;(*number of EEPROM-saved motion-task*)
    imotion_blocktype  : WORD:=16#2000;(*optional Parameters of motion tasks, default:SI-values*)
    iJogModeBasicValue : INT;(*BasicVelocity for JogMode*)
    iCalVelo           : WORD;(* basic velocity of Calibration*)
    bSetRefPoint       : BOOL;(* set Reference Point*)
    bStart             : BOOL;(*START*)
    bStop              : BOOL;(*STOP*)
    bShortStop         : BOOL;(* break of motion task*)
    iSlaveAddress      : BYTE;(* Station Address of the Slave *)
    iFC310xDeviceId    : WORD;(* Device-ID of the FCxxxx *)
    bErrorResume       : BOOL;(*Error resume*)
    tTimeOut           : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

bInit: Initialisierung des Antriebes. Bei bInit TRUE wird im Antrieb die Betriebsart 2 'Positionierung' eingestellt.

bMode_DigitalSpeed: wird gesetzt, wenn der Antrieb bei der Initialisierung in die Betriebsart 'Drehzahl digital' versetzt werden soll.

iDigitalSpeed: Drehzahl in der Betriebsart 'Drehzahl digital'.

iVelocity: Der Parameter enthält die geforderte Fahrgeschwindigkeit für einen nachfolgenden Fahrauftrag z.B. µm/s.

iPosition: Zielposition.

iRunningMode: 0: drehzahl digital, 1: motiontask, 2: JogMode, 3: Calibration.

imotion_tasknumber: Fahrsatznummer. Mit diesem Eingang kann ein vorher im Speicher des Antriebes abgelegter Fahrsatz ausgewählt werden.

imotion_blocktype: Fahrsatzart (optional) Mit diesem Eingang können Eigenschaften eines Direktfahrauftrages verändert werden.

iJogModeBasicValue: Basisgeschwindigkeit für den Tippbetrieb, die tatsächliche Geschwindigkeit ergibt sich aus der Basisgeschwindigkeit und dem Faktor "v-Tippbetrieb" des Antriebes.

iCalVelo: Basisgeschwindigkeit der Referenzfahrt. Die Endgeschwindigkeit setzt sich aus der Basisgeschwindigkeit und dem Faktor "v-Tippbetrieb" des Antriebes zusammen.

bSetRefPoint: Setzen des Referenzpunktes.

bStart: Starten der Aktion je nach Zustand von iRunningMode.

bStop: Stoppen der Aktion je nach Zustand von iRunningMode.

bShortStop:

iSlaveAddress: Stationsadresse.

iFC310xDeviceId: Device-Id.

bErrorResume: Mit einer positiven Flanke an diesem boolschen Eingang wird ein "AX200X-Fehler" zurückgesetzt (kein TimeOut-Fehler).

tTimeOut: Maximale Zeit die bei der Ausführung des Befehls nicht überschritten werden soll.

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy        : BOOL;
    bError       : BOOL;(*Errorstatus of Servo*)
    iErrID       : DWORD;
    bTimeOutErr  : BOOL;
    bInitOK      : BOOL;(*Initialization OK*)
    iactPosition : DINT;(*actual Position SI-value*)
END_VAR
```

bBusy: Dieser Ausgang bleibt solange auf TRUE, bis der Baustein eine Befehlsanforderung ausführt. Während Busy = TRUE wird an den Eingängen kein neuer Befehl angenommen. Bitte beachten Sie, dass nicht die Ausführung des Dienstes, sondern nur dessen Annahme zeitlich überwacht wird.

bError: Dieser Ausgang zeigt den Fehlerstatus an.

iErrID: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Wird durch das Ausführen eines Befehls an den Eingängen auf 0 zurückgesetzt.

bTimeOutErr: TimeOut-Fehler.

bInitOK: Initialisierungszustand des Antriebes, bInit:= TRUE: Antrieb ist Initialisiert und in der Betriebsart 2 'Positionierung'.

iactPosition: Aktuelle Positionsanzeige im RunningMode 1: Motiontask.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO Hardware	Einzubindende SPS-Bibliotheken (Kategoriegruppe)
TwinCAT v3.1.0	PC oder CX (x86)	AX2000 Profibus box	Tc2_IoFunctions (IO)

3.4 Beckhoff Lightbus

3.4.1 IOF_LB_BreakLocationTest

Der Funktionsbaustein IOF_LB_BreakLocationTest führt ein Bruchstellentest in einem Beckhoff Lightbus-Lichtwellenleiterring durch und kann eventuelle Bruchstellen lokalisieren. Wurde während des Tests keine Bruchstelle im Ring erkannt, dann liefert die Ausgangsvariable **BOXNO** die aktuelle Anzahl der Lightbus-Module im Ring. Bei einer erkannten Bruchstelle vor dem NN-ten Modul vor dem Empfängereingang wird das Flag **BREAK** gesetzt und die Modulnummer über die Ausgangsvariable **BOXNO** ausgegeben. Liefert die **BOXNO**-Variable einen **0xFF**-Wert, dann liegt die Bruchstelle direkt vor dem Empfängereingang und kann nicht lokalisiert werden.

VAR_INPUT

```
VAR_INPUT
    NETID    : T_AmsNetId;
    DEVICEID : UDINT;
```

```
    START     : BOOL;
    TMOUT     : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

NETID: Hier kann die Netzwerkadresse des TwinCAT-Rechners angegeben werden, auf dem die Funktion ausgeführt werden soll (Typ: T_AmsNetID). Für den lokalen Rechner kann auch ein Leerstring angegeben werden.

DEVICEID: Über die DeviceId (Geräte-Id) wird das Gerät spezifiziert, auf dem die Funktion ausgeführt werden soll. Die Geräte-Ids werden während der Hardware-Konfiguration vom TwinCAT System festgelegt.

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

TMOUT: Gibt die Timeout-Zeit an, die bei der Ausführung des Kommandos nicht überschritten werden darf.

VAR_OUTPUT

```
VAR_OUTPUT
    BUSY      : BOOL;
    ERR       : BOOL;
    ERRID     : UDINT;
    BREAK     : BOOL;
    BOXNO     : WORD;
END_VAR
```

BUSY: Bei der Aktivierung des Funktionsbausteins wird dieser Ausgang gesetzt und bleibt gesetzt, bis eine Rückmeldung erfolgt.

ERR: Sollte ein Fehler bei der Übertragung des Kommandos erfolgen, dann wird dieser Ausgang gesetzt, nachdem der BUSY-Ausgang zurückgesetzt wurde.

ERRID: Liefert bei einem gesetzten ERR-Ausgang die ADS-Fehlernummer [▶ 154].

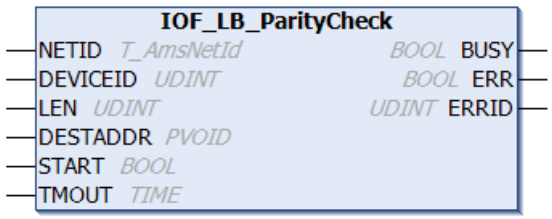
BREAK: Dieses Flag wird gesetzt, wenn in dem Lichwellenleiter-Ring eine Bruchstelle erkannt wurde.

BOXNO: Die Modulnummer vor dem Empfängereingang, vor dem die Bruchstelle erkannt wurde.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO Hard-ware	Einzubindende SPS-Bibliotheken (Kategorie-gruppe)
TwinCAT v3.1.0	Keine, diese Funktionalität wird zur Zeit von TwinCAT 3 nicht unterstützt!	Beckhoff Lightbus Master: C1220 ISA; FC200x PCI	Tc2_IoFunctions (IO)

3.4.2 IOF_LB_ParityCheck



Der Funktionsbaustein IOF_LB_ParityCheck liest die Parityfehler-Zähler der Beckhoff Lightbus-Module (z.B. BK2000). Im Gegensatz zu dem IOF_LB_ParityCheckWithReset [▶ 47] Funktionsbaustein werden die Zählerstände nicht zurückgesetzt. Für jedes Modul hält der Master einen 8-Bit Fehlerzähler bereit. Die Zähler arbeiten ohne Überlauf. Es können maximal **256**-Byte Daten und somit **256** Zähler gelesen werden.

Die Anzahl der zu lesenden Fehlerzähler wird durch die Eingangsvariablen: **LEN** und **DESTADDR** festgelegt. Existieren z.B. nur 5 Module im Ring, dann kann für den **DESTADDR**-Parameter eine Adresse auf einen 5 Byte großen Datenpuffer übergeben werden und dem **LEN**-Parameter der Wert 5.

VAR_INPUT

```
VAR_INPUT
    NETID     : T_AmsNetId;
    DEVICEID  : UDINT;
    LEN       : UDINT;
    DESTADDR  : PVOID;
    START     : BOOL;
    TMOUT     : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

NETID: Hier kann die Netzwerkadresse des TwinCAT-Rechners angegeben werden, auf dem die Funktion ausgeführt werden soll (Typ: T_AmsNetID). Für den lokalen Rechner kann auch ein Leerstring angegeben werden.

DEVICEID: Über die DeviceId (Geräte-Id) wird das Gerät spezifiziert, auf dem die Funktion ausgeführt werden soll. Die Geräte-Ids werden während der Hardware-Konfiguration vom TwinCAT System festgelegt.

LEN: Länge in Bytes der zu lesenden Daten.

DESTADDR: Die Adresse des Datenpuffers, in den die Paritydaten geschrieben werden sollen.

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

TMOUT: Gibt die Timeout-Zeit an, die bei der Ausführung des Kommandos nicht überschritten werden darf.

VAR_OUTPUT

```
VAR_OUTPUT
    BUSY    : BOOL;
    ERR     : BOOL;
    ERRID   : UDINT;
END_VAR
```

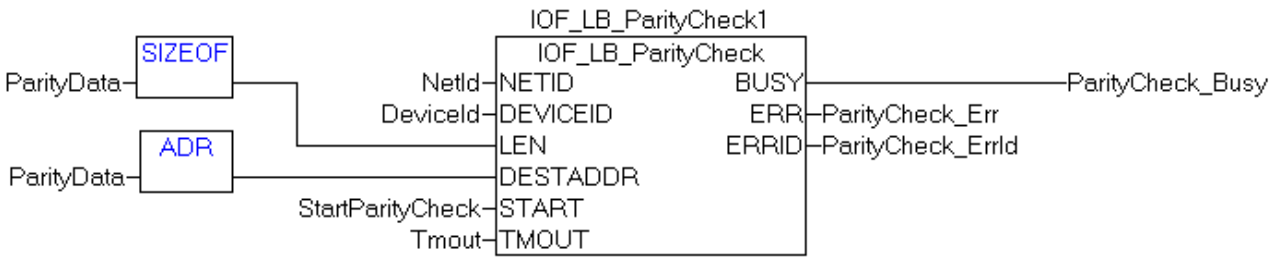
BUSY: Bei der Aktivierung des Funktionsbausteins wird dieser Ausgang gesetzt und bleibt gesetzt, bis eine Rückmeldung erfolgt.

ERR: Sollte ein Fehler bei der Übertragung des Kommandos erfolgen, dann wird dieser Ausgang gesetzt, nachdem der BUSY-Ausgang zurückgesetzt wurde.

ERRID: Liefert bei einem gesetzten ERR-Ausgang die ADS-Fehlernummer [▶ 154].

Beispiel:

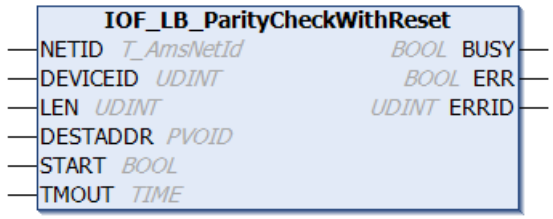
```
PROGRAM MAIN
VAR
    IOF_LB_ParityCheck1 : IOF_LB_ParityCheck;
    ParityData          : ARRAY[1..256] OF BYTE;
    StartParityCheck    : BOOL;
    ParityCheck_Busy    : BOOL;
    ParityCheck_Err     : BOOL;
    ParityCheck_ErrId   : UDINT;
END_VAR
```



Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO Hard-ware	Einzubindende SPS-Bibliotheken (Kategorie-gruppe)
TwinCAT v3.1.0	Keine, diese Funktionalität wird zur Zeit von TwinCAT 3 nicht unterstützt!	Beckhoff Lightbus Master: C1220 ISA; FC200x PCI	Tc2_IoFunctions (IO)

3.4.3 IOF_LB_ParityCheckWithReset



Der Funktionsbaustein IOF_LB_ParityCheckWithReset liest die Parityfehler-Zähler der Beckhoff Lightbus-Module (z.B. BK2000). Anschließend werden die Zähler zurückgesetzt. Für jedes Modul hält der Master einen 8-Bit Fehlerzähler bereit. Die Zähler arbeiten ohne Überlauf. Es können maximal **256**-Byte Daten und somit **256** Zähler gelesen werden. Die Anzahl der zu lesenden Fehlerzähler wird durch die Eingangsvariablen: **LEN** und **DESTADDR** festgelegt. Existieren z.B. nur 5 Module im Ring, dann kann für den **DESTADDR**-Parameter eine Adresse auf einen 5 Byte großen Datenpuffer übergeben werden und dem **LEN**-Parameter der Wert 5.

VAR_INPUT

```
VAR_INPUT
    NETID     : T_AmsNetId;
    DEVICEID  : UDINT;
    LEN       : UDINT;
    DESTADDR  : PVOID;
    START     : BOOL;
    TMOUT     : TIME;
END_VAR
```

NETID: Hier kann die Netzwerkadresse des TwinCAT-Rechners angegeben werden, auf dem die Funktion ausgeführt werden soll (Typ: T_AmsNetID). Für den lokalen Rechner kann auch ein Leerstring angegeben werden.

DEVICEID: Über die DeviceId (Geräte-Id) wird das Gerät spezifiziert, auf dem die Funktion ausgeführt werden soll. Die Geräte-Ids werden während der Hardware-Konfiguration vom TwinCAT-System festgelegt.

LEN: Länge in Bytes der zu lesenden Daten.

DESTADDR: Die Adresse des Datenpuffers, in den die Paritydaten geschrieben werden sollen.

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

TMOUT: Gibt die Timeout-Zeit an, die bei der Ausführung des Kommandos nicht überschritten werden darf.

VAR_OUTPUT

```
VAR_OUTPUT
    BUSY    : BOOL;
    ERR     : BOOL;
    ERRID   : UDINT;
END_VAR
```

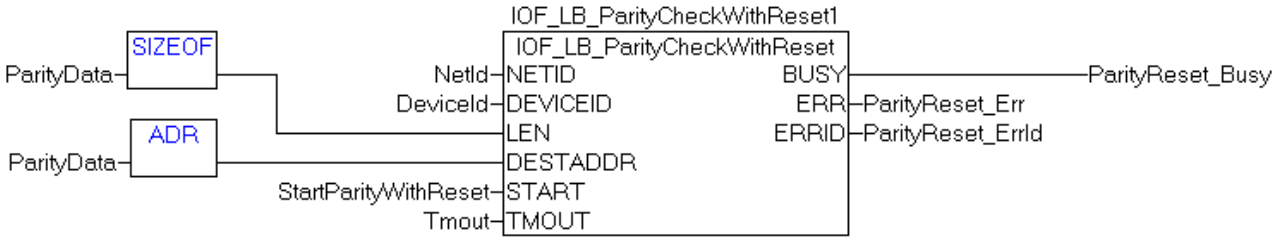
BUSY: Bei der Aktivierung des Funktionsbausteins wird dieser Ausgang gesetzt und bleibt gesetzt, bis eine Rückmeldung erfolgt.

ERR: Sollte ein Fehler bei der Übertragung des Kommandos erfolgen, dann wird dieser Ausgang gesetzt, nachdem der BUSY-Ausgang zurückgesetzt wurde.

ERRID: Liefert bei einem gesetzten ERR-Ausgang die ADS-Fehlernummer [▶ 154].

Beispiel:

```
PROGRAM MAIN
VAR
    IOF_LB_ParityCheckWithReset1 : IOF_LB_ParityCheckWithReset;
    ParityData                   : ARRAY[1..256] OF BYTE;
    StartParityWithReset         : BOOL;
    ParityReset_Busy             : BOOL;
    ParityReset_Err              : BOOL;
    ParityReset_ErrId            : UDINT;
END_VAR
```



Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO Hardware	Einzubindende SPS-Bibliotheken (Kategoriegruppe)
TwinCAT v3.1.0	Keine, diese Funktionalität wird zur Zeit von TwinCAT 3 nicht unterstützt!	Beckhoff Lightbus Master: C1220 ISA; FC200x PCI	Tc2_IoFunctions (IO)

3.5 Beckhoff USV (konfiguriert mit Windows USV service)

3.5.1 FB_GetUPSStatus

Voraussetzungen:

- Die Beckhoff USV Softwarekomponenten wurden installiert:
  - Windows 7, Windows Embedded Standard 7 und höher: Konfigurationsdialog unter „*Start->Programme->Beckhoff->UPS Software Components*“.
  - NT4, Win2K, WinXP, WinXP embedded: Zusätzlicher Reiter unter „*Systemsteuerung->Energieoptionen->Beckhoff UPS Configuration*“ oder „*Systemsteuerung->Energieoptionen->USV*“.
  - Beckhoff CE Geräte mit 24V USV-Unterstützung werden mit einem speziellen Beckhoff Battery Driver für Windows CE ausgeliefert. Der Treiber ist bei diesen Geräten in dem Standard CE Image enthalten.

- Die USV wurde aktiviert und konfiguriert. Weitere Informationen zur USV-Konfiguration finden Sie in der entsprechenden weiterführenden USV-Software und Gerätedokumentation.
  - Windows 7, Windows Embedded Standard 7 und höher: Konfigurationsdialog unter „*Start->Programme->Beckhoff->UPS Software Components*“.
  - NT4, Win2K, WinXP, WinXP embedded: Konfigurationsdialog unter „*Systemsteuerung->Energieoptionen->Beckhoff UPS Configuration*“.
  - Windows CE: Die USV-Funktion ist standardmäßig deaktiviert und muss über ein RegFile aktiviert werden. Neuere Images beinhalten ein Konfigurationsdialog unter *„Start->Systemsteuerung->BECKHOFF UPS Configuration*“.

Der Funktionsbaustein FB_GetUPSStatus liest aus der SPS den Status der USV-Hardware. Der Baustein wird Levelgetriggert, d.h. nur bei dem gesetzten *bEnable* -Eingang werden die Statusinformationen der USV zyklisch gelesen. Um dabei die Systemauslastung niedrig zu halten werden die Statusinformationen alle ~4,5s neu gelesen. Bei einem gesetzten *bValid*-Ausgang sind die zuletzt gelesenen Daten gültig. D.h. der letzte Lesezyklus wurde fehlerfrei durchgeführt. Beim Auftreten eines Fehlers wird der Lesezyklus wiederholt und der Fehler automatisch zurückgesetzt sobald die Fehlerursache behoben wurde (z.B. keine Kommunikation zur USV).

VAR_INPUT

```
VAR_INPUT
    sNetId : T_AmsNetId;
    nPort  : T_AmsPort;(* 0 = Windows UPS service / Windows Battery Driver *)
 bEnable   : BOOL;
END_VAR
```

sNetId: Hier kann ein String mit der Netzwerkadresse des TwinCAT-Rechners angegeben werden, dessen USV-Status gelesen werden soll (Typ: T_AmsNetID). Für den lokalen Rechner kann auch ein Leerstring angegeben werden.

nPort: Die ADS-Portnummer (Typ: T_AmsPort). Setzen Sie diesen Wert auf Null. Andere Portnummern sind für zukünftige Anwendungen reserviert.

bEnable: Bei einem gesetzten Eingang wird der USV-Status zyklisch gelesen.

VAR_OUTPUT

```
VAR_OUTPUT
    bValid     :BOOL;
    bError     :BOOL;
    nErrId     :UDINT;
    stStatus   :ST_UPSStatus;
END_VAR
```

bValid: Wenn dieser Ausgang gesetzt ist sind die Daten in der ST_UPSStatus-Struktur gültig (beim letzten Lesezyklus ist kein Fehler aufgetreten).

bError: Sollte ein Fehler bei der Ausführung der Funktion erfolgen, dann wird dieser Ausgang gesetzt.

nErrId: Liefert bei einem gesetzten *bError*-Ausgang die ADS-Fehlernummer [▶ 154] oder einen Befehlsspezifischen Fehlercode zurück (Tabelle).

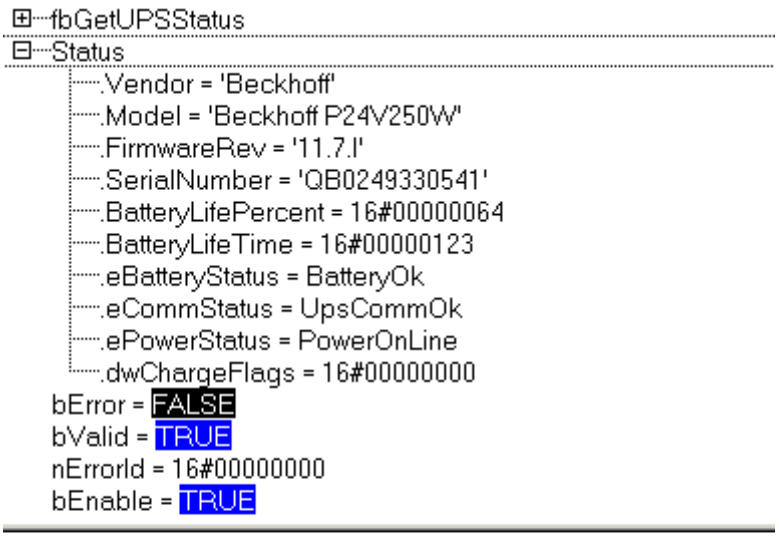
Fehlercodes	Fehlerbeschreibung
0x0000	Kein Fehler
0x8001	USV-Konfigurationsfehler. Möglicherweise ist die USV nicht richtig oder gar keine USV konfiguriert.
0x8002	Kommunikationsfehler. Die Kommunikation zu der USV wurde unterbrochen.
0x8003	Fehler beim Lesen der Statusdaten.

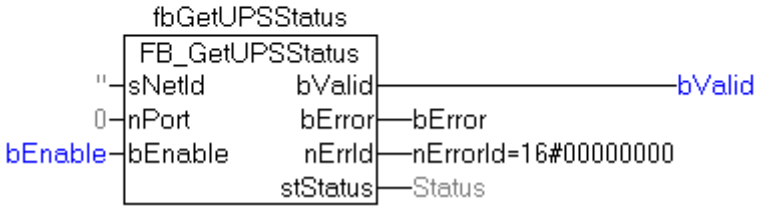
stStatus: Struktur mit den Statusinformationen der USV (Typ: ST_UPSStatus [▶ 140]).

Nicht alle USV-Geräte können alle Statusinformationen liefern. Einige Geräte können z.B. keine *BatteryLifeTime* oder keinen *BatteryReplace*-Status liefern.

Beispiel:

Online-Daten mit Statusinformationen einer USV:



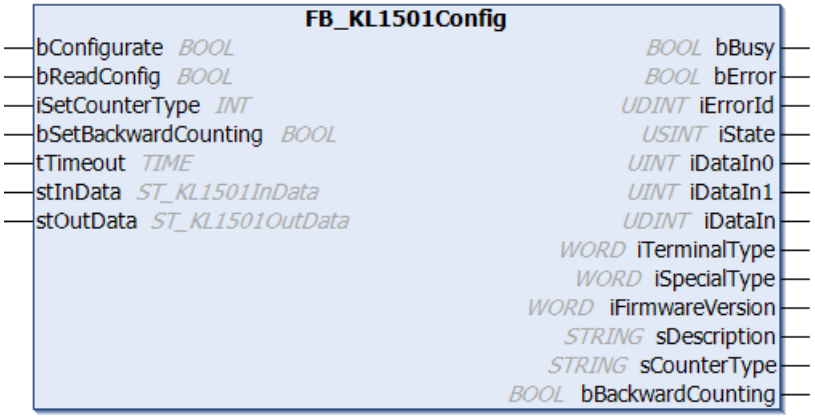


Voraussetzungen

Entwicklungsumgebung	Zielplattform	USV Hardware	Einzubindende SPS-Bibliotheken (Kategoriegruppe)
TwinCAT v3.1.0	PC oder CX (x86, x64, ARM)	• Beckhoff BAPI v1; • Beckhoff P24Vxxxx; • Beckhoff CP903x-Karte (PCI/ ISA); • Beckhoff CX2100-09x4 Modelle (z.B. CX2100-0904 oder CX2100-0914 + "Smart Battery" CX2900-0192); • Die mit Beckhoff Industrie-PC ausgelieferten APC-Geräte die das Smartprotokoll unterstützen und mit dem Windows USV-Dienst konfiguriert werden können;	Tc2_IoFunctions (IO)

3.6 Busklemmen-Konfiguration

3.6.1 FB_KL1501Config



Funktionsbaustein zur Parametrierung eine KL1501: 1-Kanal Zählerklemme.

Dieser Baustein berücksichtigt nicht das alternative Ausgabeformat, da sich bei Umstellung auf dieses Format das Prozessabbild verschiebt.

VAR_INPUT

```
VAR_INPUT
    bConfigurate         : BOOL;
    bReadConfig          : BOOL;
    iSetCounterType      : INT;
    bSetBackwardCounting : BOOL;
    tTimeout             : TIME;
END_VAR
```

bConfigurate: Eine steigende Flanke startet die Konfigurationssequenz. Zunächst werden die allgemeinen Klemmendaten "Klemmbezeichnung", "Sondervariante" und "Firmwarestand" ausgelesen. Danach werden die angegebenen Einstellungen in die entsprechenden Register geschrieben und abschließend zur Sicherheit und Information noch einmal ausgelesen. Die gelesenen Informationen werden an den Bausteinausgängen angezeigt. Während des Ablaufs dieser Sequenz steht der Ausgang *bBusy* auf TRUE und es wird kein weiterer Befehl, wie etwa *bReadConfig*, entgegen genommen.

bReadConfig: Eine steigende Flanke startet lediglich eine Lesesequenz. Es werden die allgemeinen Klemmendaten "Klemmbezeichnung", "Sondervariante" und "Firmwarestand" ausgelesen und im Anschluss daran die eingestellten Konfigurationsparameter. Die gelesenen Informationen werden an den Bausteinausgängen angezeigt. Während der Lesesequenz steht der Ausgang *bBusy* auf TRUE und es wird kein weiterer Befehl, wie etwa *bConfigurate*, entgegen genommen.

iSetCounterType: Eingabe des Zählertyps. Die Einstellung erfolgt nach unten aufgeführter Tabelle.

bSetBackwardCounting: Ein TRUE an diesem Eingang kehrt die Zählrichtung um.

tTimeout: Innerhalb der hier eingetragenen Zeit muss die Konfiguration der Klemme bzw. das Auslesen der Konfiguration abgeschlossen sein. Anderenfalls wird ein Fehler mit entsprechender Fehlernummer an den Ausgängen *bError* und *iErrorId* ausgegeben.

iSetCounterType	Zählertyp
0	32-Bit-Vorwärts/Rückwärts-Zähler
1	2 x 16-Bit Vorwärts-Zähler
2	32-bit Gated-Counter, Gate-Eingang Low sperrt den Zähler
3	32-bit Gated-Counter, Gate-Eingang High sperrt den Zähler

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy             : BOOL;
    bError            : BOOL;
    iErrorId          : UDINT;
    iState            : USINT;
    iDataIn0          : UINT;
    iDataIn1          : UINT;
    iDataIn           : UDINT;
    iTerminalType     : WORD;
    iSpecialType      : WORD;
    iFirmwareVersion  : WORD;
    sDescription      : STRING;
    sCounterType      : STRING;
    bBackwardCounting : BOOL;
END_VAR
```

bBusy: Solange eine Lese- oder Konfigurationssequenz abgearbeitet wird, steht dieser Ausgang auf TRUE.

bError: Dieser Ausgang wird auf TRUE geschaltet, wenn bei der Ausführung eines Befehls (Konfigurieren oder Lesen) ein Fehler aufgetreten ist. Der befehlsspezifische Fehlercode ist in *iErrorId* enthalten.

iErrorId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Wird durch das erneute Aktivieren des Bausteins über die Eingänge *bConfigurate* oder *bReadConfig* wieder auf 0 zurückgesetzt. Siehe Fehlercodes [▶ 154].

iState: Entspricht der Statusvariablen der Prozessdaten *stInData.iState*, siehe VAR_IN_OUT. Während der Befehlsausführung (*bBusy* = TRUE) jedoch ist dieser Ausgang auf 0 gesetzt. Damit eignet sich dieser Ausgang zur Statusbeurteilung im Normalbetrieb der Klemme. Störende Zustände während des Konfigurierens und Lesens durch die Registerkommunikation werden ausgeblendet.

iDataIn0: Entspricht der Datenvariablen der Prozessdaten *stInData.arrDataIn[0]*, siehe VAR_IN_OUT. Während der Befehlsausführung (*bBusy* = TRUE) jedoch behält dieser Ausgang den Wert, den er vor dem Befehlsaufruf innehatte. Damit eignet sich dieser Ausgang zur direkten Prozessdatenverarbeitung im Normalbetrieb der Klemme. Störende Zustände während des Konfigurierens und Lesens durch die Registerkommunikation werden ausgeblendet.

iDataIn1: Entspricht der Datenvariablen der Prozessdaten *stInData.arrDataIn[1]*, siehe VAR_IN_OUT. Während der Befehlsausführung (*bBusy* = TRUE) jedoch behält dieser Ausgang den Wert, den er vor dem Befehlsaufruf innehatte. Damit eignet sich dieser Ausgang zur direkten Prozessdatenverarbeitung im Normalbetrieb der Klemme. Störende Zustände während des Konfigurierens und Lesens durch die Registerkommunikation werden ausgeblendet.

iDataIn: Diese Variable vom Typ UDINT dient der besseren Auswertung, falls ein 32-bit Zähler angewählt ist. Sie setzt sich aus den beiden o.a. Variablen *iDataIn0* und *iDataIn1* (jeweils vom Typ UINT) zusammen. *iDataIn0* nimmt dabei den niederwertigen, *iDataIn1* den höherwertigen Teil ein.

iTerminalType: Inhalt des Registers 8 (Klemmenbezeichnung). Bei Anwendung mit der richtigen Klemme sollte der Inhalt 0x05DD (1501dez) sein.

iSpecialType: Inhalt des Registers 29 (Sondervariante).

iFirmwareVersion: Inhalt des Registers 9 (Firmware-Stand).

sDescription: Klemmenbezeichnung, Sondervariante und die Version der Firmware als String (z.B. 'Terminal KL1501-0000 / Firmware 1C').

sCounterType: Eingestellter Zählermodus als Klartext.

bBackwardCounting: TRUE: Die Zählrichtung wurde umgekehrt.

VAR_IN_OUT

```
VAR_IN_OUT
    stInData  : ST_KL1501InData;
    stOutData : ST_KL1501OutData;
END_VAR
```

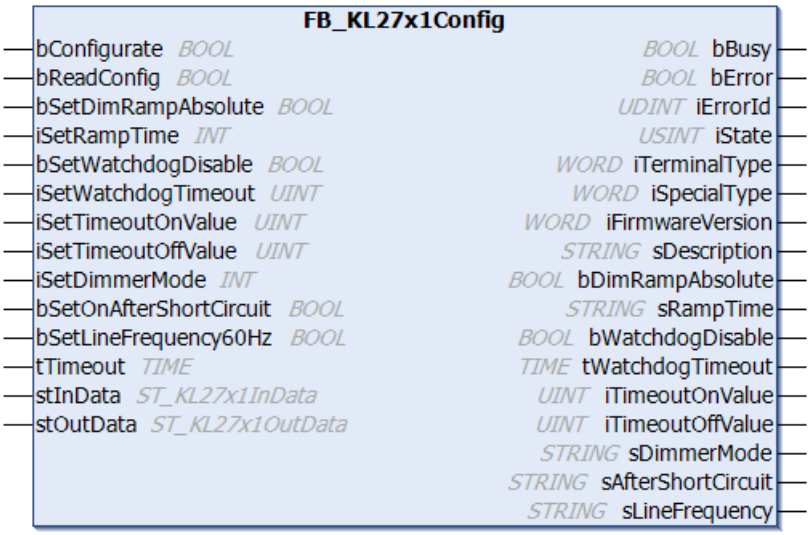
stInData: Verweis auf die Struktur des Eingangsprozessabbildes (Typ: ST_KL1501InData [▶ 142]).

stOutData: Verweis auf die Struktur des Ausgangsprozessabbildes (Typ: ST_KL1501OutData [▶ 143]).

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO-Hardware	Einzubindende SPS-Bibliotheken
TwinCAT v3.1.4018.26	PC/CX	KL1501	Tc2_IoFunctions ab v3.3.5.0

3.6.2 FB_KL27x1Config



Funktionsbaustein zur Parametrierung einer KL2751 / KL2761: 1-Kanal Dimmerklemme.

VAR_INPUT

```
VAR_INPUT
    bConfigurate            : BOOL;
    bReadConfig             : BOOL;
    iSetSensorType          : INT;
    bSetDimRampAbsolute     : BOOL;
    iSetRampTime            : INT;
    bSetWatchdogDisable     : BOOL;
    iSetWatchdogTimeout     : UINT;
    iSetTimeoutOnValue      : UINT;
    iSetTimeoutOffValue     : UINT;
    iSetDimmerMode          : INT;
    bSetOnAfterShortCircuit : BOOL;
    bSetLineFrequency60Hz   : BOOL;
    tTimeout                : TIME;
END_VAR
```

bConfigurate: Eine steigende Flanke startet die Konfigurationssequenz. Zunächst werden die allgemeinen Klemmendaten "Klemmbezeichnung", "Sondervariante" und "Firmwarestand" ausgelesen. Danach werden die angegebenen Einstellungen in die entsprechenden Register geschrieben und abschließend zur Sicherheit und Information noch einmal ausgelesen. Die gelesenen Informationen werden an den Bausteinausgängen angezeigt. Während des Ablaufs dieser Sequenz steht der Ausgang *bBusy* auf TRUE und es wird kein weiterer Befehl, wie etwa *bReadConfig*, entgegen genommen.

bReadConfig: Eine steigende Flanke startet lediglich eine Lesesequenz. Es werden die allgemeinen Klemmendaten "Klemmbezeichnung", "Sondervariante" und "Firmwarestand" ausgelesen und im Anschluss daran die eingestellten Konfigurationsparameter. Die gelesenen Informationen werden an den Bausteinausgängen angezeigt. Während der Lesesequenz steht der Ausgang *bBusy* auf TRUE und es wird kein weiterer Befehl, wie etwa *bConfigurate*, entgegen genommen.

bSetDimRampAbsolute: FALSE: Die eingestellte Rampenzeit *iSetRampTime* bezieht sich auf den kompletten Datenbereich (0 - 32767). Je kleiner der Sprung, desto kürzer die Rampenzeit. TRUE: Jeder Schaltschritt, egal wie groß, benötigt dieselbe Zeit, die unter *iSetRampTime* eingetragen ist.

iSetRampTime: Eingabe der Rampenzeit. Die Einstellung erfolgt nach unten aufgeführter Tabelle.

bSetWatchdogDisable: Der interne Watchdog wird deaktiviert.

iSetWatchdogTimeout: Einstellung der Watchdog-Zeit als Vielfaches von 10ms.

iSetTimeoutOnValue: Dieser Eingang legt den Lichtwert fest, der bei einem Feldbusfehler und aktuellen Prozessdaten > 0 ausgegeben wird.

iSetTimeoutOffValue: Dieser Eingang legt den Lichtwert fest, der bei einem Feldbusfehler und aktuellen Prozessdaten = 0 ausgegeben wird.

iSetDimmerMode: An diesem Eingang ist der Dimmermodus einzustellen. Die Einstellung erfolgt nach unten aufgeführter Tabelle.

bSetOnAfterShortCircuit: FALSE: Nach einem Kurzschluss bleibt das Licht ausgeschaltet. TRUE: Das Licht wird nach einem Kurzschluss wieder eingeschaltet.

bSetLineFrequency60Hz: FALSE: Netzfrequenz = 50 Hz. TRUE: Netzfrequenz = 60 Hz.

tTimeout: Innerhalb der hier eingetragenen Zeit muss die Konfiguration der Klemme bzw. das Auslesen der Konfiguration abgeschlossen sein. Anderenfalls wird ein Fehler mit entsprechender Fehlernummer an den Ausgängen *bError* und *iErrorId* ausgegeben.

iSetRampTime	Element
0	50 ms
1	100 ms
2	200 ms
3	500 ms
4	1 s
5	2 s
6	5 s
7	10 s

iSetDimmerMode	Element
0	Automatische Erkennung
1	Phasenabschnitt
2	Phasenanschnitt
3	Gleichrichterbetrieb, positiv (positive Halbwelle mit Phasenanschnitt)
4	Gleichrichterbetrieb, negativ (negative Halbwelle mit Phasenanschnitt)

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy            : BOOL;
    bError           : BOOL;
    iErrorId         : UDINT;
    iState           : USINT;
    iDataIn          : INT;
    iTerminalType    : WORD;
    iSpecialType     : WORD;
    iFirmwareVersion : WORD;
    sDescription     : STRING;
    sSensorType      : STRING;
END_VAR
```

bBusy: Solange eine Lese- oder Konfigurationssequenz abgearbeitet wird, steht dieser Ausgang auf TRUE.

bError: Dieser Ausgang wird auf TRUE geschaltet, wenn bei der Ausführung eines Befehls (Konfigurieren oder Lesen) ein Fehler aufgetreten ist. Der befehlsspezifische Fehlercode ist in *iErrorId* enthalten.

iErrorId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Wird durch das erneute Aktivieren des Bausteins über die Eingänge *bConfigurate* oder *bReadConfig* wieder auf 0 zurückgesetzt. Siehe Fehlercodes [▶ 154].

iState: Entspricht der Statusvariable der Prozessdaten *stInData.iState*, siehe VAR_IN_OUT. Während der Befehlsausführung (*bBusy* = TRUE) jedoch ist dieser Ausgang auf 0 gesetzt. Damit eignet sich dieser Ausgang zur Statusbeurteilung im Normalbetrieb der Klemme. Störende Zustände während des Konfigurierens und Lesens durch die Registerkommunikation werden ausgeblendet.

iDataIn: Entspricht der Datenvariable der Prozessdaten *stInData.iDataIn*, siehe VAR_IN_OUT. Während der Befehlsausführung (*bBusy* = TRUE) jedoch behält dieser Ausgang den Wert, den er vor dem Befehlsaufruf innehatte. Damit eignet sich dieser Ausgang zur direkten Prozessdatenverarbeitung im Normalbetrieb der Klemme. Störende Zustände während des Konfigurierens und Lesens durch die Registerkommunikation werden ausgeblendet.

iTerminalType: Inhalt des Registers 8 (Klemmenbezeichnung). Bei Anwendung mit der richtigen Klemme sollte der Inhalt 0x0ABF (2751dez) oder 0x0AC9 (2761dez)sein.

iSpecialType: Inhalt des Registers 29 (Sondervariante).

iFirmwareVersion: Inhalt des Registers 9 (Firmware-Stand).

sDescription: Klemmenbezeichnung, Sondervariante und die Version der Firmware als String (z.B. 'Terminal KL27x1-0000 / Firmware 1C').

bDimRampAbsolute: TRUE: Dimmrampe ist als absolut eingestellt, d.h. jeder Schaltschritt benötigt dieselbe Rampenzeit, welche unter *iSetRampTime* eingestellt ist.

sRampTime: Eingestellte Rampenzeit als Klartext.

bWatchdogDisable: TRUE: Watchdog ist deaktiviert.

tWatchdogTimeout: Eingestellte Watchdogzeit.

iTimeoutOnValue: Eingestellter Lichtwert, der bei einem Feldbusfehler und aktuellen Prozessdaten > 0 ausgegeben wird.

iTimeoutOffValue: Eingestellter Lichtwert, der bei einem Feldbusfehler und aktuellen Prozessdaten = 0 ausgegeben wird.

sDimmerMode: Eingestellter Dimmermodus als Klartext.

sAfterShortCircuit: Eingestelltes Verhalten nach Kurzschluss als Klartext.

sLineFrequency: Eingestellte Netzfrequenz als Klartext.

VAR_IN_OUT

```
VAR_IN_OUT
    stInData  : ST_KL27x1InData;
    stOutData : ST_KL27x1OutData;
END_VAR
```

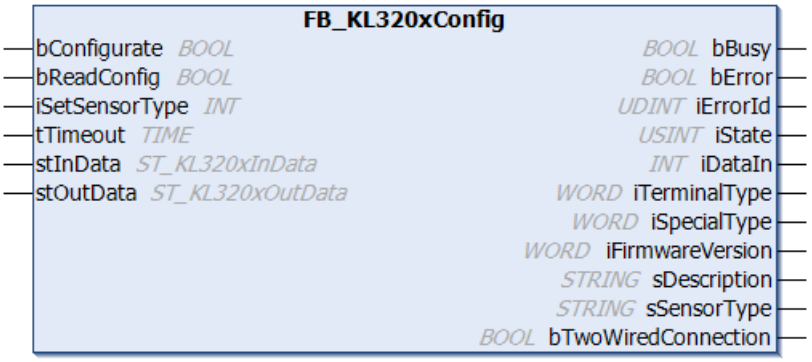
stInData: Verweis auf die Struktur des Eingangsprozessabbildes (Typ: ST_KL27x1InData [▶ 143]).

stOutData: Verweis auf die Struktur des Ausgangsprozessabbildes (Typ: ST_KL27x1OutData [▶ 143]).

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO-Hardware	Einzubindende SPS-Bibliotheken
TwinCAT v3.1.4018.26	PC/CX	KL2751, KL2761	Tc2_IoFunctions ab v3.3.5.0

3.6.3 FB_KL320xConfig



Funktionsbaustein zur Parametrierung einer KL3201, KL3202 oder KL3204: Eingangsklemme für Widerstandssensoren.

> **i** Der Baustein parametriert nur einen Klemmenkanal. Zur Parametrierung aller Kanäle ist die entsprechende Anzahl von Bausteinen zu instanziieren. Eine Mischkonfiguration (z.B. unterschiedliche Sensortypen) ist möglich.

VAR_INPUT

```
VAR_INPUT
    bConfigurate    : BOOL;
    bReadConfig     : BOOL;
    iSetSensorType  : INT;
    tTimeout        : TIME;
END_VAR
```

bConfigurate: Eine steigende Flanke startet die Konfigurationssequenz. Zunächst werden die allgemeinen Klemmendaten "Klemmbezeichnung", "Sondervariante" und "Firmwarestand" ausgelesen. Danach werden die angegebenen Einstellungen in die entsprechenden Register geschrieben und abschließend zur Sicherheit und Information noch einmal ausgelesen. Die gelesenen Informationen werden an den Bausteinausgängen angezeigt. Während des Ablaufs dieser Sequenz steht der Ausgang *bBusy* auf TRUE und es wird kein weiterer Befehl, wie etwa *bReadConfig*, entgegen genommen.

bReadConfig: Eine steigende Flanke startet lediglich eine Lesesequenz. Es werden die allgemeinen Klemmendaten "Klemmbezeichnung", "Sondervariante" und "Firmwarestand" ausgelesen und im Anschluss daran die eingestellten Konfigurationsparameter. Die gelesenen Informationen werden an den Bausteinausgängen angezeigt. Während der Lesesequenz steht der Ausgang *bBusy* auf TRUE und es wird kein weiterer Befehl, wie etwa *bConfigurate*, entgegen genommen.

iSetSensorType: An diesem Eingang ist der verwendete Sensor einzustellen. Die Einstellung erfolgt nach unten aufgeführter Tabelle.

tTimeout: Innerhalb der hier eingetragenen Zeit muss die Konfiguration der Klemme bzw. das Auslesen der Konfiguration abgeschlossen sein. Anderenfalls wird ein Fehler mit entsprechender Fehlernummer an den Ausgängen *bError* und *iErrorId* ausgegeben.

iSetSensorType	Element
0	PT100
1	NI100
2	PT1000
3	PT500
4	PT200
5	NI1000
6	NI120
7	Ausgabe 10,0 Ω - 5000,0 Ω
8	Ausgabe 10,0 Ω - 1200,0 Ω
9	PT1000 - Zwei-Leiter-Anschluss - **nicht zulässig bei Verwendung einer KL3204!**

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy               : BOOL;
    bError              : BOOL;
    iErrorId            : UDINT;
    iState              : USINT;
    iDataIn             : INT;
    iTerminalType       : WORD;
    iSpecialType        : WORD;
    iFirmwareVersion    : WORD;
    sDescription        : STRING;
    sSensorType         : STRING;
    bTwoWiredConnection : BOOL;
END_VAR
```

bBusy: Solange eine Lese- oder Konfigurationssequenz abgearbeitet wird, steht dieser Ausgang auf TRUE.

bError: Dieser Ausgang wird auf TRUE geschaltet, wenn bei der Ausführung eines Befehls (Konfigurieren oder Lesen) ein Fehler aufgetreten ist. Der befehlsspezifische Fehlercode ist in *iErrorId* enthalten.

iErrorId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Wird durch das erneute Aktivieren des Bausteins über die Eingänge *bConfigurate* oder *bReadConfig* wieder auf 0 zurückgesetzt. Siehe Fehlercodes [▶ 154].

iState: Entspricht der Statusvariable der Prozessdaten *stInData.iState*, siehe VAR_IN_OUT. Während der Befehlsausführung (*bBusy* = TRUE) jedoch ist dieser Ausgang auf 0 gesetzt. Damit eignet sich dieser Ausgang zur Statusbeurteilung im Normalbetrieb der Klemme. Störende Zustände während des Konfigurierens und Lesens durch die Registerkommunikation werden ausgeblendet.

iDataIn: Entspricht der Datenvariable der Prozessdaten *stInData.iDataIn*, siehe VAR_IN_OUT. Während der Befehlsausführung (*bBusy* = TRUE) jedoch behält dieser Ausgang den Wert, den er vor dem Befehlsaufruf innehatte. Damit eignet sich dieser Ausgang zur direkten Prozessdatenverarbeitung im Normalbetrieb der Klemme. Störende Zustände während des Konfigurierens und Lesens durch die Registerkommunikation werden ausgeblendet.

iTerminalType: Inhalt des Registers 8 (Klemmenbezeichnung). Der Registerinhalt muss der verwendeten Klemme entsprechen: 0xC81 für KL3201, 0xC82 für KL3202 und 0xC84 für KL3204.

iSpecialType: Inhalt des Registers 29 (Sondervariante).

iFirmwareVersion: Inhalt des Registers 9 (Firmware-Stand).

sDescription: Klemmenbezeichnung, Sondervariante und die Version der Firmware als String (z.B. 'Terminal KL320x-0010 / Firmware 1C').

sSensorType: Eingestellter Sensortyp als Klartext.

bTwoWiredConnection: Sensortyp ist im Zweileiter-Anschluss parametriert.

VAR_IN_OUT

```
VAR_IN_OUT
    stInData  : ST_KL320xInData;
    stOutData : ST_KL320xOutData;
END_VAR
```

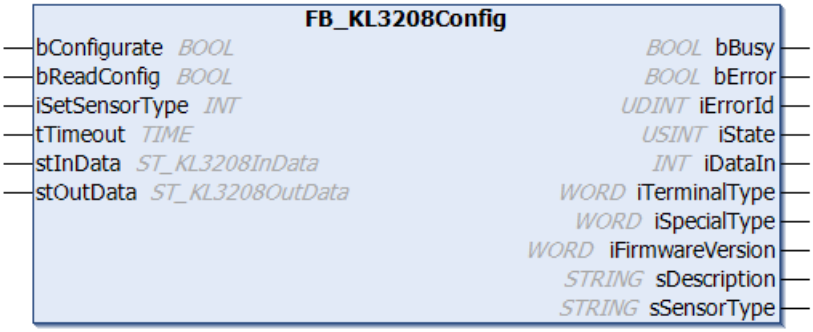
stInData: Verweis auf die Struktur des Eingangsprozessabbildes (Typ: ST_KL320xInData [▶ 144]).

stOutData: Verweis auf die Struktur des Ausgangsprozessabbildes (Typ: ST_KL320xOutData [▶ 144]).

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO-Hardware	Einzubindende SPS-Bibliotheken
TwinCAT v3.1.4018.26	PC/CX	KL3201, KL3202, KL3204	Tc2_IoFunctions ab v3.3.5.0

3.6.4 FB_KL3208Config



Funktionsbaustein zur Parametrierung einer KL3208-0010: 8-Kanal Eingangsklemme für Widerstandssensoren.

> Der Baustein parametriert nur einen Klemmenkanal. Zur Parametrierung aller Kanäle ist die entsprechende Anzahl von Bausteinen zu instanziieren. Eine Mischkonfiguration (z.B. unterschiedliche Sensortypen) ist möglich.

VAR_INPUT

```
VAR_INPUT
    bConfigurate   : BOOL;
    bReadConfig    : BOOL;
    iSetSensorType : INT;
    tTimeout       : TIME;
END_VAR
```

bConfigurate: Eine steigende Flanke startet die Konfigurationssequenz. Zunächst werden die allgemeinen Klemmendaten "Klemmbezeichnung", "Sondervariante" und "Firmwarestand" ausgelesen. Danach werden die angegebenen Einstellungen in die entsprechenden Register geschrieben und abschließend zur Sicherheit und Information noch einmal ausgelesen. Die gelesenen Informationen werden an den Bausteinausgängen angezeigt. Während des Ablaufs dieser Sequenz steht der Ausgang *bBusy* auf TRUE und es wird kein weiterer Befehl, wie etwa *bReadConfig*, entgegen genommen.

bReadConfig: Eine steigende Flanke startet lediglich eine Lesesequenz. Es werden die allgemeinen Klemmendaten "Klemmbezeichnung", "Sondervariante" und "Firmwarestand" ausgelesen und im Anschluss daran die eingestellten Konfigurationsparameter. Die gelesenen Informationen werden an den Bausteinausgängen angezeigt. Während der Lesesequenz steht der Ausgang *bBusy* auf TRUE und es wird kein weiterer Befehl, wie etwa *bConfigurate*, entgegen genommen.

iSetSensorType: An diesem Eingang ist der verwendete Sensor einzustellen. Die Einstellung erfolgt nach unten aufgeführter Tabelle.

tTimeout: Innerhalb der hier eingetragenen Zeit muss die Konfiguration der Klemme bzw. das Auslesen der Konfiguration abgeschlossen sein. Anderenfalls wird ein Fehler mit entsprechender Fehlernummer an den Ausgängen *bError* und *iErrorId* ausgegeben.

iSetSensorType	Element
0	PT1000
1	NI1000
2	RSNI1000 (NI1000 nach Landis&Staefa-Charakteristik: 1000 Ω bei 0 °C und 1500 Ω bei 100 °C.)
3	NTC1K8
4	NTC1K8_TK
5	NTC2K2
6	NTC3K
7	NTC5K
8	NTC10K
9	NTC10KPRE
10	NTC10K_3204
11	NTC10KTYP2
12	NTC10KTYP3
13	NTC10KDALE
14	NTC10K3A221
15	NTC20K
16	Poti, Auflösung 0,1 Ω
17	Poti, Auflösung 1 Ω
18	NTC100K

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy            : BOOL;
    bError           : BOOL;
    iErrorId         : UDINT;
    iState           : USINT;
    iDataIn          : INT;
    iTerminalType    : WORD;
    iSpecialType     : WORD;
    iFirmwareVersion : WORD;
    sDescription     : STRING;
    sSensorType      : STRING;
END_VAR
```

bBusy: Solange eine Lese- oder Konfigurationssequenz abgearbeitet wird, steht dieser Ausgang auf TRUE.

bError: Dieser Ausgang wird auf TRUE geschaltet, wenn bei der Ausführung eines Befehls (Konfigurieren oder Lesen) ein Fehler aufgetreten ist. Der befehlsspezifische Fehlercode ist in *iErrorId* enthalten.

iErrorId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Wird durch das erneute Aktivieren des Bausteins über die Eingänge *bConfigurate* oder *bReadConfig* wieder auf 0 zurückgesetzt. Siehe Fehlercodes [▶ 154].

iState: Entspricht der Statusvariable der Prozessdaten *stInData.iState*, siehe VAR_IN_OUT. Während der Befehlsausführung (*bBusy* = TRUE) jedoch ist dieser Ausgang auf 0 gesetzt. Damit eignet sich dieser Ausgang zur Statusbeurteilung im Normalbetrieb der Klemme. Störende Zustände während des Konfigurierens und Lesens durch die Registerkommunikation werden ausgeblendet.

iDataIn: Entspricht der Datenvariable der Prozessdaten *stInData.iDataIn*, siehe VAR_IN_OUT. Während der Befehlsausführung (*bBusy* = TRUE) jedoch behält dieser Ausgang den Wert, den er vor dem Befehlsaufruf innehatte. Damit eignet sich dieser Ausgang zur direkten Prozessdatenverarbeitung im Normalbetrieb der Klemme. Störende Zustände während des Konfigurierens und Lesens durch die Registerkommunikation werden ausgeblendet.

iTerminalType: Inhalt des Registers 8 (Klemmenbezeichnung). Bei Anwendung mit der richtigen Klemme sollte der Inhalt 0x0C88 (3208dez) sein.

iSpecialType: Inhalt des Registers 29 (Sondervariante).

iFirmwareVersion: Inhalt des Registers 9 (Firmware-Stand).

sDescription: Klemmenbezeichnung, Sondervariante und die Version der Firmware als String (z.B. 'Terminal KL3208-0010 / Firmware 1C').

sSensorType: Eingestellter Sensortyp als Klartext.

VAR_IN_OUT

```
VAR_IN_OUT
    stInData  : ST_KL3208InData;
    stOutData : ST_KL3208OutData;
END_VAR
```

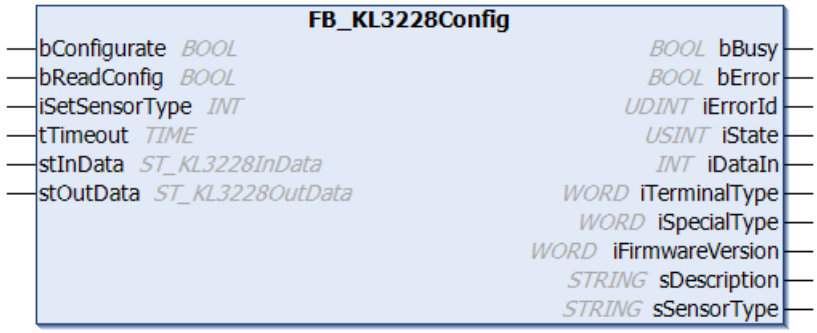
stInData: Verweis auf die Struktur des Eingangsprozessabbildes (Typ: ST_KL3208InData [▶ 144]).

stOutData: Verweis auf die Struktur des Ausgangsprozessabbildes (Typ: ST_KL3208OutData [▶ 145]).

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO-Hardware	Einzubindende SPS-Bibliotheken
TwinCAT v3.1.4018.26	PC/CX	KL3208	Tc2_IoFunctions ab v3.3.5.0

3.6.5 FB_KL3228Config



Funktionsbaustein zur Parametrierung einer KL3228: 8-Kanal Eingangsklemme für Widerstandssensoren.

Der Baustein parametriert nur einen Klemmenkanal. Zur Parametrierung aller Kanäle ist die entsprechende Anzahl von Bausteinen zu instanziieren. Eine Mischkonfiguration (z.B. unterschiedliche Sensortypen) ist möglich.

VAR_INPUT

```
VAR_INPUT
    bConfigurate    : BOOL;
    bReadConfig     : BOOL;
    iSetSensorType  : INT;
    tTimeout        : TIME;
END_VAR
```

bConfigurate: Eine steigende Flanke startet die Konfigurationssequenz. Zunächst werden die allgemeinen Klemmendaten "Klemmbezeichnung", "Sondervariante" und "Firmwarestand" ausgelesen. Danach werden die angegebenen Einstellungen in die entsprechenden Register geschrieben und abschließend zur Sicherheit und Information noch einmal ausgelesen. Die gelesenen Informationen werden an den Bausteinausgängen angezeigt. Während des Ablaufs dieser Sequenz steht der Ausgang *bBusy* auf TRUE und es wird kein weiterer Befehl, wie etwa *bReadConfig*, entgegen genommen.

bReadConfig: Eine steigende Flanke startet lediglich eine Lesesequenz. Es werden die allgemeinen Klemmendaten "Klemmbezeichnung", "Sondervariante" und "Firmwarestand" ausgelesen und im Anschluss daran die eingestellten Konfigurationsparameter. Die gelesenen Informationen werden an den Bausteinausgängen angezeigt. Während der Lesesequenz steht der Ausgang *bBusy* auf TRUE und es wird kein weiterer Befehl, wie etwa *bConfigurate*, entgegen genommen.

iSetSensorType: An diesem Eingang ist der verwendete Sensor einzustellen. Die Einstellung erfolgt nach unten aufgeführter Tabelle.

tTimeout: Innerhalb der hier eingetragenen Zeit muss die Konfiguration der Klemme bzw. das Auslesen der Konfiguration abgeschlossen sein. Anderenfalls wird ein Fehler mit entsprechender Fehlernummer an den Ausgängen *bError* und *iErrorId* ausgegeben.

iSetSensorType	Element
0	PT1000
1	NI1000
2	RSNI1000 (NI1000 nach Landis&Staefa-Charakteristik: 1000 Ω bei 0 °C und 1500 Ω bei 100 °C.)

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy            : BOOL;
    bError           : BOOL;
    iErrorId         : UDINT;
    iState           : USINT;
    iDataIn          : INT;
    iTerminalType    : WORD;
    iSpecialType     : WORD;
    iFirmwareVersion : WORD;
    sDescription     : STRING;
    sSensorType      : STRING;
END_VAR
```

bBusy: Solange eine Lese- oder Konfigurationssequenz abgearbeitet wird, steht dieser Ausgang auf TRUE.

bError: Dieser Ausgang wird auf TRUE geschaltet, wenn bei der Ausführung eines Befehls (Konfigurieren oder Lesen) ein Fehler aufgetreten ist. Der befehlsspezifische Fehlercode ist in *iErrorId* enthalten.

iErrorId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Wird durch das erneute Aktivieren des Bausteins über die Eingänge *bConfigurate* oder *bReadConfig* wieder auf 0 zurückgesetzt. Siehe Fehlercodes [▶ 154].

iState: Entspricht der Statusvariable der Prozessdaten *stInData.iState*, siehe VAR_IN_OUT. Während der Befehlsausführung (*bBusy* = TRUE) jedoch ist dieser Ausgang auf 0 gesetzt. Damit eignet sich dieser Ausgang zur Statusbeurteilung im Normalbetrieb der Klemme. Störende Zustände während des Konfigurierens und Lesens durch die Registerkommunikation werden ausgeblendet.

iDataIn: Entspricht der Datenvariable der Prozessdaten *stInData.iDataIn*, siehe VAR_IN_OUT. Während der Befehlsausführung (*bBusy* = TRUE) jedoch behält dieser Ausgang den Wert, den er vor dem Befehlsaufruf innehatte. Damit eignet sich dieser Ausgang zur direkten Prozessdatenverarbeitung im Normalbetrieb der Klemme. Störende Zustände während des Konfigurierens und Lesens durch die Registerkommunikation werden ausgeblendet.

iTerminalType: Inhalt des Registers 8 (Klemmenbezeichnung). Bei Anwendung mit der richtigen Klemme sollte der Inhalt 0x0C9C (3228dez) sein.

iSpecialType: Inhalt des Registers 29 (Sondervariante).

iFirmwareVersion: Inhalt des Registers 9 (Firmware-Stand).

sDescription: Klemmenbezeichnung, Sondervariante und die Version der Firmware als String (z.B. 'Terminal KL3228-0000 / Firmware 1C').

sSensorType: Eingestellter Sensortyp als Klartext.

VAR_IN_OUT

```
VAR_IN_OUT
    stInData  : ST_KL3228InData;
    stOutData : ST_KL3228OutData;
END_VAR
```

stInData: Verweis auf die Struktur des Eingangsprozessabbildes (Typ: ST_KL3228InData [▶ 145]).

stOutData: Verweis auf die Struktur des Ausgangsprozessabbildes (Typ: ST_KL3228OutData [▶ 145]).

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO-Hardware	Einzubindende SPS-Bibliotheken
TwinCAT v3.1.4018.26	PC/CX	KL3228	Tc2_IoFunctions ab v3.3.5.0

3.7 CANopen

3.7.1 IOF_CAN_Layer2Command

Der Funktionsbaustein IOF_CAN_Layer2Command sendet ein 10 Byte langes Kommando an die Schicht 2 eines CAN-Masters.

VAR_INPUT

```
VAR_INPUT
    NETID     : T_AmsNetId;
    DEVICEID  : UDINT;
    LEN       : UDINT;
    SRCADDR   : PVOID;
    START     : BOOL;
    TMOUT     : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

NETID: Hier kann die Netzwerkadresse des TwinCAT-Rechners angegeben werden, auf dem die Funktion ausgeführt werden soll (Typ: T_AmsNetID). Für den lokalen Rechner kann auch ein Leerstring angegeben werden.

DEVICEID: Über die DeviceId (Geräte-Id) wird das Gerät (CAN-Master) spezifiziert, auf dem die Funktion ausgeführt werden soll. Die Geräte-Ids werden während der Hardware-Konfiguration von TwinCAT-System festgelegt.

LEN: Die Bytelänge des Layer 2 Kommandos.

SRCADDR: Die Adresse von dem ersten Datenwort des CAN-Layer 2 Kommandos.

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

TMOUT: Gibt die Timeout-Zeit an, die bei der Ausführung des Kommandos nicht überschritten werden darf.

VAR_OUTPUT

```
VAR_OUTPUT
    BUSY    : BOOL;
    ERR     : BOOL;
    ERRID   : UDINT;
END_VAR
```

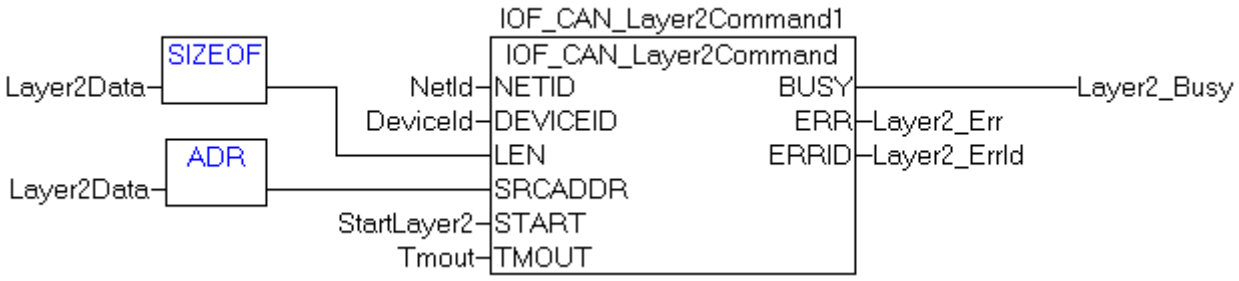
BUSY: Bei der Aktivierung des Funktionsbausteins wird dieser Ausgang gesetzt und bleibt gesetzt, bis eine Rückmeldung erfolgt.

ERR: Sollte ein Fehler bei der Übertragung des Kommandos erfolgen, dann wird dieser Ausgang gesetzt, nachdem der BUSY-Ausgang zurückgesetzt wurde.

ERRID: Liefert bei einem gesetzten ERR-Ausgang die ADS-Fehlernummer [▶ 154].

Beispiel:

```
PROGRAM MAIN
VAR
    IOF_CAN_Layer2Command1 : IOF_CAN_Layer2Command;
    Layer2Data             : ARRAY[1..5] OF WORD;
    StartLayer2            : BOOL;
    Layer2_Busy            : BOOL;
    Layer2_Err             : BOOL;
    Layer2_ErrId           : UDINT;
END_VAR
```



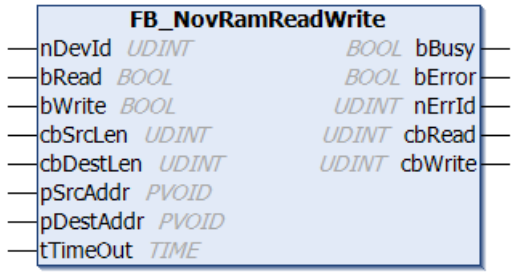
Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO Hardware	Einzubindende SPS-Bibliotheken (Kategoriegruppe)
TwinCAT v3.1.0	Keine, diese Funktionalität wird zur Zeit von TwinCAT 3 nicht unterstützt!	HILSCHER CIF3xx COM master card	Tc2_IoFunctions (IO)

3.8 NOV/DP-RAM

3.8.1 FB_NovRamReadWrite

Die folgende Beschreibung bezieht sich auf die TwinCAT Version 2.8. Ab der TwinCAT Version 2.9 [Build 927] wird kein Funktionsbaustein zum Schreiben bzw. Lesen von zu sichernden SPS-Daten ins NOVRAM mehr benötigt.



Der Funktionsbaustein *FB_NovRamReadWrite* greift aus einem SPS-Programm auf das NOV-RAM der FCxxxx-000**2** Feldbuskarten zu. Bei einer steigenden Flanke am *bRead* oder *bWrite*-Eingang wird der Funktionsbaustein aktiviert und eine entsprechende Anzahl Datenbytes aus dem NOV-RAM gelesen bzw. in das NOV-RAM geschrieben. Wurden gleichzeitig beide Eingänge: *bRead* und *bWrite* gesetzt, dann werden die Daten zuerst in das NOV-RAM geschrieben und dann zurückgelesen.

Bemerkungen:

Um den Addresspointer auf das NOV-RAM zu ermitteln, wird von dem *FB_NovRamReadWrite*-Funktionsbaustein intern eine Instanz des ADSREAD-Funktionsbausteines verwendet. Dieser Addresspointer wird aber nur beim ersten Aufruf des *FB_NovRamReadWrite*-Funktionsbausteins und bei einer Änderung von *nDevId* neu ermittelt. Dafür werden mehrere SPS-Zyklen benötigt. Um die Daten in das NOV-RAM zu schreiben oder aus dem NOV-RAM zu lesen wird die Funktion MEMCPY verwendet. Dadurch können die Daten im gleichen SPS-Zyklus geschrieben bzw. gelesen werden. Intern wird auch die maximale Bytelänge des NOV-RAM ermittelt, und die maximale Länge der Daten, die gelesen oder geschrieben werden können, auf diese Länge begrenzt.

VAR_INPUT

```
VAR_INPUT
    nDevId      : UDINT;
    bRead       : BOOL;
    bWrite      : BOOL;
    cbSrcLen    : UDINT;
    cbDestLen   : UDINT;
    pSrcAddr    : PVOID;
    pDestAddr   : PVOID;
    tTimeOut    : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

nDevId: Die Geräte-Id einer NOV-RAM-Karte. Über die Id wird das NOVRAM einer FCxxxx-000**2** Karte spezifiziert, auf das mit dem Funktionsbaustein schreibend oder lesend zugegriffen werden soll. Die Geräte-Ids werden während der Hardware-Konfiguration vom TwinCAT System festgelegt.

bRead: Bei einer positiven Flanke an diesem Eingang wird der Baustein aktiviert und *cbDestLen*-Daten aus dem NOV-RAM (ab dem Addressoffset NULL) in den Puffer mit der Adresse *pDestAddr* hineinkopiert.

bWrite: Bei einer positiven Flanke an diesem Eingang wird der Baustein aktiviert und *cbSrcLen*-Daten aus dem Puffer mit der Adresse *pSrcAddr* in das NOV-RAM (ab dem Addressoffset NULL) hineinkopiert.

cbSrcLen: : Die Bytelänge der Daten, die in das NOV-RAM geschrieben werden sollen.

cbDestLen: Die Bytelänge der Daten, die aus dem NOV-RAM gelesen werden sollen.

pSrcAddr: Der Addresspointer auf einen Datenpuffer mit den Daten, die in das NOV-RAM geschrieben werden sollen. Der Addresspointer kann mit dem ADR-Operator ermittelt werden.

pDestAddr: Der Addresspointer auf einen Datenpuffer, in den die gelesenen NOV-RAM-Daten hineinkopiert werden sollen.

tTimeOut: Gibt die Timeout-Zeit an, die bei der Ausführung des Kommandos/Funktion nicht überschritten werden darf.

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy       : BOOL;
    bError      : BOOL;
```

```
    nErrId    : UDINT;
    cbRead    : UDINT;
    cbWrite   : UDINT;
END_VAR
```

bBusy: Bei der Aktivierung des Funktionsbausteins wird dieser Ausgang gesetzt und bleibt gesetzt, bis die Ausführung abgeschlossen wurde.

bError: Sollte ein Fehler bei der Ausführung erfolgen, dann wird dieser Ausgang gesetzt.

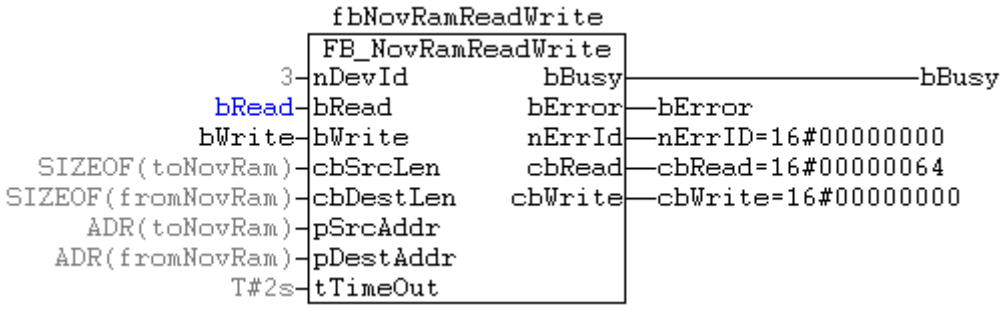
nErrId: Liefert bei einem gesetzten *bError*-Ausgang die ADS-Fehlernummer [▶ 154].

cbRead: Anzahl der erfolgreich gelesenen Datenbytes.

cbWrite: Anzahl der erfolgreich geschriebenen Datenbytes.

Beispiel:

```
PROGRAM MAIN
VAR
    fbNovRamReadWrite : FB_NovRamReadWrite;
    bRead             : BOOL;
    bWrite            : BOOL;
    fromNovRam        : ARRAY[1..100] OF BYTE;
    toNovRam          : ARRAY[1..100] OF BYTE;
    bBusy             : BOOL;
    bError            : BOOL;
    nErrID            : UDINT;
    cbRead            : UDINT;
    cbWrite           : UDINT;
END_VAR
```

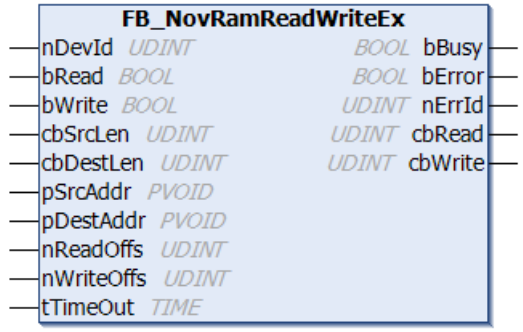


Im Beispiel wurden bei einer steigenden Flanke am *bRead*-Eingang 100 Byte Daten aus dem NOV-RAM ausgelesen und in das Array *fromNovRam* hineinkopiert.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO-Hardware	Einzubindende SPS-Bibliotheken (Kategoriegruppe)
TwinCAT v3.1.0	PC oder CX (x86)	FCxxxx cards mit NOV-RAM (FCxxxx-0002)	Tc2_IoFunctions (IO)

3.8.2 FB_NovRamReadWriteEx



Der Funktionsbaustein *FB_NovRamReadWriteEx* greift aus einem SPS-Programm auf das NOV-RAM zu (z.B. der FCxxxx-0002 Feldbuskarten, CX9000 NOV-RAM etc.). Bei einer steigenden Flanke am *bRead* oder *bWrite*-Eingang wird der Funktionsbaustein aktiviert und eine entsprechende Anzahl Datenbytes aus dem NOV-RAM gelesen bzw. in das NOV-RAM geschrieben. Wurden gleichzeitig beide Eingänge: *bRead* und *bWrite* gesetzt, dann werden die Daten zuerst in das NOV-RAM geschrieben und dann zurück gelesen. Der Unterschied zum Baustein FB_NovRamReadWrite ist, dass der Adressoffset im NOV-RAM für Schreib- und Lesezugriffe angegeben werden kann. Außerdem überprüft der Baustein die erlaubte Zugriffsart auf den NOV-RAM Speicher und kopiert wenn nötig die Daten byteweise in den NOV-RAM Speicher anstatt ein MEMCPY zu verwenden. Das NOV-RAM vom CX9000 erlaubt z.B. nur Byte-Zugriffe und der Baustein FB_NovRamReadWrite würde in diesem Fall einen Fehler zurückliefern.

Bemerkungen:

Um den Addresspointer auf das NOV-RAM zu ermitteln, wird von dem *FB_NovRamReadWriteEx*-Funktionsbaustein intern eine Instanz des ADSREAD-Funktionsbausteines verwendet. Dieser Addresspointer wird aber nur beim ersten Aufruf des *FB_NovRamReadWriteEx*-Funktionsbausteins und bei einer Änderung von *nDevId* neu ermittelt. Dafür werden mehrere SPS-Zyklen benötigt. Um die Daten in das NOV-RAM zu schreiben oder aus dem NOV-RAM zu lesen wird direkt auf den NOV-RAM Speicher zugegriffen. Dadurch können die Daten im gleichen SPS-Zyklus geschrieben bzw. gelesen werden. Intern wird auch die maximale Bytelänge des NOV-RAM ermittelt, und die maximale Länge der Daten, die gelesen oder geschrieben werden können, auf diese Länge begrenzt.

VAR_INPUT

```
VAR_INPUT
    nDevId     : UDINT;
    bRead      : BOOL;
    bWrite     : BOOL;
    cbSrcLen   : UDINT;
    cbDestLen  : UDINT;
    pSrcAddr   : PVOID;
    pDestAddr  : PVOID;
    nReadOffs  : UDINT;
    nWriteOffs : UDINT;
    tTimeOut   : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

nDevId: Die Geräte-Id einer NOV-RAM-Karte. Über die Id wird das NOVRAM einer FCxxxx-000**2** Karte spezifiziert, auf das mit dem Funktionsbaustein schreibend oder lesend zugegriffen werden soll. Die Geräte-Ids werden während der Hardware-Konfiguration vom TwinCAT System Manager festgelegt.

bRead: Bei einer positiven Flanke an diesem Eingang wird der Baustein aktiviert und *cbDestLen*-Daten aus dem NOV-RAM (ab dem Adressoffset *nReadOffs*) in den Puffer mit der Adresse *pDestAddr* hineinkopiert.

bWrite: Bei einer positiven Flanke an diesem Eingang wird der Baustein aktiviert und *cbSrcLen*-Daten aus dem Puffer mit der Adresse *pSrcAddr* in das NOV-RAM (ab dem Adressoffset *nWriteOffs*) hineinkopiert.

cbSrcLen: : Die Bytelänge der Daten, die in das NOV-RAM geschrieben werden sollen.

cbDestLen: Die Bytelänge der Daten, die aus dem NOV-RAM gelesen werden sollen.

pSrcAddr: Der Addresspointer auf einen Datenpuffer mit den Daten, die in das NOV-RAM geschrieben werden sollen. Der Addresspointer kann mit dem ADR-Operator ermittelt werden.

pDestAddr: Der Addresspointer auf einen Datenpuffer, in den die gelesenen NOV-RAM-Daten hineinkopiert werden sollen.

nReadOffs: Der Adressoffset im NOV-RAM ab dem gelesen werden soll.

nWriteOffs: Der Adressoffset im NOV-RAM ab dem geschrieben werden soll.

tTimeOut: Gibt die Timeout-Zeit an, die bei der Ausführung des Kommandos/Funktion nicht überschritten werden darf.

VAR_OUTPUT

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

bBusy: Bei der Aktivierung des Funktionsbausteins wird dieser Ausgang gesetzt und bleibt gesetzt, bis die Ausführung abgeschlossen wurde.

bError: Sollte ein Fehler bei der Ausführung erfolgen, dann wird dieser Ausgang gesetzt..

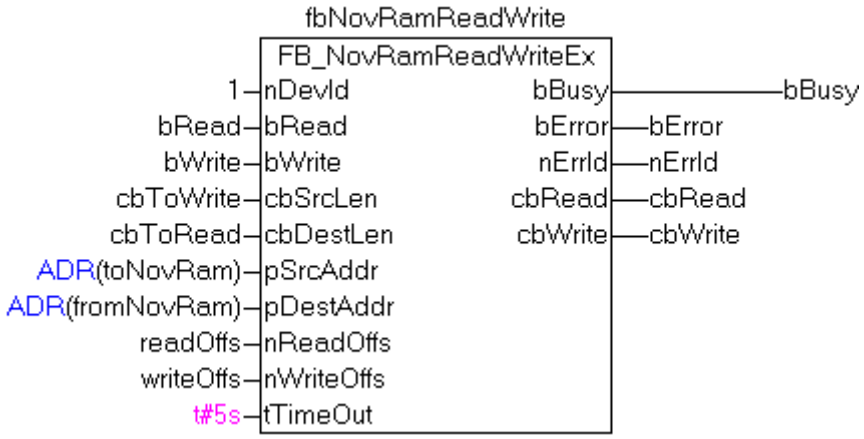
nErrId: Liefert bei einem gesetzten *bError*-Ausgang die ADS-Fehlernummer [▶ 154].

cbRead: Anzahl der erfolgreich gelesenen Datenbytes.

cbWrite: Anzahl der erfolgreich geschriebenen Datenbytes.

Beispiel:

```
PROGRAM MAIN
VAR
    fbNovRamReadWrite : FB_NovRamReadWriteEx;
    bRead             : BOOL;
    bWrite            : BOOL;
    fromNovRam        : ARRAY[1..100] OF BYTE;
    toNovRam          : ARRAY[1..100] OF BYTE;
    bBusy             : BOOL;
    bError            : BOOL;
    nErrID            : UDINT;
    cbRead            : UDINT;
    cbWrite           : UDINT;
    readOffs          : UDINT :=0;
    writeOffs         : UDINT:=0;
    cbToWrite         : UDINT := 100;
    cbToRead          : UDINT := 100;
END_VAR
```



Im Beispiel wurden bei einer steigenden Flanke am *bRead*-Eingang 100 Byte Daten aus dem NOV-RAM ausgelesen und in das Array *fromNovRam* hineinkopiert.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO-Hardware	Einzubindende SPS-Bibliotheken (Kategoriegruppe)
TwinCAT v3.1.0	PC oder CX (x86)	FCxxxx cards mit NOV-RAM (FCxxxx-0002)	Tc2_IoFunctions (IO)

3.8.3 FB_GetDPRAMInfo

Der Funktionsbaustein *FB_GetDPRAMInfo* ermittelt den Addresspointer und die konfigurierte Größe des NOV/DP-RAM einer Feldbuskarte. Der Addresspointer kann z.B. dazu benutzt werden, um direkt auf das NOV-RAM der FCxxx-0002 (Beckhoff) Karten oder DPRAM der von TwinCAT nicht unterstützten Karten (Drittthersteller) schreibend oder lesend zugreifen zu können. Die Karte muss vorher in TwinCAT System als allgemeines NOV/DP-RAM konfiguriert werden. Im SPS-Programm können dann die MEMCPY-, MEMSET- oder MEMCMP-Funktionen benutzt werden um auf einen beliebigen Speicheroffset schreibend/lesend zugreifen zu können.

Wenn das NOV/DP-RAM einen speziellen Zugriff (z.B. BYTE, aligned WORD) erfordert gibt dieser Funktionsbaustein einen *Service Not Supported* (16#701) Fehler zurück. In diesem Fall kann der Baustein FB_NovRamReadWriteEx verwendet werden um Daten aus dem NOV/DP-RAM zu lesen oder Daten in das NOV/DP-RAM zu schreiben. Z.B. das NOV-RAM vom *CX9000* erlaubt nur BYTE-Zugriffe, so dass in diesem Falle der Baustein FB_NovRamReadWriteEx verwendet werden sollte.

VAR_INPUT

```
VAR_INPUT
    nDevId      : UDINT;
    bExecute    : BOOL;
    tTimeOut    : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

nDevId: Die Geräte-Id einer NOV/DP-RAM-Karte. Über die Id wird die Karte spezifiziert, deren Informationen gelesen werden sollen. Die Geräte-Ids werden während der Hardware-Konfiguration vom TwinCAT System festgelegt.

bExecute: Bei einer positiven Flanke an diesem Eingang wird der Baustein aktiviert.

tTimeOut: Gibt die Timeout-Zeit an, die bei der Ausführung des Kommandos/Funktion nicht überschritten werden darf.

VAR_OUTPUT

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

bBusy: Bei der Aktivierung des Funktionsbausteins wird dieser Ausgang gesetzt und bleibt gesetzt, bis die Ausführung abgeschlossen wurde.

bError: Sollte ein Fehler bei der Ausführung erfolgen, dann wird dieser Ausgang gesetzt..

nErrId: Liefert bei einem gesetzten *bError*-Ausgang die ADS-Fehlernummer [▶ 154].

stInfo: Eine Struktur mit der Adresse und Größe des NOV/DP-RAM (Typ: ST_NovRamAddrInfo [▶ 130]).

Beispiel:

In MAIN wird ein Pointer auf eine Instanz der ST_NOVRAM-Strukturvariablen deklariert. Dieser Pointer wird beim Programmstart mit der Adresse des NOV-/DP-RAM initialisiert. Durch Referenzieren des Pointers kann auf die einzelnen Strukturelemente (und das NOV-/DP-RAM) schreibend oder lesend zugegriffen werden.

```
TYPE ST_NOVRAM :
STRUCT
    dwParam_0 : DWORD;
    dwParam_1 : DWORD;
    dwParam_2 : DWORD;
    dwParam_3 : DWORD;

    cbSize : DWORD;
    wCounter : WORD;
    func : BYTE;
    sMsg : STRING(20);

    (* ...other NOV-/DP-RAM variables *)
END_STRUCT
END_TYPE
```

```
PROGRAM MAIN
VAR
    pNOVRAM      : POINTER TO ST_NOVRAM;
    cbNOVRAM     : DWORD;
    fbGetInfo    : FB_GetDPRAMInfo;
    state        : INT := 0;
    bInit        : BOOL := FALSE;
    timer        : TON;
    bReset       : BOOL;
END_VAR
```

```
CASE state OF
0:
    IF NOT bInit THEN
        state := 1;
    END_IF

1:
    fbGetInfo( bExecute := FALSE ); (* reset fb *)
 fbGetInfo( nDevId:= 3,
    bExecute:= TRUE, (* start fb execution *)
 tTimeOut:= T#10s );
    state := 2;
2:
    fbGetInfo( bExecute := FALSE );
    IF NOT fbGetInfo.bBusy THEN
        IF NOT fbGetInfo.bError THEN
            IF fbGetInfo.stInfo.pCardAddress <> 0 THEN
                pNOVRAM := fbGetInfo.stInfo.pCardAddress;
                cbNOVRAM := fbGetInfo.stInfo.iCardMemSize;
                bInit := TRUE;
                state := 0; (* ready, go to the idle step *)
         ELSE
                state := 100; (* error *)
         END_IF
        ELSE
            state := 100; (* error *)
     END_IF
    END_IF

100:
    ; (* error, stay here *)
END_CASE


IF bInit THEN
 (* read from or write to NOV-/DP-RAM*)
 IF bReset THEN (* reset all bytes to 0 *)
     bReset := FALSE;
        MEMSET( pNOVRAM, 0, cbNOVRAM );
    END_IF

    timer( IN := TRUE, PT := T#1s );
    IF timer.Q THEN
        timer( IN := FALSE );
        pNOVRAM^.dwParam_0 := pNOVRAM^.dwParam_0 + 1;
        pNOVRAM^.dwParam_1 := pNOVRAM^.dwParam_1 - 1;
    END_IF
END_IF
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO-Hardware	Einzubindende SPS-Bibliotheken (Kategoriegruppe)
TwinCAT v3.1.0	PC oder CX (x86)	FCxxxx cards mit NOV-RAM (FCxxxx-0002)	Tc2_IoFunctions (IO)

3.8.4 FB_GetDPRAMInfoEx

Der Funktionsbaustein FB_GetDPRAMInfoEx ermittelt den Addresspointer und die konfigurierte Größe des NOV/DP-RAM einer Feldbuskarte. Der Addresspointer kann z.B. dazu benutzt werden, um direkt auf das NOV-RAM der Fcxxx-0002 (Beckhoff) Karten oder DPRAM der von TwinCAT nicht unterstützen Karten (Dritthersteller) schreiben oder lesend zugreifen zu können. Die Karte muss vorher im TwinCAT-System als allgemeines NOV/DP-RAM konfiguriert werden.

Der Ausgang stInfo liefert neben der Adresse (pCardAddress) und der Größe (iCardMemSize) des NOV/DP-RAM auch die Art des Zugriffs (eAccessType).

Wenn das NOV/DP-RAM keinen speziellen Zugriff (BYTE oder WORD aligned) erfordert, können im SPS-Programm die MEMCPY-, MEMSET-oder MEMCMP-Funktionen benutzt werden, um auf einen beliebigen Speicheroffset schreibend/lesend zugreifen zu können. Erfordert das NOV/DP-RAM einen speziellen Zugriff (BYTE-oder WORD-aligned), dann darf nur mit passender Datengröße zugegriffen werden. Hierfür sollte der FB_NovRamReadWriteEx verwendet werden.

```
VAR_INPUT
    nDevId      :    UDINT; (*Device id of the FCxxxx or other DPRAM card (Map the FC card as generi
c DPRAM/NOVRAM card im System Manager first)*)
    bExecute    :    BOOL; (*Rising edge starts function block execution*)
    tTimeOut    :    TIME; (*Max. timeout for this command*)
END_VAR
```

nDevId: Die Geräte-Id einer NOV/DP-RAM-Karte. Über die Id wird die Karte spezifiziert, deren Informationen gelesen werden sollen. Die Geräte-Ids werden während der Hardware-Konfiguration vom TwinCAT System festgelegt.

bExecute: Bei einer positiven Flanke an diesem Eingang wird der Baustein aktiviert.

tTimeOut: Gibt die Timeout-Zeit an, die bei der Ausführung des Kommandos/Funktion nicht überschritten werden darf.

```
VAR_OUTPUT
    bBusy     :    BOOL;
    bError    :    BOOL;
    nErrid    :    UDINT;
    stInfo    :    ST_NovRamAddrInfoEx;
END_VAR
```

bBusy: Bei der Aktivierung des Funktionsbaustein wird dieser Ausgang gesetzt und bleibt gesetzt bis die Ausführung abgeschlossen wurde.

bError: Dieser Ausgang wird gesetzt, wenn ein Fehler bei der Ausführung auftritt.

nErrId: Liefert bei einem gesetzten bError-Ausgang die ADS-Fehlernummer [▶ 154].

stInfo: Eine Struktur mit der Adresse und Größe des NOV/DP-RAM (Typ: ST_NovRamAddrInfoEx [▶ 131])

Beispiel

In MAIN wird ein Pointer auf einen BYTE deklariert. Dieser Pointer wird beim Programmstart mit der Adresse des NOV-/DP-RAM initialisiert. Durch Referenzieren des Pointers kann auf das NOV-/DP-RAM schreibend oder lesend zugegriffen werden.

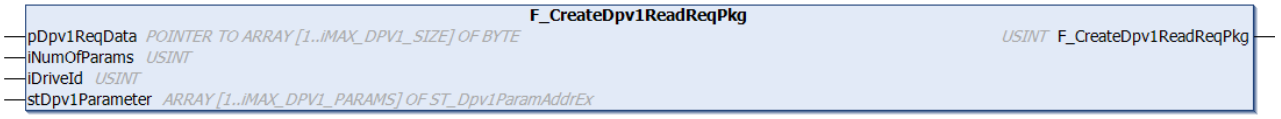
```
PROGRAM MAIN
VAR
    nDevId       : UDINT := 1; //device 1, see ID of NOV/DP-RAM device
    pNOVRAM      : POINTER TO BYTE;
    cbNOVRAM     : DWORD;
    fbGetInfo    : FB_GetDPRAMInfoEx;
    bInit        : BOOL := FALSE;
    eAccessType  : E_IOACCESSTYPE;
    bByteAccess  : BOOL;
    bWordAccess  : BOOL;
    bDWordAccess : BOOL;
END_VAR

IF NOT bInit THEN
    fbGetInfo(
        nDevId   := nDevId,
        bExecute := TRUE,
        tTimeOut := T#5S
    );

    IF NOT fbGetInfo.bBusy THEN
        IF NOT fbGetInfo.bError THEN
            pNOVRAM     := fbGetInfo.stInfo.pCardAddress;
            cbNOVRAM    := fbGetInfo.stInfo.iCardMemSize;
            eAccessType := fbGetInfo.stInfo.eAccessType;

            bDWordAccess := FALSE;
            bByteAccess  := FALSE;
            bWordAccess  := FALSE;

            CASE eAccessType OF
                eIOAccess_Default:
                    bDWordAccess := TRUE;
                    //access via MEMCPY,  MEMSET,  MEMCMP possible
                eIOAccess_Byte:
                    bByteAccess := TRUE;
                    //access via POINTER to BYTE possible
                eIOAccess_WordSwap:
                    bWordAccess := TRUE;
                    //access via POINTER to WORD +
                    //swapping of high and low byte possible
            END_CASE

            bInit := TRUE;
        END_IF
        fbGetInfo(bExecute := FALSE);
    END_IF
END_IF
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO-Hardware	Einzubindende SPS-Bibliotheken (Kategoriegruppe)
TwinCAT v3.1.0	PC oder CX (x86)	FCxxxx cards mit NOV-RAM (FCxxxx-0002)	Tc2_IoFunctions (IO)

3.9 Profibus DPV1 (Sinamics)

3.9.1 F_CreateDpv1ReadReqPkg



Die Funktion "F_CreateDpv1ReadReqPkg" erzeugt ein DPV1 Telegramm für einen FB_Dpv1Read [▶ 75] eines oder mehrerer Parameter eines Antriebs oder der Controller Unit eines Sinamics Profidrives (Profidrive Specification 3.1). Da Profidrives das Motorola-Format und IPCs das Intel-Format nutzen, führt die Funktion automatisch das Vertauschen der Bytereihenfolge im DPV1 Telegramm von Parametern mit Datentypen mit mehr als einem Byte durch.

Die Funktion liefert die aktuelle Länge des DPV1 Telegramms in Bytes (max. 240 Bytes) zurück.

FUNCTION F_CreateDpv1ReadReqPkg : USINT

VAR_INPUT

```
VAR_INPUT
    pDpv1ReqData : POINTER TO ARRAY [1..iMAX_DPV1_SIZE] OF BYTE;(* DPV1 read request *)
    iNumOfParams : USINT; (* 1..39; else: reserved *)
    iDriveId     : USINT;
END_VAR
```

pDpv1ReqData: Zeiger auf ein Array von 240 Bytes, das das DPV1 Lesetelegramm enthält. Dieses Telegram wird von der Funktion erstellt.

iNumOfParams: Anzahl der zu lesenden Parameter (1 to 39). Eine weitere Begrenzung ist die Telegrammgröße von 240 Bytes.

iDriveID: Die ID ist 1 für die ControllerUnit, 2 für das Antriebsobjekt A, 3 für das Antriebsobjekt B eines Doppel/Dreifach-Antriebs. Die Drive ID wird in der Starter Software gesetzt. 1..16 ist möglich.

VAR_IN_OUT

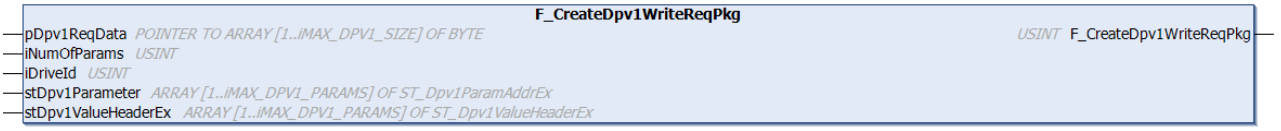
```
VAR_IN_OUT
    stDpv1Parameter : ARRAY [1..iMAX_DPV1_PARAMS] OF ST_Dpv1ParamAddrEx;(* list of parameters *)
END_VAR
```

stDpv1Parameter: Array von 39 Parametern, die zum DPV1 Lesetelegramm zugefügt werden sollen (Typ: ST_Dpv1ParamAddrEx [▶ 129]).

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO Hardware	Einzubindende SPS-Bibliotheken (Kategoriegruppe)
TwinCAT v3.1.0	PC oder CX (x86)	Beckhoff FC310x PCI, CX1500-M310, EL6731	Tc2_IoFunctions (IO)

3.9.2 F_CreateDpv1WriteReqPkg



Die Funktion "F_CreateDpv1WriteReqPkg" erzeugt ein DPV1 Telegramm für einen FB_Dpv1Write [▶ 78] eines oder mehrerer Parameter eines Antriebs oder der Controller Unit eines Sinamics Profidrives (Profidrive Specification 3.1). Da Profidrives das Motorola-Format und IPCs das Intel-Format nutzen, führt die Funktion automatisch das Vertauschen der Bytereihenfolge im DPV1 Telegramm von Parametern mit Datentypen mit mehr als einem Byte durch.

Die Funktion liefert die aktuelle Länge des DPV1 Telegramms in Bytes (max. 240 Bytes) zurück.

FUNCTION F_CreateDpv1WriteReqPkg : USINT

VAR_INPUT

```
VAR_INPUT
    pDpv1ReqData : POINTER TO ARRAY [1..iMAX_DPV1_SIZE] OF BYTE; (* DPV1 write request *)
    iNumOfParams : USINT;(* 1..39; else: reserved *)
    iDriveId     : USINT;
END_VAR
```

pDpv1ReqData: Zeiger auf ein Array von 240 Bytes, das das DPV1 Schreibtelegramm enthält. Dieses Telegram wird von der Funktion erstellt.

iNumOfParams: Anzahl der zu schreibenden Parameter (1 to 39). Eine weitere Begrenzung ist die Telegrammgröße von 240 Bytes.

iDriveID: Die ID ist 1 für die ControllerUnit, 2 für das Antriebsobjekt A, 3 für das Antriebsobjekt B eines Doppel/Dreifach-Antriebs. Die Drive ID wird in der Starter Software gesetzt. 1..16 ist möglich.

VAR_IN_OUT

```
VAR_IN_OUT
    stDpv1Parameter     : ARRAY [1..iMAX_DPV1_PARAMS] OF ST_Dpv1ParamAddrEx;(* list of parameters *)
    stDpv1ValueHeaderEx : ARRAY [1..iMAX_DPV1_PARAMS] OF ST_Dpv1ValueHeaderEx;(* list of values *)
END_VAR
```

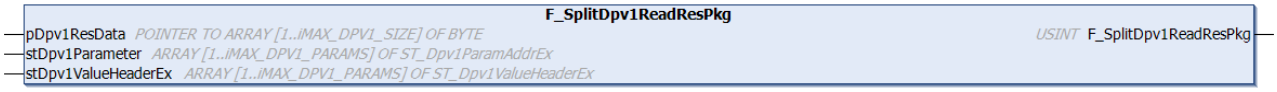
stDpv1Parameter: Array von 39 Parametern, die zum DPV1 Schreibtelegramm zugefügt werden sollen (Typ: ST_Dpv1ParamAddrEx [▶ 129]).

stDpv1ValueHeaderEx: Array of 39 Parameterwerten, die zum DPV1 Schreibtelegramm zugefügt werden sollen (Typ: ST_Dpv1ValueHeaderEx [▶ 130]).

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO Hardware	Einzubindende SPS-Bibliotheken (Kategoriegruppe)
TwinCAT v3.1.0	PC oder CX (x86)	Beckhoff FC310x PCI, CX1500-M310, EL6731	Tc2_IoFunctions (IO)

3.9.3 F_SplitDpv1ReadResPkg



Die Funktion "F_SplitDpv1ReadResPkg" wertet ein DPV1 Telegramm eines FB_Dpv1Read [▶ 75] eines oder mehrerer Parameter eines Antriebs oder der Controller Unit eines Sinamics Profidrives (Profidrive Specification 3.1) aus. Da Profidrives das Motorola-Format und IPCs das Intel-Format nutzen, führt die Funktion automatisch das Vertauschen der Bytereihenfolge im DPV1 Telegramm von Parametern mit Datentypen mit mehr als einem Byte durch.

Die Funktion liefert die aktuelle Länge des DPV1 Telegramms in Bytes (max. 240 Bytes) zurück.

FUNCTION F_SplitDpv1ReadResPkg : USINT

VAR_INPUT

```
VAR_INPUT
    pDpv1ResData : POINTER TO ARRAY [1..iMAX_DPV1_SIZE] OF BYTE; (* DPV1 read response *)
END_VAR
```

pDpv1ResData: Zeiger auf ein Array von 240 Bytes, das das DPV1 Leseantworttelegramm enthält. Dieses Telegramm wird von der Funktion ausgewertet.

VAR_IN_OUT

```
VAR_IN_OUT
    stDpv1Parameter     : ARRAY [1..iMAX_DPV1_PARAMS] OF ST_Dpv1ParamAddrEx;(* list of parameters *)
    stDpv1ValueHeaderEx : ARRAY [1..iMAX_DPV1_PARAMS] OF ST_Dpv1ValueHeaderEx;(* list of values *)
END_VAR
```

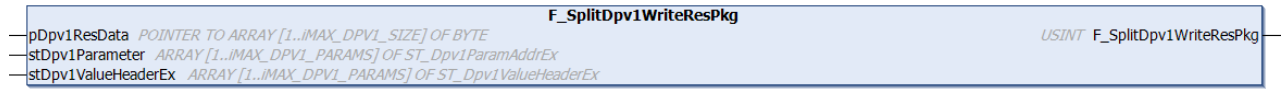
stDpv1Parameter: Array von 39 Parametern, die zum DPV1 Lesetelegramm zugefügt wurden (Typ: ST_Dpv1ParamAddrEx [▶ 129]).

stDpv1ValueHeaderEx: Array von 39 Parameterwerten, die vom Antrieb gelesen wurden (Typ: ST_Dpv1ValueHeaderEx [▶ 130]).

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO Hardware	Einzubindende SPS-Bibliotheken (Kategoriegruppe)
TwinCAT v3.1.0	PC oder CX (x86)	Beckhoff FC310x PCI, CX1500-M310, EL6731	Tc2_IoFunctions (IO)

3.9.4 F_SplitDpv1WriteResPkg



Die Funktion "F_SplitDpv1WriteResPkg" wertet ein DPV1 Telegramm eines FB_Dpv1Write [▶ 78] eines oder mehrerer Parameter eines Antriebs oder der Controller Unit eines Sinamics Profidrives (Profidrive Specification 3.1) aus. Da Profidrives das Motorola-Format und IPCs das Intel-Format nutzen, führt die Funktion automatisch das Vertauschen der Bytereihenfolge im DPV1 Telegramm von Parametern mit Datentypen mit mehr als einem Byte durch.

Die Funktion liefert die aktuelle Länge des DPV1 Telegramms in Bytes (max. 240 Bytes) zurück.

FUNCTION F_SplitDpv1WriteResPkg : USINT

VAR_INPUT

```
VAR_INPUT
    pDpv1ResData : POINTER TO ARRAY [1..iMAX_DPV1_SIZE] OF BYTE;(* DPV1 write response *)
END_VAR
```

pDpv1ResData: Zeiger auf ein Array von 240 Bytes, das das DPV1 Schreibantworttelegramm enthält. Dieses Telegramm wird von der Funktion ausgewertet.

VAR_IN_OUT

```
VAR_IN_OUT
    stDpv1Parameter     : ARRAY [1..iMAX_DPV1_PARAMS] OF ST_Dpv1ParamAddrEx;(* list of parameters *)
    stDpv1ValueHeaderEx : ARRAY [1..iMAX_DPV1_PARAMS] OF ST_Dpv1ValueHeaderEx;(* list of values *)
END_VAR
```

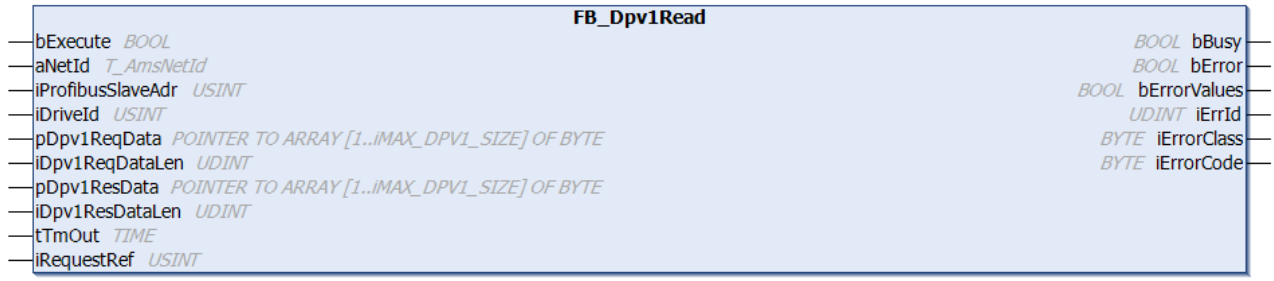
stDpv1Parameter: Array von 39 Parametern, die zum DPV1 Schreibtelegramm zugefügt wurden (Typ: ST_Dpv1ParamAddrEx [▶ 129]).

stDpv1ValueHeaderEx: Array von 39 Parameterwerten, die vom Antrieb gelesen wurden (Typ: ST_Dpv1ValueHeaderEx [▶ 130]).

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO Hardware	Einzubindende SPS-Bibliotheken (Kategoriegruppe)
TwinCAT v3.1.0	PC oder CX (x86)	Beckhoff FC310x PCI, CX1500-M310, EL6731	Tc2_IoFunctions (IO)

3.9.5 FB_Dpv1Read



Der Funktionsbaustein "FB_Dpv1Read" liest einen oder mehrere Parameter eines Sinamics Profidrive via DPV1 (Profidrive Specification 3.1). Das DPV1 Lesetelegramm muss mit F_CreateDpv1ReadReqPkg [▶ 72] erstellt werden, bevor an bExecute eine steigende Flanke ansteht. Das DPV1 Antworttelegramm muss mit F_SplitDpv1ReadResPkg [▶ 73] ausgewertet werden, nachdem bBusy eine fallende Flanke anzeigt.

Die Ausführung dieses Funktionsbausteins benötigt einige Zeit, abhängig von der Anzahl der Parameter, die gelesen werden sollen. Der Funktionsbaustein sendet das DPV1 Telegramm und pollt nach einem Antworttelegramm.

VAR_INPUT

```
VAR_INPUT
    bExecute          : BOOL;
    aNetId            : T_AmsNetId; (* NetID of Profibus Master FC310x/EL6731 *)
    iProfibusSlaveAdr : USINT; (* DP address of ProfiDrive *)
    iDriveId          : USINT; (* 1..16 possible *)
    pDpv1ReqData      : POINTER TO ARRAY [1..iMAX_DPV1_SIZE] OF BYTE;
    iDpv1ReqDataLen   : UDINT;
    pDpv1ResData      : POINTER TO ARRAY [1..iMAX_DPV1_SIZE] OF BYTE;
    iDpv1ResDataLen   : UDINT;
    tTmOut            : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

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

aNetId: Die Netzwerkadresse des Profibus Master Gerätes (siehe ADS-Tab des Profibus Master Gerätes in der I/O-Konfiguration im TwinCAT System, Typ: T_AmsNetID).

iProfibusSlaveAdr: Die Profibus slave DP-Adresse des Antriebs. Das ist eine Adresse für mehrere Achsen, spezifiziert im TwinCAT System in der I/O-Konfiguration.

iDriveID: Die ID ist 1 für die ControllerUnit, 2 für das Antriebsobjekt A, 3 für das Antriebsobjekt B eines Doppel/Dreifach-Antriebs. Die Drive ID wird in der Starter Software gesetzt. 1..16 ist möglich.

pDpv1ReqData: Zeiger auf ein Array von 240 Bytes, das das DPV1 Lesetelegramm enthält. Dieses Telegramm muss von der Funktion F_CreateDpv1ReadReqPkg [▶ 72] erstellt werden, bevor das DPV1 Lesen via bExecute aktiviert wird.

iDv1ReqDataLen: Maximale Länge des DPV1 Datapuffer (240 bytes).

pDpv1ResData: Zeiger auf ein Array von 240 Bytes, das das DPV1 Leseantworttelegramms enthält. Dieses Telegramm muss von der Funktion F_SplitDpv1ReadResPkg [▶ 73] ausgewertet werden nachdem auf bBusy eine negative Flanke erscheint.

iDv1ResDataLen: Maximale Länge des DPV1 Antwort-Datapuffers (240 bytes).

tTmOut: Maximale Zeit die bei der Ausführung des Kommandos nicht überschritten werden soll.

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy        : BOOL;
    bError       : BOOL;
    bErrorValues : BOOL;
    iErrId       : UDINT;
    iErrorClass  : BYTE;
    iErrorCode   : BYTE;
END_VAR
```

bBusy: Der Ausgang geht auf TRUE sobald der Funktionsbaustein via bExecute aktiviert wurde und bleibt so lange TRUE, wie der Baustein keine Antwort erhalten hat.

bError: Bei Fehlern geht der Ausgang auf TRUE und bBusy auf FALSE.

bErrorValues: Ist TRUE wenn der DPV1 Read nicht oder nur teilweise erfolgreich war. Die Fehlerursachen werden über die Fehler-ID geliefert (sowie Class und Code).

nErrId: Liefert die ADS Fehlernummer [▶ 154] oder funktionsbausteinspezifische Fehlernummern, wenn bError = TRUE.

nErrClass: Profidrive Fehlerklasse.

nErrCode: Profidrive Fehlercode.

Funktionbausteinspezifische Fehlercodes	Beschreibung
0x2	falsche Antwortreferenz
0x3	DPV1 Read fehlerhaft oder teilweise fehlerhaft
0x4	falsche Antwort-ID
other error IDs	siehe ADS Fehlercodes

Fehlerklassen	Beschreibung	Fehlercode
0x0 - 0x9	reserviert	-
0xA	Anwendungsfehler	0x0: Lesefehler 0x1: Schreibfehler 0x2: Modulfehler 0x3 - 0x7: reserviert 0x8: Versionskonflikt 0x9: nicht unterstützt 0xA - 0xF: benutzerabhängig
0xB	Zugriffsfehler	0x0: ungültiger Index (kein Datenblock DB47, Parameterzugriff wird nicht unterstützt) 0x1: Schreiblängenfehler 0x2: ungültiger Slot 0x3: Typkonflikt 0x4: ungültiger Bereich 0x5: Zustandskonflikt (Zugriff auf DB47 temporär nicht möglich wegen interner Prozesszustände) 0x6: Zugriff verweigert 0x7: ungültiger Bereich (Schreibfehler im DB47 Header) 0x8: ungültiger Parameter 0x9: ungültiger Typ 0xA - 0xF: benutzerabhängig
0xC	Resourcefehler	0x0: Lesekonflikt 0x1: Schreibkonflikt 0x2: Resource beschäftigt 0x3: Resource nicht erreichbar 0x4 - 0x7: reserviert 0x8 - 0xF: benutzerabhängig
0xD - 0xF	Benutzerdefinierte Fehler	-

VAR_IN_OUT

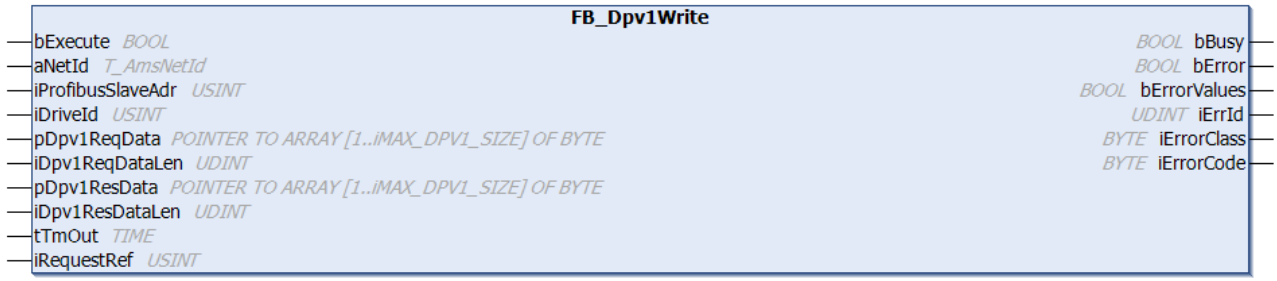
```
VAR_OUTPUT
    iRequestRef : USINT; (* 1..127; 0: reserved *)
END_VAR
```

iRefRequest: Referenz, die mit jedem Telegramm automatisch hochgezählt wird. Die Referenz wird für die Zuordnung der Antworten auf die Schreib/Lese-Anforderungen benötigt.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO Hardware	Einzubindende SPS-Bibliotheken (Kategoriegruppe)
TwinCAT v3.1.0	PC oder CX (x86)	Beckhoff FC310x PCI, CX1500-M310, EL6731	Tc2_IoFunctions (IO)

3.9.6 FB_Dpv1Write



Der Funktionsbaustein "FB_Dpv1Write" schreibt einen oder mehrere Parameter eines Sinamics Profidrive via DPV1 (Profidrive Specification 3.1). Das DPV1 Schreibtelegramm muss mit F_CreateDpv1WriteReqPkg [▶ 72] erstellt werden, bevor an bExecute eine steigende Flanke ansteht. Das DPV1 Antworttelegramm muss mit F_SplitDpv1WriteResPkg [▶ 74] ausgewertet werden, nach dem bBusy eine fallende Flanke anzeigt.

Die Ausführung dieses Funktionsbausteins dauert einige Zeit, abhängig von der Anzahl der Parameter, die gelesen werden sollen. Der Funktionsbaustein sendet das DPV1 Telegramm und pollt nach einem Antworttelegramm.

VAR_INPUT

```
VAR_INPUT
    bExecute          : BOOL;
    aNetId            : T_AmsNetId; (* NetID of Profibus Master FC310x/EL6731 *)
    iProfibusSlaveAdr : USINT; (* DP address of ProfiDrive *)
    iDriveId          : USINT; (* 1..16 possible *)
    pDpv1ReqData      : POINTER TO ARRAY [1..iMAX_DPV1_SIZE] OF BYTE;
    iDpv1ReqDataLen   : UDINT;
    pDpv1ResData      : POINTER TO ARRAY [1..iMAX_DPV1_SIZE] OF BYTE;
    iDpv1ResDataLen   : UDINT;
    tTmOut            : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

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

aNetId: Die Netzwerkadresse des Profibus Master Gerätes (siehe ADS-Tab des Profibus Master Gerätes in der I/O-Konfiguration im TwinCAT System, Typ: T_AmsNetID).

iProfibusSlaveAdr: Die Profibus slave DP-Adresse des Antriebs. Das ist eine Adresse für mehrere Achsen, spezifiziert im TwinCAT System in der I/O-Konfiguration.

iDriveID: Die ID ist 1 für die ControllerUnit, 2 für das Antriebsobjekt A, 3 für das Antriebsobjekt B eines Doppel/Dreifach-Antriebs. Die Drive ID wird in der Starter Software gesetzt. 1..16 ist möglich.

pDpv1ReqData: Zeiger auf ein Array von 240 Bytes, das das DPV1 Schreibtelegramm enthält. Dieses Telegramm muss von der Funktion F_CreateDpv1WriteReqPkg [▶ 72] erstellt werden, bevor das DPV1 Schreiben via bExecute aktiviert wird.

iDv1ReqDataLen: Maximale Länge des DPV1 Datapuffer (240 bytes).

pDpv1ResData: Zeiger auf ein Array von 240 Bytes, das das DPV1 Schreibantworttelegramms enthält. Dieses Telegramm muss von der Funktion F_SplitDpv1WriteResPkg [▶ 74] ausgewertet werden nachdem auf bBusy eine negative Flanke erscheint.

iDv1ResDataLen: Maximale Länge des DPV1 Antwort-Datapuffers (240 bytes).

tTmOut: Maximale Zeit die bei der Ausführung des Kommandos nicht überschritten werden soll.

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy        : BOOL;
    bError       : BOOL;
    bErrorValues : BOOL;
    iErrId       : UDINT;
```

```
    iErrorClass  : BYTE;
    iErrorCode   : BYTE;
END_VAR
```

bBusy: Der Ausgang geht auf TRUE sobald der Funktionsbaustein via bExecute aktiviert wurde und bleibt so lange TRUE, wie der Baustein keine Antwort erhalten hat.

bError: Bei Fehlern geht der Ausgang auf TRUE und bBusy auf FALSE.

bErrorValues: Ist TRUE wenn der DPV1 Write nicht oder nur teilweise erfolgreich war. Die Fehlerursachen werden über die Fehler-ID geliefert (sowie Class und Code).

nErrId: Liefert die ADS Fehlernummer [▶ 154] oder funktionsbausteinspezifische Fehlernummern, wenn bError = TRUE.

nErrClass: Profidrive Fehlerklasse.

nErrCode: Profidrive Fehlercode.

Funktionbausteinspezifische Fehlercodes	Beschreibung
0x2	falsche Antwortreferenz
0x3	DPV1 Write fehlerhaft oder teilweise fehlerhaft
0x4	falsche Antwort-ID
other error IDs	siehe ADS Fehlercodes

Fehlerklassen	Beschreibung	Fehlercode
0x0 - 0x9	reserviert	-
0xA	Anwendungsfehler	0x0: Lesefehler 0x1: Schreibfehler 0x2: Modulfehler 0x3 - 0x7: reserviert 0x8: Versionskonflikt 0x9: nicht unterstützt 0xA - 0xF: benutzerabhängig
0xB	Zugriffsfehler	0x0: ungültiger Index (kein Datenblock DB47, Parameterzugriff wird nicht unterstützt) 0x1: Schreiblängenfehler 0x2: ungültiger Slot 0x3: Typkonflikt 0x4: ungültiger Bereich 0x5: Zustandskonflikt (Zugriff auf DB47 temporär nicht möglich wegen interner Prozesszustände) 0x6: Zugriff verweigert 0x7: ungültiger Bereich (Schreibfehler im DB47 Header) 0x8: ungültiger Parameter 0x9: ungültiger Typ 0xA - 0xF: benutzerabhängig
0xC	Resourcefehler	0x0: Lesekonflikt 0x1: Schreibkonflikt 0x2: Resource beschäftigt 0x3: Resource nicht erreichbar 0x4 - 0x7: reserviert 0x8 - 0xF: benutzerabhängig
0xD - 0xF	Benutzerdefinierte Fehler	-

VAR_IN_OUT

```
VAR_OUTPUT
    iRequestRef : USINT; (* 1..127; 0: reserved *)
END_VAR
```

iRefRequest: Referenz, die mit jedem Telegramm automatisch hochgezählt wird. Die Referenz wird für die Zuordnung der Antworten auf die Schreib/Lese-Anforderungen benötigt.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO Hardware	Einzubindende SPS-Bibliotheken (Kategoriegruppe)
TwinCAT v3.1.0	PC oder CX (x86)	Beckhoff FC310x PCI, CX1500-M310, EL6731	Tc2_IoFunctions (IO)

3.10 Profinet DPV1 (Sinamics)

3.10.1 F_CreateDpv1ReadReqPkgPNET

Die Funktion "F_CreateDpv1ReadReqPkg" erzeugt ein DPV1 Telegramm für einen FB_Dpv1ReadPNET [▶ 83] eines oder mehrerer Parameter eines Antriebs oder der Controller Unit eines Sinamics Profidrives (Profidrive Specification 3.1), das über Profinet angeschlossen ist. Da Profidrives das Motorola-Format und IPCs das Intel-Format nutzen, führt die Funktion automatisch das Vertauschen der Bytereihenfolge im DPV1 Telegramm von Parametern mit Datentypen mit mehr als einem Byte durch.

Die Funktion liefert die aktuelle Länge des DPV1 Telegramms in Bytes (max. 240 Bytes) zurück.

FUNCTION F_CreateDpv1ReadReqPkgPNET : USINT

VAR_INPUT

```
VAR_INPUT
    pDpv1ReqData : POINTER TO ARRAY [1..iMAX_DPV1_SIZE] OF BYTE;(* DPV1 read request *)
    iNumOfParams : USINT; (* 1..39; else: reserved *)
    iDriveId     : USINT;
END_VAR
```

pDpv1ReqData: Zeiger auf ein Array von 240 Bytes, das das DPV1 Lesetelegramm enthält. Dieses Telegram wird von der Funktion erstellt.

iNumOfParams: Anzahl der zu lesenden Parameter (1 to 39). Eine weitere Begrenzung ist die Telegrammgröße von 240 Bytes.

iDriveID: Die ID ist 1 für die ControllerUnit, 2 für das Antriebsobjekt A, 3 für das Antriebsobjekt B eines Doppel/Dreifach-Antriebs. Die Drive ID wird in der Starter Software gesetzt. 1..16 ist möglich.

VAR_IN_OUT

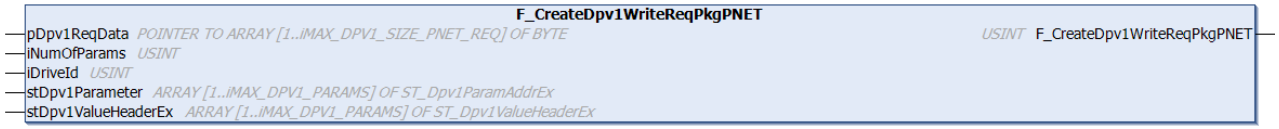
```
VAR_IN_OUT
    stDpv1Parameter : ARRAY [1..iMAX_DPV1_PARAMS] OF ST_Dpv1ParamAddrEx;(* list of parameters *)
END_VAR
```

stDpv1Parameter: Array von 39 Parametern, die zum DPV1 Lesetelegramm zugefügt werden sollen (Typ: ST_Dpv1ParamAddrEx [▶ 129]).

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO Hardware	Einzubindende SPS-Bibliotheken (Kategoriegruppe)
TwinCAT v3.1.0	PC oder CX (x86)	Beckhoff EL6632	Tc2_IoFunctions (IO)

3.10.2 F_CreateDpv1WriteReqPkgPNET



Die Funktion "F_CreateDpv1WriteReqPkgPNET" erzeugt ein DPV1 Telegramm für einen FB_Dpv1WritePNET [▶ 85] eines oder mehrerer Parameter eines Antriebs oder der Controller Unit eines Sinamics Profidrives (Profidrive Specification 3.1), das über Profinet angeschlossen ist. Da Profidrives das Motorola-Format und IPCs das Intel-Format nutzen, führt die Funktion automatisch das Vertauschen der Bytereihenfolge im DPV1 Telegramm von Parametern mit Datentypen mit mehr als einem Byte durch.

Die Funktion liefert die aktuelle Länge des DPV1 Telegramms in Bytes (max. 240 Bytes) zurück.

FUNCTION F_CreateDpv1WriteReqPkgPNET : USINT

VAR_INPUT

```
VAR_INPUT
    pDpv1ReqData : POINTER TO ARRAY [1..iMAX_DPV1_SIZE] OF BYTE; (* DPV1 write request *)
    iNumOfParams : USINT; (* 1..39; else: reserved *)
    iDriveId     : USINT;
END_VAR
```

pDpv1ReqData: Zeiger auf ein Array von 240 Bytes, das das DPV1 Schreibtelegramm enthält. Dieses Telegram wird von der Funktion erstellt.

iNumOfParams: Anzahl der zu schreibenden Parameter (1 to 39). Eine weitere Begrenzung ist die Telegrammgröße von 240 Bytes.

iDriveID: Die ID ist 1 für die ControllerUnit, 2 für das Antriebsobjekt A, 3 für das Antriebsobjekt B eines Doppel/Dreifach-Antriebs. Die Drive ID wird in der Starter Software gesetzt. 1..16 ist möglich.

VAR_IN_OUT

```
VAR_IN_OUT
    stDpv1Parameter     : ARRAY [1..iMAX_DPV1_PARAMS] OF ST_Dpv1ParamAddrEx;(* list of parameters *)
    stDpv1ValueHeaderEx : ARRAY [1..iMAX_DPV1_PARAMS] OF ST_Dpv1ValueHeaderEx;(* list of values *)
END_VAR
```

stDpv1Parameter: Array von 39 Parametern, die zum DPV1 Schreibtelegramm zugefügt werden sollen (Typ: ST_Dpv1ParamAddrEx [▶ 129]).

stDpv1ValueHeaderEx: Array of 39 Parameterwerten, die zum DPV1 Schreibtelegramm zugefügt werden sollen (Typ: ST_Dpv1ValueHeaderEx [▶ 130]).

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO Hardware	Einzubindende SPS-Bibliotheken (Kategoriegruppe)
TwinCAT v3.1.0	PC oder CX (x86)	Beckhoff EL6632	Tc2_IoFunctions (IO)

3.10.3 F_SplitDpv1ReadResPkgPNET

Die Funktion "F_SplitDpv1ReadResPkgPNET" wertet ein DPV1 Telegramm eines FB_Dpv1ReadPNET [▶ 83] eines oder mehrerer Parameter eines Antriebs oder der Controller Unit eines Sinamics Profidrives (Profidrive Specification 3.1) aus, das über Profinet angeschlossen ist. Da Profidrives das Motorola-Format und IPCs das Intel-Format nutzen, führt die Funktion automatisch das Vertauschen der Bytereihenfolge im DPV1 Telegramm von Parametern mit Datentypen mit mehr als einem Byte durch.

Die Funktion liefert die aktuelle Länge des DPV1 Telegramms in Bytes (max. 240 Bytes) zurück.

FUNCTION F_SplitDpv1ReadResPkgPNET : USINT

VAR_INPUT

```
VAR_INPUT
    pDpv1ResData : POINTER TO ARRAY [1..iMAX_DPV1_SIZE] OF BYTE; (* DPV1 read response *)
END_VAR
```

pDpv1ResData: Zeiger auf ein Array von 240 Bytes, das das DPV1 Leseantworttelegramm enthält. Dieses Telegramm wird von der Funktion ausgewertet.

VAR_IN_OUT

```
VAR_IN_OUT
    stDpv1Parameter     : ARRAY [1..iMAX_DPV1_PARAMS] OF ST_Dpv1ParamAddrEx;(* list of parameters *)
    stDpv1ValueHeaderEx : ARRAY [1..iMAX_DPV1_PARAMS] OF ST_Dpv1ValueHeaderEx;(* list of values *)
END_VAR
```

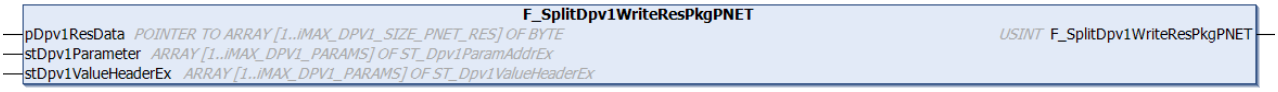
stDpv1Parameter: Array von 39 Parametern, die zum DPV1 Lesetelegramm zugefügt wurden (Typ: ST_Dpv1ParamAddrEx [▶ 129]).

stDpv1ValueHeaderEx: Array von 39 Parameterwerten, die vom Antrieb gelesen wurden (Typ: ST_Dpv1ValueHeaderEx [▶ 130]).

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO Hardware	Einzubindende SPS-Bibliotheken (Kategoriegruppe)
TwinCAT v3.1.0	PC oder CX (x86)	Beckhoff EL6632	Tc2_IoFunctions (IO)

3.10.4 F_SplitDpv1WriteResPkgPNET



Die Funktion "F_SplitDpv1WriteResPkgPNET" wertet ein DPV1 Telegramm eines FB_Dpv1WritePNET [▶ 85] eines oder mehrerer Parameter eines Antriebs oder der Controller Unit eines Sinamics Profidrives (Profidrive Specification 3.1) aus, das über Profinet angeschlossen ist. Da Profidrives das Motorola-Format und IPCs das Intel-Format nutzen, führt die Funktion automatisch das Vertauschen der Bytereihenfolge im DPV1 Telegramm von Parametern mit Datentypen mit mehr als einem Byte durch.

Die Funktion liefert die aktuelle Länge des DPV1 Telegramms in Bytes (max. 240 Bytes) zurück.

FUNCTION F_SplitDpv1WriteResPkgPNET : USINT

VAR_INPUT

```
VAR_INPUT
    pDpv1ResData : POINTER TO ARRAY [1..iMAX_DPV1_SIZE] OF BYTE; (* DPV1 write response *)
END_VAR
```

pDpv1ResData: Zeiger auf ein Array von 240 Bytes, das das DPV1 Schreibantworttelegramm enthält. Dieses Telegramm wird von der Funktion ausgewertet.

VAR_IN_OUT

```
VAR_IN_OUT
    stDpv1Parameter      : ARRAY [1..iMAX_DPV1_PARAMS] OF ST_Dpv1ParamAddrEx; (* list of parameters
*)
    stDpv1ValueHeaderEx  : ARRAY [1..iMAX_DPV1_PARAMS] OF ST_Dpv1ValueHeaderEx; (* list of values *)
END_VAR
```

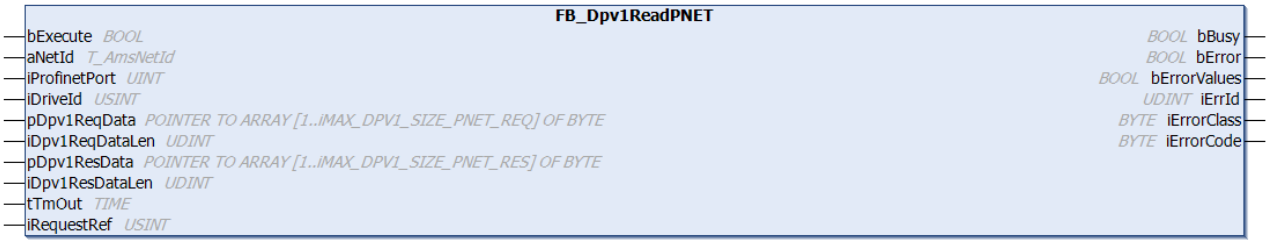
stDpv1Parameter: Array von 39 Parametern, die zum DPV1 Schreibtelegramm zugefügt wurden (Typ: ST_Dpv1ParamAddrEx [▶ 129]).

stDpv1ValueHeaderEx: Array von 39 Parameterwerten, die vom Antrieb gelesen wurden (Typ: ST_Dpv1ValueHeaderEx [▶ 130]).

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO Hardware	Einzubindende SPS-Bibliotheken (Kategoriegruppe)
TwinCAT v3.1.0	PC oder CX (x86)	Beckhoff EL6632	Tc2_IoFunctions (IO)

3.10.5 FB_Dpv1ReadPNET



Der Funktionsbaustein "FB_Dpv1ReadPNET" liest einen oder mehrere Parameter eines Sinamics Profidrive via DPV1 (Profidrive specification 3.1) via Profinet. Das DPV1 Lesetelegramm muss mit F_CreateDpv1ReadReqPkgPNET [▶ 80] erstellt werden, bevor an bExecute eine steigende Flanke ansteht. Das DPV1 Antworttelegramm muss mit F_SplitDpv1ReadResPkgPNET [▶ 82] ausgewertet werden, nach dem bBusy eine fallende Flanke anzeigt.

Die Ausführung dieses Funktionsbausteins benötigt einige Zeit, abhängig von der Anzahl der Parameter, die gelesen werden sollen. Der Funktionsbaustein sendet das DPV1 Telegramm und pollt nach einem Antworttelegramm.

VAR_INPUT

```
VAR_INPUT
    bExecute        : BOOL;(* drive access info *)
    aNetId          : T_AmsNetId; (* NetID of Profibus Master EL6631 *)
    iProfinetPort   : UINT; (* Port of ProfiDrive *)
    iDriveId        : USINT; (* 0..255 possible *)
    pDpv1ReqData    : POINTER TO ARRAY [1..iMAX_DPV1_SIZE_PNET_REQ] OF BYTE;
    iDpv1ReqDataLen : UDINT;
    pDpv1ResData    : POINTER TO ARRAY [1..iMAX_DPV1_SIZE_PNET_RES] OF BYTE;
```

```
    iDpv1ResDataLen : UDINT;
    tTmOut          : TIME;
END_VAR
```

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

aNetId: Die Netzwerkadresse des Profibus Master Gerätes (siehe ADS-Tab des Profibus Master Gerätes in der I/O-Konfiguration im TwinCAT System, Typ: T_AmsNetID).

iProfinetPort: Die Profinet Port-Nummer des Antriebs. Das ist eine Adresse für mehrere Achsen, spezifiziert im TwinCAT System in der I/O-Konfiguration.

iDriveID: Die ID ist 1 für die ControllerUnit, 2 für das Antriebsobjekt A, 3 für das Antriebsobjekt B eines Doppel/Dreifach-Antriebs. Die Drive ID wird in der Starter Software gesetzt. 1..16 ist möglich.

pDpv1ReqData: Zeiger auf ein Array von 240 Bytes, das das DPV1 Lesetelegramm enthält. Dieses Telegramm muss von der Funktion F_CreateDpv1ReadReqPkg [▶ 72] erstellt werden, bevor das DPV1 Lesen via bExecute aktiviert wird.

iDv1ReqDataLen: Maximale Länge des DPV1 Datapuffer (240 bytes).

pDpv1ResData: Zeiger auf ein Array von 240 Bytes, das das DPV1 Leseantworttelegramms enthält. Dieses Telegramm muss von der Funktion F_SplitDpv1ReadResPkg [▶ 73] ausgewertet werden nachdem auf bBusy eine negative Flanke erscheint.

iDv1ResDataLen: Maximale Länge des DPV1 Antwort-Datapuffers (240 bytes).

tTmOut: Maximale Zeit die bei der Ausführung des Kommandos nicht überschritten werden soll.

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy        : BOOL;
    bError       : BOOL;
    bErrorValues : BOOL;
    iErrId       : UDINT;
    iErrorClass  : BYTE;
    iErrorCode   : BYTE;
END_VAR
```

bBusy: Der Ausgang geht auf TRUE sobald der Funktionsbaustein via bExecute aktiviert wurde und bleibt so lange TRUE, wie der Baustein keine Antwort erhalten hat.

bError: Bei Fehlern geht der Ausgang auf TRUE und bBusy auf FALSE.

bErrorValues: Ist TRUE wenn der DPV1 Read nicht oder nur teilweise erfolgreich war. Die Fehlerursachen werden über die Fehler-ID geliefert (sowie Class und Code).

nErrId: Liefert die ADS Fehlernummer [▶ 154] oder funktionsbausteinspezifische Fehlernummern, wenn bError = TRUE.

nErrClass: Profidrive Fehlerklasse.

nErrCode: Profidrive Fehlercode.

Funktionbausteinspezifische Fehlercodes	Beschreibung
0x2	falsche Antwortreferenz
0x3	DPV1 Read fehlerhaft oder teilweise fehlerhaft
0x4	falsche Antwort-ID
other error IDs	siehe ADS Fehlercodes

Fehlerklassen	Beschreibung	Fehlercode
0x0 - 0x9	reserviert	-
0xA	Anwendungsfehler	0x0: Lesefehler 0x1: Schreibfehler 0x2: Modulfehler 0x3 - 0x7: reserviert 0x8: Versionskonflikt 0x9: nicht unterstützt 0xA - 0xF: benutzerabhängig
0xB	Zugriffsfehler	0x0: ungültiger Index (kein Datenblock DB47, Parameterzugriff wird nicht unterstützt) 0x1: Schreiblängenfehler 0x2: ungültiger Slot 0x3: Typkonflikt 0x4: ungültiger Bereich 0x5: Zustandskonflikt (Zugriff auf DB47 temporär nicht möglich wegen interner Prozesszustände) 0x6: Zugriff verweigert 0x7: ungültiger Bereich (Schreibfehler im DB47 Header) 0x8: ungültiger Parameter 0x9: ungültiger Typ 0xA - 0xF: benutzerabhängig
0xC	Resourcefehler	0x0: Lesekonflikt 0x1: Schreibkonflikt 0x2: Resource beschäftigt 0x3: Resource nicht erreichbar 0x4 - 0x7: reserviert 0x8 - 0xF: benutzerabhängig
0xD - 0xF	Benutzerdefinierte Fehler	-

VAR_IN_OUT

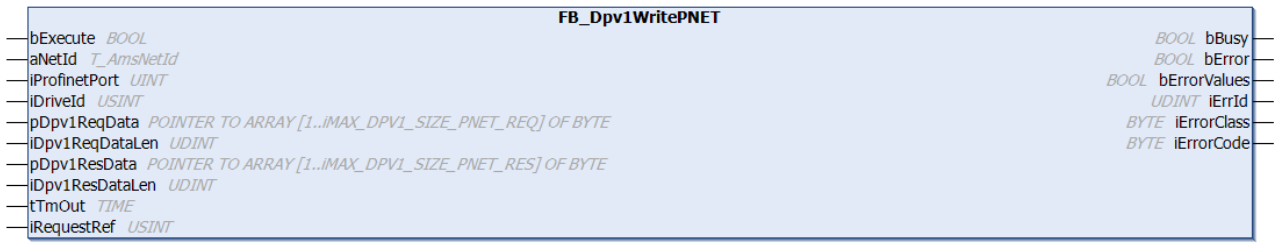
```
VAR_OUTPUT
    iRequestRef : USINT; (* 1..127; 0: reserved *)
END_VAR
```

iRefRequest: Referenz, die mit jedem Telegramm automatisch hochgezählt wird. Die Referenz wird für die Zuordnung der Antworten auf die Schreib/Lese-Anforderungen benötigt.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO Hardware	Einzubindende SPS-Bibliotheken (Kategoriegruppe)
TwinCAT v3.1.0	PC oder CX (x86)	Beckhoff EL6632	Tc2_IoFunctions (IO)

3.10.6 FB_Dpv1WritePNET



Der Funktionsbaustein "FB_Dpv1WritePNET" schreibt einen oder mehrere Parameter eines Sinamics Profidrive via DPV1 (Profidrive Specification 3.1) via Profinet. Das DPV1 Schreibtelegramm muss mit F_CreateDpv1WriteReqPkgPNET [▶ 81] erstellt werden, bevor an bExecute eine steigende Flanke ansteht. Das DPV1 Antworttelegramm muss mit F_SplitDpv1WriteResPkgPNET [▶ 82] ausgewertet werden, nachdem bBusy eine fallende Flanke anzeigt.

Die Ausführung dieses Funktionsbausteins benötigt einige Zeit, abhängig von der Anzahl der Parameter, die gelesen werden sollen. Der Funktionsbaustein sendet das DPV1 Telegramm und pollt nach einem Antworttelegramm.

VAR_INPUT

```
VAR_INPUT
    bExecute        : BOOL;(* drive access info *)
    aNetId          : T_AmsNetId; (* NetID of Profinet Master EL6631 *)
    iProfinetPort   : UINT; (* Port of ProfiDrive *)
    iDriveId        : USINT; (* 0..255 possible *)
    pDpv1ReqData    : POINTER TO ARRAY [1..iMAX_DPV1_SIZE_PNET_REQ] OF BYTE;
    iDpv1ReqDataLen : UDINT;
    pDpv1ResData    : POINTER TO ARRAY [1..iMAX_DPV1_SIZE_PNET_RES] OF BYTE;
    iDpv1ResDataLen : UDINT;
    tTmOut          : TIME;
END_VAR
```

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

aNetId: Die Netzwerkadresse des Profibus Master Gerätes (siehe ADS-Tab des Profibus Master Gerätes in der I/O-Konfiguration im TwinCAT System, Typ: T_AmsNetID).

iProfinetPort: Die Profinet Port Nummer des Antriebs. Das ist eine Adresse für mehrere Achsen, spezifiziert im TwinCAT System in der I/O-Konfiguration.

iDriveID: Die ID ist 1 für die ControllerUnit, 2 für das Antriebsobjekt A, 3 für das Antriebsobjekt B eines Doppel/Dreifach-Antriebs. Die Drive ID wird in der Starter Software gesetzt. 1..16 ist möglich.

pDpv1ReqData: Zeiger auf ein Array von 240 Bytes, das das DPV1 Schreibtelegramm enthält. Dieses Telegramm muss von der Funktion F_CreateDpv1WriteReqPkgPNET [▶ 81] erstellt werden, bevor das DPV1 Schreiben via bExecute aktiviert wird.

iDv1ReqDataLen: Maximale Länge des DPV1 Datapuffer (240 bytes).

pDpv1ResData: Zeiger auf ein Array von 240 Bytes, das das DPV1 Schreibantworttelegramms enthält. Dieses Telegramm muss von der Funktion F_SplitDpv1WriteResPkgPNET [▶ 82] ausgewertet werden nachdem auf dem bBusy eine negative Flanke erscheint.

iDv1ResDataLen: Maximale Länge des DPV1 Antwort-Datapuffers (240 bytes).

tTmOut: Maximale Zeit die bei der Ausführung des Kommandos nicht überschritten werden soll.

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy        : BOOL;
    bError       : BOOL;
    bErrorValues : BOOL;
    iErrId       : UDINT;
    iErrorClass  : BYTE;
    iErrorCode   : BYTE;
END_VAR
```

bBusy: Der Ausgang geht auf TRUE sobald der Funktionsbaustein via bExecute aktiviert wurde und bleibt so lange TRUE, wie der Baustein keine Antwort erhalten hat.

bError: Bei Fehlern geht der Ausgang auf TRUE und bBusy auf FALSE.

bErrorValues: Ist TRUE wenn der DPV1 Write nicht oder nur teilweise erfolgreich war. Die Fehlerursachen werden über die Fehler-ID geliefert (sowie Class und Code).

nErrId: Liefert die ADS Fehlernummer [▶ 154] oder funktionsbausteinspezifische Fehlernummern, wenn bError = TRUE.

nErrClass: Profidrive Fehlerklasse.

nErrCode: Profidrive Fehlercode.

Funktionbausteinspezifische Fehlercodes	Beschreibung
0x2	falsche Antwortreferenz
0x3	DPV1 Write fehlerhaft oder teilweise fehlerhaft
0x4	falsche Antwort-ID
other error IDs	siehe ADS Fehlercodes

Fehlerklassen	Beschreibung	Fehlercode
0x0 - 0x9	reserviert	-
0xA	Anwendungsfehler	0x0: Lesefehler 0x1: Schreibfehler 0x2: Modulfehler 0x3 - 0x7: reserviert 0x8: Versionskonflikt 0x9: nicht unterstützt 0xA - 0xF: benutzerabhängig
0xB	Zugriffsfehler	0x0: ungültiger Index (kein Datenblock DB47, Parameterzugriff wird nicht unterstützt) 0x1: Schreiblängenfehler 0x2: ungültiger Slot 0x3: Typkonflikt 0x4: ungültiger Bereich 0x5: Zustandskonflikt (Zugriff auf DB47 temporär nicht möglich wegen interner Prozesszustände) 0x6: Zugriff verweigert 0x7: ungültiger Bereich (Schreibfehler im DB47 Header) 0x8: ungültiger Parameter 0x9: ungültiger Typ 0xA - 0xF: benutzerabhängig
0xC	Resourcefehler	0x0: Lesekonflikt 0x1: Schreibkonflikt 0x2: Resource beschäftigt 0x3: Resource nicht erreichbar 0x4 - 0x7: reserviert 0x8 - 0xF: benutzerabhängig
0xD - 0xF	Benutzerdefinierte Fehler	-

VAR_IN_OUT

```
VAR_OUTPUT
    iRequestRef : USINT;(* 1..127; 0: reserved *)
END_VAR
```

iRefRequest: Referenz, die mit jedem Telegramm automatisch hochgezählt wird. Die Referenz wird für die Zuordnung der Antworten auf die Schreib/Lese-Anforderungen benötigt.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO Hardware	Einzubindende SPS-Bibliotheken (Kategoriegruppe)
TwinCAT v3.1.0	PC oder CX (x86)	Beckhoff EL6632	Tc2_IoFunctions (IO)

3.11 RAID Controller

3.11.1 FB_RAIDFindCntlr

Diese Funktion gibt den Zählerstand des RAID-Controllers und die entsprechenden Controller-IDs zurück.

HINWEIS

Der Funktionsbaustein sollte nur einmal in einem SPS-Programm aufgerufen werden!

Die Systemperformance kann durch zyklisches Aufrufen dieses Funktionsblocks dramatisch beeinflusst werden.

VAR_INPUT

```
VAR_INPUT
    sNETID    : T_AmsNetId;
    bWrtRd    : BOOL;
    tTimeOut  : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

sNETID: Ist ein String, der die AMS-Netzwerk-ID des Zielgerätes enthält, zu dem das ADS-Kommando geschickt wird (Typ: T_AmsNetID).

bWrtRd: Das ADS-Kommando wird von der steigenden Flanke dieses Inputs getriggert.

tTimeOut: Gibt die Timeout-Zeit an, die bei der Ausführung des Kommandos nicht überschritten werden darf.

VAR_OUTPUT

```
VAR_OUTPUT
    stRAIDCntlrFound : ST_RAIDCntlrFound;
    nBytesRead       : UDINT;
    bBusy            : BOOL;
    bError           : BOOL;
    nErrorID         : UDINT;
END_VAR
```

stRAIDCntlrFound: Enthält die Anzahl der gefundenen RAID-Controller und ihre RAID-Controller-Ids (Typ: ST_RAIDCntlrFound [▶ 136]).

nBytesRead: Anzahl der erfolgreich zurückgegebenen Datenbytes.

bBusy: Dieser Ausgang bleibt solange TRUE bis der Block ein Kommando ausgeführt hat, aber längstens für die Zeit, die am Timeout'-Input ansteht. Solange Busy = TRUE, wird kein neues Kommando an den Inputs angenommen werden. Bitte beachten Sie, dass es nicht die Ausführung des Dienstes ist, sondern die Zeit in der er ausgeführt werden darf, die hier gemonitort wird.

bError: Dieser Ausgang wird auf TRUE gesetzt, wenn ein Fehler während der Ausführung eines Kommandos auftritt. Der Kommando-spezifische Error-Code steht in 'nErrorId'. Wenn der Block einen Timeout-Error hat, wird 'bError' auf TRUE gesetzt und 'nErrorId' ist 1861 (hexadezimal 0x745). Es wird auf FALSE gesetzt, wenn ein Kommando an den Eingängen ausgeführt wird.

nErrorID: Enthält den Kommando-spezifischen Error-Code der zuletzt ausgeführten Kommandos, wird durch ein Kommando an den Eingängen auf 0 zurückgesetzt.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken (Kategoriegruppe)
TwinCAT v3.1.0	PC oder CX (x86, x64, ARM)	Tc2_IoFunctions (IO)

3.11.2 FB_RAIDGetInfo

Dieser Funktionsblock gibt ein RAID-Info zurück, das die Anzahl der RAID-Controller-Sets und die maximale Anzahl von Antrieben pro Set enthält.

HINWEIS

Der Funktionsbaustein sollte nur einmal in einem SPS-Programm aufgerufen werden!

Die Systemperformance kann durch zyklisches Aufrufen dieses Funktionsblocks dramatisch beeinflusst werden.

VAR_INPUT

```
VAR_INPUT
    sNETID       : T_AmsNetId;
    bWrtRd       : BOOL;
    nRAIDCntlrID : UDINT;
    tTimeOut     : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

sNETID: Ist ein String, der die AMS-Netzwerk-ID des Zielgerätes enthält, zu dem das ADS-Kommando geschickt wird (Typ: T_AmsNetID).

bWrtRd: Das ADS-Kommando wird von der steigenden Flanke dieses Inputs getriggert.

nRAIDCntlrID: Die RAID-Controller-ID. (Hinweis : kann mit FB_RAIDCntlrFind gelesen werden)

tTimeOut: Gibt die Timeout-Zeit an, die bei der Ausführung des Kommandos nicht überschritten werden darf.

VAR_OUTPUT

```
VAR_OUTPUT
    stRAIDInfo   : ST_RAIDInfo;
    nBytesRead   : UDINT;
    bBusy        : BOOL;
    bError       : BOOL;
    nErrorID     : UDINT;
END_VAR
```

stRAIDInfo: Gibt ein RAID-Info zurück, das die Anzahl von RAID-Controller-Sets und die maximale Anzahl von Antrieben pro Set enthält (Typ: ST_RAIDInfo [▶ 137]).

nBytesRead: Anzahl der erfolgreich zurückgegebenen Datenbytes.

bBusy: Dieser Ausgang bleibt solange TRUE bis der Block ein Kommando ausgeführt hat, aber längstens für die Zeit, die am Timeout'-Input ansteht. Solange Busy = TRUE, wird kein neues Kommando an den Inputs angenommen werden. Bitte beachten Sie, dass es nicht die Ausführung des Dienstes ist, sondern die Zeit in der er ausgeführt werden darf, die hier gemonitort wird.

bError: Dieser Ausgang wird auf TRUE gesetzt, wenn ein Fehler während der Ausführung eines Kommandos auftritt. Der Kommando-spezifische Error-Code steht in 'nErrorId'. Wenn der Block einen Timeout-Error hat, wird 'bError' auf TRUE gesetzt und 'nErrorId' ist 1861 (hexadezimal 0x745). Es wird auf FALSE gesetzt, wenn ein Kommando an den Eingängen ausgeführt wird.

nErrorID: Enthält den Kommando-spezifischen Error-Code der zuletzt ausgeführten Kommandos, wird durch ein Kommando an den Eingängen auf 0 zurückgesetzt.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken (Kategoriegruppe)
TwinCAT v3.1.0	PC oder CX (x86, x64, ARM)	Tc2_IoFunctions (IO)

3.11.3 FB_RAIDGetStatus

Dieser Funktionsblock gibt den RAID-Set-Index, den RAID-Typ, den RAID-Status, die Anzahl der RAID-Antriebe und den Status der RAID-Antriebe zurück.

HINWEIS

Höchstens 1mal pro Sekunde aufrufen!

Die Systemperformance kann durch zyklisches Aufrufen dieses Funktionsblocks dramatisch beeinflusst werden.

VAR_INPUT

```
VAR_INPUT
    sNETID          : T_AmsNetId;
    bWrtRd          : BOOL;
    stRAIDConfigReq : ST_RAIDConfigReq;
    tTimeOut        : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

sNETID: Ist ein String, der die AMS-Netzwerk-ID des Zielgerätes enthält, zu dem das ADS-Kommando geschickt wird (Typ: T_AmsNetID).

bWrtRd: Das Kommando wird von der steigenden Flanke dieses Inputs getriggert.

stRAIDConfigReq: RAID-Konfiguration Request-Parameters werden in dieser Struktur festgelegt (Typ: ST_RAIDConfigReq [▶ 136]). Sie enthält die Controller ID und den Index des RAID-Sets.

tTimeOut: Gibt die Timeout-Zeit an, die bei der Ausführung des Kommandos nicht überschritten werden darf.

VAR_OUTPUT

```
VAR_OUTPUT
    stRAIDStatusRes  : ST_RAIDStatusRes;
    nBytesRead       : UDINT;
    bBusy            : BOOL;
    bError           : BOOL;
    nErrorID         : UDINT;
END_VAR
```

stRAIDStatusRes: Diese Struktur enthält die RAID-Status-Antwort, den RAID-Set-Index, den RAID-Typ, den RAID-Status, die Anzahl der RAID-Antriebe, den Status der RAID-Antriebe (Typ: ST_RAIDStatusRes [▶ 137]).

nBytesRead: Anzahl der erfolgreich zurückgegebenen Datenbytes.

bBusy: Dieser Ausgang bleibt solange TRUE bis der Block ein Kommando ausgeführt hat, aber längstens für die Zeit, die am Timeout'-Input ansteht. Solange Busy = TRUE, wird kein neues Kommando an den Inputs angenommen werden. Bitte beachten Sie, dass es nicht die Ausführung des Dienstes ist, sondern die Zeit in der er ausgeführt werden darf, die hier gemonitort wird.

bError: Dieser Ausgang wird auf TRUE gesetzt, wenn ein Fehler während der Ausführung eines Kommandos auftritt. Der Kommando-spezifische Error-Code steht in 'nErrorId'. Wenn der Block einen Timeout-Error hat, wird 'bError' auf TRUE gesetzt und 'nErrorId' ist 1861 (hexadezimal 0x745). Es wird auf FALSE gesetzt, wenn ein Kommando an den Eingängen ausgeführt wird.

nErrorID: Enthält den Kommando-spezifischen Error-Code der zuletzt ausgeführten Kommandos, wird durch ein Kommando an den Eingängen auf 0 zurückgesetzt.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken (Kategoriegruppe)
TwinCAT v3.1.0	PC oder CX (x86, x64, ARM)	Tc2_IoFunctions (IO)

3.12 SERCOS

3.12.1 IOF_SER_GetPhase

Der Funktionsbaustein "IOF_SER_GetPhase" ermittelt die aktuelle Kommunikationsphase auf dem SERCOS-Ring. Die Kommunikationsphasen können die Werte von 0 bis 4 annehmen.

VAR_INPUT

```
VAR_INPUT
    NETID     : T_AmsNetId;
    DEVICEID  : UDINT;
    GET       : BOOL;
    TMOUT     : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

NETID: Hier kann die Netzwerkadresse des TwinCAT-Rechners angegeben werden, auf dem das ADS-Kommando durchgeführt werden soll (Typ: T_AmsNetID). Für den lokalen Rechner kann auch ein Leerstring angegeben werden.

DEVICEID: Über die DeviceId (Geräte-Id) wird der SERCOS-Master spezifiziert, dessen Kommunikationsphase ermittelt werden soll. Die Geräte-Ids werden während der Hardware-Konfiguration von TwinCAT-System festgelegt.

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

TMOUT: Gibt die Timeout-Zeit an, die bei der Ausführung des Kommandos nicht überschritten werden darf.

VAR_OUTPUT

```
VAR_OUTPUT
    BUSY    : BOOL;
    ERR     : BOOL;
    ERRID   : UDINT;
    PHASE   : BYTE;
END_VAR
```

BUSY: Bei der Aktivierung des Funktionsbausteins wird dieser Ausgang gesetzt und bleibt gesetzt, bis eine Rückmeldung erfolgt.

ERR: Sollte ein Fehler bei der Übertragung des Kommandos erfolgen, dann wird dieser Ausgang gesetzt, nachdem der BUSY-Ausgang zurückgesetzt wurde.

ERRID: Liefert bei einem gesetzten ERR-Ausgang die ADS-Fehlernummer [▶ 154].

PHASE: Die aktuelle Kommunikationsphase im SERCOS-Ring.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO Hardware	Einzubindende SPS-Bibliotheken (Kategoriegruppe)
TwinCAT v3.1.0	PC (x86)	Sercans SCS-P ISA; Sercans SCS-P PCI; Beckhoff FC750x PCI	Tc2_IoFunctions (IO)

3.12.2 IOF_SER_SaveFlash

Der Funktionsbaustein "IOF_SER_SaveFlash" überprüft die im DPRAM-Speicher stehenden Systemparameter. Wenn kein Fehler vorliegt aktiviert und speichert er sie ins EEPROM. Der Funktionsbaustein kann Systemparameter im EEPROM von der Steuerung passend zur Applikation einstellen.

HINWEIS

Das EEPROM besitzt eine maximale Wiederbeschreibbarkeit von 100 000 Schreibzyklen.

Die SPS sollte diesen Funktionsbaustein nicht automatisch, sondern nur durch den Anwender gezielt aktivieren.

VAR_INPUT

```
VAR_INPUT
    NETID     : T_AmsNetId;
    DEVICEID  : UDINT;
    SAVE      : BOOL;
    TMOUT     : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

NETID: Hier kann die Netzwerkadresse des TwinCAT-Rechners angegeben werden, auf dem das ADS-Kommando durchgeführt werden soll (Typ: T_AmsNetID). Für den lokalen Rechner kann auch ein Leerstring angegeben werden.

DEVICEID: Über die DeviceId (Geräte-Id) wird der SERCOS-Master spezifiziert, dessen Systemparameter gespeichert werden sollen. Die Geräte-Ids werden während der Hardware-Konfiguration vom TwinCAT-System festgelegt.

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

TMOUT: Gibt die Timeout-Zeit an, die bei der Ausführung des Kommandos nicht überschritten werden darf.

VAR_OUTPUT

```
VAR_OUTPUT
    BUSY    : BOOL;
    ERR     : BOOL;
    ERRID   : UDINT;
END_VAR
```

BUSY: Bei der Aktivierung des Funktionsbausteins wird dieser Ausgang gesetzt und bleibt gesetzt, bis eine Rückmeldung erfolgt.

ERR: Sollte ein Fehler bei der Übertragung des Kommandos erfolgen, dann wird dieser Ausgang gesetzt, nachdem der BUSY-Ausgang zurückgesetzt wurde.

ERRID: Liefert bei einem gesetzten ERR-Ausgang die ADS-Fehlernummer [▶ 154].

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO Hardware	Einzubindende SPS-Bibliotheken (Kategoriegruppe)
TwinCAT v3.1.0	PC (x86)	Sercans SCS-P ISA; Sercans SCS-P PCI; Beckhoff FC750x PCI	Tc2_IoFunctions (IO)

3.12.3 IOF_SER_ResetErr

Der Funktionsbaustein "IOF_SER_ResetErr" löscht folgende Fehler eines SERCOS-Masters:

- Die Fehler in den vorhandenen Antrieben;
- Den Diagnosestatus im Diagnosekanal der vorhandenen Antriebe;
- Den Systemfehler;

VAR_INPUT

```
VAR_INPUT
    NETID     : T_AmsNetId;
    DEVICEID  : UDINT;
    RESET     : BOOL;
    TMOUT     : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

NETID: Hier kann die Netzwerkadresse des TwinCAT-Rechners angegeben werden, auf dem das ADS-Kommando durchgeführt werden soll (Typ: T_AmsNetID). Für den lokalen Rechner kann auch ein Leerstring angegeben werden.

DEVICEID: Über die DeviceId (Geräte-Id) wird der SERCOS-Master spezifiziert, dessen Fehler gelöscht werden sollen. Die Geräte-Ids werden während der Hardware-Konfiguration von TwinCAT-System festgelegt.

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

TMOUT: Gibt die Timeout-Zeit an, die bei der Ausführung des Kommandos nicht überschritten werden darf.

VAR_OUTPUT

```
VAR_OUTPUT
    BUSY    : BOOL;
    ERR     : BOOL;
    ERRID   : UDINT;
END_VAR
```

BUSY: Bei der Aktivierung des Funktionsbausteins wird dieser Ausgang gesetzt und bleibt gesetzt, bis eine Rückmeldung erfolgt.

ERR: Sollte ein Fehler bei der Übertragung des Kommandos erfolgen, dann wird dieser Ausgang gesetzt, nachdem der BUSY-Ausgang zurückgesetzt wurde.

ERRID: Liefert bei einem gesetzten ERR-Ausgang die ADS-Fehlernummer [▶ 154].

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO Hardware	Einzubindende SPS-Bibliotheken (Kategoriegruppe)
TwinCAT v3.1.0	PC (x86)	Sercans SCS-P ISA; Sercans SCS-P PCI; Beckhoff FC750x PCI	Tc2_IoFunctions (IO)

3.12.4 IOF_SER_SetPhase

Der Funktionsbaustein "IOF_SER_SetPhase" führt den Phasenhochlauf im SERCOS-Ring auf einen bestimmten Wert durch.

VAR_INPUT

```
VAR_INPUT
    NETID     : T_AmsNetId;
    DEVICEID  : UDINT;
    PHASE     : BYTE;
    SET       : BOOL;
    TMOUT     : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

NETID: Hier kann die Netzwerkadresse des TwinCAT-Rechners angegeben werden, auf dem das ADS-Kommando durchgeführt werden soll (Typ: T_AmsNetID). Für den lokalen Rechner kann auch ein Leerstring angegeben werden.

DEVICEID: Über die DeviceId (Geräte-Id) wird der SERCOS-Master spezifiziert, dessen Kommunikationsphase gesetzt werden soll. Die Geräte-Ids werden während der Hardware-Konfiguration vom TwinCAT-System festgelegt.

PHASE: Die zu setzende Kommunikationsphase im SERCOS-Ring.

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

TMOUT: Gibt die Timeout-Zeit an, die bei der Ausführung des Kommandos nicht überschritten werden darf.

VAR_OUTPUT

```
VAR_OUTPUT
    BUSY    : BOOL;
    ERR     : BOOL;
    ERRID   : UDINT;
END_VAR
```

BUSY: Bei der Aktivierung des Funktionsbausteins wird dieser Ausgang gesetzt und bleibt gesetzt, bis eine Rückmeldung erfolgt.

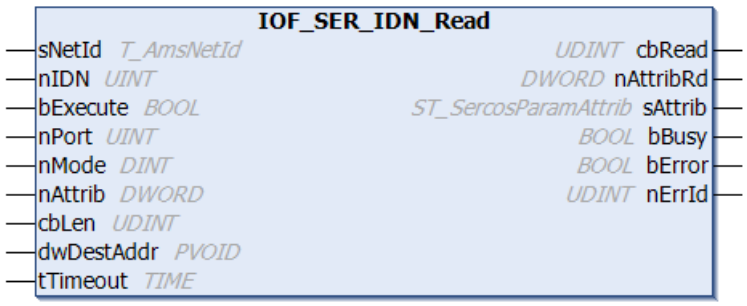
ERR: Sollte ein Fehler bei der Übertragung des Kommandos erfolgen, dann wird dieser Ausgang gesetzt, nachdem der BUSY-Ausgang zurückgesetzt wurde.

ERRID: Liefert bei einem gesetzten ERR-Ausgang die ADS-Fehlernummer [▶ 154].

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO Hardware	Einzubindende SPS-Bibliotheken (Kategoriegruppe)
TwinCAT v3.1.0	PC (x86)	Sercans SCS-P ISA; Sercans SCS-P PCI; Beckhoff FC750x PCI	Tc2_IoFunctions (IO)

3.12.5 IOF_SER_IDN_Read



Der Funktionsbaustein "IOF_SER_IDN_Read" erlaubt das Lesen eines Wertes aus einem S- oder P-Parameter eines Sercos-Antriebes. Datentyp und Größe werden automatisch anhand des Attributes bestimmt.

VAR_INPUT

```
VAR_INPUT
    sNetId      : T_AmsNetId;
    nIDN        : UINT;
    bExecute    : BOOL;
    nPort       : UINT;
    nMode       : DINT;
    nAttrib     : DWORD;
    cbLen       : UDINT;
    dwDestAddr  : PVOID;
    tTimeout    : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

sNetId: Hier kann die Netzwerkadresse des TwinCAT-Rechners angegeben werden, auf dem das ADS-Kommando durchgeführt werden soll (Typ: T_AmsNetID). Für den lokalen Rechner kann auch ein Leerstring angegeben werden.

nIDN: beinhaltet die Sercos-Parameter-Nummer, auf die lesend zugegriffen werden soll. Für S-Parameters muß nIDN zwischen 0 und 32767 liegen, für P-Parameters zwischen 32768 und 65535.

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

nPort: Die Port-Number nPort wird vom TwinCAT System während der Hardware-Konfiguration vergeben.

nMode: Der Lesemodus bestimmt, welcher Teil des Parameters gelesen werden soll.
nMode = 0: Wert
nMode = 2: Name
nMode = 3: Attribut (wird immer gelesen, um Datentyp und Größe zu bestimmen, es sei denn, nAttrib ist <> 0)
nMode = 4: Einheit (nicht für jeden Parameter verfügbar)
nMode = 5: Minimum (nicht für jeden Parameter verfügbar)
nMode = 6: Maximum (nicht für jeden Parameter verfügbar)

nAttrib: Attribut des Parameter, wenn es bekannt ist. Wenn nAttrib = 0 dann liest IOF_SER_IDN_Write erst das Parameter-Attribut vom Antrieb, bevor der Wert in den Parameter des Antriebs geschrieben wird.

cbLen: Maximale Länge des Datapuffers, der den Wert aufnehmen soll.

dwDestAddr: Adresse des Datapuffers, der den Wert aufnehmen soll.

tTimeout: Gibt die Timeout-Zeit an, die bei der Ausführung des Kommandos nicht überschritten werden darf.

VAR_OUTPUT

```
VAR_OUTPUT
    cbRead    : UDINT;
    nAttribRd : DWORD;
    sAttrib   : ST_SercosParamAttrib;
    bBusy     : BOOL;
    bError    : BOOL;
    nErrId    : UDINT;
END_VAR
```

cbRead: Anzahl der gelesenen und nach dwDestAddr kopierten Bytes.

nAttribRd: Attribut des Parameter, kann für den nächsten Zugriff (nAttrib) auf den Parameter gespeichert werden.
sAttrib: beinhaltet das Attribut nAttribRd des Sercos-Parameters in einzelne Variablen zerlegt (Typ: ST_SercosParamAttrib [▶ 138]).
bBusy: Bei der Aktivierung des Funktionsbausteins wird dieser Ausgang gesetzt und bleibt gesetzt, bis eine Rückmeldung erfolgt.

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

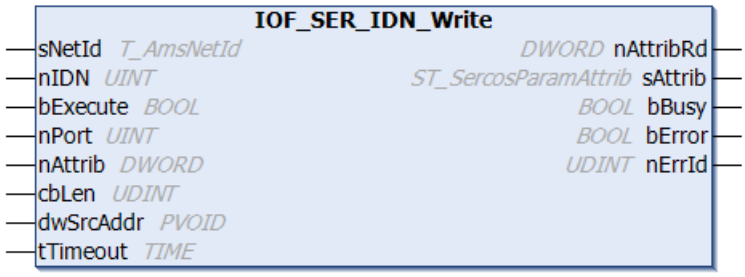
nErrId: Liefert bei einem gesetzten ERR-Ausgang die ADS-Fehlernummer [▶ 154] bzw. die spezifische Funktionsbaustein-Fehlernummer.

spezifische Functionsbaustein- Fehlernummer	Beschreibung
0x1003	Falscher Parameter-Mode
0x1004	Falsche Parameterdatengröße

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO Hardware	Einzubindende SPS-Bibliotheken (Kategoriegruppe)
TwinCAT v3.1.0	PC (x86)	Sercans SCS-P ISA; Sercans SCS-P PCI; Beckhoff FC750x PCI	Tc2_IoFunctions (IO)

3.12.6 IOF_SER_IDN_Write



Der Funktionsbaustein "IOF_SER_IDN_Write" erlaubt das Schreiben eines Wertes in einen S- oder P-Parameter eines Sercos-Antriebes. Datentyp und Größe werden automatisch anhand des Attributes bestimmt.

VAR_INPUT

```
VAR_INPUT
    sNetId    : T_AmsNetId;
    nIDN      : UINT; (* S: 0******* ********, P: 1******* ******** *)
    bExecute  : BOOL;
    nPort     : UINT;
    nAttrib   : DWORD;
    cbLen     : UDINT;
    dwSrcAddr : PVOID;
    tTimeout  : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

sNetId: Hier kann die Netzwerkadresse des TwinCAT-Rechners angegeben werden, auf dem das ADS-Kommando durchgeführt werden soll (Typ: T_AmsNetID). Für den lokalen Rechner kann auch ein Leerstring angegeben werden.

nIDN: beinhaltet die Sercos-Parameter-Nummer, auf die schreibend zugegriffen werden soll. Für S-Parameters muss nIDN zwischen 0 und 32767 liegen, für P-Parameters zwischen 32768 und 65535.

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

nPort: Die Port-Number nPort wird vom TwinCAT System während der Hardware-Konfiguration vergeben.

nAttrib: Attribut des Parameter, wenn es bekannt ist. Wenn nAttrib = 0 dann liest IOF_SER_IDN_Write erst das Parameter-Attribut vom Antrieb, bevor der Wert in den Parameter des Antriebs geschrieben wird.

cbLen: Länge des Datapuffers, der den Wert enthält.

dwSrcAddr: Adresse des Datapuffers, der den Wert enthält.

tTimeout: Gibt die Timeout-Zeit an, die bei der Ausführung des Kommandos nicht überschritten werden darf.

VAR_OUTPUT

```
VAR_OUTPUT
    nAttribRd : DWORD;
    sAttrib   : ST_SercosParamAttrib;
    bBusy     : BOOL;
    bError    : BOOL;
    nErrId    : UDINT;
END_VAR
```

nAttribRd: Attribut des Parameter, kann für den nächsten Zugriff (nAttrib) auf den Parameter gespeichert werden.

sAttrib: beinhaltet das Attribut nAttribRd des Sercos-Parameters in einzelne Variablen zerlegt (Typ: ST_SercosParamAttrib [▶ 138]).

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

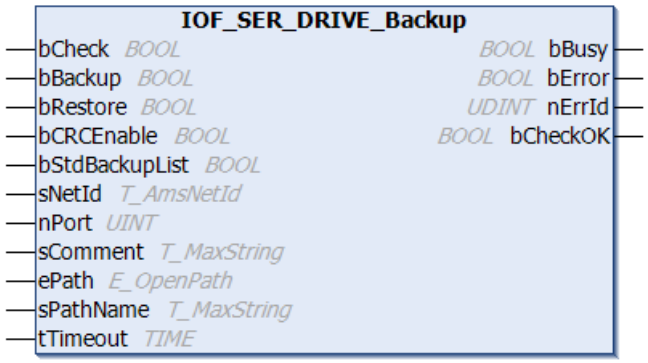
bError: Sollte ein Fehler bei der Übertragung des Kommandos erfolgen, dann wird dieser Ausgang gesetzt, nachdem der bBusy-Ausgang zurückgesetzt wurde.

nErrId: Liefert bei einem gesetzten ERR-Ausgang die ADS-Fehlernummer [▶ 154].

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO Hardware	Einzubindende SPS-Bibliotheken (Kategoriegruppe)
TwinCAT v3.1.0	PC (x86)	Sercans SCS-P ISA; Sercans SCS-P PCI; Beckhoff FC750x PCI	Tc2_IoFunctions (IO)

3.12.7 IOF_SER_DRIVE_Backup



Der Funktionsbaustein "IOF_SER_DRIVE_Backup" erlaubt das Backup und Restore der Antriebsdaten (S- und P-Parameter) von der SPS in eine Binärdatei. Die Liste der zu sichernden S- und P-Parameterdaten wird standardmäßig dem Sercos-Parameter IDN192 entnommen. Backup und Restore erfordern den SERCOS-Parameter-Mode (Phase 2).

Wenn bStdBackupList = TRUE (Standard) dann wird der Parameter IDN192 als Liste der zu sichernden Daten genommen, anderenfalls IDN17, die Liste aller Sercos-Parameter. Restore erfordert eine Backup-Datei, die mit Parameter IDN192 erstellt wurde, da einige Parameter der Liste IDN17 schreibgeschützt sind.

Backup und Restore erzeugen eine CRC16-CCITT und eine 16 bit Check-Summe und speichern diese in Parameter IDN142, wenn verfügbar.

Das Dateiformat der Backup-Datei ist in Backup-Dateiformat [▶ 147] beschrieben.

VAR_INPUT

```
VAR_INPUT
    bCheck        : BOOL;
    bBackup       : BOOL;
    bRestore      : BOOL;
    bCRCEnable    : BOOL := TRUE;
    bStdBackupList: BOOL := TRUE;
    sNetId        : T_AmsNetId;
    nPort         : UINT;
    sComment      : T_MaxString;
    ePath         : E_OpenPath := PATH_BOOTPATH;
    sPathName     : T_MaxString := 'DRIVEPAR.BIN';
    tTimeout      : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

bCheck: Über eine positive Flanke an diesem Eingang wird die Überprüfung per CRC und Checksumme aktiviert. CRC und Checksumme werden persistent und im Parameter IDN142 nach einem Backup oder Restore gespeichert. Wenn der Wert aus Parameter IDN142 und die persistenten Daten übereinstimmen wird bCheckOK auf TRUE gesetzt, anderenfalls wird bCheckOK auf FALSE gesetzt.

bBackup: Über eine positive Flanke an diesem Eingang wird das Backup aktiviert.

bRestore: Über eine positive Flanke an diesem Eingang wird das Restore aktiviert.

bCRCEnable: Der CRC16-CCITT und die 16 bit Checksumme werden über bCRCEnable = TRUE aktiviert. Die CRC und die Checksumme werden in Parameter IDN142 gespeichert, wenn bCRCEnable = TRUE.

bStdBackupList: bestimmt, welche Parameterliste für das Backup benutzt wird. Standardmäßig wird IDN192 (bStdBackupList = TRUE) für das Backup benutzt, wenn bStdBackupList = FALSE dann wird die Liste aller Parameter IDN017 benutzt. Restore benötigt eine Backup-Datei, die mit der Liste IDN192 erzegt wurde.

sNetId: Hier kann die Netzwerkadresse des TwinCAT-Rechners angegeben werden, auf dem das ADS-Kommando durchgeführt werden soll (Typ: T_AmsNetID). Für den lokalen Rechner kann auch ein Leerstring angegeben werden.

nPort: Die Port-Number nPort wird vom TwinCAT System während der Hardware-Konfiguration vergeben.

sComment: ist ein Kommentar, der in den Datei-Header der Backup-Datei geschrieben wird (Typ: T_MaxString).

ePath: bestimmt den Pfad der Backup-Datei. Wenn ePath = PATH_BOOTPATH dann wird der TwinCAT BOOT-Pfad genommen, bei ePath = PATH_GENERIC wird der in sPathName spezifizierte Pfad genommen (Typ: E_OpenPath).

sPathName: beinhaltet den Dateinamen (bei Verwendung des Boot-Pfades) bzw. den kompletten Pfad und Dateinamen bei Verwendung des generischen Pfades (Typ: T_MaxString).

tTimeout: Gibt die Timeout-Zeit an, die bei der Ausführung des Kommandos nicht überschritten werden darf.

VAR_OUTPUT

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

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

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

nErrId: Liefert bei einem gesetzten ERR-Ausgang die ADS-Fehlernummer [▶ 154] bzw. die spezifische Funktionsbaustein-Fehlernummer.

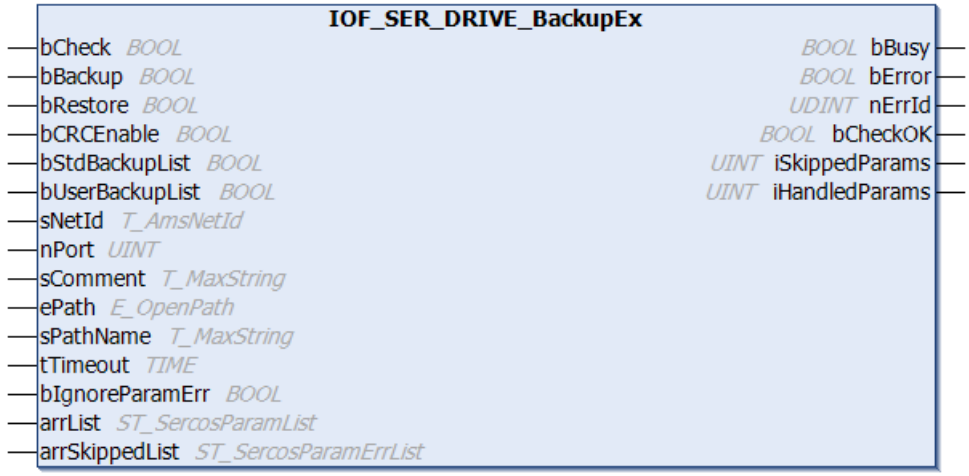
spezifische Functionsbaustein- Fehlernummer	Beschreibung
0x1003	Falscher Parameter-Mode
0x1004	Falsche Parameterdatengröße
0x1005	Falscher Backup Parameter Typ
0x1006	Backup Parameterliste war nicht IDN 192

bCheckOk: Ist WAHR, wenn der Checksummen-Test erfolgreich war.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO Hardware	Einzubindende SPS-Bibliotheken (Kategoriegruppe)
TwinCAT v3.1.0	PC (x86)	Sercans SCS-P ISA; Sercans SCS-P PCI; Beckhoff FC750x PCI	Tc2_IoFunctions (IO)

3.12.8 IOF_SER_DRIVE_BackupEx



Der Funktionsbaustein "IOF_SER_DRIVE_BackupEx" erlaubt das Backup (Sichern) und Restore (Wiederherstellen) der Antriebsdaten (S- und P-Parameter) über die SPS in eine Binärdatei bzw. zurück in den Antrieb. Die Liste der zu sichernden S- und P-Parameterdaten wird standardmäßig dem Sercos-Parameter IDN192 entnommen. Backup und Restore erfordern den SERCOS-Parameter-Mode (Phase 2).

Wenn bStdBackupList = TRUE (Standard) ist, dann wird der Parameter IDN192 als Liste der zu sichernden Daten genommen.
Wenn bUserBackupList = TRUE ist, dann wird der die Parameterliste arrList als Liste der zu sichernden Daten genommen.
Anderenfalls wird IDN17 verwendet, die Liste aller Sercos-Parameter.

Restore erfordert eine Backup-Datei, die mit Parameter IDN192 oder mit einer Userparameterliste erstellt wurde. Einige Parameter der Liste IDN17 sind schreibgeschützt.

Backup und Restore können eine CRC16-CCITT und eine 16 bit Check-Summe erstellen und speichern diese in Parameter IDN142, wenn verfügbar. Die Option bCRCEnable ist standardmäßig deaktiviert (FALSE).

Das Dateiformat der Backup-Datei ist in Backup-Dateiformat [▶ 147] beschrieben.

VAR_INPUT

```
VAR_INPUT
    bCheck          : BOOL;
    bBackup         : BOOL;
    bRestore        : BOOL;
    bCRCEnable      : BOOL;
    bStdBackupList  : BOOL     := TRUE;
    bUserBackupList : BOOL;
    sNetId          : T_AmsNetId;
    nPort           : UINT;
    sComment        : T_MaxString;
    ePath           : E_OpenPath := PATH_BOOTPATH;
    sPathName       : T_MaxString := 'DRIVEPAR.BIN';
    tTimeout        : TIME := DEFAULT_ADS_TIMEOUT;
    bIgnoreParamErr : BOOL;
END_VAR
```

bCheck: Über eine positive Flanke an diesem Eingang wird die Überprüfung per CRC und Checksumme aktiviert. CRC und Checksumme werden persistent und im Parameter IDN142 nach einem Backup oder Restore gespeichert. Wenn der Wert aus Parameter IDN142 und die persistenten Daten übereinstimmen wird bCheckOK auf TRUE gesetzt, anderenfalls wird bCheckOK auf FALSE gesetzt.

bBackup: Über eine positive Flanke an diesem Eingang wird das Backup aktiviert.

bRestore: Über eine positive Flanke an diesem Eingang wird das Restore aktiviert.

bCRCEnable: Der CRC16-CCITT und die 16 bit Checksumme werden über bCRCEnable = TRUE (WAHR) aktiviert. Die CRC und die Checksumme werden in Parameter IDN142 gespeichert, wenn bCRCEnable = TRUE.

bStdBackupList: Bestimmt, welche Parameterliste für das Backup benutzt wird. Standardmäßig wird IDN192 (bStdBackupList = TRUE) für das Backup benutzt, wenn bStdBackupList = FALSE (FALSCH) ist, dann wird die Liste aller Parameter IDN017 benutzt. Restore benötigt eine Backup-Datei, die mit der Liste IDN192 erzegt wurde.

bUserBackupList: Bestimmt, ob eine benutzerdefinerte Parameterliste arrList für das Backup benutzt wird. Standardmäßig wird IDN192 (bStdBackupList = TRUE) für das Backup benutzt, wenn bStdBackupList = FALSE und bUserBackupList = TRUE ist, dann wird die Liste arrList benutzt. Restore benötigt eine Backup-Datei, die mit der Liste IDN192 oder einer benutzerdefinierten Parameterliste erzegt wurde.

sNetId: Hier kann die Netzwerkadresse des TwinCAT-Rechners angegeben werden, auf dem das ADS-Kommando durchgeführt werden soll (Typ: T_AmsNetID). Für den lokalen Rechner kann auch ein Leerstring angegeben werden.

nIDN: Beinhaltet die Sercos-Parameter-Nummer, auf die schreibend zugegriffen werden soll. Für S-Parameters muß nIDN zwischen 0 und 32767 liegen, für P-Parameters zwischen 32768 und 65535.

nPort: Die Port-Number nPort wird vom TwinCAT System während der Hardware-Konfiguration vergeben.

sComment: Ist ein Kommentar, der in den Datei-Header der Backup-Datei geschrieben wird (Typ: T_MaxString).

ePath: Bestimmt den Pfad der Backup-Datei (Typ: E_OpenPath). Wenn ePath = PATH_BOOTPATH dann wird der TwinCAT BOOT-Pfad genommen, bei ePath = PATH_GENERIC wird der in sPathName spezifizierte Pfad genommen.

sPathName: Beinhaltet den Dateinamen (bei Verwendung des Boot-Pfades) bzw. den kompletten Pfad und Dateinamen bei Verwendung des generischen Pfades (Typ: T_MaxString).

tTimeout: Gibt die Timeout-Zeit an, die bei der Ausführung des Kommandos nicht überschritten werden darf.

bIgnoreParamErr: Bestimmt, ob bei Parameterlese- oder Parameterschreibfehlern das Backup/Restore fortgeführt oder abgebrochen werden soll. Standardmäßig wird bei Fehlern abgebrochen (bIgnoreParamErr = FALSE). Ist das Ignorieren von Fehlern aktiviert (bIgnoreParamErr = TRUE), dann werden in der Liste der übersprungenen Parameter arrSkippedList die Parameternummer und die Fehlernummern hinterlegt.

VAR_IN_OUT

```
VAR_IN_OUT
    arrList        : ST_SercosParamList;
    arrSkippedList : ST_SercosParamErrList;
END_VAR
```

arrList: Bei Standardbackup über IDN192 (bStdBackupList = TRUE) stehen in dieser Liste nach dem Backup die Backup-Parameter aus IDN192. Bei benutzerdefiniertem Backup (bUserBackupList = TRUE und bStdBackupList = FALSE) müssen in dieser Liste vor dem Backup die Liste der zu sichernden Parameter stehen. Bei Backup über IDN17 (bStdBackupList = FALSE und bStdBackupList = FALSE) stehen in dieser Liste nach dem Backup die Liste vorhandenen Parameter aus IDN17 (Typ: ST_SercosParamList [▶ 139]).

arrSkippedList: Enthält eine Liste der übersprungenen Parameter (die Parameternummer und die Fehlernummern, Typ: ST_SercosParamErrList [▶ 139]).

Siehe Strukturdefinitionen unten.

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy          : BOOL;
    bError         : BOOL;
    nErrId         : UDINT;
    bCheckOK       : BOOL;
    iSkippedParams : UINT;
    iHandledParams : UINT;
END_VAR
```

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

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

nErrId: Liefert bei einem gesetzten ERR-Ausgang die ADS-Fehlernummer [▶ 154] bzw. die spezifische Funktionsbaustein-Fehlernummer.

spezifische Functionsbaustein- Fehlernummer	Beschreibung
0x1003	Falscher Parameter-Mode
0x1004	Falsche Parameterdatengröße
0x1005	Falscher Backup Parameter Typ
0x1006	Backup Parameterliste war nicht IDN 192 oder benutzerdefiniert

bCheckOk: Ist WAHR, wenn der Checksummen-Test erfolgreich war.

iSkippedParams: Enthält die Anzahl der übersprungenen Parameter (siehe arrSkippedList), falls das Ignorieren der Parameterfehler aktiv war (bIgnoreParamErr = TRUE).

iHandledParams: Enthält die Anzahl der erfolgreich gesicherten/wiederhergestellten Parameter.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO Hardware	Einzubindende SPS-Bibliotheken (Kategoriegruppe)
TwinCAT v3.1.0	PC (x86)	Sercans SCS-P ISA; Sercans SCS-P PCI; Beckhoff FC750x PCI	Tc2_IoFunctions (IO)

3.12.9 IOF_SER_DRIVE_Reset

Der Funktionsbaustein "IOF_SER_DRIVE_Reset" führt einen Antriebs-Reset eines Sercos-Antriebes durch. Antriebsfehler werden gelöscht.

VAR_INPUT

```
VAR_INPUT
    sNetId    : T_AmsNetId;
    nPort     : UINT;
    bReset    : BOOL;
    tTimeout  : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

sNetId: Hier kann die Netzwerkadresse des TwinCAT-Rechners angegeben werden, auf dem das ADS-Kommando durchgeführt werden soll (Typ: T_AmsNetID). Für den lokalen Rechner kann auch ein Leerstring angegeben werden.

nPort: Die Port-Number nPort wird vom TwinCAT System während der Hardware-Konfiguration vergeben.

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

tTimeout: Gibt die Timeout-Zeit an, die bei der Ausführung des Kommandos nicht überschritten werden darf.

VAR_OUTPUT

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

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

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

nErrId: Liefert bei einem gesetzten ERR-Ausgang die ADS-Fehlernummer [▶ 154].

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO Hardware	Einzubindende SPS-Bibliotheken (Kategoriegruppe)
TwinCAT v3.1.0	PC (x86)	Sercans SCS-P ISA; Sercans SCS-P PCI; Beckhoff FC750x PCI	Tc2_IoFunctions (IO)

3.13 TcTouchLock

TcTouchLock ist eine Funktion der Beckhoff IPC Panel.

Weitere Informationen dazu finden Sie in der Dokumentation Touch_Lock.

Die Hardware-Voraussetzungen finden Sie ebenfalls dort, in dem Kapitel Voraussetzungen.

3.13.1 FB_TcTouchLock_AcquireFocus

Der Funktionsbaustein FB_TcTouchLock_AcquireFocus dient der Vermeidung von parallelen, sich störenden Eingaben über mehrere Multitouch-Control-Panel, die an einen IPC angeschlossen sind. Zu diesem Zweck wird ein Fokus auf eines der angeschlossenen Control-Panel gelegt und damit die Eingabe über alle anderen angeschlossenen Control-Panel gesperrt. Mit dem Funktionsbaustein FB_TcTouchLock_AcquireFocus kann dieser Fokus angefordert und freigegeben werden.

Wird der Fokus an einem Multitouch-Control-Panel angefordert, wenn ein anderes Multitouch-Control-Panel ihn derzeit besitzt, so muss der Fokus von diesem zunächst frei gegeben werden. Sobald die Freigabe erfolgt ist, wird der Fokus automatisch auf das im Wartezustand befindliche Gerät gesetzt.

Die Multitouch-Control-Panel, auf die durch den Funktionsblock zugegriffen werden, müssen zuvor durch die Kommandozeilen-Applikation TcTouchLockService.exe konfiguriert werden. Dabei muss jedem Gerät eine spezifische Identifikationsnummer zugewiesen werden.

VAR_INPUT

```
VAR_INPUT
    bEnable     : BOOL;
    sSetID      : STRING(32);
    tLEDTime    : TIME := 200;
END_VAR
```

bEnable: TRUE = Fokus anfordern, FALSE = Fokus abgeben

sSetID: ID des Gerätes

tLEDTime: Die Ausgabe-LED blinkt im festgelegten Intervall (100ms – 1s), während der Fokus angefordert wird

VAR_OUTPUT

```
VAR_OUTPUT
    bAcquired  : BOOL := FALSE;(* Focus status information *)
    bLED       : BOOL := FALSE;(* LED control output *)
    bBusy      : BOOL;(* TRUE => function in progress *)
    bError:    : BOOL;(* Error flag *)
    nErrID     : UDINT;(* Error code *)
END_VAR
```

bAcquired: TRUE, wenn der Client den Fokus besitzt und FALSE, wenn er ihn verliert.

bLED: Dieser Ausgang hat je nach Modus folgende Bedeutung:

Modus	Bedeutung
Konstant TRUE	Das Panel besitzt den Fokus
Konstant FALSE	Das Panel besitzt den Fokus nicht
Toggelt	Das Panel wartet darauf den Fokus zu erhalten

bBusy: TRUE solange der Funktionsbaustein aktiv ist.

bError: TRUE, wenn ein ADS-Fehler bei der Übertragung des Kommandos auftritt. Der bBusy-Ausgang wird zuvor zurückgesetzt.

nErrId: Liefert bei einem gesetzten *bError*-Ausgang die ADS-Fehlernummer [▶ 154] oder einen befehlsspezifischen Fehlercode zurück (Tabelle).

Fehlercodes	Fehlerbeschreibung
0x0000	Kein Fehler
0x0006	Ziel-Port nicht gefunden

Beispiel: Touch-Focus über Sondertaste steuern

Das manuelle Setzen des Fokus kann z.B. über eine Sondertaste des Panels gesteuert werden. Da der Fokus auch bei gesperrter Eingabe über den Touchscreen angefordert werden soll, ist eine Eingabemöglichkeit außerhalb des sperrbaren Touchscreens zu berücksichtigen. Über den TwinCAT System Manager wird die Sondertaste mit der entsprechenden Input-Variable des PLC Programms verknüpft. Pro Panel wird eine FB_TcTouchLock_AquireFocus Instanz erzeugt und mit der ID des Panels konfiguriert. Nach Drücken der Sondertaste an einem Panel, wobei der Baustein R_TRIG die steigende Flanke detektiert, versucht das PLC Programm über die entsprechende FB_TcTouchLock_AquireFocus Instanz den Touch-Fokus zu setzen. Der Funktionsbaustein kann außerdem einen Ausgang ansteuern (z.B. eine LED), der signalisiert, ob der Touch-Fokus erfolgreich gesetzt wurde oder ob noch versucht wird den Fokus zu holen. Erneutes Drücken der Sondertaste setzt den Touch-Fokus wieder zurück und ermöglicht damit ein Setzen des Touch-Fokus an anderen Panels.

Das PLC-Programm sieht für zwei Panels folgendermaßen aus:

```
PROGRAM MAIN
VAR
    button1 AT%IX0.0 : BOOL;
    button2 AT%IX0.1 : BOOL;

    led1 AT%QX0.0 : BOOL;
    led2 AT%QX0.1 : BOOL;

    fbPanel1 : FB_TcTouchLock_AcquireFocus := ( sSetID := 'A' );
    fbPanel2 : FB_TcTouchLock_AcquireFocus := ( sSetID := 'B' );

    trigger1 : R_TRIG;
    trigger2 : R_TRIG;
END_VAR
```

```
(* Panel 1 *)
trigger1( CLK := button1 );
IF trigger1.Q THEN
fbPanel1.bEnable := NOT fbPanel1.bEnable;
END_IF
fbPanel1(bLED=>LED1);

(* Panel 2 *)
trigger2( CLK := button2 );
IF trigger2.Q THEN
fbPanel2.bEnable := NOT fbPanel2.bEnable;
END_IF
fbPanel2(bLED=>LED2 );
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken (Kategoriegruppe)
TwinCAT v3.1. >= 4022.31	PC oder CX (x86, x64)	Tc2_IoFunctions (IO)

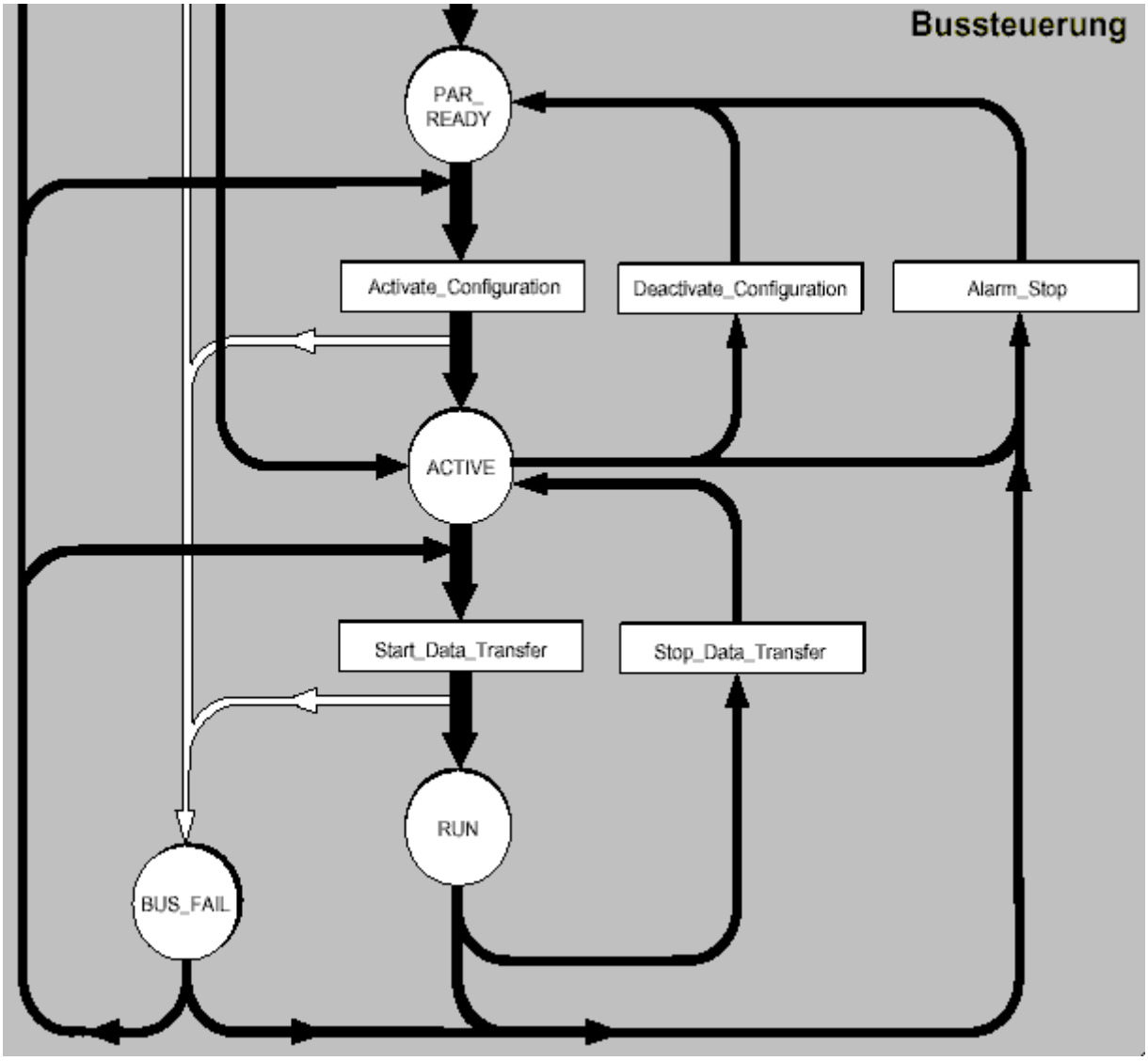
3.14 Drittherstellergeräte

3.14.1 Phoenix IBS SC/I-T

3.14.1.1 Übersicht

Die Bibliothek bietet eine komfortable Möglichkeit die wichtigsten Firmwaredienste der Phoenix Interbuskarte IBS SC/I-T zur Bussteuerung von der TwinCAT PLC auszuführen. Im folgenden Bild sind die Zustände und Übergangsbedingungen der Bussteuerung dargestellt.

Bussteuerung



SCIT_ActivateConfiguration [▶ 107]: Führt den Befehl **Activate_Configuration** aus.

SCIT_DeactivateConfiguration [▶ 108]: Führt den Befehl **Deactivate_Configuration** aus.

SCIT_StartDataTransfer [▶ 109]: Führt den Befehl **Start_Data_Transfer** aus.

SCIT_StopDataTransfer [▶ 110]: Führt den Befehl **Stop_Data_Transfer** aus.

SCIT_AlarmStop [▶ 111]: Führt den Befehl **Alarm_Stop** aus.

Konfiguration

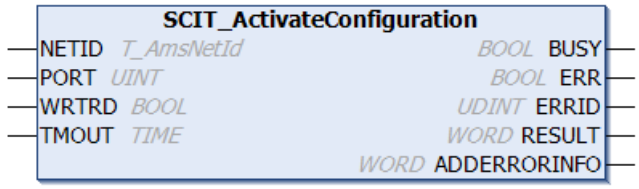
SCIT_ControlActiveConfiguration [▶ 112]: Dient zur Beeinflussung der aktiven Konfiguration der Interbus-Teilnehmer. Dieses Kommando kann sowohl im Zustand *PAR_READY* als auch im Zustand *ACTIVE* und *RUN* ausgeführt werden. Hierüber können einzelne, abhängige und gruppierte Teilnehmer aktiviert und deaktiviert werden.

Fehlerdiagnose

SCIT_GetErrorInfo [▶ 113]: Liefert Fehlerart und Fehlerort eines Interbus-Teilnehmers nach einem Busfehler.

SCIT_ConfDevErrAll [▶ 114]: Peripheriestörungen aller Geräte quittieren.

3.14.1.2 SCIT_ActivateConfiguration



Der Funktionsbaustein "SCIT_ActivateConfiguration" dient als Hilfsbaustein um einen **Activate_Configuration** auf der Interbuskarte durchzuführen, die mit der NETID und dem PORT adressiert wird. Durch einen **Activate_Configuration** wird die Karte in den Zustand ACTIVE [▶ 106] versetzt.

VAR_INPUT

```
VAR_INPUT
    NETID    : T_AmsNetId;
    PORT     : T_AmsPort;
    WRTRD    : BOOL;
    TMOUT    : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

NETID: Hier kann die Netzwerkadresse des Rechners angegeben werden, in dem die Karte eingebaut ist (Typ: T_AmsNetID). Befindet sich die Karte auf dem selben System kann auch ein Leerstring angegeben werden.

PORT: Beinhaltet die ADS-Portnummer der Karte, die vom TwinCAT System vergeben wurde (Typ: T_AmsPort).

WRTRD: Über eine positive Flanke an diesem Wert wird der Baustein aktiviert.

TMOUT: Maximale Zeit die bei der Ausführung des Kommandos nicht überschritten werden soll.

VAR_OUTPUT

```
VAR_INPUT
    BUSY       : BOOL;
    ERR        : BOOL;
    ERRID      : UDINT;
    RESULT     : WORD;
    ADDERRINFO : WORD;
END_VAR
```

BUSY: Nach dem Aktivieren des Bausteins liegt das Busy-Signal solange an, bis eine Rückmeldung erfolgt.

ERR: Sollte ein Fehler bei der Übertragung des Kommandos erfolgen wird der ERR Ausgang nach Rücknahme des BUSY-Signals gesetzt.

ERRID: Liefert beim Fehler die ADS-Fehlernummer [▶ 154].

RESULT: Liefert das Ergebnis von der Karte zurück (Voraussetzung ist ein fehlerfreier ADS-Transport (ERR = FALSE)). RESULT = 0 kennzeichnet eine erfolgreiche Ausführung des Befehls. Ein Wert ungleich 0 beinhaltet die Fehlernummer der Phoenixkarte.

ADDERRINFO: Enthält bei negativer Rückmeldung der Karte zusätzliche Fehlerinformationen (vgl. Befehlsbeschreibung der Phoenixkarte).

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO-Hardware	Einzubindende SPS-Bibliotheken (Kategoriegruppe)
TwinCAT v3.1.0	Keine, diese Funktionalität wird zur Zeit von TwinCAT 3 nicht unterstützt!	Phoenix: IBS SC/I-T; IBS SC/RI/RT-LK	Tc2_IoFunctions (IO)

3.14.1.3 SCIT_DeactivateConfiguration

Der Funktionsbaustein "SCIT_DeactivateConfiguration" dient als Hilfsbaustein um einen **Deactivate_Configuration** auf der Interbuskarte durchzuführen, die mit der NETID und dem PORT adressiert wird. Durch einen **Deactivate_Configuration** wird die Karte in den Zustand PAR_READY [▶ 106] versetzt und alle Ausgänge zurückgenommen.

VAR_INPUT

```
VAR_INPUT
    NETID    : T_AmsNetId;
    PORT     : T_AmsPort;
    WRTRD    : BOOL;
    TMOUT    : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

NETID: Hier kann die Netzwerkadresse des Rechners angegeben werden, in dem die Karte eingebaut ist (Typ: T_AmsNetID). Befindet sich die Karte auf dem selben System kann auch ein Leerstring angegeben werden.

PORT: Beinhaltet die ADS-Portnummer der Karte, die vom TwinCAT System vergeben wurde (Typ: T_AmsPort).

WRTRD: Über eine positive Flanke an diesem Wert wird der Baustein aktiviert.

TMOUT: Maximale Zeit die bei der Ausführung des Kommandos nicht überschritten werden soll.

VAR_OUTPUT

```
VAR_OUTPUT
    BUSY       : BOOL;
    ERR        : BOOL;
    ERRID      : UDINT;
    RESULT     : WORD;
    ADDERRINFO : WORD;
END_VAR
```

BUSY: Nach dem Aktivieren des Bausteins liegt das Busy-Signal solange an, bis eine Rückmeldung erfolgt.

ERR: Sollte ein Fehler bei der Übertragung des Kommandos erfolgen wird der ERR Ausgang nach Rücknahme des BUSY-Signals gesetzt.

ERRID: Liefert beim Fehler die ADS-Fehlernummer [▶ 154].

RESULT: Liefert das Ergebnis von der Karte zurück (Voraussetzung ist ein fehlerfreier ADS-Transport (ERR = FALSE)). RESULT = 0 kennzeichnet eine erfolgreiche Ausführung des Befehls. Ein Wert ungleich 0 beinhaltet die Fehlernummer der Phoenixkarte.

ADDERRINFO: Enthält bei negativer Rückmeldung der Karte zusätzliche Fehlerinformationen (vgl. Befehlsbeschreibung der Phoenixkarte).

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO-Hardware	Einzubindende SPS-Bibliotheken (Kategoriegruppe)
TwinCAT v3.1.0	Keine, diese Funktionalität wird zur Zeit von TwinCAT 3 nicht unterstützt!	Phoenix: IBS SC/I-T; IBS SC/RI/RT-LK	Tc2_IoFunctions (IO)

3.14.1.4 SCIT_StartDataTransfer

Der Funktionsbaustein "SCIT_StartDataTransfer" dient als Hilfsbaustein um einen **Start_Data_Transfer** auf der Interbuskarte durchzuführen, die mit der NETID und dem PORT adressiert wird. Durch einen **Start_Data_Transfer** wird die Karte in den Zustand RUN [▶ 106] versetzt.

VAR_INPUT

```
VAR_INPUT
    NETID : T_AmsNetId;
    PORT  : T_AmsPort;
    WRTRD : BOOL;
    TMOUT : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

NETID: Hier kann die Netzwerkadresse des Rechners angegeben werden, in dem die Karte eingebaut ist (Typ: T_AmsNetID). Befindet sich die Karte auf dem selben System kann auch ein Leerstring angegeben werden.

PORT: Beinhaltet die ADS-Portnummer der Karte, die vom TwinCAT System vergeben wurde (Typ: T_AmsPort).

WRTRD: Über eine positive Flanke an diesem Wert wird der Baustein aktiviert.

TMOUT: Maximale Zeit die bei der Ausführung des Kommandos nicht überschritten werden soll.

VAR_OUTPUT

```
VAR_OUTPUT
    BUSY       : BOOL;
    ERR        : BOOL;
    ERRID      : UDINT;
    RESULT     : WORD;
    ADDERRINFO : WORD;
END_VAR
```

BUSY: Nach dem Aktivieren des Bausteins liegt das Busy-Signal solange an, bis eine Rückmeldung erfolgt.

ERR: Sollte ein Fehler bei der Übertragung des Kommandos erfolgen wird der ERR Ausgang nach Rücknahme des BUSY-Signals gesetzt.

ERRID: Liefert beim Fehler die ADS-Fehlernummer [▶ 154].

RESULT: Liefert das Ergebnis von der Karte zurück (Voraussetzung ist ein fehlerfreier ADS-Transport (ERR = FALSE)). RESULT = 0 kennzeichnet eine erfolgreiche Ausführung des Befehls. Ein Wert ungleich 0 beinhaltet die Fehlernummer der Phoenixkarte.

ADDERRINFO: Enthält bei negativer Rückmeldung der Karte zusätzliche Fehlerinformationen (vgl. Befehlsbeschreibung der Phoenixkarte).

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO-Hardware	Einzubindende SPS-Bibliotheken (Kategoriegruppe)
TwinCAT v3.1.0	Keine, diese Funktionalität wird zur Zeit von TwinCAT 3 nicht unterstützt!	Phoenix: IBS SC/I-T; IBS SC/RI/RT-LK	Tc2_IoFunctions (IO)

3.14.1.5 SCIT_StopDataTransfer

Der Funktionsbaustein "SCIT_StopDataTransfer" dient als Hilfsbaustein um einen **Stop_Data_Transfer** auf der Interbuskarte durchzuführen, die mit der NETID und dem PORT adressiert wird. Durch einen **Stop_Data_Transfer** wird die Karte in den Zustand ACTIVE [▶ 106] versetzt, die Ausgänge werden *nicht* zurückgenommen.

VAR_INPUT

```
VAR_INPUT
    NETID : T_AmsNetId;
    PORT  : T_AmsPort;
    WRTRD : BOOL;
    TMOUT : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

NETID: Hier kann die Netzwerkadresse des Rechners angegeben werden, in dem die Karte eingebaut ist (Typ: T_AmsNetID). Befindet sich die Karte auf dem selben System kann auch ein Leerstring angegeben werden.

PORT: Beinhaltet die ADS-Portnummer der Karte, die vom TwinCAT System vergeben wurde (Typ: T_AmsPort).

WRTRD: Über eine positive Flanke an diesem Wert wird der Baustein aktiviert.

TMOUT: Maximale Zeit die bei der Ausführung des Kommandos nicht überschritten werden soll.

VAR_OUTPUT

```
VAR_OUTPUT
    BUSY       : BOOL;
    ERR        : BOOL;
    ERRID      : UDINT;
    RESULT     : WORD;
    ADDERRINFO : WORD;
END_VAR
```

BUSY: Nach dem Aktivieren des Bausteins liegt das Busy-Signal solange an, bis eine Rückmeldung erfolgt.

ERR: Sollte ein Fehler bei der Übertragung des Kommandos erfolgen wird der ERR Ausgang nach Rücknahme des BUSY-Signals gesetzt.

ERRID: Liefert beim Fehler die ADS-Fehlernummer [▶ 154].

RESULT: Liefert das Ergebnis von der Karte zurück (Voraussetzung ist ein fehlerfreier ADS-Transport (ERR = FALSE)). RESULT = 0 kennzeichnet eine erfolgreiche Ausführung des Befehls. Ein Wert ungleich 0 beinhaltet die Fehlernummer der Phoenixkarte.

ADDERRINFO: Enthält bei negativer Rückmeldung der Karte zusätzliche Fehlerinformationen (vgl. Befehlsbeschreibung der Phoenixkarte).

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO-Hardware	Einzubindende SPS-Bibliotheken (Kategoriegruppe)
TwinCAT v3.1.0	Keine, diese Funktionalität wird zur Zeit von TwinCAT 3 nicht unterstützt!	Phoenix: IBS SC/I-T; IBS SC/RI/RT-LK	Tc2_IoFunctions (IO)

3.14.1.6 SCIT_AlarmStop

Der Funktionsbaustein "SCIT_AlarmStop" dient als Hilfsbaustein um einen **Alarm_Stop** auf der Interbuskarte durchzuführen, die mit der NETID und dem PORT adressiert wird. Durch einen **Alarm_Stop** wird die Karte in den Zustand PAR_READY [▶ 106] versetzt und alle Ausgänge zurückgenommen.

VAR_INPUT

```
VAR_INPUT
    NETID    : T_AmsNetId;
    PORT     : T_AmsPort;
    WRTRD    : BOOL;
    TMOUT    : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

NETID: Hier kann die Netzwerkadresse des Rechners angegeben werden, in dem die Karte eingebaut ist (Typ: T_AmsNetID). Befindet sich die Karte auf dem selben System kann auch ein Leerstring angegeben werden.

PORT: Beinhaltet die ADS-Portnummer der Karte, die vom TwinCAT System vergeben wurde (Typ: T_AmsPort).

WRTRD: Über eine positive Flanke an diesem Wert wird der Baustein aktiviert.

TMOUT: Maximale Zeit die bei der Ausführung des Kommandos nicht überschritten werden soll.

VAR_OUTPUT

```
VAR_OUTPUT
    BUSY       : BOOL;
    ERR        : BOOL;
    ERRID      : UDINT;
    RESULT     : WORD;
    ADDERRINFO : WORD;
END_VAR
```

BUSY: Nach dem Aktivieren des Bausteins liegt das Busy-Signal solange an, bis eine Rückmeldung erfolgt.

ERR: Sollte ein Fehler bei der Übertragung des Kommandos erfolgen wird der ERR Ausgang nach Rücknahme des BUSY-Signals gesetzt.

ERRID: Liefert beim Fehler die ADS-Fehlernummer [▶ 154].

RESULT: Liefert das Ergebnis von der Karte zurück (Voraussetzung ist ein fehlerfreier ADS-Transport (ERR = FALSE)). RESULT = 0 kennzeichnet eine erfolgreiche Ausführung des Befehls. Ein Wert ungleich 0 beinhaltet die Fehlernummer der Phoenixkarte.

ADDERRINFO: Enthält bei negativer Rückmeldung der Karte zusätzliche Fehlerinformationen (vgl. Befehlsbeschreibung der Phoenixkarte).

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO-Hardware	Einzubindende SPS-Bibliotheken (Kategoriegruppe)
TwinCAT v3.1.0	Keine, diese Funktionalität wird zur Zeit von TwinCAT 3 nicht unterstützt!	Phoenix: IBS SC/I-T; IBS SC/RI/RT-LK	Tc2_IoFunctions (IO)

3.14.1.7 SCIT_ControlActiveConfiguration

Der Funktionsbaustein "SCIT_ControlActiveConfiguration" dient als Hilfsbaustein um einen **Control_Active_Configuration** auf der Interbuskarte durchzuführen, die mit der NETID und dem PORT adressiert wird. Durch einen **Control_Active_Configuration** kann der Zustand eines Teilnehmers (oder mehrerer, wenn der angegebene Teilnehmer Teil einer Gruppe ist) verändert werden.

VAR_INPUT

```
VAR_INPUT
    NETID         : T_AmsNetId;
    PORT          : T_AmsPort;
    WRTRD         : BOOL;
    TMOUT         : TIME := DEFAULT_ADS_TIMEOUT;
    SWITCH_CODE   : WORD;
    DEVICE_NO     : WORD;
END_VAR
```

NETID: Hier kann die Netzwerkadresse des Rechners angegeben werden, in dem die Karte eingebaut ist (Typ: T_AmsNetID). Befindet sich die Karte auf dem selben System kann auch ein Leerstring angegeben werden.

PORT: Beinhaltet die ADS-Portnummer der Karte, die vom TwinCAT System vergeben wurde (Typ: T_AmsPort).

WRTRD: Über eine positive Flanke an diesem Wert wird der Baustein aktiviert.

TMOUT: Maximale Zeit die bei der Ausführung des Befehls nicht überschritten werden soll.

SWITCH_CODE: Gibt an welche Aktion mit dem Teilnehmer ausgeführt werden soll:

0 = Segment Off

1 = Segment On

2 = Device_Off

3 = Device_On

4 = Device_Disable

5 = Device_Enable

DEVICE_NO: Gibt die Gerätenummer des angesprochenen Teilnehmers an. Für den Teilnehmer 3.1 muss z.B. ein Wert von 16#0301 angegeben werden.

VAR_OUTPUT

```
VAR_OUTPUT
    BUSY       : BOOL;
    ERR        : BOOL;
    ERRID      : UDINT;
    RESULT     : WORD;
    ADDERRINFO : WORD;
END_VAR
```

BUSY: Nach dem Aktivieren des Bausteins liegt das Busy-Signal solange an, bis eine Rückmeldung erfolgt.

ERR: Sollte ein Fehler bei der Übertragung des Kommandos erfolgen wird der ERR Ausgang nach Rücknahme des BUSY-Signals gesetzt.

ERRID: Liefert beim Fehler die ADS-Fehlernummer [▶ 154].

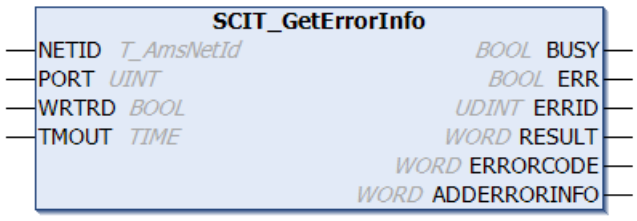
RESULT: Liefert das Ergebnis von der Karte zurück (Voraussetzung ist ein fehlerfreier ADS-Transport (ERR = FALSE)). RESULT = 0 kennzeichnet eine erfolgreiche Ausführung des Befehls. Ein Wert ungleich 0 beinhaltet die Fehlernummer der Phoenixkarte.

ADDERRINFO: Enthält bei negativer Rückmeldung der Karte zusätzliche Fehlerinformationen (vgl. Befehlsbeschreibung der Phoenixkarte).

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO-Hardware	Einzubindende SPS-Bibliotheken (Kategoriegruppe)
TwinCAT v3.1.0	Keine, diese Funktionalität wird zur Zeit von TwinCAT 3 nicht unterstützt!	Phoenix: IBS SC/I-T; IBS SC/RI/RT-LK	Tc2_IoFunctions (IO)

3.14.1.8 SCIT_GetErrorInfo



Der Funktionsbaustein "SCIT_GetErrorInfo" liest die genaue Fehlerursache und den genauen Fehlerort eines zuvor aufgetretenen Busfehlers aus der Interbuskarte, die mit der NETID und dem PORT adressiert wird.

VAR_INPUT

```
VAR_INPUT
    NETID  : T_AmsNetId;
    PORT   : T_AmsPort;
    WRTRD  : BOOL;
    TMOUT  : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

NETID: Hier kann die Netzwerkadresse des Rechners angegeben werden, in dem die Karte eingebaut ist (Typ: T_AmsNetID). Befindet sich die Karte auf dem selben System kann auch ein Leerstring angegeben werden.

PORT: Beinhaltet die ADS-Portnummer der Karte, die vom TwinCAT System vergeben wurde (Typ: T_AmsPort).

WRTRD: Über eine positive Flanke an diesem Wert wird der Baustein aktiviert.

TMOUT: Maximale Zeit die bei der Ausführung des Kommandos nicht überschrirtten werden soll.

VAR_OUTPUT

```
VAR_OUTPUT
    BUSY          : BOOL;
    ERR           : BOOL;
    ERRID         : UDINT;
    RESULT        : WORD;
    ERRORCODE     : WORD;
    ADDERRORINFO  : WORD;
END_VAR
```

BUSY: Nach dem Aktivieren des Bausteins liegt das Busy-Signal solange an, bis eine Rückmeldung erfolgt.

ERR: Sollte ein Fehler bei der Übertragung des Kommandos erfolgen wird der ERR Ausgang nach Rücknahme des BUSY-Signals gesetzt.

ERRID: Liefert beim Fehler die ADS-Fehlernummer [▶ 154].

RESULT: Liefert das Ergebnis von der Karte zurück (Voraussetzung ist ein fehlerfreier ADS-Transport (ERR = FALSE)). RESULT = 0 kennzeichnet eine erfolgreiche Ausführung des Befehls. Ein Wert ungleich 0 beinhaltet die Fehlernummer der Phoenix-Karte.

ERRORCODE: Liefert Informationen zur Fehlerart (vgl. Fehlerbeschreibung der Phoenix-Karte).

ADDERRORINFO: Enthält bei negativer Rückmeldung der Karte den Fehlerort (vgl. Fehlerbeschreibung der Phoenixkarte).

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO-Hardware	Einzubindende SPS-Bibliotheken (Kategoriegruppe)
TwinCAT v3.1.0	Keine, diese Funktionalität wird zur Zeit von TwinCAT 3 nicht unterstützt!	Phoenix: IBS SC/I-T; IBS SC/RI/RT-LK	Tc2_IoFunctions (IO)

3.14.1.9 SCIT_ConfDevErrAll

Der Funktionsbaustein "SCIT_ConfDevErrAll" quittiert Peripheriestörungen aller vorhandenen Geräte gleichzeitig.Intern wird die **Control_Device_Function** der Interbuskarte aufgerufen. Die Interbuskarte wird mit der NETID und dem PORT adressiert.

VAR_INPUT

```
VAR_INPUT
    NETID : T_AmsNetId;
    PORT  : T_AmsPort;
```

```
    WRTRD : BOOL;
    TMOUT : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

NETID: Hier kann die Netzwerkadresse des Rechners angegeben werden, in dem die Karte eingebaut ist (Typ: T_AmsNetID). Befindet sich die Karte auf dem selben System kann auch ein Leerstring angegeben werden.

PORT: Beinhaltet die ADS-Portnummer der Karte, die vom TwinCAT System vergeben wurde (Typ: T_AmsPort).

WRTRD: Über eine positive Flanke an diesem Wert wird der Baustein aktiviert.

TMOUT: Maximale Zeit die beim Ausführen des Kommandos nicht überschritten werden soll.

VAR_OUTPUT

```
VAR_OUTPUT
    BUSY         : BOOL;
    ERR          : BOOL;
    ERRID        : UDINT;
    RESULT       : WORD;
    ADDERRORINFO : WORD;
END_VAR
```

BUSY: Nach dem Aktivieren des Bausteins liegt das Busy-Signal solange an, bis eine Rückmeldung erfolgt.

ERR: Sollte ein Fehler bei der Übertragung des Kommandos erfolgen wird der ERR Ausgang nach Rücknahme des BUSY-Signals gesetzt.

ERRID: Liefert beim Fehler die ADS-Fehlernummer [▶ 154].

RESULT: Liefert das Ergebnis von der Karte zurück (Voraussetzung ist ein fehlerfreier ADS-Transport (ERR = FALSE)). RESULT = 0 kennzeichnet eine erfolgreiche Ausführung des Befehls. Ein Wert ungleich 0 beinhaltet die Fehlernummer der Phoenixkarte.

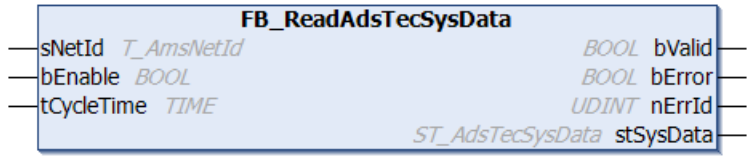
ADDERRORINFO: Enthält bei negativer Rückmeldung der Karte zusätzliche Fehlerinformationen (vgl. Befehlsbeschreibung der Phoenixkarte).

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO-Hardware	Einzubindende SPS-Bibliotheken (Kategoriegruppe)
TwinCAT v3.1.0	Keine, diese Funktionalität wird zur Zeit von TwinCAT 3 nicht unterstützt!	Phoenix: IBS SC/I-T; IBS SC/RI/RT-LK	Tc2_IoFunctions (IO)

3.14.2 ads-tec

3.14.2.1 FB_ReadAdsTecSysData



Der Funktionsbaustein *FB_ReadAdsTecSysData* liest die Systemdaten/Diagnosedaten eines ads-tec Industrie-PCs aus. Der Baustein wird Levelgetriggert, d.h. nur beim gesetzten *bEnable* -Eingang werden die Systemdaten zyklisch gelesen. Um dabei die Systemauslastung niedrig zu halten wird der Lesezyklus automatisch alle ~100ms (Defaultwert) wiederholt. Bei einem gesetzten *bValid*-Ausgang sind die zuletzt gelesenen Daten gültig (d.h. der letzte Lesezyklus wurde fehlerfrei durchgeführt). Beim Auftreten eines

Fehlers wird der *bError*-Ausgang gesetzt und das zyklische Lesen gestoppt. Mit einer erneuten steigenden Flanke am *bEnable*-Eingang können vorhandene Fehler gelöscht und das zyklische Lesen neu gestartet werden.

VAR_INPUT

```
VAR_INPUT
    sNetId     : T_AmsNetId;
    bEnable    : BOOL;
    tCycleTime : TIME := T#100ms;
END_VAR
```

sNetId: Hier kann ein String mit der Netzwerkadresse des TwinCAT-Rechners angegeben werden, dessen Systemdaten gelesen werden sollen (Typ: T_AmsNetID). Für den lokalen Rechner kann auch ein Leerstring angegeben werden.

bEnable: Mit einer steigenden Flanke wird der Baustein zurückgesetzt (vorherige Fehler am Ausgang *bError* und *nErrId* gelöscht). Bei einem gesetzten Eingang werden die Systemdaten zyklisch gelesen.

tCycleTime: Das zyklische Leseintervall.

VAR_OUTPUT

```
VAR_OUTPUT
    bValid    : BOOL;
    bError    : BOOL;
    nErrId    : UDINT;
    stSysData : ST_AdsTecSysData;
END_VAR
```

bValid: Wenn dieser Ausgang gesetzt ist sind die Daten in der *ST_AdsTecSysData*-Struktur gültig (beim letzten Lesezyklus ist kein Fehler aufgetreten).

bError: Sollte ein Fehler bei der Ausführung der Funktion erfolgen, dann wird dieser Ausgang gesetzt. Mit einer steigenden Flanke am *bEnable*-Eingang wird der Fehler zurückgesetzt.

nErrId: Liefert bei einem gesetzten *bError*-Ausgang die ADS-Fehlernummer [▶ 154].

stSysData: Struktur mit den Systemdaten/Diagnosedaten (Typ: ST_AdsTecSysData [▶ 128]).

Voraussetzungen

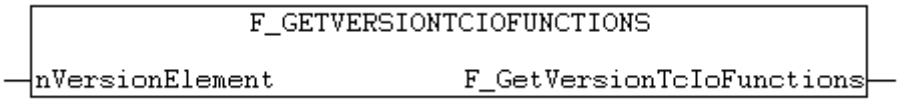
Entwicklungsumgebung	Zielplattform	IO Hardware	Einzubindende SPS-Bibliotheken (Kategoriegruppe)
TwinCAT v3.1.0	Keine, diese Funktionalität wird zur Zeit von TwinCAT 3 nicht unterstützt!	ads-tec PC	Tc2_IoFunctions (IO)

4 Funktionen

4.1 [veraltete Funktionen]

4.1.1 F_GetVersionTcIoFunctions

Diese Funktion ist veraltet und sollte nicht verwendet werden. Verwenden Sie bitte die globale Konstante: stLibVersion_Tc2_IoFunctions [▶ 146] um Versionsinformationen der SPS-Bibliothek auszulesen.



Diese Funktion liest Versionsinformationen der SPS-Bibliothek aus.

FUNCTION F_GetVersionTcIoFunctions : 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	IO Hardware	Einzubindende SPS-Bibliotheken (Kategoriegruppe)
TwinCAT v3.1.0	PC oder CX (x86, x64, ARM)	Alle IO-Geräte	Tc2_IoFunctions (IO)

4.1.2 F_GetVersionRAIDController

Diese Funktion ist veraltet und sollte nicht verwendet werden. Verwenden Sie bitte die globale Konstante: stLibVersion_Tc2_IoFunctions [▶ 146] um Versionsinformationen der SPS-Bibliothek auszulesen.

Diese Funktion liest Versionsinformationen der SPS-Bibliothek aus.

FUNCTION F_GetVersionRAIDController : 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	IO Hardware	Einzubindende SPS-Bibliotheken (Kategoriegruppe)
TwinCAT v3.1.0	PC oder CX (x86, x64, ARM)	Alle IO-Geräte	Tc2_IoFunctions (IO)

5 Datentypen

5.1 E_PD_Dpv1Error

E_PD_Dpv1Error lists the DPV1-Error IDs:

```
TYPE E_PD_Dpv1Error :
(
    ePD_Err_ParamNumber               := 0,(* Unzulässige Parameternummer *)
    ePD_Err_ParamReadOnly             := 1,(* Parameterwert nicht änderbar *)
    ePD_Err_ValueOutOfRange           := 2,(* Untere oder obere Wertgrenze überschritten *)
    ePD_Err_InvalidSubIndex           := 3,(* Fehlerhafter Subindex *)
    ePD_Err_NoArray                   := 4,(* Kein Array *)
    ePD_Err_WrongDataType             := 5,(* Falscher Datentyp *)
    ePD_Err_OnlyResetPermitted        := 6,(* Kein Setzen errlaubt (nur Rücksetzen) *)
    ePD_Err_DescNotChangable          := 7,(* Beschreibungselement nicht änderbar *)
    ePD_Err_DescNotFound              := 9,(* Beschreibungselement nicht vorhanden *)
    ePD_Err_NoPermissionToChange      := 11,(* Keine Bedienhoheit *)
    ePD_Err_NoTextArray               := 15,(* Kein Textarray vorhanden *)
    ePD_Err_JobNotExecutable          := 17,(* Auftrag wegen Betriebszustand nicht ausführbar *)
    ePD_Err_ValueInvalid              := 20,(* Wert unzulässig *)
    ePD_Err_ResponseToLong            := 21,(* Antwort zu lang *)
    ePD_Err_ParamAddrInvalid          := 22,(* Parameteradresse unzulässig *)
    ePD_Err_FormatInvalid             := 23,(* Format unzulässig *)
    ePD_Err_NumOfValuesInvalid        := 24,(* Anzahl Werte nicht konsistent *)
    ePD_Err_DriveObjNotExisting       := 25,(* Antriebsobjekt existiert nicht *)
    ePD_Err_ParamDeactivated          := 101,(* Parameter momentan deaktiviert *)
    ePD_Err_ParamNoWrIfEnabled        := 107,(* Kein Schreibzugriff bei freigegebenem Regler *)
    ePD_Err_ParamInvalidUnit          := 108,(* Unbekannte Einheit *)
    ePD_Err_ParamNoWrIfNotInitFbk     := 109,(* Schreibzugriff nur in Inbetriebnahmezustand Geber *)
    ePD_Err_ParamWrIfInitMtr          := 110,(* Schreibzugriff nur in Inbetriebnahmezustand Motor *)
    ePD_Err_ParamWrIfInitDrv          := 111,
(* Schreibzugriff nur in Inbetriebnahmezustand Leistungsteil *)
    ePD_Err_ParamWrIfFastInit         := 112,(* Schreibzugriff nur in Schnellinbetriebnahme *)
    ePD_Err_ParamWrIfReady            := 113,(* Schreibzugriff nur in Bereit *)
    ePD_Err_ParamWrIfInitParamReset   := 114,
(* Schreibzugriff nur in Inbetriebnahmezustand Parameterreset *)
    ePD_Err_ParamWrIfInitSafety       := 115,
(* Schreibzugriff nur in Inbetriebnahmezustand Safety *)
    ePD_Err_ParamWrIfInitTechApp      := 116,
(* Schreibzugriff nur in Inbetriebnahmezustand Tech.Appl./Einheiten *)
    ePD_Err_ParamWrIfInit             := 117,(* Schreibzugriff nur in Inbetriebnahmezustand *)
    ePD_Err_ParamWrIfInitDwnLd        := 118,
(* Schreibzugriff nur in Inbetriebnahmezustand Download *)
    ePD_Err_ParamNoWrtIfDwnLd         := 119,(* Darf im Download nicht geschrieben werden *)
    ePD_Err_ParamWrIfInitDrvCfg       := 120,
(* Schreibzugriff nur in Inbetriebnahmezustand Antriebskonfiguration *)
    ePD_Err_ParamWrIfInitSetDrvType   := 121,
(* Schreibzugriff nur in Inbetriebnahmezustand Festlegung Antriebstyp *)
    ePD_Err_ParamWrIfInitDatasetCfg   := 122,
(* Schreibzugriff nur in Inbetriebnahmezustand Datensatz-Basiskonfiguration *)
    ePD_Err_ParamWrIfInitDevCfg       := 123,
(* Schreibzugriff nur in Inbetriebnahmezustand Gerätekonfiguration *)
    ePD_Err_ParamWrIfInitDevDwnLd     := 124,
(* Schreibzugriff nur in Inbetriebnahmezustand Gerätedownload *)
    ePD_Err_ParamWrIfInitDevPrmReset  := 125,
(* Schreibzugriff nur in Inbetriebnahmezustand Geräteparameterreset *)
    ePD_Err_ParamWrIfInitDevReady     := 126,
(* Schreibzugriff nur in Inbetriebnahmezustand Gerät bereit *)
    ePD_Err_ParamWrIfInitDevice       := 127,(* Schreibzugriff nur in Inbetriebnahmezustand Gerät *)
    ePD_Err_ParamNoWriteIfDwnLd       := 129,(* darf im Download nicht geschrieben werden *)
    ePD_Err_CtrlTakeOverBlocked       := 130,(* Übernahme der Steuerungshoheit über BICO gesperrt *)
    ePD_Err_ParamBicoSetInvalid       := 131,(* gewünschte BICO-Verschaltung unmöglich *)
    ePD_Err_ParamChangeBlocked        := 132,(* Parameteränderung gesperrt *)
    ePD_Err_ParamNoAccessDefined      := 133,(* Keine Zugriffsmethode definiert *)
    ePD_Err_BelowDefinedMinimum       := 200,(* Unterhalb aktuell gültiger Grenze *)
    ePD_Err_AboveDefinedMaximum       := 201,(* Oberhalb aktuell gültiger Grenze *)
    ePD_Err_WriteNotPermitted         := 204(* Schreiben nicht erlaubt *)
);
END_TYPE
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO Hardware	Einzubindende SPS-Bibliotheken (Kategoriegruppe)
TwinCAT v3.1.0	PC oder CX (x86, x64, ARM)	Beckhoff FC310x PCI, CX1500-M310, EL6731, EL6632	Tc2_IoFunctions (IO)

5.2 E_BatteryStatus

Batteriestatus.

```
TYPE E_BatteryStatus :
(
    BatteryUnknownStatus,
    BatteryOk,
    BatteryReplace
);
END_TYPE
```

Name	Wert	Bedeutung
BatteryUnknownStatus	0	Der Akkustatus ist unbekannt.
BatteryOk	1	Der Akkustatus ist OK.
BatteryReplace	2	Der Akku soll ausgewechselt werden.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	USV Hardware	Einzubindende SPS-Bibliotheken (Kategoriegruppe)
TwinCAT v3.1.0	PC oder CX (x86, x64, ARM)	• Beckhoff BAPI v1; • Beckhoff P24Vxxxx; • Beckhoff CP903x-Karte (PCI/ ISA); • Beckhoff CX2100-09x4 Modelle (z.B. CX2100-0904 oder CX2100-0914 + "Smart Battery" CX2900-0192); • Die mit Beckhoff Industrie-PC ausgelieferten APC-Geräte die das Smartprotokoll unterstützen und mit dem Windows USV-Dienst konfiguriert werden können;	Tc2_IoFunctions (IO)

5.3 E_PD_Datatype

E_PD_Datatype enthält die möglichen Datentypen eines Profidrive-Parameters.

```
TYPE E_PD_Datatype :
(
    ePD_UNDEFINED    := 0,
    ePD_BOOL         := 1,(* 0/1 (not impl.) *)
    ePD_INT08        := 2,(* -128 .. 127 *)
    ePD_INT16        := 3,(* -32768 .. 32767 *)
    ePD_INT32        := 4,(* -2147483648 .. 2147483647 *)
    ePD_UINT08       := 5,(* 0 .. 255 *)
    ePD_UINT16       := 6,(* 0 .. 65535 *)
```

```
    ePD_UINT32       := 7,(* 0 .. 4294967295 *)
    ePD_FLOAT        := 8,(* IEEE 754 *)
    ePD_VSTRING      := 9,(* ISO/IEC 646, variable length *)
    ePD_OCTSTRING    := 10,(* bytearray, variable length *)
    ePD_TIMEOFDAY_WDI := 12,(* 6 Bytes: 4 bytes ms  + 2 bytes day since 1.1.1984 *)
    ePD_TIMEDIFF     := 13,(* 4|6 Bytes: 4 bytes ms  + optional 2 bytes days*)
    ePD_N2_16BIT     := 33,
    ePD_N4_32BIT     := 34,
    ePD_V2_BITSEQ    := 35,
    ePD_L2_NIBBLE    := 36,
    ePD_R2_RECIP_TC  := 37,
    ePD_T2_TC_16BIT  := 38,
    ePD_T2_TC_32BIT  := 39,
    ePD_D2_TC        := 40,
    ePD_E2_FIXPT_16  := 41,
    ePD_C2_FIXPT_32  := 42,
    ePD_X2_NV_16     := 43,
    ePD_X4_NV_32     := 44,
    ePD_DATE         := 50,
(* 7 Bytes: 2 bytes ms + 2 bits (res.), 6 bits(minutes) + 1 bit (0: StdTime/1:DaylightSavingTime), 2
 bits (res.), 5 bits(hours) + 3 bits (DayOfWeek), 5 bits(DayOfMonth) + 2 bits (res.), 6 bits(month)
+ 1 bit (res.), 7 bits (year)*)
    ePD_TIMEOFDAY_NODI := 52, (* 0 .. 268435455 ms *)
    ePD_TIMEDIFF_WDI := 53,(* 6 Bytes: 4 bytes ms + 2 bytes days *)
    ePD_TIMEDIFF_NODI:= 54,(* 0 .. 4294967295 ms *)
    ePD_ZERO         := 64,
    ePD_BYTE         := 65,
    ePD_WORD         := 66,
    ePD_DWORD        := 67,
    ePD_ERROR        := 68
);
END_TYPE
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO Hardware	Einzubindende SPS-Bibliotheken (Kategoriegruppe)
TwinCAT v3.1.0	PC oder CX (x86, x64, ARM)	Beckhoff FC310x PCI, CX1500-M310, EL6731, EL6632	Tc2_IoFunctions (IO)

5.4 E_RAIDDriveStatus

```
TYPE E_RAIDDriveStatus :
(
    eRAID_DRIVE_STATUS_OK         := 0,
    eRAID_DRIVE_STATUS_REBUILDING := 1,
    eRAID_DRIVE_STATUS_FAILED     := 2,
    eRAID_DRIVE_STATUS_DEGRADED   := 3
);
END_TYPE
```

Name	Wert	Bedeutung
eRAID_DRIVE_STATUS_OK	0	Zeigt an, dass der physische Antrieb betriebsbereit ist.
eRAID_DRIVE_STATUS_DEGRADED	1	Zeigt an, dass der physische Antrieb eine SMART-Meldung an den Controller geschickt hat.
eRAID_DRIVE_STATUS_REBUILDING	2	Zeigt an, dass der physische Antrieb der Zielantrieb eines RAID-Set-Rebuilds ist. Wenn der Rebuild erfolgreich absolviert ist, ändert sich der Status zu **eRAID_DRIVE_STATUS_OK**. Wenn der Rebuild fehlschlägt, wird der Status entsprechend upgedated.
eRAID_DRIVE_STATUS_FAILED	3	Zeigt an, dass der physische Antrieb nicht behebbare Fehler an den Controller gemeldet hat oder der Antrieb hat eine Anbieter-spezifische Aktion gestartet, um den Antrieb aus dem RAID-Set zu entfernen. Es gibt keine Garantie für die Betriebsbereitschaft des Antriebs und Datenverlust ist aufgetreten oder droht.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken (Kategoriegruppe)
TwinCAT v3.1.0	PC oder CX (x86, x64, ARM)	Tc2_IoFunctions (IO)

5.5 E_RAIDDriveUsage

```
TYPE E_RAIDDriveUsage :
(
    eRAID_DRIVE_CONFIG_NOT_USED   := 0,
    eRAID_DRIVE_CONFIG_MEMBER     := 1,
    eRAID_DRIVE_CONFIG_SPARE      := 2
);
END_TYPE
```

Name	Wert	Bedeutung
eRAID_DRIVE_CONFIG_NOT_USED	0	Zeigt an, dass der physische Antrieb nicht Teil eines RAID-Sets ist.
eRAID_DRIVE_CONFIG_MEMBER	1	Zeigt an, dass der physische Antrieb Teil eines RAID-Sets ist.
eRAID_DRIVE_CONFIG_SPARE	2	Zeigt an, dass der physische Antrieb als "hot swap spare" Teil dieses RAID-Sets ist.

„hot swap spare“ -> Fällt ein Laufwerk innerhalb des RAID-Verbundes aus, wird es im laufenden Betrieb (hot swap) durch das Reservelaufwerk (hot spare) ersetzt. Dadurch ist die Redundanz schnellstmöglich wiederhergestellt.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken (Kategoriegruppe)
TwinCAT v3.1.0	PC oder CX (x86, x64, ARM)	Tc2_IoFunctions (IO)

5.6 E_RAIDStatus

```
TYPE E_RAIDStatus :
(
    eRAID_SET_STATUS_OK         := 0,
    eRAID_SET_STATUS_DEGRADED   := 1,
    eRAID_SET_STATUS_REBUILDING := 2,
    eRAID_SET_STATUS_FAILED     := 3
);
END_TYPE
```

Name	Wert	Bedeutung
eRAID_SET_STATUS_OK	0	Zeigt an, dass das RAID-Set betriebsbereit ist.
eRAID_SET_STATUS_DEGRADED	1	Zeigt an, dass das RAID-Set nicht mehr in einem fehlertoleranten Modus arbeitet.
eRAID_SET_STATUS_REBUILDING	2	Zeigt an, dass das RAID-Set im Rebuild ist. Dies bedeutet einen eingeschränkten Betrieb. Wenn der Rebuild erfolgreich abgeschlossen ist, ändert sich der Status zu **eRAID_SET_STATUS_OK**. Wenn der Rebuild fehlschlägt, wird der Status entsprechend upgedated.
eRAID_SET_STATUS_FAILED	3	Zeigt an, dass das RAID-Set fehlgeschlagen ist. Es gibt keine Garantie für die Betriebsbereitschaft des RAID-Sets und Datenverlust ist aufgetreten oder droht.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken (Kategoriegruppe)
TwinCAT v3.1.0	PC oder CX (x86, x64, ARM)	Tc2_IoFunctions (IO)

5.7 E_RAIDType

```
TYPE E_RAIDType :
(
    eRAID_TYPE_NONE  := 0,
    eRAID_TYPE_0     := 1,
    eRAID_TYPE_1     := 2,
    eRAID_TYPE_10    := 3,
    eRAID_TYPE_5     := 4,
    eRAID_TYPE_15    := 5,
    eRAID_TYPE_OTHER := 255
);
END_TYPE
```

Name	Wert	Bedeutung
eRAID_TYPE_NONE	0	Zeigt an, dass das RAID-Set aus einem einzigen Antrieb besteht. Kein Set mit der angegebenen Nummer existiert.
eRAID_TYPE_0	1	Zeigt an, dass das RAID-Set ein gestriptes Set ist ohne Fehlertoleranz.
eRAID_TYPE_1	2	Zeigt an, dass das RAID-Set ein gemirrortes Set ist.
eRAID_TYPE_10	3	Zeigt an, dass das RAID-Set ein gestriptes und gemirrortes Set ist.
eRAID_TYPE_5	4	Zeigt an, dass das RAID-Set ein Parity-Set ist.
eRAID_TYPE_15	5	Zeigt an, dass das RAID-Set ein Advanced-Parity-Set ist.
eRAID_TYPE_OTHER	255	Zeigt an, dass die Konfiguration des RAID-Sets nicht den Standardtypen entspricht.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken (Kategoriegruppe)
TwinCAT v3.1.0	PC oder CX (x86, x64, ARM)	Tc2_IoFunctions (IO)

5.8 E_SercosAttribLen

```
TYPE E_SercosAttribLen : (
    eLEN_2BYTE  := 1, (* 2 bytes, fixed length *)
    eLEN_4BYTE  := 2, (* 4 bytes, fixed length *)
    eLEN_8BYTE  := 3, (* 8 bytes, fixed length *)
    eLEN_V1BYTE := 4, (* 1 bytes, variable length *)
    eLEN_V2BYTE := 5, (* 2 bytes, variable length *)
    eLEN_V4BYTE := 6, (* 4 bytes, variable length *)
    eLEN_V8BYTE := 7 (* 8 bytes, variable length *)
);
END_TYPE
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO Hardware	Einzubindende SPS-Bibliotheken (Kategoriegruppe)
TwinCAT v3.1.0	PC (x86)	Sercans SCS-P ISA; Sercans SCS-P PCI; Beckhoff FC750x PCI	Tc2_IoFunctions (IO)

5.9 E_SercosAttribType

```
TYPE E_SercosAttribType :
(
    eType_BIN      := 0,
    eType_UNSIGNED := 1,
    eType_SIGNED   := 2,
    eType_HEX      := 3,
    eType_ASCII    := 4,
    eType_ID       := 5,
    eType_FLOAT    := 6
);
END_TYPE
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO Hardware	Einzubindende SPS-Bibliotheken (Kategoriegruppe)
TwinCAT v3.1.0	PC (x86)	Sercans SCS-P ISA; Sercans SCS-P PCI; Beckhoff FC750x PCI	Tc2_IoFunctions (IO)

5.10 E_UpsCommStatus

Kommunikationsstatus zur USV-Hardware.

```
TYPE E_UpsCommStatus :
(
    UpsCommUnknownStatus,
    UpsCommOk,
    UpsCommFailed
);
END_TYPE
```

Name	Wert	Bedeutung
UpsCommUnknownStatus	0	Der Kommunikationsstatus ist unbekannt.
UpsCommOk	1	Die Kommunikation zur USV ist hergestellt.
UpsCommFailed	2	Die Kommunikation zur USV wurde unterbrochen.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	USV Hardware	Einzubindende SPS-Bibliotheken (Kategoriegruppe)
TwinCAT v3.1.0	PC oder CX (x86, x64, ARM)	• Beckhoff BAPI v1; • Beckhoff P24Vxxxx; • Beckhoff CP903x-Karte (PCI/ ISA); • Beckhoff CX2100-09x4 Modelle (z.B. CX2100-0904 oder CX2100-0914 + "Smart Battery" CX2900-0192); • Die mit Beckhoff Industrie-PC ausgelieferten APC-Geräte die das Smartprotokoll unterstützen und mit dem Windows USV-Dienst konfiguriert werden können;	Tc2_IoFunctions (IO)

5.11 E_UpsPowerStatus

Status der Spannungsversorung.

```
TYPE E_UpsPowerStatus :
(
    PowerUnknownStatus,
    PowerOnLine,
    PowerOnBattery
);
END_TYPE
```

Name	Wert	Bedeutung
PowerUnknownStatus	0	Der Status der Spannungsversorgung ist unbekannt
PowerOnLine	1	Netzspannungsversorgung.
PowerOnBattery	2	Akkuspannungsversorgung.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	USV Hardware	Einzubindende SPS-Bibliotheken (Kategoriegruppe)
TwinCAT v3.1.0	PC oder CX (x86, x64, ARM)	• Beckhoff BAPI v1; • Beckhoff P24Vxxxx; • Beckhoff CP903x-Karte (PCI/ ISA); • Beckhoff CX2100-09x4 Modelle (z.B. CX2100-0904 oder CX2100-0914 + "Smart Battery" CX2900-0192); • Die mit Beckhoff Industrie-PC ausgelieferten APC-Geräte die das Smartprotokoll unterstützen und mit dem Windows USV-Dienst konfiguriert werden können;	Tc2_IoFunctions (IO)

5.12 IODEVICETYPES

```
TYPE IODEVICETYPES:
(
IODEVICETYPE_UNKNOWN            := 0,(* Unknown device *)
IODEVICETYPE_C1220              := 1,(* Beckhoff Lightbus-Master *)
IODEVICETYPE_C1200              := 2,(* Beckhoff Lightbus-Master *)
IODEVICETYPE_SPC3               := 3,(* ProfiBus Slave (Siemens) *)
IODEVICETYPE_CIF30DPM           := 4,(* ISA ProfiBus-Master 2 kByte (Hilscher) *)
IODEVICETYPE_CIF40IBSM          := 5,(* ISA Interbus-S-Master 2 kByte (Hilscher) *)
IODEVICETYPE_BKHFPC             := 6,(* Beckhoff PC C2001*)
IODEVICETYPE_CP5412A2           := 7,(* ProfiBus-Master (Siemens)*)
IODEVICETYPE_SERCANSISA         := 8,(* Sercos Master (Indramat)*)
IODEVICETYPE_LPTPORT            := 9,(* Lpt Port*)
IODEVICETYPE_DPRAM              := 10,(* Generic DPRAM*)
IODEVICETYPE_COMPORT            := 11,(* COM Port*)
IODEVICETYPE_CIF30CAN           := 12,(* ISA CANopen-Master (Hilscher)*)
IODEVICETYPE_CIF30PB            := 13,(* ISA ProfiBus-Master 8 kByte (Hilscher)*)
IODEVICETYPE_BKHFCP2030         := 14,(* Beckhoff CP2030 (Pannel-Link)*)
IODEVICETYPE_IBSSCIT            := 15,(* Interbus-S-Master (Phoenix)*)
IODEVICETYPE_CIF30IBM           := 16,(* ISA Interbus-S-Master (Hilscher)*)
IODEVICETYPE_CIF30DNM           := 17,(* ISA DeviceNet-Master (Hilscher)*)
IODEVICETYPE_FCXXXX             := 18,(* Beckhoff-Filedbus card *)
IODEVICETYPE_CIF50PB            := 19,(* PCI ProfiBus-Master 8 kByte (Hilscher)*)
IODEVICETYPE_CIF50IBM           := 20,(* PCI Interbus-S-Master (Hilscher)*)
IODEVICETYPE_CIF50DNM           := 21,(* PCI DeviceNet-Master (Hilscher)*)
IODEVICETYPE_CIF50CAN           := 22,(* PCI CANopen-Master (Hilscher)*)
IODEVICETYPE_CIF60PB            := 23,(* PCMCIA ProfiBus-Master (Hilscher)*)
IODEVICETYPE_CIF60DNM           := 24,(* PCMCIA DeviceNet-Master (Hilscher)*)
IODEVICETYPE_CIF60CAN           := 25,(* PCMCIA CANopen-Master (Hilscher)*)
IODEVICETYPE_CIF104DP           := 26,(* PC104 ProfiBus-Master 2 kByte (Hilscher)*)
IODEVICETYPE_C104PB             := 27,(* PC104 ProfiBus-Master 8 kByte (Hilscher)*)
IODEVICETYPE_C104IBM            := 28,(* PC104 Interbus-S-Master 2 kByte (Hilscher)*)
IODEVICETYPE_C104CAN            := 29,(* PC104 CANopen-Master (Hilscher)*)
IODEVICETYPE_C104DNM            := 30,(* PC104 DeviceNet-Master (Hilscher)*)
IODEVICETYPE_BKHFCP9030         := 31,(* Beckhoff CP9030 (Pannel-Link with UPS)*)
IODEVICETYPE_SMB                := 32,(* Motherboard System Management Bus*)
IODEVICETYPE_PBMON              := 33,(* Beckhoff-PROFIBUS-Monitor*)
IODEVICETYPE_CP5613             := 34,(* PCI ProfiBus-Master (Siemens)*)
```

```
IODEVICETYPE_CIF60IBM           := 35,(* PCMCIA Interbus-S-Master (Hilscher)*)
IODEVICETYPE_FC200X             := 36,(* Beckhoff-Lightbus-I/II-PCI-Karte*)
IODEVICETYPE_FC3100_OLD         := 37,(* obsolete: dont use*)
IODEVICETYPE_FC3100             := 38,(* Beckhoff-Profibus-PCI*)
IODEVICETYPE_FC5100             := 39,(* Beckhoff-CanOpen-PCI*)
IODEVICETYPE_FC5200             := 41,(* Beckhoff-DeviceNet-PCI*)
IODEVICETYPE_BKHFNCBP           := 43,(* Beckhoff NC back plane*)
IODEVICETYPE_SERCANSPCI         := 44,(* Sercos Master (SICAN/IAM PCI)*)
IODEVICETYPE_ETHERNET           := 45,(* Virtual Ethernet Device*)
IODEVICETYPE_SERCONPCI          := 46,(* Sercon 410B or 816 Chip Master or Slave (PCI)*)
IODEVICETYPE_IBSSCRIRTLK        := 47,(* Interbus-S-Master with Slave-Module LWL Basis (Phoenix)*)
IODEVICETYPE_FC7500             := 48,(* Beckhoff-SERCOS-PCI*)
IODEVICETYPE_CIF30IBS           := 49,(* ISA Interbus-S-Slave (Hilscher)*)
IODEVICETYPE_CIF50IBS           := 50,(* PCI Interbus-S-Slave (Hilscher)*)
IODEVICETYPE_C104IBS            := 51,(* PC104 Interbus-S-Slave (Hilscher)*)
IODEVICETYPE_BKHFCP9040         := 52,(* Beckhoff CP9040 (CP-PC) *)
IODEVICETYPE_BKHFAH2000         := 53,(* Beckhoff AH2000 (Hydr. Backplane) *)
IODEVICETYPE_BKHFCP9035         := 54,(* Beckhoff CP9035 (PCI, Pannel-Link with UPS) *)
IODEVICETYPE_AH2000MC           := 55,(* Beckhoff-AH2000 with Profibus-MC *)
IODEVICETYPE_FC3100MON          := 56,(* Beckhoff-Profibus-Monitor-PCI *)
IODEVICETYPE_USB                := 57,(* Virtual USB Device *)
IODEVICETYPE_FC5100MON          := 58,(* Beckhoff-CANopen-Monitor-PCI *)
IODEVICETYPE_FC5200MON          := 59,(* Beckhoff-DeviceNet-Monitor-PCI *)
IODEVICETYPE_FC3100SLV          := 60,(* Beckhoff-Profibus-PCI Slave *)
IODEVICETYPE_FC5100SLV          := 61,(* Beckhoff-CanOpen-PCI Slave *)
IODEVICETYPE_FC5200SLV          := 62,(* Beckhoff-DeviceNet-PCI Slave *)
IODEVICETYPE_IBSSCITPCI         := 63,(* PCI Interbus-S-Master (Phoenix) *)
IODEVICETYPE_IBSSCRIRTLKPCI     := 64,(* PCI Interbus-S-Master with Slave-
Modulel LWL Basis (Phoenix) *)
IODEVICETYPE_CX1100_BK          := 65,(* Beckhoff-CX1100 terminal bus power supply *)
IODEVICETYPE_ENETRTMP           := 66,(* Ethernet real time miniport *)
IODEVICETYPE_CX1500_M200        := 67,(* PC104 Lightbus-Master *)
IODEVICETYPE_CX1500_B200        := 68,(* PC104 Lightbus-Slave *)
IODEVICETYPE_CX1500_M310        := 69,(* PC104 ProfiBus-Master *)
IODEVICETYPE_CX1500_B310        := 70,(* PC104 ProfiBus-Slave *)
IODEVICETYPE_CX1500_M510        := 71,(* PC104 CANopen-Master *)
IODEVICETYPE_CX1500_B510        := 72,(* PC104 CANopen-Slave *)
IODEVICETYPE_CX1500_M520        := 73,(* PC104 DeviceNet-Master *)
IODEVICETYPE_CX1500_B520        := 74,(* PC104 DeviceNet-Slave *)
IODEVICETYPE_CX1500_M750        := 75,(* PC104 Sercos-Master *)
IODEVICETYPE_CX1500_B750        := 76,(* PC104 Sercos-Slave *)
IODEVICETYPE_BX_BK              := 77,(* BX terminal bus interface *)
IODEVICETYPE_BX_M510            := 78,(* BX SSB-Master *)
IODEVICETYPE_BX_B310            := 79,(* BX ProfiBus-Slave *)
IODEVICETYPE_IBSSCRIRTPCI       := 80,(* PCI Interbus-S-
Master with slave module copper basis (Phoenix) *)
IODEVICETYPE_BX_B510            := 81,(* BX CANopen Slave *)
IODEVICETYPE_BX_B520            := 82,(* BX DeviceNet Slave *)
IODEVICETYPE_BC3150             := 83,(* BCxx50 ProfiBus Slave *)
IODEVICETYPE_BC5150             := 84,(* BCxx50 CANopen Slave *)
IODEVICETYPE_BC5250             := 85,(* BCxx50 DeviceNet Slave *)
IODEVICETYPE_EL6731             := 86,(* Beckhoff Profibus-EtherCAT Terminal *)
IODEVICETYPE_EL6751             := 87,(* Beckhoff CanOpen-EtherCAT Terminal *)
IODEVICETYPE_EL6752             := 88,(* Beckhoff DeviceNet-EtherCAT Terminal *)
IODEVICETYPE_COMPB              := 89,(* COM ProfiBus Master 8 kByte (Hilscher) *)
IODEVICETYPE_COMIBM             := 90,(* COM Interbus-S Master (Hilscher) *)
IODEVICETYPE_COMDNM             := 91,(* COM DeviceNet Master (Hilscher) *)
IODEVICETYPE_COMCAN             := 92,(* COM CANopen Master (Hilscher) *)
IODEVICETYPE_COMIBS             := 93,(* COM CANopen Slave (Hilscher) *)
IODEVICETYPE_ETHERCAT           := 94,(* EtherCAT in direct mode *)
IODEVICETYPE_PROFINETIOCONTROLLER:= 95,(* PROFINET Master *)
IODEVICETYPE_PROFINETIODEVICE   := 96,(* PROFINET Slave *)
IODEVICETYPE_EL6731SLV          := 97,(* Beckhoff Profibus Slave EtherCAT Terminal *)
IODEVICETYPE_EL6751SLV          := 98,(* Beckhoff CanOpen Slave EtherCAT Terminal *)
IODEVICETYPE_EL6752SLV          := 99,(* Beckhoff DeviceNet Slave EtherCAT Terminal *)
IODEVICETYPE_C104PPB            := 100,(* PC104+ ProfiBus Master 8 kByte (Hilscher) *)
IODEVICETYPE_C104PCAN           := 101,(* PC104+ CANopen Master (Hilscher) *)
IODEVICETYPE_C104PDNM           := 102,(* PC104+ DeviceNet Master (Hilscher) *)
IODEVICETYPE_BC8150             := 103,(* BCxx50 Serial Slave *)
IODEVICETYPE_BX9000             := 104,(* BX9000 Ethernet Slave *)
IODEVICETYPE_CX9000_BK          := 105,(* Beckhoff-CX9000 K-Bus Power Supply *)
IODEVICETYPE_EL6601             := 106,(* Beckhoff-RT-Ethernet-EtherCAT-Terminal *)
IODEVICETYPE_BC9050             := 107,(* BC9050 Ethernet Slave *)
IODEVICETYPE_BC9120             := 108,(* BC9120 Ethernet Slave *)
IODEVICETYPE_ENETADAPTER        := 109,(* Ethernet Miniport Adapter *)
IODEVICETYPE_BC9020             := 110,(* BC9020 Ethernet Slave *)
IODEVICETYPE_ETHERCATPROT       := 111,(* EtherCAT Protocol in direct mode *)
IODEVICETYPE_ETHERNETNVPROT     := 112,(* *)
IODEVICETYPE_ETHERNETPNMPROT    := 113,(* Profinet Controller *)
IODEVICETYPE_EL6720             := 114,(* Beckhoff-Lightbus-EtherCAT-Terminal *)
```

```
IODEVICETYPE_ETHERNETPNSPROT    := 115,(* Profinet Device*)
IODEVICETYPE_BKHFCP6608         := 116,(* Beckhoff CP6608(IXP PC) *)
IODEVICETYPE_PTP_IEEE1588       := 117,(* *)
IODEVICETYPE_EL6631SLV          := 118,(* EL6631-0010 Profinet Slave terminal *)
IODEVICETYPE_EL6631             := 119,(* EL6631 Profinet Master terminal *)
IODEVICETYPE_CX5000_BK          := 120,(* Beckhoff-CX5100 K-Bus power supply *)
IODEVICETYPE_PCIDEVICE          := 121,(* Generic PCI DPRAM (TCOM) *)
IODEVICETYPE_ETHERNETUPDPROT    := 122,(* UDP Protocol *)
IODEVICETYPE_ETHERNETAUTOPROT   := 123,(* Automation Protocol *)
IODEVICETYPE_CCAT               := 124,(* CCAT *)
IODEVICETYPE_CPLINK3            := 125,(* Virtuelles USB Device (remote via CPLINK3) *)
IODEVICETYPE_EL6632             := 126,(* EL6632 *)
IODEVICETYPE_CCAT_PBM           := 127,(* CCAT Profibus Master *)
IODEVICETYPE_CCAT_PBS           := 128,(* CCAT Profibus Slave *)
IODEVICETYPE_CCAT_CNM           := 129,(* CCAT CANopen Master *)
IODEVICETYPE_ETHERCATSLAVE      := 130,(* EtherCAT Slave *)
IODEVICETYPE_BACNET             := 131,(* BACnet device *)
IODEVICETYPE_CCAT_CNS           := 132,(* CCAT CANopen Slave *)
IODEVICETYPE_ETHIP_SCANNER      := 133,(* ETHERNET IP Master *)
IODEVICETYPE_ETHIP_ADAPTER      := 134,(* ETHERNET IP Slave (OLD) *)
IODEVICETYPE_CX8000_BK          := 135,(* Beckhoff-CX8100 Klemmenbus Netzteil -
 LEGACY use IODEVICETYPE_CX_BK *)
IODEVICETYPE_ETHERNETUDPPROT    := 136,(* Upd Protocol *)
IODEVICETYPE_BC9191             := 137,(* BC9191 Etherent Slave *)
IODEVICETYPE_ENETPROTOCOL       := 138,(* Real-Time Ethernet Protocol (BK90xx, AX2000-B900) *)
IODEVICETYPE_ETHIP_ADAPTEREX    := 139,(* ETHERNET IP Slave (NEW) *)
IODEVICETYPE_PNCONTR_CCAT_RT    := 140,(* Profinet Controller CCAT RT *)
IODEVICETYPE_PNCONTR_CCAT_IRT   := 141,(* Profinet Controller CCAT RT + IRT *)
IODEVICETYPE_PNDEV_CCAT_RT      := 142,(* Profinet Device CCAT RT *)
IODEVICETYPE_PNDEV_CCAT_IRT     := 143,(* Profinet Device CCAT RT + IRT *)
IODEVICETYPE_ETHERCATSIMULATION := 144,(* EtherCAT-Simulation *)
IODEVICETYPE_EL6652SLV          := 145,(* EL6652-0010 *)
IODEVICETYPE_PTP_VIA_CCAT       := 146,(* PTP CLock via CCAT *)
IODEVICETYPE_BACNETR9           := 147,(* BACnet Rev9 device *)
IODEVICETYPE_ETHERCATXFC        := 148,(* EtherCAT in xfc mode *)
IODEVICETYPE_CX2500_0030        := 149,(* CX2500-0030 RS232 Serial Communication Port *)
IODEVICETYPE_CX2500_0031        := 150,(* CX2500-0031 RS422/RS485 Serial Communication Port *)
IODEVICETYPE_EL6652MST          := 151,(* EL6652 *)

(* Reserved for new devices*)


IODEVICETYPE_MAX
);
END_TYPE
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO Hardware	Einzubindende SPS-Bibliotheken (Kategoriegruppe)
TwinCAT v3.1.0	PC oder CX (x86, x64, ARM)	Alle IO-Geräte	Tc2_IoFunctions (IO)

5.13 ST_AdsTecSysData

```
TYPE ST_AdsTecSysData
STRUCT
    bShiftKey    : BOOL;(* TRUE == Shift key pressed*)
    bRMouseKey   : BOOL;(* TRUE == Right mouse key pressed *)
    bHotKey      : BOOL;(* TRUE == Hotkey pressed *)
    bTaskChaKey  : BOOL;(* TRUE == Task change key pressed *)
    bABCKey      : BOOL;(* TRUE == ABC soft keyboard key pressed*)
    bRsrv1       : BOOL;
    bRsrv2       : BOOL;
    bRsrv3       : BOOL;
    bMainFanErr  : BOOL;(* TRUE == Main fan error*)
    bCpuFanErr   : BOOL;(* TRUE == CPU fan error*)
    bTempErr     : BOOL;(* TRUE == Internal temperature error ( temp > 50°C)*)
    bBatteryErr  : BOOL;(* TRUE == Battery error *)
    bRsrv4       : BOOL;
    bRsrv5       : BOOL;
    bRsrv6       : BOOL;
    bRsrv7       : BOOL;
    nMainNtcTemp : SINT;(* Main NTC temperature (-127°C .. + 127°C) *)
```

```
    nExtNtcTemp  : SINT;(* External NTC temperature (-127°C .. + 127°C) *)
    nRsrv8       : ARRAY[1..12] OF BYTE;
END_STRUCT
END_TYPE
```

bShiftKey: "Shift"-Taste gedrückt (Taste ganz rechts in der Front).

bRMouseKey: "Rechte Maus"-Taste gedrückt.

bHotKey: "Hotkey"-Taste gedrückt.

bTaskChaKey: "Taskwechsel"-Taste gedrückt.

bABCKey: "ABC Softkeyboard"-Taste gedrückt.

bMainFanErr: Fehler Hauptlüfter.

bCpuFanErr: Fehler CPU-Lüfter.

bTempErr: Temperaturfehler (Innentemperatur > 50°C).

bBatteryErr: Batteriefehler (derzeit reserviert).

nMainNtcTemp: Temperaturwert 1 (eingelöteter NTC-127°C .. + 127°C).

nExtNtcTemp: Temperaturwert 2 (anschließbarer NTC, nicht bei jedem Gerät vorhanden).

bRsrv1 - bRsrv7: Reserviert.

nRsrv8: Reserviert.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO Hardware	Einzubindende SPS-Bibliotheken (Kategoriegruppe)
TwinCAT v3.1.0	Keine, diese Funktionalität wird zur Zeit von TwinCAT 3 nicht unterstützt!	ads-tec PC	Tc2_IoFunctions (IO)

5.14 ST_Dpv1ParamAddrEx

ST_Dpv1ParamAddrEx enthält die Daten eines Profidrive-Parameters.

```
TYPE ST_Dpv1ParamAddrEx :
STRUCT
    iAttribute       : USINT;
    iNumOfElements   : USINT;
    iParameterNumber : UINT;
    iSubIndex        : UINT;
    iDataAddr        : PVOID;
    iDataSize        : UDINT;
    eFormat          : E_PD_Datatype;
    iNumOfValues     : UINT;
    iErrorValue      : UDINT;
    stError          : ST_PD_Dpv1Error;
END_STRUCT
END_TYPE
```

iAttribute: 0x10: Wert; 0x20: Beschreibung; 0x30: Text; 0x80..F0: herstellerspezifisch; andere Werte sind reserviert.

iNumOfElements: 1..234: Anzahl der Elemente; 0: Spezialfunktionen; andere Werte sind reserviert.

iParameterNumber: 1..65535: Parameternummer; 0: reserviert.

iSubIndex: 0..65535: Unterindex.

iDataAddr: Adresse des Puffers/Adresse der SPS-Variablen.

iDataSize: Größe des Puffers/Größe der SPS-Variablen.

eFormat: 0x01..0x36: Datentyp; 0x40: ZERO; 0x41: BYTE; 0x42: WORD; 0x43: DWORD; 0x44: Fehler; andere Werte sind reserviert. (Typ: E_PD_Datatype [▶ 120]).

iNumOfValues: 0..234: Anzahl der Werte; andere Werte sind reserviert.

iErrorValue: DPV1 Fehlerwert.

stError: DPV1 Fehlerflag, DPV1 Fehleraufzählungstyp. (Typ: ST_PD_Dpv1Error [▶ 133]).

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO Hardware	Einzubindende SPS-Bibliotheken (Kategoriegruppe)
TwinCAT v3.1.0	PC oder CX (x86, x64, ARM)	Beckhoff FC310x PCI, CX1500-M310, EL6731, EL6632	Tc2_IoFunctions (IO)

5.15 ST_Dpv1ValueHeaderEx

ST_Dpv1ValueHeaderEx enthält die Daten eines Parameters im DPV1 Telegramm und seine String-Repräsentation.

```
TYPE ST_Dpv1ValueHeaderEx :
STRUCT
    eFormat      : E_PD_Datatype;
    iNumOfValues : USINT;
    iOffset      : USINT;
    iDataLen     : UINT;
    strData      : STRING;
END_STRUCT
END_TYPE
```

eFormat: 0x01..0x36: Datentyp; 0x40: ZERO; 0x41: BYTE; 0x42: WORD; 0x43: DWORD; 0x44: Fehler; andere Werte sind reserviert. (Typ: E_PD_Datatype [▶ 120]).

iNumOfValues: 0..234: Anzahl der Werte; andere Werte sind reserviert.

iOffset: Offset im DPV1 Antworttelegramm.

iDataLen: Datenlänge.

strData: Daten als STRING.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO Hardware	Einzubindende SPS-Bibliotheken (Kategoriegruppe)
TwinCAT v3.1.0	PC oder CX (x86, x64, ARM)	Beckhoff FC310x PCI, CX1500-M310, EL6731, EL6632	Tc2_IoFunctions (IO)

5.16 ST_NovRamAddrInfo

```
TYPE ST_NovRamAddrInfo:
STRUCT
    pCardAddress : POINTER TO DWORD;
```

```
    iCardMemSize : UDINT;
END_STRUCT
END_TYPE
```

pCardAddress: Der Addresspointer vom NOV/DP-RAM.

iCardMemSize: Die Konfigurierte NOV/DP-RAM-Größe in Bytes.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO Hardware	Einzubindende SPS-Bibliotheken (Kategoriegruppe)
TwinCAT v3.1.0	PC oder CX (x86, x64, ARM)	Alle IO-Geräte	Tc2_IoFunctions (IO)

5.17 ST_NovRamAddrInfoEx

```
TYPE ST_NovRamAddrInfoEx:
STRUCT
   pCardAddress  :  POINTER TO DWORD;
   iCardMemSize  :  UDINT;
   eAccessType   :  E_IOACCESSTYPE
END_STRUCT
END_TYPE
```

pCardAddress: Der Adresspointer vom NOV/DP-RAM.

iCardMemSize: Die konfigurierte NOV/DP-RAM- Größe in Bytes.

eAccessType: Die Zugriffsart auf NOV/DP-RAM.

eIOAccess_Default: normaler Zugriff via MEMCPY-Funktion möglich

eIOAccess_Byte: nur BYTE-Zugriff via FOR-Schleife möglich

eIOAccess_WordSwap: nur WORT-Zugriff + High/Low-Byte-Swapping via FOR-Schleife möglich

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO Hardware	Einzubindende SPS-Bibliotheken (Kategoriegruppe)
TwinCAT v3.1.0	PC oder CX (x86, x64, ARM)	Alle IO-Geräte	Tc2_IoFunctions (IO)

5.18 ST_Parameter_IN

Eingangsdaten von der ASI-Klemme.

```
TYPE ST_ParameterBuffer :
STRUCT
    iParameterStatus    : WORD;
    iParameterReadValue : DWORD;
END_STRUCT
END_TYPE
```

Byte-Offset	Bit-Offset	Beschreibung
0	0-5	Status-Bits (wie bei bisherigen Klemmen)
0	6	0: keine Diagnose, 1: Diagnose (wie bei bisherigen Klemmen)
0	7	immer 0: keine Registerkommunikation
1	0-3	0-3 Reserviert für Erweiterungen
1	4	Toggle-Bit, um Auftrag zu quittieren (bei Cyclic wird das Bit 6 aus Byte 0 kopiert)
1	5	Quittung (0: NoError, 1: Error)
1	6	0: Cyclic, 1: Acyclic
1	7	0: Parameterzugriff, 1: ADS
2		Input-Daten (Cyclic), Parameterwert (Acyclic) oder Fehlernummer Bit 0-7
3		Input-Daten (Cyclic), Parameterwert (Acyclic) oder Fehlernummer Bit 8-15
4		Input-Daten (Cyclic), Parameterwert (Acyclic) oder Fehlernummer Bit 16-23
5		Input-Daten (Cyclic), Parameterwert (Acyclic) oder Fehlernummer Bit 24-31

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO Hardware	Einzubindende SPS-Bibliotheken (Kategoriegruppe)
TwinCAT v3.1.0	PC oder CX (x86)	ASI Masterklemme	Tc2_IoFunctions (IO)

5.19 ST_Parameter_OUT

Ausgangsdaten zur ASI-Klemme.

```
TYPE ST_ParameterBuffer :
STRUCT
    ParameterControl : WORD;
    iParametervalue  : DWORD;
END_STRUCT
END_TYPE
```

Byte-Offset	Bit-Offset	Beschreibung
0	0-5	Parameternummer Bit 0-5 (oder Parameter-Offset)
0	6	Bei Acyclic: 0: Read, 1:Write, bei Cyclic (immer Read/Write) wird das Bit in die Inputdaten kopiert, um eine direkte Zuordnung zu haben (dann könnten die Cyclic-Daten auch geändert werden)
0	7	0: Parameterzugriff, 1: Registerkommunikation
1	0-5	Parameternummer Bit 6-11 (oder Parameter-Page)
1	6	0: Cyclic, 1: Acyclic
1	7	0: Parameterzugriff, 1: ADS
2		Output-Daten (Cyclic) oder Parameterwert (Acyclic) Bit 0-7
3		Output-Daten (Cyclic) oder Parameterwert (Acyclic) Bit 8-15
4		Output-Daten (Cyclic) oder Parameterwert (Acyclic) Bit 16-23
5		Output-Daten (Cyclic) oder Parameterwert (Acyclic) Bit 24-32

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO Hardware	Einzubindende SPS-Bibliotheken (Kategoriegruppe)
TwinCAT v3.1.0	PC oder CX (x86)	ASI Masterklemme	Tc2_IoFunctions (IO)

5.20 ST_ParameterBuffer

Datenpuffer für die E/A-Daten der ASI-Klemme

```
TYPE ST_ParameterBuffer :
STRUCT
    ParameterControl    : ARRAY[0..50] OF WORD;
    iParametervalue     : ARRAY[0..50] OF DWORD;
    iParameterStatus    : ARRAY[0..50] OF WORD;
    iParameterReadValue : ARRAY[0..50] OF DWORD;
    icounterState       : INT;
    icounterControl     : INT;
END_STRUCT
END_TYPE
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO Hardware	Einzubindende SPS-Bibliotheken (Kategoriegruppe)
TwinCAT v3.1.0	PC oder CX (x86)	ASI Masterklemme	Tc2_IoFunctions (IO)

5.21 ST_PD_Dpv1Error

```
TYPE ST_PD_Dpv1Error :
STRUCT
    bError   : BOOL;
```

```
    eErrorId : E_PD_Dpv1Error;
END_STRUCT
END_TYPE
```

bError: Fehlerflag (TRUE => Fehler, FALSE => kein Fehler).

eErrorID: Fehlercode. (Typ: E_PD_Dpv1Error [▶ 119]).

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO Hardware	Einzubindende SPS-Bibliotheken (Kategoriegruppe)
TwinCAT v3.1.0	PC oder CX (x86, x64, ARM)	Beckhoff FC310x PCI, CX1500-M310, EL6731, EL6632	Tc2_IoFunctions (IO)

5.22 ST_PNET_CCDSTS

```
TYPE ST_PNET_CCDSTS :
STRUCT
    iCycleCounter  : UINT;
    iDataState     : USINT;
    iTransferState : USINT;
END_STRUCT
END_TYPE
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO Hardware	Einzubindende SPS-Bibliotheken (Kategoriegruppe)
TwinCAT v3.1.0	PC oder CX (x86)	Beckhoff EL6632	Tc2_IoFunctions (IO)

5.23 ST_PNIOConfigRecord

```
TYPE ST_PNIOConfigRecord :
STRUCT
    iRW          : UINT;
    iNumOfAR     : UINT;
    iAPI         : UDINT;
    iSlot        : UINT;
    iSubSlot     : UINT;
    stPNIORecord : ST_PNIORecord;
END_STRUCT
END_TYPE
```

iRW: 0: Read, 1: Write.

iNumOfAR: Anzahl der Argumente.

iAPI: API-Nummer.

iSlot: Slot-Nummer.

iSubSlot: SubSlot-Nummer.

stPNIORecord: (Typ: ST_PNIORecord [▶ 135]).

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO Hardware	Einzubindende SPS-Bibliotheken (Kategoriegruppe)
TwinCAT v3.1.0	PC oder CX (x86)	Beckhoff EL6632	Tc2_IoFunctions (IO)

5.24 ST_PNIORecord

```
TYPE ST_PNIORecord :
STRUCT
    iIndex     : UINT;
    iLength    : UINT; (* 0 for READ *)
    iTransfSeq : UINT;
    iAligned   : UINT;
END_STRUCT
END_TYPE
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO Hardware	Einzubindende SPS-Bibliotheken (Kategoriegruppe)
TwinCAT v3.1.0	PC oder CX (x86)	Beckhoff EL6632	Tc2_IoFunctions (IO)

5.25 ST_PNIOState

```
TYPE ST_PNIOState :
STRUCT
    bInDataExchange : BOOL; (* bit 0 *)
    bApplRunning    : BOOL; (* bit 2 *)
    bDiagIndicator  : BOOL; (* bit 3 *)
END_STRUCT
END_TYPE
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO Hardware	Einzubindende SPS-Bibliotheken (Kategoriegruppe)
TwinCAT v3.1.0	PC oder CX (x86)	Beckhoff EL6632	Tc2_IoFunctions (IO)

5.26 ST_PZD_IN

Datenwörter vom Antrieb zur PLC.

```
TYPE ST_PZD_IN :
STRUCT
    wSTW :WORD;
    wHIW :WORD;
    PZD3 :WORD;
    PZD4 :WORD;
    PZD5 :WORD;
    PZD6 :WORD;
END_STRUCT
END_TYPE
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO Hardware	Einzubindende SPS-Bibliotheken (Kategoriegruppe)
TwinCAT v3.1.0	PC oder CX (x86)	AX2000 Profibus box	Tc2_IoFunctions (IO)

5.27 ST_PZD_OUT

Datenwörter von der PLC zum Antrieb.

```
TYPE ST_PZD_OUT :
STRUCT
    wCtrlW :WORD;
    PZD2   :WORD;
    PZD3   :WORD;
    PZD4   :WORD;
    PZD5   :WORD;
    PZD6   :WORD;
END_STRUCT
END_TYPE
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO Hardware	Einzubindende SPS-Bibliotheken (Kategoriegruppe)
TwinCAT v3.1.0	PC oder CX (x86)	AX2000 Profibus box	Tc2_IoFunctions (IO)

5.28 ST_RAIDCntlrFound

```
TYPE ST_RAIDCntlrFound :
STRUCT
    nRAIDCntlrCount : UDINT;
    nRAIDCntlrIds   : ARRAY [1..g_nMAX_NUMBER_OF_RAID_CNTLRS] OF UDINT;
END_STRUCT
END_TYPE
```

nRAIDCntlrCount: Anzahl von RAID-Controllern

nRAIDCntlrIds: ID von jedem RAID-Controller (Default-Wert ist 4294967295 und daher ungültig).

g_nMAX_NUMBER_OF_RAID_CNTLRS ist die maximale Anzahl von RAID-Controllern und ist definiert als globale Konstante = 10.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken (Kategoriegruppe)
TwinCAT v3.1.0	PC oder CX (x86, x64, ARM)	Tc2_IoFunctions (IO)

5.29 ST_RAIDConfigReq

```
TYPE ST_RAIDConfigReq :
STRUCT
    nRAIDCntlrID  : UDINT;
    nRAIDSetIndex : UDINT;
END_STRUCT
END_TYPE
```

nRAIDCntlrID: ID des RAID-Controllers

nRAIDSetIndex: Enthält die Nummer des RAID-Sets für das Information angefordert wird. Bitte beachten Sie, dass im Fall von Beckhoff Boards CBx051 das Sets mit Index 0 beginnt.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken (Kategoriegruppe)
TwinCAT v3.1.0	PC oder CX (x86, x64, ARM)	Tc2_IoFunctions (IO)

5.30 ST_RAIDDriveStatus

```
TYPE ST_RAIDDriveStatus :
STRUCT
    eRAIDDriveStatus  : E_RAIDDriveStatus;
    eRAIDDriveUsage   : E_RAIDDriveUsage;
    nSATAPort         : UINT;
    sRAIDDriveSerial  : STRING [39];
END_STRUCT
END_TYPE
```

eRAIDDriveStatus: Enthält den Status des physischen Antriebs (Typ: E_RAIDDriveStatus [▶ 121]).

eRAIDDriveUsage: Besagt, ob der physische Antrieb Teil des RAID-Sets ist (Typ: E_RAIDDriveUsage [▶ 122]).

nSATAPort: Enthält die SAS-Addresse des physischen Antriebs. Wenn der Antrieb keine SAS-Addresse hat, wie z. B. mit einem direkt angehängten SATA Antrieb, dann sollte dieses Feld mit 0 gefüllt werden.

sRAIDDriveSerial: Seriennummer des RAID-Antriebs (40 Buchstaben).

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken (Kategoriegruppe)
TwinCAT v3.1.0	PC oder CX (x86, x64, ARM)	Tc2_IoFunctions (IO)

5.31 ST_RAIDInfo

```
TYPE ST_RAIDInfo :
STRUCT
    nNumRAIDSets     : UDINT;
    nMaxDrivesPerSet : UDINT;
    bReserved        : ARRAY [1..92] OF BYTE;
END_STRUCT
END_TYPE
```

nNumRAIDSets: Anzahl der aktuell definierten RAID-Sets. Falls bislang keine Sets definiert worden sind, wird der Wert 0 zurückgegeben.

nMaxDriverPerSet: Maximale Anzahl der physischen Antriebe in einem logischen RAID-Set. Dies kann ein absolutes Maximum sein oder das aktuell für alle Sets definierte Maximum.

bReserved: Reserviert für interne Zwecke.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken (Kategoriegruppe)
TwinCAT v3.1.0	PC oder CX (x86, x64, ARM)	Tc2_IoFunctions (IO)

5.32 ST_RAIDStatusRes

```
TYPE ST_RAIDStatusRes :
STRUCT
    nRAIDSetIndex     : UDINT;
    eRAIDType         : E_RAIDType;
```

```
    eRAIDStatus       : E_RAIDStatus;
    nRAIDDriveCount   : UINT;
    nReserved         : UINT;
    stRAIDDriveStatus : ARRAY [1..g_nMAX_NUMBER_OF_RAID_DRIVES] OF ST_RAIDDriveStatus;
END_STRUCT
END_TYPE
```

nRAIDSetIndex: RAID-Set-ID, wie beim Input.

eRAIDType: Enthält den Basis-RAID-Typ des RAID-Sets (Typ: E_RAIDType [▶ 123]). **eRAID_TYPE_NONE**, Zeigt an, dass das RAID-Set aus einem einzigen Antrieb besteht, also kein Set mit der angegebenen Nummer existiert.

eRAIDStatus: Enthält den Status des RAID-Sets (Typ: E_RAIDStatus [▶ 123]).

nRAIDDriveCount: Enthält die Anzahl von Antrieben im RAID-Set

nReserved: Reserviert.

stRAIDDriveStatus: Enthält den Status des physischen Antriebs und die Information ob der physische Antrieb Teil des RAID-Sets ist (Typ: ST_RAIDDriveStatus [▶ 137]).

g_nMAX_NUMBER_OF_RAID_DRIVES ist die Anzahl der RAID-Antriebs-Status die gelesen werden können und ist definiert als globale Konstante = 10.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken (Kategoriegruppe)
TwinCAT v3.1.0	PC oder CX (x86, x64, ARM)	Tc2_IoFunctions (IO)

5.33 ST_SercosParamAttrib

ST_SercosParamAttrib beinhaltet das Attribut nAttrib des Sercos-Parameters in einzelne Variablen zerlegt.

```
TYPE ST_SercosParamAttrib :
STRUCT
    nFactor       : UINT;
    eLength       : E_SercosAttribLen;
    bCommand      : BOOL;
    eType         : E_SercosAttribType;
    bReserved1    : BOOL;
    nComma        : USINT;
    bWriteProtCP2 : BOOL;
    bWriteProtCP3 : BOOL;
    bWriteProtCP4 : BOOL;
    bReserved2    : BOOL;
END_STRUCT
END_TYP
```

nFactor: Bits 0..15.

eLength: Bits 16..18. (Typ: E_SercosAttribLen [▶ 124]).

bCommand: Bit 19.

eType: Bits 20..22. (Typ: E_SercosAttribType [▶ 124]).

bReserved1: Bit 23.

nComma: Bits 24..27.

bWriteProtCP2: Bit 28.

bWriteProtCP3: Bit 29.

bWriteProtCP4: Bit 30.

bReserved2: Bit 31.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO Hardware	Einzubindende SPS-Bibliotheken (Kategoriegruppe)
TwinCAT v3.1.0	PC (x86)	Sercans SCS-P ISA; Sercans SCS-P PCI; Beckhoff FC750x PCI	Tc2_IoFunctions (IO)

5.34 ST_SercosParamErrList

```
TYPE ST_SercosParamErrList :
STRUCT
    iActCount    : UINT;
    iMaxCount    : UINT;
    iList        : ARRAY [0..2047] OF UINT;
    iError       : ARRAY [0..2047] OF UDINT;
END_STRUCT
END_TYPE
```

iActCount: Ist die Anzahl der übersprungenen Parameter (hier bedeutet 3 = 3 Parameterfehler).

iMaxCount: Ist die Anzahl der übersprungenen Parameter (hier bedeutet 3 = 3 Parameterfehler).

iList: Ist ein Feld von bis zu 2048 Parameternummern, bei denen Zugriffsfehler auftraten.

iError: Ist ein Feld von bis zu 2048 Zugriffsfehlernummern.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO Hardware	Einzubindende SPS-Bibliotheken (Kategoriegruppe)
TwinCAT v3.1.0	PC (x86)	Sercans SCS-P ISA; Sercans SCS-P PCI; Beckhoff FC750x PCI	Tc2_IoFunctions (IO)

5.35 ST_SercosParamList

```
TYPE ST_SercosParamList :
STRUCT
    iActCount    : UINT;
    iMaxCount    : UINT;
    iList        : ARRAY [0..2047] OF UINT;
END_STRUCT
END_TYPE
```

iActCount: Ist die aktuelle Anzahl der Parameter einer Liste * 2. Sercos speichert hier die Anzahl der Bytes, eine Parameternummer besteht aus zwei Bytes, z.B. 6 bedeutet 3 Parameter.

iMaxCount: Ist die max. Anzahl der Parameter einer Liste * 2. Sercos speichert hier die Anzahl der Bytes, eine Parameternummer besteht aus zwei Bytes, z.B. 6 bedeutet 3 Parameter.

iList: Ist ein Feld von bis zu 2048 Parameternummern.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO Hardware	Einzubindende SPS-Bibliotheken (Kategoriegruppe)
TwinCAT v3.1.0	PC (x86)	Sercans SCS-P ISA; Sercans SCS-P PCI; Beckhoff FC750x PCI	Tc2_IoFunctions (IO)

5.36 ST_UPSStatus

```
TYPE ST_UPSStatus
STRUCT
    Vendor             : STRING; (* UPS vendor name  *)
    Model              : STRING; (* UPS model name *)
    FirmwareRev        : STRING; (* UPS firmware revision *)
    SerialNumber       : STRING; (* UPS serial number *)
    BatteryLifePercent : DWORD; (* The percent of battery capacity remaining in the UPS (0..100%) *)
    BatteryLifeTime    : DWORD; (* Remaining UPS run time, in minutes *)
    eBatteryStatus     : E_BatteryStatus; (* UPS battery state *)
    eCommStatus        : E_UpsCommStatus; (* Status of the communication path to the UPS *)
    ePowerStatus       : E_UpsPowerStatus; (* Status of utility-supplied power into the UPS *)
    nPowerFailCnt      : DWORD; (* Power Fail counter. Increments every time the UPS service detects
 power fail *)
    dwChargeFlags      : DWORD; (* Battery charge status flags. This member can be one or more of th
e following values.
    Bits0..7 := General battery status flags (if all bits are set to 0 => unknown status)
        Bit0 := High (bit set => high battery charge)
        Bit1 := Low (bit set => low battery charge)
        Bit2 := Critical (bit set  => battery is near empty)
        Bit3 := Charging (bit set  => battery is charging)
        Bits4..6 := reserved (all bits are 0)
        Bit7 := No Battery (bit set => battery not found or not connected, bit not set => battery is
 existing or unknown state)
   Bits8..15 := Special status information (if all bits are set to 0 => state ok or unknown state)
        Bit8 := UPS Fan Error (bit set => fan hardware reports an error, bit not set => fan is ok)
        Bit9 := Over Temperature (bit set => over temperature detected, bit not set => temperature i
s ok)
        Bit10 := Service Interval Notify (bit set => service interval time expired, bit not set =>se
rvice interval time not expired )
        Bit11 := Under Temperature (bit set => under temperature detected , bit not set => temperatu
re is ok )
        Bit12 := Fuse Not Ok (bit set => fuse broken or missed, bit not set => fuse ok)
        Bit13 := Alarm1 (reserved for later use, bit is 0)
        Bit14 := Alarm2 (reserved for later use, bit is 0)
        Bit15 := Alarm3 (reserved for later use, bit is 0)
    Bits16..31 := (reserved for later use, all bits are 0)
 *)
END_STRUCT
END_TYPE
```

Nicht alle USV-Modelle können alle Statusinformationen liefern.

X: Die Statusinformation ist bei diesem Modell vorhanden.

*) Nur vorhanden, wenn das Model "Smart Signaling to any APC UPS & TwinCAT" konfiguriert wurde.

Statusinformation	Beckhoff BAPI v1	Beckhoff P24Vxxxx	Beckhoff CP903x ISA/ PCI-Karte	Beckhoff CX2100-09x4	APC Back-UPS Pro 280	APC Smart-UPS 420	Beschreibung
Vendor	X	X	X	X	X	X	Herstellername.
Model	X	X	X	X	X	X	Modelstring. Leerstring, wenn keine USV konfiguriert wurde.
Firmware-Rev	X	X	X	X	X	X	Versionsinformationen zur USV-Firmware. Leerstring, wenn die USV diesen Parameter nicht unterstützt.
SerialNumber	X	X	Keine	X	X	X	Seriennummer der USV. Leerstring, wenn die USV diesen Parameter nicht unterstützt.
BatteryLifePercent	X	X	Keine	X	X	X	Verbliebene Akkulaufzeit in Prozent. Der Wert ist immer Null wenn die USV diesen Parameter nicht liefern kann.
BatteryLifeTime	X	X	Keine	X	X	X	Verbliebene Akkulaufzeit in Minuten. Der Wert ist immer Null wenn die USV diesen Parameter nicht liefern kann.
eBatteryStatus	**BatteryOk**	**BatteryUnknownStatus** wenn kein Akku vorhanden ist, ab USV-Softwareversion >=2.0.0.6 und USV-Firmware >= 25.1.I **BatteryOk**	**BatteryUnknownStatus** wenn kein Akku vorhanden ist. **BatteryOk**	**BatteryUnknownStatus** wenn kein Akku vorhanden ist (gilt nur für das Model mit "Smart Battery" und nicht mit Kondensatoren). **BatteryOk**	X	X	Akku-Status (Typ: E_BatteryStatus).
eCommStatus	X	X	X	X	X	X	Status der Kommunikation zur USV (Typ: E_UpsCommStatus).
ePowerStatus	X	X	X	X	X	X	Status der externen Spannungsversorgung (Typ: E_UpsPowerStatus).
nPowerFailCnt	X	X	X	X	*X	*X	Power-Fail-Zähler. Der Zähler wird inkrementiert wenn ein Spannungsausfall vom USV-Service erkannt wurde.
dwChargeFlags	**No Battery** (Bit 7 gesetzt) ab USV-Firmware >= 33.12-0 wenn kein Akku angeschlossen. **Service-Interval Notify** (Bit 10 gesetzt). Der konfigurierte Akkuwechsel Intervall Service ist abgelaufen.	**No Battery** (Bit 7 gesetzt) ab USV-Softwareversion >=2.0.0.6 und Firmware >= 25.1.I Die Existenz des Akkus wird jede Minute überprüft. **UPS Fan Error** (Bit 8 gesetzt) ab USV-Softwareversion >=2.0.0.7 und Firmware >=40.1.I Der USV Lüfterstatus wird jede Minute überprüft. **Erfordert eine neuere (zweite) Hardwarerevision!**	**High** (Bit 0 gesetzt) wenn Akku voll geladen. **Charging** (Bit 3 gesetzt) **No Battery** (Bit 7 gesetzt) wenn kein Akku gefunden wurde.	**No Battery** (Bit 7 gesetzt). Keine Kommunikation zum Akku (gilt nur für das Model mit "Smart Battery" und nicht mit Kondensatoren). **Over Temperature** (Bit 9 gesetzt) wenn Übertemperatur detektiert wurde und das Laden des Akkus unterbrochen wurde. **Erfordert eine neuere (zweite) Hardwarerevision**. Implementiert in der USV-Softwareversion >= 3.0.0.18.	Keine	Keine	Akku-Ladestatus-Flags und spezielle Statusinformationen.

Statusinformation	Beckhoff BAPI v1	Beckhoff P24Vxxxx	Beckhoff CP903x ISA/ PCI-Karte	Beckhoff CX2100-09x4	APC Back-UPS Pro 280	APC Smart-UPS 420	Beschreibung
		Service Interval Notify (Bit 10 gesetzt). Der konfigurierte Akkuwechsel Intervall Service ist abgelaufen. Implementiert in der USV-Softwareversion >= 3.0.0.8;		**Service Interval Notify** (Bit 10 gesetzt) Die konfigurierte Akku-Service-Intervallzeit ist abgelaufen. **Under Temperature** (Bit 11 gesetzt) wenn Untertemperatur detektiert wurde und das Laden des Akkus unterbrochen wurde. **Erfordert eine neuere (zweite) Hardwarerevision**. Implementiert in der USV-Softwareversion >= 3.0.0.18. **Fuse Not Ok** (Bit 12 gesetzt) Die "Smart Battery"-Sicherung ist defekt oder nicht vorhanden. **Erfordert eine neuere (zweite) Hardwarerevision**. Implementiert in der USV-Softwareversion >= 3.0.0.18.			

Voraussetzungen

Entwicklungsumgebung	Zielplattform	USV Hardware	Einzubindende SPS-Bibliotheken (Kategoriegruppe)
TwinCAT v3.1.0	PC oder CX (x86, x64, ARM)	• Beckhoff BAPI v1; • Beckhoff P24Vxxxx; • Beckhoff CP903x-Karte (PCI/ ISA); • Beckhoff CX2100-09x4 Modelle (z.B. CX2100-0904 oder CX2100-0914 + "Smart Battery" CX2900-0192); • Die mit Beckhoff Industrie-PC ausgelieferten APC-Geräte die das Smartprotokoll unterstützen und mit dem Windows USV-Dienst konfiguriert werden können;	Tc2_IoFunctions (IO)

5.37 ST_KL1501InData

Struktur zur Verknüpfung im Eingangs-Prozessabbild.

```
TYPE ST_KL1501InData :
STRUCT
  iStatus   : USINT;
  arrDataIn : ARRAY[0..1] OF UINT;
END_STRUCT
END_TYPE
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO-Hardware	Einzubindende SPS-Bibliotheken
TwinCAT v3.1.4018.26	PC/CX	KL1501	Tc2_IoFunctions ab v3.3.5.0

5.38 ST_KL1501OutData

Struktur zur Verknüpfung im Ausgangs-Prozessabbild.

```
TYPE ST_KL1501OutData :
STRUCT
  iCtrl      : USINT;
  arrDataOut : ARRAY[0..1] OF UINT;
END_STRUCT
END_TYPE
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO-Hardware	Einzubindende SPS-Bibliotheken
TwinCAT v3.1.4018.26	PC/CX	KL1501	Tc2_IoFunctions ab v3.3.5.0

5.39 ST_KL27x1InData

Struktur zur Verknüpfung im Eingangs-Prozessabbild.

```
TYPE ST_KL27x1InData :
STRUCT
  iStatus : USINT;
  iDataIn : INT;
END_STRUCT
END_TYPE
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO-Hardware	Einzubindende SPS-Bibliotheken
TwinCAT v3.1.4018.26	PC/CX	KL2751, KL2761	Tc2_IoFunctions ab v3.3.5.0

5.40 ST_KL27x1OutData

Struktur zur Verknüpfung im Ausgangs-Prozessabbild.

```
TYPE ST_KL27x1OutData :
STRUCT
  iCtrl    : USINT;
  iDataOut : INT;
END_STRUCT
END_TYPE
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO-Hardware	Einzubindende SPS-Bibliotheken
TwinCAT v3.1.4018.26	PC/CX	KL2751, KL2761	Tc2_IoFunctions ab v3.3.5.0

5.41 ST_KL320xInData

Struktur zur Verknüpfung im Eingangs-Prozessabbild.

```
TYPE ST_KL320xInData :
STRUCT
  iStatus : USINT;
  iDataIn : INT;
END_STRUCT
END_TYPE
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO-Hardware	Einzubindende SPS-Bibliotheken
TwinCAT v3.1.4018.26	PC/CX	KL3201, KL3202, KL3204	Tc2_IoFunctions ab v3.3.5.0

5.42 ST_KL320xOutData

Struktur zur Verknüpfung im Ausgangs-Prozessabbild.

```
TYPE ST_KL320xOutData :
STRUCT
  iCtrl    : USINT;
  iDataOut : INT;
END_STRUCT
END_TYPE
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO-Hardware	Einzubindende SPS-Bibliotheken
TwinCAT v3.1.4018.26	PC/CX	KL3201, KL3202, KL3204	Tc2_IoFunctions ab v3.3.5.0

5.43 ST_KL3208InData

Struktur zur Verknüpfung im Eingangs-Prozessabbild.

```
TYPE ST_KL3208InData :
STRUCT
  iStatus : USINT;
  iDataIn : INT;
END_STRUCT
END_TYPE
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO-Hardware	Einzubindende SPS-Bibliotheken
TwinCAT v3.1.4018.26	PC/CX	KL3208	Tc2_IoFunctions ab v3.3.5.0

5.44 ST_KL3208OutData

Struktur zur Verknüpfung im Ausgangs-Prozessabbild.

```
TYPE ST_KL3208OutData :
STRUCT
  iCtrl    : USINT;
  iDataOut : INT;
END_STRUCT
END_TYPE
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO-Hardware	Einzubindende SPS-Bibliotheken
TwinCAT v3.1.4018.26	PC/CX	KL3208	Tc2_IoFunctions ab v3.3.5.0

5.45 ST_KL3228InData

Struktur zur Verknüpfung im Eingangs-Prozessabbild.

```
TYPE ST_KL3228InData :
STRUCT
  iStatus : USINT;
  iDataIn : INT;
END_STRUCT
END_TYPE
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO-Hardware	Einzubindende SPS-Bibliotheken
TwinCAT v3.1.4018.26	PC/CX	KL3228	Tc2_IoFunctions ab v3.3.5.0

5.46 ST_KL3228OutData

Struktur zur Verknüpfung im Ausgangs-Prozessabbild.

```
TYPE ST_KL3228OutData :
STRUCT
  iCtrl    : USINT;
  iDataOut : INT;
END_STRUCT
END_TYPE
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO-Hardware	Einzubindende SPS-Bibliotheken
TwinCAT v3.1.4018.26	PC/CX	KL3228	Tc2_IoFunctions ab v3.3.5.0

6 Globale Konstanten

6.1 Bibliotheksversion

Alle Bibliotheken haben eine bestimmte Version. Diese Version ist u. a. im SPS-Bibliotheks-Repository zu sehen. Eine globale Konstante enthält die Information über die Bibliotheksversion:

Global_Version

```
VAR_GLOBAL CONSTANT
    stLibVersion_Tc2_IoFunctions : ST_LibVersion;
END_VAR
```

stLibVersion_Tc2_IoFunctions: Versionsnummer der Tc2_IoFunctions-Bibliothek (Typ: ST_LibVersion).

Um zu sehen, ob die Version, die Sie haben auch die Version ist, die Sie brauchen, benutzen Sie die Funktion F_CmpLibVersion (definiert in der Tc2_System-Bibliothek)

Alle anderen Möglichkeiten Bibliotheksversionen zu vergleichen, die Sie von TwinCAT 2 kennen, sind veraltet!

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO Hardware	Einzubindende SPS-Bibliotheken (Kategoriegruppe)
TwinCAT v3.1.0	PC oder CX (x86, x64, ARM)	Alle IO-Geräte	Tc2_IoFunctions (IO)

7 Anhang

7.1 SERCOS Dateiformat der Backup-Datei

1. Datei-Header vom Typ ST_SercosFileHeader

2. n * Daten
a) Parameter-Header vom Type ST_ParamHeader
b) Parameter-Daten als Bytes

Beispiel für n Parameter

```
1 * ST_SercosFileHeader (268
Bytes)


-----------------------------------------------------

nVersion     ( 4 Bytes)

nListType    ( 4 Bytes)

cbCommentLen ( 4 Bytes)

sComment     ( 256 Bytes)


n * (ST_SercosParamHeader + Data)

-----------------------------------------------------

nIDN         ( 2 Bytes)

cbSize       ( 2 Bytes)

nAttrib      ( 4 Bytes)

arrData      (cbSize Bytes),
kann für jeden Parameter verschieden groß sein, je nach Typ oder
Listenlänge
```

Beispiel für 3 Parameter

```
ST_SercosFileHeader (268 Bytes)
-------------------------------------------------------------------------------
nVersion ( 4 Bytes), i.e. = 01 00 00 00 (= 1)
nListType ( 4 Bytes), i.e. = 00 00 00 00
cbCommentLen ( 4 Bytes), i.e. = 00 00 00 00 (= 0)
sComment ( 256 Bytes), i.e. = 00 00 00 00 00 00 00 ... 00
(256 * 00)
1st parameter ST_SercosParamHeader + Data (10 Bytes)
-------------------------------------------------------------------------------
nIDN ( 2 Bytes), i.e. = nnnncbSize( 2 Bytes), i.e. = 02 00 (= 2)nAttrib ( 4 Bytes), i.e. = xx xx xx
xx
arrData     (2 Bytes), i.e. = 12 342nd parameter ST_SercosParamHeader + Data (16 Bytes)

-------------------------------------------------------------------------------
nIDN ( 2 Bytes), i.e. = nnnncbSize( 2 Bytes), i.e. = 08 00 (= 8)nAttrib( 4 Bytes), i.e. = xx xx xx x
x
arrData ( 8 Bytes), i.e. = 12 34 56 78 9a bc de f03rd parameter ST_SercosParamHeader + Data (12 Byte
s)
-------------------------------------------------------------------------------
nIDN ( 2 Bytes), i.e. = nn nncbSize ( 2 Bytes), i.e. = 04 00 (= 4)nAttrib ( 4 Bytes), i.e. = xx xx x
x xx
arrData ( 4 Bytes), i.e. = 12 34 56 78
```

TYPE ST_SercosFileHeader (268 Bytes)

Der Datei-Header der Sercos-Backup-Datei basiert auf der Struktur ST_SercosFileHeader.

```
TYPE ST_SercosFileHeader :
STRUCT
    nVersion      : UDINT;(* 4 Bytes *)
```

```
    nListType     : UDINT;(* 4 Bytes *)
    cbCommentLen  : UDINT;(* 4 Bytes *)
    sComment      : T_MaxString;(* 256 Bytes *)
END_STRUCT
END_TYPE
```

nVersion: beinhaltet die Dateiversion, momentan 1.

nListType: beinhaltet die IDN-Parameterliste, die für das Backup benutzt wurde. Der Standard-Wert ist 192 (Liste aller Backup-Parameter), bei benutzerdefinerter Backupliste steht hier 0. Alternativ kann die Liste aller Sercos-Parameter (IDN 17) verwendet werden. Das Restore erfordert allerdings die Liste aus Parameter 192 oder über die benutzerdefinierte Liste (0) erfolgen.

cbCommentLen: Länge des Kommentars der Backup-Datei.

sComment: Kommentar der Backup-Datei. Der String wird mit allen 256 Zeichen geschrieben.

TYPE ST_SercosParamHeader (8 Bytes)

Im Anschluß an den Datei-Header folgt in der Backup-Datei je Parameter ein Parameter Header vom Typ ST_SercosParamHeader.

```
TYPE ST_SercosParamHeader :
STRUCT
    nIDN     : UINT;(* 2 Bytes *)
    cbSize   : UINT;(* 2 Bytes *)
    nAttrib  : DWORD;(* 4 Bytes *)
END_STRUCT
END_TYPE
```

nIDN: Sercos-Parameter-Nummer.

cbSize: Länge der Daten in Bytes, die diesem Header folgen. Kann für jeden Parameter verschieden sein, ja nach Parameter Typ oder Listenlänge.

nAttrib: Attribut des Sercos-Parameters (siehe ST_SercosParamData [▶ 138]), wird zur Bestimmung von Länge und Datentyp benötigt.

Parameter Daten (cbSize Bytes)

Auf jeden Sercos-Parameter-Header in der Backup-Datei folgen unmittelbar die Daten. Die Anzahl der Daten-Bytes ist im Parameter-Header in cbSize gespeichert.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO Hardware	Einzubindende SPS-Bibliotheken (Kategoriegruppe)
TwinCAT v3.1.0	PC (x86)	Sercans SCS-P ISA; Sercans SCS-P PCI; Beckhoff FC750x PCI	Tc2_IoFunctions (IO)

7.2 AX200x Profibus Parameternummer

PNU	Datentyp	Zugriff	Kurzbeschreibung	SERVOSTAR™ ASCII-Befehl
Profilparameter				
904	UINT32	ro	Nummer des unterstützen PPO-Write, immer 2	-
911	UINT32	ro	Nummer des unterstützen PPO-Read, immer 2	-
918	UINT32	ro	Teilnehmeradresse am PROFIBUS	ADDR
930	UINT32	r/w	Auswahlschalter für Betriebsart	-
963	UINT32	ro	PROFIBUS-Baudrate	-
965	Octet-String2	ro	Nummer des PROFIDRIVE-Profils (0302H)	-
970	UINT32	wo	Defaultparameters atz laden	RSTVAR
971	UINT32	wo	Parameter nichtflüchtig speichern	SAVE
Herstellerspezifische Parameter SERVOSTAR™				
Allgemeine Parameter				
1000	Visible String4	ro	Gerätekennung	-
1001	UINT32	ro	Herstellerspezifisches Fehlerregister	ERRCODE
1002	UINT32	ro	Herstellerspezifisches Statusregister	-
Drehzahlreglerparameter				
1200	UINT32	r/w	Kp – Verstärkungsfaktor des Drehzahlreglers	GV
1201	UINT32	r/w	Tn – Nachstellzeit des Drehzahlreglers	GVTN
1202	UINT32	r/w	PID – T2 – Zeitkonstante des Drehzahlreglers	GVT2
1203	UINT32	r/w	Sollwertrampe+, Drehzahlregler	ACC
1204	UINT32	r/w	Sollwertrampe-, Drehzahlregler	DEC
1205	UINT32	r/w	Not-Rampe, Drehzahlregler	DECSTOP
1206	UINT32	r/w	Maximale Drehzahl	VLIM
1207	UINT32	r/w	Überdrehzahl	VOSPD
1208	UINT32	r/w	Zählrichtung	DIR
Lagereglerparameter				
1250	UINT32	r/w	Multiplikator für Geschwindigkeiten Tippen/Ref.	VMUL

PNU	Datentyp	Zugriff	Kurzbeschreibung	SERVOSTAR™ASCII-Befehl
1251	UINT32	r/w	Achstyp	POSCNFG
1251	INTEGER32	r/w	In-Position-Fenster	PEINPOS
1253	INTEGER32	r/w	Schleppfehlerfenster	PEMAX
1254	INTEGER32	r/w	Positionsregister 1	SWE1
1255	INTEGER32	r/w	Positionsregister 2	SWE2
1256	INTEGER32	r/w	Positionsregister 3	SWE3
1257	INTEGER32	r/w	Positionsregister 4	SWE4
1258	UINT32	r/w	Auflösung Nenner	PGEARO
1259	UINT32	r/w	Auflösung Zähler	PGEARI
1260	UINT32	r/w	Minimale Beschleunigungs-, Bremszeit	PTMIN
1261	UINT32	r/w	FeedForward-Faktor Lageregler	GPFFV
1262	UINT32	r/w	KV - Faktor Lageregler	GP
1263	UINT32	r/w	KP - Faktor Lageregler	GPV
1264	UINT32	r/w	Tn - Nachstellzeit Lageregler	GPTN
1265	UINT32	r/w	Maximale Geschwindigkeit für Positionierbetrieb	PVMAX
1266	UINT32	r/w	Konfigurationsvariable für Softwareschalter	SWCNFG
1267	UINT32	r/w	Konfigurationsvariable 2 für Softwareschalter	SWCNFG2
Positionierdaten für den Lagereglermodus				
1300	INTEGER32	r/w	Position	O_P
1301	INTEGER16	r/w	Geschwindigkeit	O_V
1302	UINT32	r/w	Fahrauftragsart	O_C
1304	UINT32	r/w	Anfahrzeit (Beschleunigung)	O_ACC1
1305	UINT32	r/w	Bremszeit (Verzögerung)	O_DEC1
1306	UINT32	r/w	Ruckbegrenzung (Beschleunigung)	O_ACC2
1307	UINT32	r/w	Ruckbegrenzung (Verzögerung)	O_DEC2
1308	UINT32	r/w	Nummer des Folgefahrauftrags	O_FN
1309	UINT32	r/w	Startverzögerung für Folgefahrauftrag	O_FT
1310	2 * UINT16	wo	Kopieren eines Fahrauftrags	OCOPY
Einrichtbetrieb Lage				
1350	UINT32	r/w	Referenzfahrtart	NREF

PNU	Datentyp	Zugriff	Kurzbeschreibung	SERVOSTAR™ASCII-Befehl
1351	UINT32	r/w	Referenzfahrtrichtung	DREF
1352	UINT32	r/w	Beschleunigungsrampe (Tippen/ Referenzieren)	ACCR
1353	UINT32	r/w	Bremsrampe	DECR
1354	UINT32	r/w	Referenzoffset	ROFFS
1355	UINT32	ro	Referenzfahrtgeschwindigkeit	VREF
1356	UINT32	ro	Tippgeschwindigkeit	VJOG
Istwerte				
1400	INTEGER32	ro	Istlage 20 Bit / Umdrehung	PRD
1401	INTEGER32	ro	Drehzahl	-
1402	INTEGER32	ro	Inkrementeller Positionsistwert	-
1403	INTEGER32	ro	SI - Positionsistwert	PFB
1404	INTEGER32	ro	SI - Geschwindigkeitsistwert	PV
1405	INTEGER32	ro	SI - Schleppfehler	PE
1406	INTEGER32	ro	Effektivstrom	I
1407	INTEGER32	ro	SI - Drehzahlistwert	V
1408	INTEGER32	ro	Kühlkörpertemperatur	TEMPH
1409	INTEGER32	ro	Innentemperatur	TEMPE
1410	INTEGER32	ro	Zwischenkreisspannung	VBUS
1411	INTEGER32	ro	Ballastleistung	PBAL
1412	INTEGER32	ro	I^2t - Belastung	I2T
1413	INTEGER32	ro	Betriebsdauer	TRUN
Digital I/O - Konfiguration				
1450	UINT32	r/w	Funktion des digitalen Eingangs 1	IN1MODE
1451	UINT32	r/w	Funktion des digitalen Eingangs 2	IN2MODE
1452	UINT32	r/w	Funktion des digitalen Eingangs 3	IN3MODE
1453	UINT32	r/w	Funktion des digitalen Eingangs 4	IN4MODE
1454	INTEGER32	r/w	Hilfsvariable für digitalen Eingang 1	IN1TRIG
1455	INTEGER32	r/w	Hilfsvariable für digitalen Eingang 2	IN2TRIG
1456	INTEGER32	r/w	Hilfsvariable für digitalen Eingang 3	IN3TRIG

PNU	Datentyp	Zugriff	Kurzbeschreibung	SERVOSTAR™ASCII-Befehl
1457	INTEGER32	r/w	Hilfsvariable für digitalen Eingang 4	IN4TRIG
1458	INTEGER32	r/w	Funktion des digitalen Ausgangs 1	O1MODE
1459	INTEGER32	r/w	Funktion des digitalen Ausgangs 2	O2MODE
1460	UINT32	r/w	Hilfsvariable für digitalen Ausgang 1	O1TRIG
1461	UINT32	r/w	Hilfsvariable für digitalen Ausgang 2	O2TRIG
1462	UINT32	r/w	Zustand von vier Digitalen Inputs, Enable, 2 digitalen Outputs	STATIO
Analog - Konfiguration				
1500	UINT32	r/w	Konfiguration der analogen Eingangsfunktionen	ANCNFG
1501	UINT32	r/w	Konfiguration Monitorfunktion Analogausgang 1	ANOUT1
1502	UINT32	r/w	Offsetspannung für Analogeingang 1	ANOFF1
1503	UINT32	r/w	Filterzeitkonstante für Analogeingang 1	AVZ1
1504	UINT32	r/w	Skalierungsfaktor Geschwindigkeit Analogeing. 1	VSCALE1
1505	UINT32	r/w	Skalierungsfaktor Strom Analogeingang 1	ISCALE1
1506	UINT32	r/w	Konfiguration Monitorfunktion Analogausgang 2	ANOUT2
1507	UINT32	r/w	Offsetspannung für Analogeingang 2	ANOFF2
1508	UINT32	r/w	Skalierungsfaktor Geschwindigkeit Analogeing. 2	VSCALE2
1509	UINT32	r/w	Skalierungsfaktor Strom Analogeingang 2	ISCALE2
Motorparameter				
1550	UINT32	r/w	Konfiguration Bremse	MBRAKE
1551	UINT32	r/w	Motornummer aus Motordatenbank	MNUMBER

7.3 Fehlercodes

ErrId (hex)	ErrId (dez)	Beschreibung
0x0	0	Kein Fehler
0x8001	32769	Timeout-Fehler bei der Konfiguration
0x8002	32770	Konfigurationsbaustein passt nicht zum Klemmentyp
0x8010	32784	FB_KL1501Config: Ungültiger Zählertyp gewählt. Siehe Eingang iSetCounterType.
0x8011	32785	FB_KL1501Config: Ungültiger Zählertyp ausgelesen. Siehe Ausgang sCounterType.
0x8018	32792	FB_KL27x1Config: Ungültige Rampenzeit gewählt. Siehe Eingang iSetRampTime.
0x8019	32793	FB_KL27x1Config: Ungültiger Dimmermodus gewählt. Siehe Eingang iSetDimmerMode.
0x801A	32794	FB_KL27x1Config: Ungültige Rampenzeit ausgelesen. Siehe Ausgang sRampTime.
0x801B	32795	FB_KL27x1Config: Ungültiger Dimmermodus ausgelesen. Siehe Ausgang sDimmerMode.
0x801C	32796	FB_KL27x1Config: Ungültiges Nach-Kurzschluss-Verhalten ausgelesen. Siehe Ausgang sAfterShortCircuit.
0x801D	32797	FB_KL27x1Config: Ungültige Netzfrequenz ausgelesen. Siehe Ausgang sLineFrequency.
0x8020	32800	FB_KL3208Config: Ungültiger Sensortyp gewählt. Siehe Eingang iSetSensorType.
0x8021	32801	FB_KL3208Config: Ungültiger Sensortyp ausgelesen. Siehe Ausgang sSensorType.
0x8028	32808	FB_KL320xConfig: Zweileiteranschluss ist für KL3204 nicht zulässig.
0x8029	32809	FB_KL320xConfig: Ungültiger Sensortyp gewählt. Siehe Eingang iSetSensorType.
0x802A	32810	FB_KL320xConfig: Ungültiger Sensortyp ausgelesen. Siehe Ausgang sSensorType.
0x8030	32816	FB_KL3228Config: Ungültiger Sensortyp gewählt. Siehe Eingang iSetSensorType.
0x8031	32817	FB_KL3228Config: Ungültiger Sensortyp ausgelesen. Siehe Ausgang sSensorType.

7.4 ADS Return Codes

Gruppierung der Fehlercodes: 0x000 [▶ 154]..., 0x500 [▶ 155]..., 0x700 [▶ 156]..., 0x1000 [▶ 158]...

Globale Fehlercodes

Hex	Dec	HRESULT	Name	Beschreibung
0x0	0	0x9811 0000	ERR_NOERROR	Kein Fehler.
0x1	1	0x9811 0001	ERR_INTERNAL	Interner Fehler.
0x2	2	0x9811 0002	ERR_NORTIME	Keine Echtzeit.
0x3	3	0x9811 0003	ERR_ALLOCLOCKEDMEM	Zuweisung gesperrt - Speicherfehler.
0x4	4	0x9811 0004	ERR_INSERTMAILBOX	Postfach voll – Es konnte die ADS Nachricht nicht versendet werden. Reduzieren der Anzahl der ADS Nachrichten pro Zyklus bringt Abhilfe.
0x5	5	0x9811 0005	ERR_WRONGRECEIVEHMSG	Falsches HMSG.
0x6	6	0x9811 0006	ERR_TARGETPORTNOTFOUND	Ziel-Port nicht gefunden – ADS Server ist nicht gestartet oder erreichbar.
0x7	7	0x9811 0007	ERR_TARGETMACHINENOTFOUND	Zielrechner nicht gefunden – AMS Route wurde nicht gefunden.
0x8	8	0x9811 0008	ERR_UNKNOWNCMDID	Unbekannte Befehl-ID.
0x9	9	0x9811 0009	ERR_BADTASKID	Ungültige Task-ID.
0xA	10	0x9811 000A	ERR_NOIO	Kein IO.
0xB	11	0x9811 000B	ERR_UNKNOWNAMSCMD	Unbekannter AMS-Befehl.
0xC	12	0x9811 000C	ERR_WIN32ERROR	Win32 Fehler.
0xD	13	0x9811 000D	ERR_PORTNOTCONNECTED	Port nicht verbunden.
0xE	14	0x9811 000E	ERR_INVALIDAMSLENGTH	Ungültige AMS-Länge.
0xF	15	0x9811 000F	ERR_INVALIDAMSNETID	Ungültige AMS Net ID.
0x10	16	0x9811 0010	ERR_LOWINSTLEVEL	Installations-Level ist zu niedrig –TwinCAT 2 Lizenzfehler.
0x11	17	0x9811 0011	ERR_NODEBUGINTAVAILABLE	Kein Debugging verfügbar.
0x12	18	0x9811 0012	ERR_PORTDISABLED	Port deaktiviert – TwinCAT System Service nicht gestartet.
0x13	19	0x9811 0013	ERR_PORTALREADYCONNECTED	Port bereits verbunden.
0x14	20	0x9811 0014	ERR_AMSSYNC_W32ERROR	AMS Sync Win32 Fehler.
0x15	21	0x9811 0015	ERR_AMSSYNC_TIMEOUT	AMS Sync Timeout.
0x16	22	0x9811 0016	ERR_AMSSYNC_AMSERROR	AMS Sync Fehler.
0x17	23	0x9811 0017	ERR_AMSSYNC_NOINDEXINMAP	Keine Index-Map für AMS Sync vorhanden.
0x18	24	0x9811 0018	ERR_INVALIDAMSPORT	Ungültiger AMS-Port.
0x19	25	0x9811 0019	ERR_NOMEMORY	Kein Speicher.
0x1A	26	0x9811 001A	ERR_TCPSEND	TCP Sendefehler.
0x1B	27	0x9811 001B	ERR_HOSTUNREACHABLE	Host nicht erreichbar.
0x1C	28	0x9811 001C	ERR_INVALIDAMSFRAGMENT	Ungültiges AMS Fragment.
0x1D	29	0x9811 001D	ERR_TLSSEND	TLS Sendefehler – Secure ADS Verbindung fehlgeschlagen.
0x1E	30	0x9811 001E	ERR_ACCESSDENIED	Zugriff Verweigert – Secure ADS Zugriff verweigert.

Router Fehlercodes

Hex	Dec	HRESULT	Name	Beschreibung
0x500	1280	0x9811 0500	ROUTERERR_NOLOCKEDMEMORY	Lockierter Speicher kann nicht zugewiesen werden.
0x501	1281	0x9811 0501	ROUTERERR_RESIZEMEMORY	Die Größe des Routerspeichers konnte nicht geändert werden.
0x502	1282	0x9811 0502	ROUTERERR_MAILBOXFULL	Das Postfach hat die maximale Anzahl der möglichen Meldungen erreicht.
0x503	1283	0x9811 0503	ROUTERERR_DEBUGBOXFULL	Das Debug Postfach hat die maximale Anzahl der möglichen Meldungen erreicht.
0x504	1284	0x9811 0504	ROUTERERR_UNKNOWNPORTTYPE	Der Porttyp ist unbekannt.
0x505	1285	0x9811 0505	ROUTERERR_NOTINITIALIZED	Router ist nicht initialisiert.
0x506	1286	0x9811 0506	ROUTERERR_PORTALREADYINUSE	Die Portnummer ist bereits vergeben.
0x507	1287	0x9811 0507	ROUTERERR_NOTREGISTERED	Der Port ist nicht registriert.
0x508	1288	0x9811 0508	ROUTERERR_NOMOREQUEUES	Die maximale Portanzahl ist erreicht.
0x509	1289	0x9811 0509	ROUTERERR_INVALIDPORT	Der Port ist ungültig.
0x50A	1290	0x9811 050A	ROUTERERR_NOTACTIVATED	Der Router ist nicht aktiv.
0x50B	1291	0x9811 050B	ROUTERERR_FRAGMENTBOXFULL	Das Postfach hat die maximale Anzahl für fragmentierte Nachrichten erreicht.
0x50C	1292	0x9811 050C	ROUTERERR_FRAGMENTTIMEOUT	Fragment Timeout aufgetreten.
0x50D	1293	0x9811 050D	ROUTERERR_TOBEREMOVED	Port wird entfernt.

Allgemeine ADS Fehlercodes

Hex	Dec	HRESULT	Name	Beschreibung
0x700	1792	0x9811 0700	ADSERR_DEVICE_ERROR	Allgemeiner Gerätefehler.
0x701	1793	0x9811 0701	ADSERR_DEVICE_SRVNOTSUPP	Service wird vom Server nicht unterstützt.
0x702	1794	0x9811 0702	ADSERR_DEVICE_INVALIDGRP	Ungültige Index-Gruppe.
0x703	1795	0x9811 0703	ADSERR_DEVICE_INVALIDOFFSET	Ungültiger Index-Offset.
0x704	1796	0x9811 0704	ADSERR_DEVICE_INVALIDACCESS	Lesen oder Schreiben nicht gestattet.
0x705	1797	0x9811 0705	ADSERR_DEVICE_INVALIDSIZE	Parametergröße nicht korrekt.
0x706	1798	0x9811 0706	ADSERR_DEVICE_INVALIDDATA	Ungültige Daten-Werte.
0x707	1799	0x9811 0707	ADSERR_DEVICE_NOTREADY	Gerät nicht betriebsbereit.
0x708	1800	0x9811 0708	ADSERR_DEVICE_BUSY	Gerät beschäftigt.
0x709	1801	0x9811 0709	ADSERR_DEVICE_INVALIDCONTEXT	Ungültiger Kontext vom Betriebssystem - Kann durch Verwendung von ADS Bausteinen in unterschiedlichen Tasks auftreten. Abhilfe kann die Multitasking-Syncronisation in der SPS geben.
0x70A	1802	0x9811 070A	ADSERR_DEVICE_NOMEMORY	Nicht genügend Speicher.
0x70B	1803	0x9811 070B	ADSERR_DEVICE_INVALIDPARM	Ungültige Parameter-Werte.
0x70C	1804	0x9811 070C	ADSERR_DEVICE_NOTFOUND	Nicht gefunden (Dateien,...).
0x70D	1805	0x9811 070D	ADSERR_DEVICE_SYNTAX	Syntax-Fehler in Datei oder Befehl.
0x70E	1806	0x9811 070E	ADSERR_DEVICE_INCOMPATIBLE	Objekte stimmen nicht überein.
0x70F	1807	0x9811 070F	ADSERR_DEVICE_EXISTS	Objekt ist bereits vorhanden.
0x710	1808	0x9811 0710	ADSERR_DEVICE_SYMBOLNOTFOUND	Symbol nicht gefunden.
0x711	1809	0x9811 0711	ADSERR_DEVICE_SYMBOLVERSIONINVALID	Symbol-Version ungültig – Kann durch einen Online-Change auftreten. Erzeuge einen neuen Handle.
0x712	1810	0x9811 0712	ADSERR_DEVICE_INVALIDSTATE	Gerät (Server) ist im ungültigen Zustand.
0x713	1811	0x9811 0713	ADSERR_DEVICE_TRANSMODENOTSUPP	AdsTransMode nicht unterstützt.
0x714	1812	0x9811 0714	ADSERR_DEVICE_NOTIFYHNDINVALID	Notification Handle ist ungültig.
0x715	1813	0x9811 0715	ADSERR_DEVICE_CLIENTUNKNOWN	Notification-Client nicht registriert.
0x716	1814	0x9811 0716	ADSERR_DEVICE_NOMOREHDLS	Keine weiteren Notification Handles verfügbar.
0x717	1815	0x9811 0717	ADSERR_DEVICE_INVALIDWATCHSIZE	Größe der Notification zu groß.
0x718	1816	0x9811 0718	ADSERR_DEVICE_NOTINIT	Gerät nicht initialisiert.
0x719	1817	0x9811 0719	ADSERR_DEVICE_TIMEOUT	Gerät hat einen Timeout.
0x71A	1818	0x9811 071A	ADSERR_DEVICE_NOINTERFACE	Interface Abfrage fehlgeschlagen.
0x71B	1819	0x9811 071B	ADSERR_DEVICE_INVALIDINTERFACE	Falsches Interface angefordert.
0x71C	1820	0x9811 071C	ADSERR_DEVICE_INVALIDCLSID	Class-ID ist ungültig.
0x71D	1821	0x9811 071D	ADSERR_DEVICE_INVALIDOBJID	Object-ID ist ungültig.
0x71E	1822	0x9811 071E	ADSERR_DEVICE_PENDING	Anforderung steht aus.
0x71F	1823	0x9811 071F	ADSERR_DEVICE_ABORTED	Anforderung wird abgebrochen.
0x720	1824	0x9811 0720	ADSERR_DEVICE_WARNING	Signal-Warnung.
0x721	1825	0x9811 0721	ADSERR_DEVICE_INVALIDARRAYIDX	Ungültiger Array-Index.
0x722	1826	0x9811 0722	ADSERR_DEVICE_SYMBOLNOTACTIVE	Symbol nicht aktiv.
0x723	1827	0x9811 0723	ADSERR_DEVICE_ACCESSDENIED	Zugriff verweigert.
0x724	1828	0x9811 0724	ADSERR_DEVICE_LICENSENOTFOUND	Fehlende Lizenz.
0x725	1829	0x9811 0725	ADSERR_DEVICE_LICENSEEXPIRED	Lizenz abgelaufen.
0x726	1830	0x9811 0726	ADSERR_DEVICE_LICENSEEXCEEDED	Lizenz überschritten.
0x727	1831	0x9811 0727	ADSERR_DEVICE_LICENSEINVALID	Lizenz ungültig.
0x728	1832	0x9811 0728	ADSERR_DEVICE_LICENSESYSTEMID	Lizenzproblem: System-ID ist ungültig.
0x729	1833	0x9811 0729	ADSERR_DEVICE_LICENSENOTIMELIMIT	Lizenz nicht zeitlich begrenzt.
0x72A	1834	0x9811 072A	ADSERR_DEVICE_LICENSEFUTUREISSUE	Lizenzproblem: Zeitpunkt in der Zukunft.
0x72B	1835	0x9811 072B	ADSERR_DEVICE_LICENSETIMETOLONG	Lizenz-Zeitraum zu lang.
0x72C	1836	0x9811 072C	ADSERR_DEVICE_EXCEPTION	Exception beim Systemstart.
0x72D	1837	0x9811 072D	ADSERR_DEVICE_LICENSEDUPLICATED	Lizenz-Datei zweimal gelesen.
0x72E	1838	0x9811 072E	ADSERR_DEVICE_SIGNATUREINVALID	Ungültige Signatur.
0x72F	1839	0x9811 072F	ADSERR_DEVICE_CERTIFICATEINVALID	Zertifikat ungültig.
0x730	1840	0x9811 0730	ADSERR_DEVICE_LICENSEOEMNOTFOUND	Public Key vom OEM nicht bekannt.
0x731	1841	0x9811 0731	ADSERR_DEVICE_LICENSERESTRICTED	Lizenz nicht gültig für diese System.ID.
0x732	1842	0x9811 0732	ADSERR_DEVICE_LICENSEDEMODENIED	Demo-Lizenz untersagt.
0x733	1843	0x9811 0733	ADSERR_DEVICE_INVALIDFNCID	Funktions-ID ungültig.
0x734	1844	0x9811 0734	ADSERR_DEVICE_OUTOFRANGE	Außerhalb des gültigen Bereiches.
0x735	1845	0x9811 0735	ADSERR_DEVICE_INVALIDALIGNMENT	Ungültiges Alignment.

Hex	Dec	HRESULT	Name	Beschreibung
0x736	1846	0x9811 0736	ADSERR_DEVICE_LICENSEPLATFORM	Ungültiger Plattform Level.
0x737	1847	0x9811 0737	ADSERR_DEVICE_FORWARD_PL	Kontext – Weiterleitung zum Passiv-Level.
0x738	1848	0x9811 0738	ADSERR_DEVICE_FORWARD_DL	Kontext – Weiterleitung zum Dispatch-Level.
0x739	1849	0x9811 0739	ADSERR_DEVICE_FORWARD_RT	Kontext – Weiterleitung zur Echtzeit.
0x740	1856	0x9811 0740	ADSERR_CLIENT_ERROR	Clientfehler.
0x741	1857	0x9811 0741	ADSERR_CLIENT_INVALIDPARM	Dienst enthält einen ungültigen Parameter.
0x742	1858	0x9811 0742	ADSERR_CLIENT_LISTEMPTY	Polling-Liste ist leer.
0x743	1859	0x9811 0743	ADSERR_CLIENT_VARUSED	Var-Verbindung bereits im Einsatz.
0x744	1860	0x9811 0744	ADSERR_CLIENT_DUPLINVOKEID	Die aufgerufene ID ist bereits in Benutzung.
0x745	1861	0x9811 0745	ADSERR_CLIENT_SYNCTIMEOUT	Timeout ist aufgetreten – Die Gegenstelle antwortet nicht im vorgegebenen ADS Timeout. Die Routeneinstellung der Gegenstelle kann falsch konfiguriert sein.
0x746	1862	0x9811 0746	ADSERR_CLIENT_W32ERROR	Fehler im Win32 Subsystem.
0x747	1863	0x9811 0747	ADSERR_CLIENT_TIMEOUTINVALID	Ungültiger Client Timeout-Wert.
0x748	1864	0x9811 0748	ADSERR_CLIENT_PORTNOTOPEN	Port nicht geöffnet.
0x749	1865	0x9811 0749	ADSERR_CLIENT_NOAMSADDR	Keine AMS Adresse.
0x750	1872	0x9811 0750	ADSERR_CLIENT_SYNCINTERNAL	Interner Fehler in Ads-Sync.
0x751	1873	0x9811 0751	ADSERR_CLIENT_ADDHASH	Überlauf der Hash-Tabelle.
0x752	1874	0x9811 0752	ADSERR_CLIENT_REMOVEHASH	Schlüssel in der Tabelle nicht gefunden.
0x753	1875	0x9811 0753	ADSERR_CLIENT_NOMORESYM	Keine Symbole im Cache.
0x754	1876	0x9811 0754	ADSERR_CLIENT_SYNCRESINVALID	Ungültige Antwort erhalten.
0x755	1877	0x9811 0755	ADSERR_CLIENT_SYNCPORTLOCKED	Sync Port ist verriegelt.

RTime Fehlercodes

Hex	Dec	HRESULT	Name	Beschreibung
0x1000	4096	0x9811 1000	RTERR_INTERNAL	Interner Fehler im Echtzeit-System.
0x1001	4097	0x9811 1001	RTERR_BADTIMERPERIODS	Timer-Wert nicht gültig.
0x1002	4098	0x9811 1002	RTERR_INVALIDTASKPTR	Task-Pointer hat den ungültigen Wert 0 (null).
0x1003	4099	0x9811 1003	RTERR_INVALIDSTACKPTR	Stack-Pointer hat den ungültigen Wert 0 (null).
0x1004	4100	0x9811 1004	RTERR_PRIOEXISTS	Die Request Task Priority ist bereits vergeben.
0x1005	4101	0x9811 1005	RTERR_NOMORETCB	Kein freier TCB (Task Control Block) verfügbar. Maximale Anzahl von TCBs beträgt 64.
0x1006	4102	0x9811 1006	RTERR_NOMORESEMAS	Keine freien Semaphoren zur Verfügung. Maximale Anzahl der Semaphoren beträgt 64.
0x1007	4103	0x9811 1007	RTERR_NOMOREQUEUES	Kein freier Platz in der Warteschlange zur Verfügung. Maximale Anzahl der Plätze in der Warteschlange beträgt 64.
0x100D	4109	0x9811 100D	RTERR_EXTIRQALREADYDEF	Ein externer Synchronisations-Interrupt wird bereits angewandt.
0x100E	4110	0x9811 100E	RTERR_EXTIRQNOTDEF	Kein externer Sync-Interrupt angewandt.
0x100F	4111	0x9811 100F	RTERR_EXTIRQINSTALLFAILED	Anwendung des externen Synchronisierungs-Interrupts ist fehlgeschlagen.
0x1010	4112	0x9811 1010	RTERR_IRQLNOTLESSOREQUAL	Aufruf einer Service-Funktion im falschen Kontext
0x1017	4119	0x9811 1017	RTERR_VMXNOTSUPPORTED	Intel VT-x Erweiterung wird nicht unterstützt.
0x1018	4120	0x9811 1018	RTERR_VMXDISABLED	Intel VT-x Erweiterung ist nicht aktiviert im BIOS.
0x1019	4121	0x9811 1019	RTERR_VMXCONTROLSMISSING	Fehlende Funktion in Intel VT-x Erweiterung.
0x101A	4122	0x9811 101A	RTERR_VMXENABLEFAILS	Aktivieren von Intel VT-x schlägt fehl.

TCP Winsock-Fehlercodes

Hex	Dec	Name	Beschreibung
0x274C	10060	WSAETIMEDOUT	Verbindungs Timeout aufgetreten - Fehler beim Herstellen der Verbindung, da die Gegenstelle nach einer bestimmten Zeitspanne nicht ordnungsgemäß reagiert hat, oder die hergestellte Verbindung konnte nicht aufrecht erhalten werden, da der verbundene Host nicht reagiert hat.
0x274D	10061	WSAECONNREFUSED	Verbindung abgelehnt - Es konnte keine Verbindung hergestellt werden, da der Zielcomputer dies explizit abgelehnt hat. Dieser Fehler resultiert normalerweise aus dem Versuch, eine Verbindung mit einem Dienst herzustellen, der auf dem fremden Host inaktiv ist—das heißt, einem Dienst, für den keine Serveranwendung ausgeführt wird.
0x2751	10065	WSAEHOSTUNREACH	Keine Route zum Host - Ein Socketvorgang bezog sich auf einen nicht verfügbaren Host.
Weitere Winsock-Fehlercodes: Win32-Fehlercodes			

Mehr Informationen:
www.beckhoff.de/te1000

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