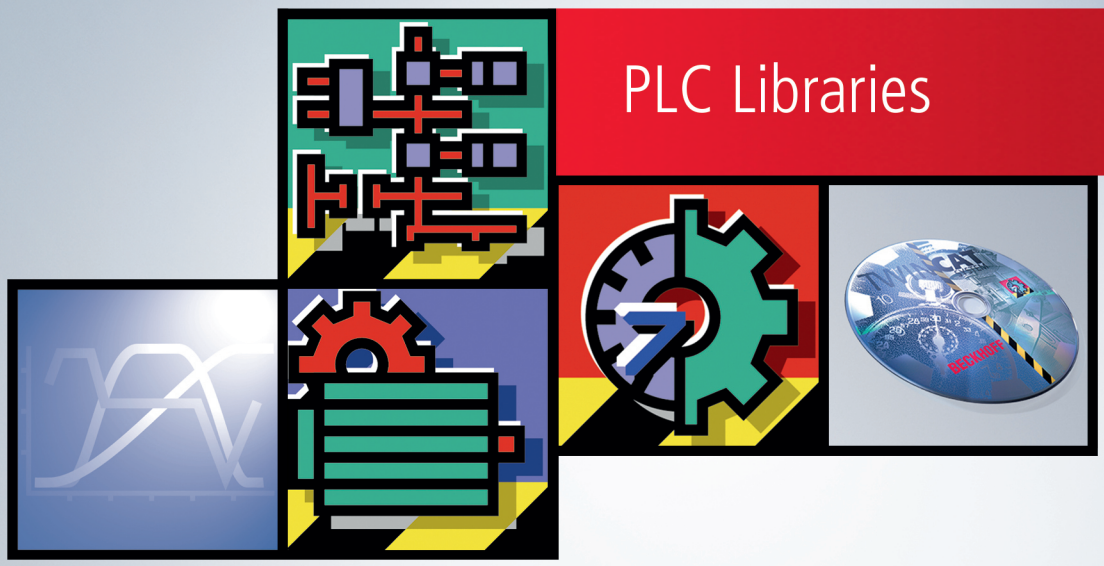
BECKHOFF New Automation Technology

Handbuch | DE

TX1200

TwinCAT 2 | PLC-Bibliothek: TcIoFunctions



18.10.2022 | Version: 1.1

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

Patente

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

EtherCAT®

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

Copyright

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.

1.3 Hinweise zur Informationssicherheit

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

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

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

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

2 Übersicht

Die 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 [▶ 19]	Die AmsNetId einer Box anhand der Feldbusadresse der Box und der Geräte-Id lesen
IOF_GetDeviceCount [▶ 20]	Die Anzahl der IO-Geräte lesen
IOF_GetDeviceIDByName [▶ 21]	Die Geräte-Id anhand der Geräte-Bezeichnung ermitteln
IOF_GetDeviceIDs [▶ 22]	Alle Geräte-Ids lesen
IOF_GetDeviceName [▶ 23]	Die Geräte-Bezeichnung anhand der Geräte-Id lesen
IOF_GetDeviceNetId [▶ 25]	Die AmsNetId anhand der Geräte-Id lesen
IOF_GetDeviceType [▶ 26]	Den Geräte-Typ anhand der Geräte-Id lesen
IOF_GetDeviceInfoByName [▶ 27]	Die Geräte-Id und die AmsNetId anhand der Geräte-Bezeichnung ermitteln

Feldbusspezifische und gerätespezifische Funktionen

CANopen

Name	Beschreibung
IOF_CAN_Layer2Command [▶ 33]	Ein Layer 2 Kommando ausführen

Beckhoff Lightbus

Name	Beschreibung
IOF_LB_BreakLocationTest [▶ 28]	Bruchstellen-Test des Lichtwellenleiterrings
IOF_LB_ParityCheck [▶ 30]	Parity-Zähler lesen
IOF_LB_ParityCheckWithReset [▶ 31]	Parity-Zähler lesen und zurücksetzen

SERCOS

Name	Beschreibung
IOF_SER_GetPhase [▶ 34]	Die aktuelle Phase lesen
IOF_SER_ResetErr [▶ 37]	Reset des Fehlerpuffers
IOF_SER_SaveFlash [▶ 36]	Parameter im Flash speichern
IOF_SER_SetPhase [▶ 38]	Die aktuelle Phase setzen
IOF_SER_IDN_Read [▶ 39]	Sercos-Drive-Parameter lesen
IOF_SER_IDN_Write [▶ 41]	Sercos-Drive-Parameter schreiben
IOF_SER_DRIVE_Backup [▶ 42]	Backup und Restore der Sercos-Drive-Parameter in/aus einer Datei

Name	Beschreibung
IOF_SER_DRIVE_BackupEx [▶ 44]	Backup und Restore der Sercos-Drive-Parameter in/aus einer Datei (erweiterte Funktionalität)
IOF_SER_DRIVE_Reset [▶ 47]	Drive-Reset eines Sercos-Drives per Kommando auf Parameter S-0-0099 (IDN99)

Profibus DPV1 (Sinamics)

Name	Beschreibung
F_CreateDpv1ReadReqPkg [▶ 73]	DPV1 Telegramm für Parameterlesen erzeugen
FB_Dpv1Read [▶ 74]	DPV1 Telegramm für Parameterlesen senden
F_SplitDpv1ReadResPkg [▶ 76]	DPV1 Antwort-Telegramm für Parameterlesen auswerten
F_CreateDpv1WriteReqPkg [▶ 77]	DPV1 Telegramm für Parameterschreiben erzeugen
FB_Dpv1Write [▶ 78]	DPV1 Telegramm für Parameterschreiben senden
F_SplitDpv1WriteResPkg [▶ 80]	DPV1 Antwort-Telegramm für Parameterschreiben auswerten

Profinet DPV1 (Sinamics)

Name	Beschreibung
F_CreateDpv1ReadReqPkgPNET [▶ 81]	DPV1 Telegramm für Parameterlesen erzeugen
FB_Dpv1ReadPNET [▶ 82]	DPV1 Telegramm für Parameterlesen senden
F_SplitDpv1ReadResPkgPNET [▶ 84]	DPV1 Antwort-Telegramm für Parameterlesen auswerten
F_CreateDpv1WriteReqPkgPNET [▶ 85]	DPV1 Telegramm für Parameterschreiben erzeugen
FB_Dpv1WritePNET [▶ 86]	DPV1 Telegramm für Parameterschreiben senden
F_SplitDpv1WriteResPkgPNET [▶ 89]	DPV1 Antwort-Telegramm für Parameterschreiben auswerten

NOV/DP-RAM

Name	Beschreibung
FB_NovRamReadWrite [▶ 48]	Daten in das NOV-RAM schreiben oder aus dem NOV-RAM lesen
FB_NovRamReadWriteEx [▶ 50]	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 [▶ 52]	Den Addresspointer und die konfigurierte Größe vom NOV/DP-RAM lesen

AX200x Profibus

Funktionsbausteine für den Zugriff auf den AX20XX über den Profibus: Übersicht [▶ 54].

ASI Master Terminal

Funktionsbausteine für den Zugriff auf eine ASI-Masterklemme: Übersicht [▶ 63].

Beckhoff USV (unter Windows USV Service)

Name	Beschreibung
FB_GetUPSStatus [▶ 90]	Den Status der USV aus der SPS lesen.

Busklemmen-Konfiguration

Name	Beschreibung
FB_KL1501Config [▶ 92]	Konfiguration einer KL1501.
FB_KL27x1Config [▶ 96]	Konfiguration einer KL2751 oder KL2761.
FB_KL320xConfig [▶ 100]	Konfiguration einer KL3201, KL3202 oder KL3204.
FB_KL3208Config [▶ 103]	Konfiguration einer KL3208.
FB_KL3228Config [▶ 106]	Konfiguration einer KL3228.

Drittherstellergeräte

INTERBUS Phoenix IBS SC/I-T Funktionen

Phoenix IBS SC/I-T Funktionen: Übersicht [▶ 109].

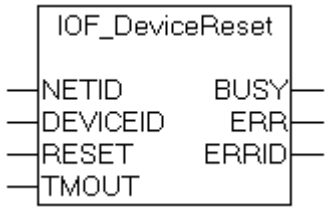
Name	Beschreibung
SCIT_ActivateConfiguration [▶ 110]	Führt den Befehl **Activate_Configuration** aus
SCIT_DeactivateConfiguration [▶ 111]	Führt den Befehl **Deactivate_Configuration** aus
SCIT_StartDataTransfer [▶ 112]	Führt den Befehl **Start_Data_Transfer** aus
SCIT_StopDataTransfer [▶ 113]	Führt den Befehl **Stop_Data_Transfer** aus
SCIT_AlarmStop [▶ 115]	Führt den Befehl **Alarm_Stop** aus
SCIT_ControlActiveConfiguration [▶ 116]	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 [▶ 117]	Liefert Fehlerart und Fehlerort eines Interbus-Teilnehmers nach einem Busfehler
SCIT_ConfDevErrAll [▶ 119]	Peripheriestörungen aller Geräte quittieren

Tab. 1: ads-tec

Name	Beschreibung
FB_ReadAdsTecSysData [▶ 120]	Liest die Systemdaten/Diagnosedaten

3 IO Funktionen

3.1 IOF_DeviceReset



Der Funktionsbaustein IOF_DeviceReset führt ein Reset eines IO-Gerätes (z.B. einer Feldbuskarte) durch. Die Funktion entspricht der IO-Reset-Funktion aus dem TwinCAT-Systemmenü.

VAR_INPUT

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

NETID: Hier kann die AmsNetId des TwinCAT-Rechners angegeben werden, auf dem die Funktion ausgeführt werden soll. 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 Manager festgelegt.

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

TMOUT: Gibt die Timeout-Zeit an, die bei der Ausführung des ADS-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 ADS-Fehler bei der Übertragung des Kommandos erfolgen, dann wird dieser Ausgang gesetzt, nachdem der BUSY-Ausgang zurückgesetzt wurde.

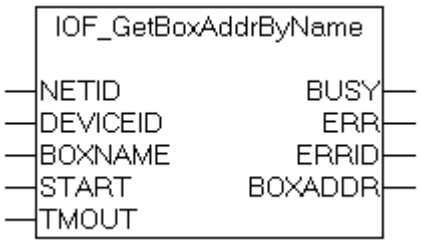
ERRID: Liefert bei einem gesetzten ERR-Ausgang die ADS-Fehlernummer.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO-Hardware	Einzubindende SPS-Bibliotheken
TwinCAT v2.7.0	PC (i386)	All IO devices	Standard.Lib; PLCSystem.Lib; TcPLCAds.Lib; TcPlcIoFunctions.Lib
TwinCAT v2.8.0	PC (i386)	All IO devices	TcIoFunctions.Lib

Entwicklungsumgebung	Zielplattform	IO-Hardware	Einzubindende SPS-Bibliotheken
			(Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib werden automatisch eingebunden)

3.2 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 Manager vom Benutzer festgelegt werden. Intern wird in dem Funktionsbaustein eine Instanz des ADSRDWRT-Funktionsbausteins aufgerufen.

VAR_INPUT

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

NETID: Hier kann die AmsNetId des TwinCAT-Rechners angegeben werden, auf dem die Funktion ausgeführt werden soll. 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 Manager festgelegt.

BOXNAME: Die Box-Bezeichnung als String.

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

TMOUT: Gibt die Timeout-Zeit an, die bei der Ausführung des ADS-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 ADS-Fehler bei der Übertragung des Kommandos erfolgen, dann wird dieser Ausgang gesetzt, nachdem der BUSY-Ausgang zurückgesetzt wurde.

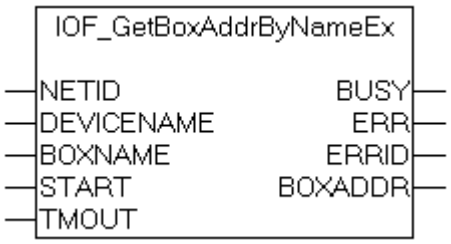
ERRID: Liefert bei einem gesetzten ERR-Ausgang die ADS-Fehlernummer.

BOXADDR: Die Feldbusadresse der Box.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO-Hardware	Einzubindende SPS-Bibliotheken
TwinCAT v2.7.0	PC (i386)	All IO devices	Standard.Lib; PLCSystem.Lib; TcPLCAds.Lib; TcPlcIoFunctions.Lib
TwinCAT v2.8.0	PC (i386)	All IO devices	TcIoFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib werden automatisch eingebunden)

3.3 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 Manager vom Benutzer festgelegt werden. Intern wird in dem Funktionsbaustein eine Instanz des ADSRDWRT-Funktionsbausteins aufgerufen.

VAR_INPUT

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

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

DEVICENAME: Die Geräte-Bezeichnung eines IO-Gerätes als String.

BOXNAME: Die Box-Bezeichnung als String.

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

TMOUT: Gibt die Timeout-Zeit an, die bei der Ausführung des ADS-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 ADS-Fehler bei der Übertragung des Kommandos erfolgen, dann wird dieser Ausgang gesetzt, nachdem der BUSY-Ausgang zurückgesetzt wurde.

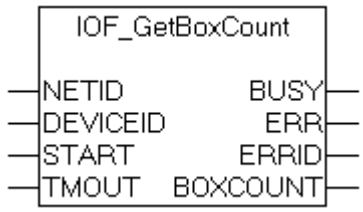
ERRID: Liefert bei einem gesetzten ERR-Ausgang die ADS-Fehlernummer.

BOXADDR: Die Feldbusadresse der Box.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO-Hardware	Einzubindende SPS-Bibliotheken
TwinCAT v2.7.0	PC (i386)	All IO devices	Standard.Lib; PLCSystem.Lib; TcPLCAds.Lib; TcPlcIoFunctions.Lib
TwinCAT v2.8.0	PC (i386)	All IO devices	TcIoFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib werden automatisch eingebunden)

3.4 IOF_GetBoxCount



Der Funktionsbaustein IOF_GetBoxCount liest die Anzahl der konfigurierten und aktiven Boxen (Box = Slave, Modul, Station) eines IO-Gerätes. Intern wird in dem Funktionsbaustein eine Instanz des ADSREAD-Funktionsbausteins aufgerufen.

VAR_INPUT

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

NETID: Hier kann die AmsNetId des TwinCAT-Rechners angegeben werden, auf dem die Funktion ausgeführt werden soll. 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 Manager festgelegt.

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

TMOUT: Gibt die Timeout-Zeit an, die bei der Ausführung des ADS-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 ADS-Fehler bei der Übertragung des Kommandos erfolgen, dann wird dieser Ausgang gesetzt, nachdem der BUSY-Ausgang zurückgesetzt wurde.

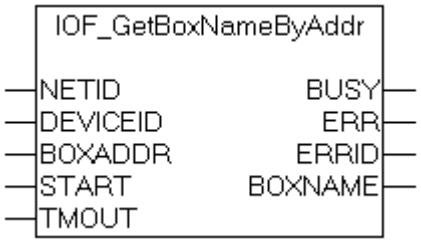
ERRID: Liefert bei einem gesetzten ERR-Ausgang die ADS-Fehlernummer.

BOXCOUNT: Anzahl der Boxen.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO-Hardware	Einzubindende SPS-Bibliotheken
TwinCAT v2.7.0	PC (i386)	All IO devices	Standard.Lib; PLCSystem.Lib; TcPLCAds.Lib; TcPlcIoFunctions.Lib
TwinCAT v2.8.0	PC (i386)	All IO devices	TcIoFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib werden automatisch eingebunden)

3.5 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-System Manager konfigurierte Box-Bezeichnung als String zurück. Intern wird in dem Funktionsbaustein eine Instanz des ADSRDWRT-Funktionsbausteins aufgerufen.

VAR_INPUT

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

NETID: Hier kann die AmsNetId des TwinCAT-Rechners angegeben werden, auf dem die Funktion ausgeführt werden soll. 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 Manager 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 ADS-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 ADS-Fehler bei der Übertragung des Kommandos erfolgen, dann wird dieser Ausgang gesetzt, nachdem der BUSY-Ausgang zurückgesetzt wurde.

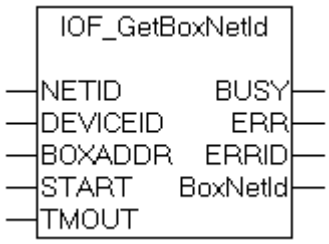
ERRID: Liefert bei einem gesetzten ERR-Ausgang die ADS-Fehlernummer.

BOXNAME: Die Box-Bezeichnung als String.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO-Hardware	Einzubindende SPS-Bibliotheken
TwinCAT v2.7.0	PC (i386)	All IO devices	Standard.Lib; PLCSystem.Lib; TcPLCAds.Lib; TcPlcIoFunctions.Lib
TwinCAT v2.8.0	PC (i386)	All IO devices	TcIoFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib werden automatisch eingebunden)

3.6 IOF_GetBoxNetId



Einigen Boxen (Slave-Module) kann eine AmsNetId während der Konfiguration im TwinCAT-System Manager zugewiesen werden. Die AmsNetId kann dann benutzt werden, um auf der Box Firmware-Funktionen ausführen zu können. Der Funktionsbaustein IOF_GetBoxNetId ermittelt die AmsNetId anhand der Geräte-Id des Masters und der Feldbusadresse oder logischen Adresse im Feldbus. Die Geräte-Ids werden während der Konfiguration vom TwinCAT-System Manager festgelegt und können nicht vom Benutzer konfiguriert werden. Intern wird in dem Funktionsbaustein eine Instanz des ADSRDWRT-Funktionsbausteins aufgerufen.

VAR_INPUT

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

NETID: Hier kann die AmsNetId des TwinCAT-Rechners angegeben werden, auf dem die Funktion ausgeführt werden soll. 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 ADS-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 ADS-Fehler bei der Übertragung des Kommandos erfolgen, dann wird dieser Ausgang gesetzt, nachdem der BUSY-Ausgang zurückgesetzt wurde.

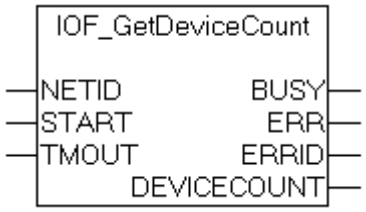
ERRID: Liefert bei einem gesetzten ERR-Ausgang die ADS-Fehlernummer.

BoxNetId: Die AmsNetId der Box als String

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO-Hardware	Einzubindende SPS-Bibliotheken
TwinCAT v2.7.0	PC (i386)	All IO boxes with NetID	Standard.Lib; PLCSystem.Lib; TcPLCAds.Lib; TcPlcIoFunctions.Lib
TwinCAT v2.8.0	PC (i386)	All IO boxes with NetID	TcIoFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib werden automatisch eingebunden)

3.7 IOF_GetDeviceCount



Der Funktionsbaustein IOF_GetDeviceCount liest die Anzahl der konfigurierten und aktiven IO-Geräte. Intern wird in dem Funktionsbaustein eine Instanz des ADSREAD-Funktionsbausteins aufgerufen.

VAR_INPUT

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

NETID: Hier kann die AmsNetId des TwinCAT-Rechners angegeben werden, auf dem die Funktion ausgeführt werden soll. 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 ADS-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 ADS-Fehler bei der Übertragung des Kommandos erfolgen, dann wird dieser Ausgang gesetzt, nachdem der BUSY-Ausgang zurückgesetzt wurde.

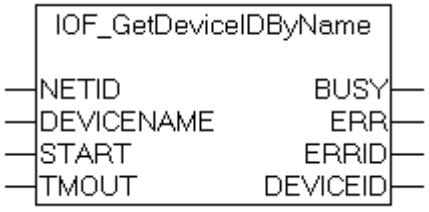
ERRID: Liefert bei einem gesetzten ERR-Ausgang die ADS-Fehlernummer.

DEVICECOUNT: Anzahl der IO-Geräte.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO-Hardware	Einzubindende SPS-Bibliotheken
TwinCAT v2.7.0	PC (i386)	All IO devices	Standard.Lib; PLCSystem.Lib; TcPLCAds.Lib; TcPlcIoFunctions.Lib
TwinCAT v2.8.0	PC (i386)	All IO devices	TcIoFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib werden automatisch eingebunden)

3.8 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 Manager während der Konfiguration festgelegte Geräte-Id. Die Geräte-Ids können vom Benutzer nicht konfiguriert werden. Intern wird in dem Funktionsbaustein eine Instanz des ADSRDWRT-Funktionsbausteins aufgerufen.

VAR_INPUT

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

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

DEVICENAME: Die Geräte-Bezeichnung eines IO-Gerätes.

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

TMOUT: Gibt die Timeout-Zeit an, die bei der Ausführung des ADS-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 ADS-Fehler bei der Übertragung des Kommandos erfolgen, dann wird dieser Ausgang gesetzt, nachdem der BUSY-Ausgang zurückgesetzt wurde.

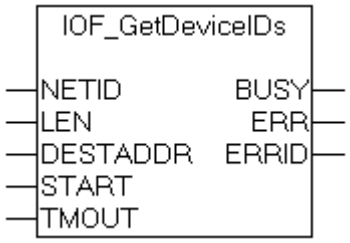
ERRID: Liefert bei einem gesetzten ERR-Ausgang die ADS-Fehlernummer.

DEVICEID: Die Geräte-Id eines IO-Gerätes.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO-Hardware	Einzubindende SPS-Bibliotheken
TwinCAT v2.7.0	PC (i386)	All IO devices	Standard.Lib; PLCSystem.Lib; TcPLCAds.Lib; TcPlcIoFunctions.Lib
TwinCAT v2.8.0	PC (i386)	All IO devices	TcIoFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib werden automatisch eingebunden)

3.9 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 Manager festgelegt und können nicht vom Benutzer konfiguriert werden. Intern wird in dem Funktionsbaustein eine Instanz des ADSREAD-Funktionsbausteins aufgerufen.

VAR_INPUT

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

NETID: Hier kann die AmsNetId des TwinCAT-Rechners angegeben werden, auf dem die Funktion ausgeführt werden soll. 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 ADS-Kommandos nicht überschritten werden darf.

VAR_OUTPUT

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

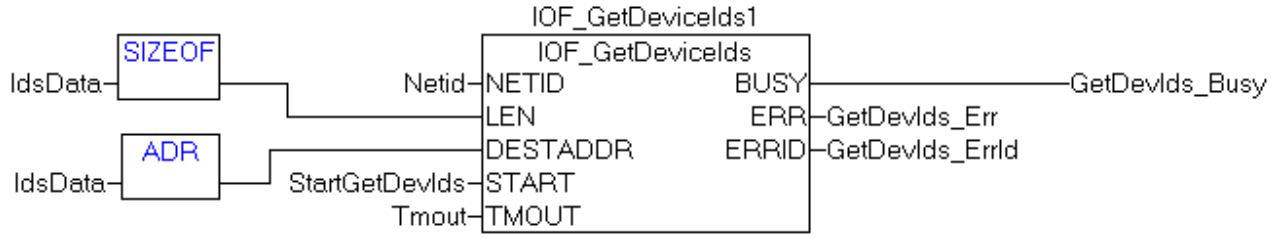
BUSY: Bei der Aktivierung des Funktionsbausteins wird dieser Ausgang gesetzt und bleibt gesetzt, bis eine Rückmeldung erfolgt.

ERR: Sollte ein ADS-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.

Beispiel für einen Aufruf in FUP:

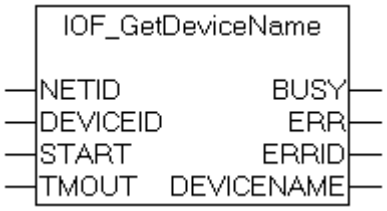
```
IOF_GetDeviceIds1   : IOF_GetDeviceIDs;
IdsData             : ARRAY[1..201] OF WORD;
StartGetDevIds      : BOOL;
GetDevIds_Busy      : BOOL;
GetDevIds_Err       : BOOL;
GetDevIds_ErrId     : UDINT;
```



Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO-Hardware	Einzubindende SPS-Bibliotheken
TwinCAT v2.7.0	PC (i386)	All IO devices	Standard.Lib; PLCSystem.Lib; TcPLCAds.Lib; TcPlcIoFunctions.Lib
TwinCAT v2.8.0	PC (i386)	All IO devices	TcIoFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib werden automatisch eingebunden)

3.10 IOF_GetDeviceName



Der Funktionsbaustein IOF_GetDeviceName liest die Gerätebezeichnung eines IO-Gerätes. Intern wird in diesem Funktionsbaustein eine Instanz des ADSREAD-Funktionsbausteins aufgerufen. Die Gerätebezeichnung kann während der Konfiguration in TwinCAT-System Manager 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;
END_VAR
```

NETID: Hier kann die AmsNetId des TwinCAT-Rechners angegeben werden, auf dem die Funktion ausgeführt werden soll. 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 Manager festgelegt.

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

TMOUT: Gibt die Timeout-Zeit an, die bei der Ausführung des ADS-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 ADS-Fehler bei der Übertragung des Kommandos erfolgen, dann wird dieser Ausgang gesetzt, nachdem der BUSY-Ausgang zurückgesetzt wurde.

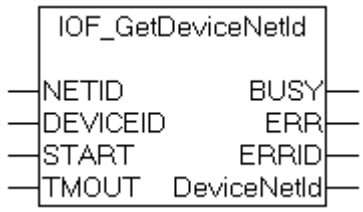
ERRID: Liefert bei einem gesetzten ERR-Ausgang die ADS-Fehlernummer.

DEVICENAME: Die Gerätebezeichnung des IO-Gerätes.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO-Hardware	Einzubindende SPS-Bibliotheken
TwinCAT v2.7.0	PC (i386)	All IO devices	Standard.Lib; PLCSystem.Lib; TcPLCAds.Lib; TcPlcIoFunctions.Lib
TwinCAT v2.8.0	PC (i386)	All IO devices	TcIoFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib werden automatisch eingebunden)

3.11 IOF_GetDeviceNetId



Einigen IO-Geräten kann eine AmsNetId während der Konfiguration im TwinCAT-System Manager zugewiesen werden (z.B. FC310x Profibuskarte oder CP9030-Karte). Die AmsNetId kann dann benutzt werden, um auf dem Gerät Firmware-Funktionen ausführen zu können. Der Funktionsbaustein IOF_GetDeviceNetId ermittelt die AmsNetId anhand der Geräte-Id (DEVICEID) . Die Geräte-Ids werden während der Konfiguration vom TwinCAT-System Manager festgelegt und können nicht vom Benutzer konfiguriert werden. Intern wird in dem Funktionsbaustein eine Instanz des ADSREAD-Funktionsbausteins aufgerufen.

VAR_INPUT

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

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

DEVICEID: Die Id des Gerätes, dessen 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 ADS-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 ADS-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.

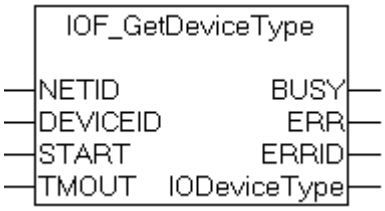
DeviceNetId: Die AmsNetId des Gerätes als String.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO-Hardware	Einzubindende SPS-Bibliotheken
TwinCAT v2.7.0	PC (i386)	All IO devices with NetID	Standard.Lib; PLCSystem.Lib; TcPLCAds.Lib; TcPlcIoFunctions.Lib
TwinCAT v2.8.0	PC (i386)	All IO devices with NetID	TcIoFunctions.Lib

Entwicklungsumgebung	Zielplattform	IO-Hardware	Einzubindende SPS-Bibliotheken
			(Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib werden automatisch eingebunden)

3.12 IOF_GetDeviceType



Der Funktionsbaustein "IOF_GetDeviceType" ermittelt den Geräte-Typ anhand der Geräte-Id. Die Geräte-Ids werden während der Konfiguration vom TwinCAT-System Manager festgelegt und können nicht vom Benutzer konfiguriert werden. Intern wird in dem Funktionsbaustein eine Instanz des ADSREAD-Funktionsbausteins aufgerufen.

VAR_INPUT

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

NETID: Hier kann die AmsNetId des TwinCAT-Rechners angegeben werden, auf dem die Funktion ausgeführt werden soll. 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 ADS-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.

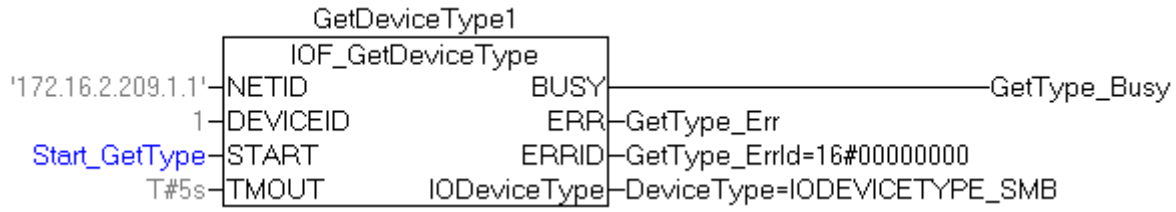
ERR: Sollte ein ADS-Fehler bei der Übertragung des Kommandos erfolgen, dann wird dieser Ausgang gesetzt, nachdem der BUSY-Ausgang zurückgesetzt wurde.

ERRID: Liefert bei einem gesetzten ERR-Ausgang die ADS-Fehlernummer.

IODeviceType: Die Geräte-Typ-Konstante [▶ 123].

Beispiel für einen Aufruf in FUP:

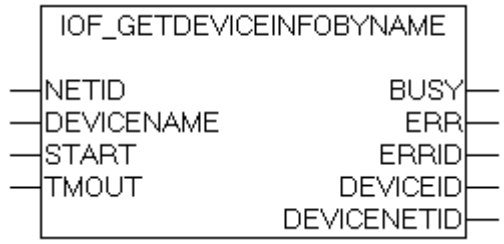
```
PROGRAM MAIN
VAR
    GetDeviceType1  : IOF_GetDeviceType;
    Start_GetType   : BOOL;
    GetType_Busy    : BOOL;
    GetType_Err     : BOOL;
    GetType_ErrId   : UDINT;
    DeviceType      : IODEVICETYPES;
END_VAR
```



Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO-Hardware	Einzubindende SPS-Bibliotheken
TwinCAT v2.7.0	PC (i386)	All IO devices	Standard.Lib; PLCSystem.Lib; TcPLCAds.Lib; TcPlcIoFunctions.Lib
TwinCAT v2.8.0	PC (i386)	All IO devices	TcIoFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib werden automatisch eingebunden)

3.13 IOF_GetDeviceInfoByName



Der Funktionsbaustein IOF_GetDeviceInfoByName ermittelt die Geräte-Id eines IO-Gerätes und dessen AmsNetId (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;
END_VAR
```

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

DEVICENAME: Die Geräte-Bezeichnung eines IO-Gerätes.

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

TMOUT: Gibt die Timeout-Zeit an, die bei der Ausführung des ADS-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 ADS-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.

DEVICEID: Die Geräte-Id eines IO-Gerätes.

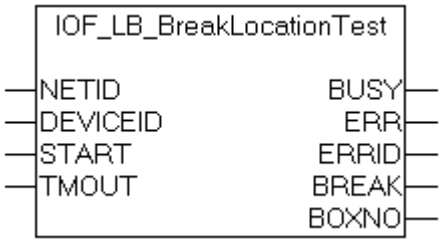
DEVICENETID: Die Netzwerkadresse eines IO-Gerätes.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO-Hardware	Einzubindende SPS-Bibliotheken
TwinCAT v2.8.0 Build > 737	PC (i386)	All IO devices	TcIoFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib werden automatisch eingebunden)

3.14 Beckhoff Lightbus

3.14.1 IOF_LB_BreakLocationTest



Der Funktionsbaustein IOF_LB_BreakLocationTest führt ein Bruchstellentest in einem Beckhoff Lightbus-Lichtwellenleiterring durch und kann eventuelle Bruchstellen lokalisieren. Intern wird eine Instanz des ADSREAD-Funktionsbausteines aufgerufen. 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;
END_VAR
```

NETID: Hier kann die AmsNetId des TwinCAT-Rechners angegeben werden, auf dem die Funktion ausgeführt werden soll. 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 Manager festgelegt.

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

TMOUT: Gibt die Timeout-Zeit an, die bei der Ausführung des ADS-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 ADS-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.

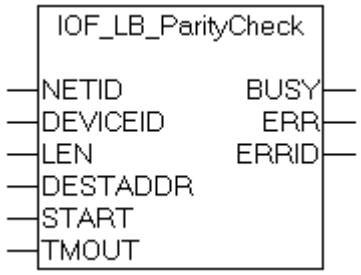
BREAK: Dieses Flag wird gesetzt, wenn in dem Lichwellenleiter-Ring eine Bruchstelle erkannt wurde.

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

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO-Hardware	Einzubindende SPS-Bibliotheken
TwinCAT v2.7.0	PC (i386)	Beckhoff Lightbus Master: C1220 ISA; FC200x PCI	Standard.Lib; PLCSystem.Lib; TcPLCAds.Lib; TcPlcIoFunctions.Lib
TwinCAT v2.8.0	PC (i386)	Beckhoff Lightbus Master: C1220 ISA; FC200x PCI	TcIoFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib werden automatisch eingebunden)

3.14.2 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 [▶ 31] Funktionsbaustein werden die Zählerstände nicht zurückgesetzt. Intern wird eine Instanz des ADSREAD-Funktionsbausteines aufgerufen. 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    : DWORD;
    START       : BOOL;
    TMOUT       : TIME;
END_VAR
```

NETID: Hier kann die AmsNetId des TwinCAT-Rechners angegeben werden, auf dem die Funktion ausgeführt werden soll. 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 Manager 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 ADS-Kommandos nicht überschritten werden darf.

VAR_OUTPUT

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

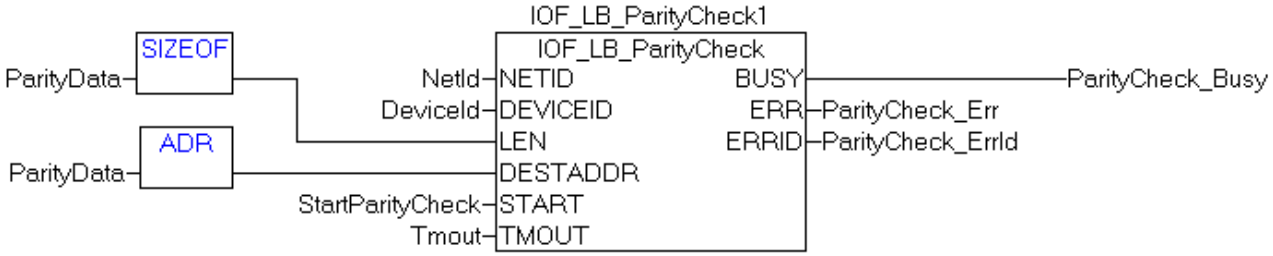
BUSY: Bei der Aktivierung des Funktionsbausteins wird dieser Ausgang gesetzt und bleibt gesetzt, bis eine Rückmeldung erfolgt.

ERR: Sollte ein ADS-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.

Beispiel für einen Aufruf in FUP:

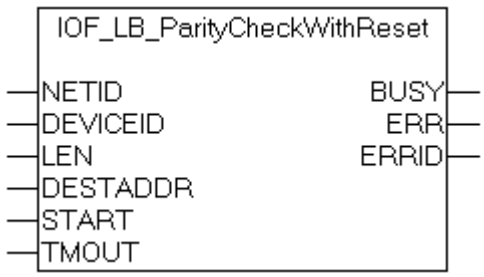
```
IOF_LB_ParityCheck1    : IOF_LB_ParityCheck;
ParityData             : ARRAY[1..256] OF BYTE;
StartParityCheck       : BOOL;
ParityCheck_Busy       : BOOL;
ParityCheck_Err        : BOOL;
ParityCheck_ErrId      : UDINT;
```



Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO-Hardware	Einzubindende SPS-Bibliotheken
TwinCAT v2.7.0	PC (i386)	Beckhoff Lightbus Master: C1220 ISA; FC200x PCI	Standard.Lib; PLCSystem.Lib; TcPLCAds.Lib; TcPlcIoFunctions.Lib
TwinCAT v2.8.0	PC (i386)	Beckhoff Lightbus Master: C1220 ISA; FC200x PCI	TcIoFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib werden automatisch eingebunden)

3.14.3 IOF_LB_ParityCheckWithReset



Der Funktionsbaustein IOF_LB_ParityCheckWithReset liest die Parityfehler-Zähler der Beckhoff Lightbus-Module (z.B. BK2000). Anschließend werden die Zähler zurückgesetzt. Intern wird eine Instanz des ADSREAD-Funktionsbausteines aufgerufen. 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    : DWORD;
    START       : BOOL;
    TMOUT       : TIME;
END_VAR
```

NETID: Hier kann die AmsNetId des TwinCAT-Rechners angegeben werden, auf dem die Funktion ausgeführt werden soll. 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 Manager 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 ADS-Kommandos nicht überschritten werden darf.

VAR_OUTPUT

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

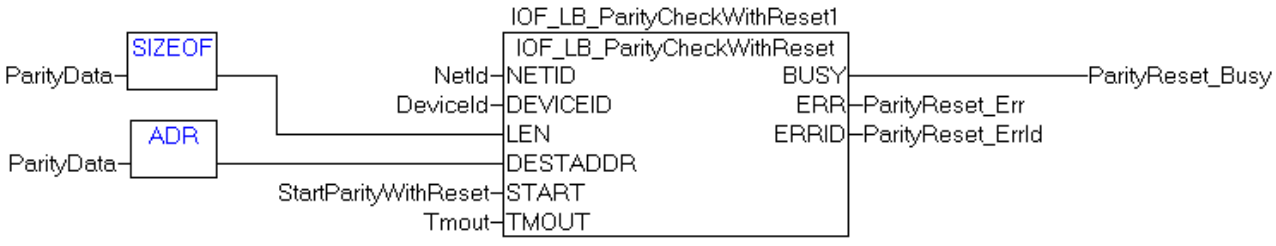
BUSY: Bei der Aktivierung des Funktionsbausteins wird dieser Ausgang gesetzt und bleibt gesetzt, bis eine Rückmeldung erfolgt.

ERR: Sollte ein ADS-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.

Beispiel für einen Aufruf in FUP:

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



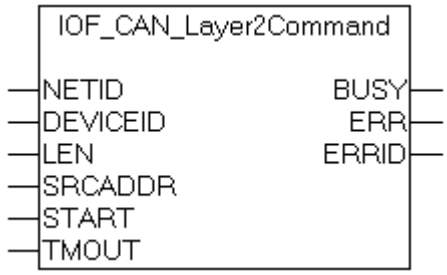
Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO-Hardware	Einzubindende SPS-Bibliotheken
TwinCAT v2.7.0	PC (i386)	Beckhoff Lightbus Master: C1220 ISA; FC200x PCI	Standard.Lib; PLCSystem.Lib; TcPLCAds.Lib; TcPlcIoFunctions.Lib
TwinCAT v2.8.0	PC (i386)	Beckhoff Lightbus Master: C1220 ISA; FC200x PCI	TcIoFunctions.Lib

Entwicklungsumgebung	Zielplattform	IO-Hardware	Einzubindende SPS-Bibliotheken
			(Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib werden automatisch eingebunden)

3.15 CANopen

3.15.1 IOF_CAN_Layer2Command



Der Funktionsbaustein IOF_CAN_Layer2Command sendet ein 10 Byte langes Kommando an die Schicht 2 eines CAN-Masters. Intern wird eine Instanz des ADSWRITE-Funktionsbausteins aufgerufen.

VAR_INPUT

```
VAR_INPUT
    NETID       : T_AmsNetId;
    DEVICEID    : UDINT;
    LEN         : UDINT;
    SRCADDR     : DWORD;
    START       : BOOL;
    TMOUT       : TIME;
END_VAR
```

NETID: Hier kann die AmsNetId des TwinCAT-Rechners angegeben werden, auf dem die Funktion ausgeführt werden soll. 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 Manager 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 ADS-Kommandos nicht überschritten werden darf.

VAR_OUTPUT

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

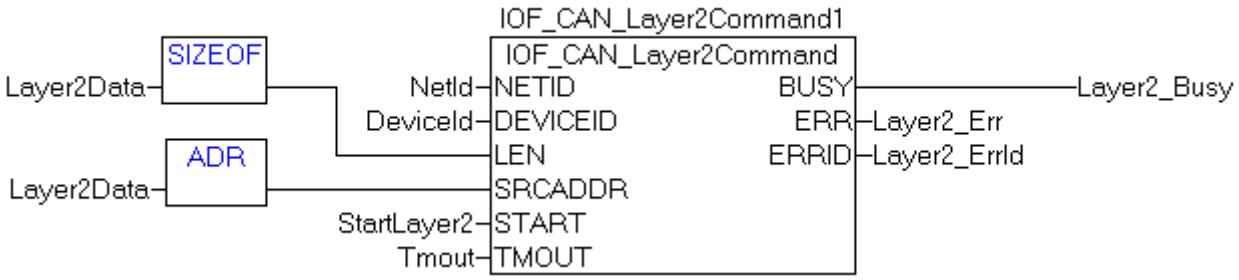
BUSY: Bei der Aktivierung des Funktionsbausteins wird dieser Ausgang gesetzt und bleibt gesetzt, bis eine Rückmeldung erfolgt.

ERR: Sollte ein ADS-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.

Beispiel für einen Aufruf in FUP:

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

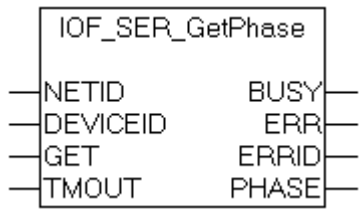


Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO-Hardware	Einzubindende SPS-Bibliotheken
TwinCAT v2.7.0	PC (i386)	HILSCHER Master-Karte CIF30 COM	Standard.Lib; PLCSystem.Lib; TcPLCAds.Lib; TcPlcIoFunctions.Lib
TwinCAT v2.8.0	PC (i386)	HILSCHER Master-Karte CIF30 COM	TcIoFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib werden automatisch eingebunden)

3.16 SERCOS

3.16.1 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. Intern wird eine Instanz des ADSREAD-Funktionsbausteins aufgerufen.

VAR_INPUT

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

NETID: Hier kann die AmsNetId des TwinCAT-Rechners angegeben werden, auf dem das ADS-Kommando durchgeführt werden soll. 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 Manager festgelegt.

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

TMOUT: Gibt die Timeout-Zeit an, die bei der Ausführung des ADS-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 ADS-Fehler bei der Übertragung des Kommandos erfolgen, dann wird dieser Ausgang gesetzt, nachdem der BUSY-Ausgang zurückgesetzt wurde.

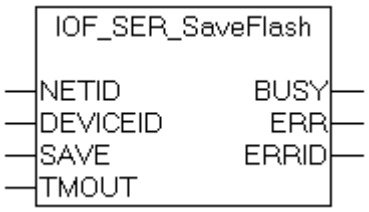
ERRID: Liefert bei einem gesetzten ERR-Ausgang die ADS-Fehlernummer.

PHASE: Die aktuelle Kommunikationsphase im SERCOS-Ring.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO-Hardware	Einzubindende SPS-Bibliotheken
TwinCAT v2.7.0	PC (i386)	Sercans SCS-P ISA; Sercans SCS-P PCI	Standard.Lib; PLCSystem.Lib; TcPLCAds.Lib; TcPlcIoFunctions.Lib
TwinCAT v2.8.0	PC (i386)	Sercans SCS-P ISA; Sercans SCS-P PCI	TcIoFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib werden automatisch eingebunden)

3.16.2 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.

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;
END_VAR
```

NETID: Hier kann die AmsNetId des TwinCAT-Rechners angegeben werden, auf dem das ADS-Kommando durchgeführt werden soll. 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 von TwinCAT-System Manager festgelegt.

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

TMOUT: Gibt die Timeout-Zeit an, die bei der Ausführung des ADS-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 ADS-Fehler bei der Übertragung des Kommandos erfolgen, dann wird dieser Ausgang gesetzt, nachdem der BUSY-Ausgang zurückgesetzt wurde.

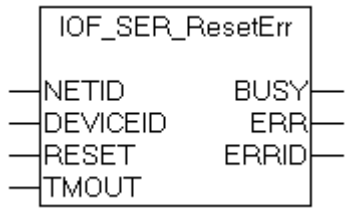
ERRID: Liefert bei einem gesetzten ERR-Ausgang die ADS-Fehlernummer.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO-Hardware	Einzubindende SPS-Bibliotheken
TwinCAT v2.7.0	PC (i386)	Sercans SCS-P ISA; Sercans SCS-P PCI	Standard.Lib; PLCSystem.Lib; TcPLCAds.Lib; TcPlcIoFunctions.Lib

Entwicklungsumgebung	Zielplattform	IO-Hardware	Einzubindende SPS-Bibliotheken
TwinCAT v2.8.0	PC (i386)	Sercans SCS-P ISA; Sercans SCS-P PCI	TcIoFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib werden automatisch eingebunden)

3.16.3 IOF_SER_ResetErr



Der Funktionsbaustein "IOF_SER_ResetErr" löscht folgende Fehler eines SERCOS-Masters:

·die Fehler in den vorhandenen Antrieben,

·der Diagnosestatus im Diagnosekanal der vorhandenen Antriebe und

·der Systemfehler.

Intern wird eine Instanz des ADSWRITE-Funktionsbausteins aufgerufen.

VAR_INPUT

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

NETID: Hier kann die AmsNetId des TwinCAT-Rechners angegeben werden, auf dem das ADS-Kommando durchgeführt werden soll. 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 Manager festgelegt.

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

TMOUT: Gibt die Timeout-Zeit an, die bei der Ausführung des ADS-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.

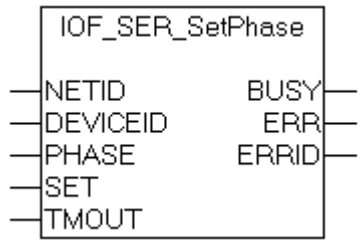
ERR: Sollte ein ADS-Fehler bei der Übertragung des Kommandos erfolgen, dann wird dieser Ausgang gesetzt, nachdem der BUSY-Ausgang zurückgesetzt wurde.

ERRID: Liefert bei einem gesetzten ERR-Ausgang die ADS-Fehlernummer.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO-Hardware	Einzubindende SPS-Bibliotheken
TwinCAT v2.7.0	PC (i386)	Sercans SCS-P ISA; Sercans SCS-P PCI	Standard.Lib; PLCSystem.Lib; TcPLCAds.Lib; TcPlcIoFunctions.Lib
TwinCAT v2.8.0	PC (i386)	Sercans SCS-P ISA; Sercans SCS-P PCI	TcIoFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib werden automatisch eingebunden)

3.16.4 IOF_SER_SetPhase



Der Funktionsbaustein "IOF_SER_SetPhase" führt den Phasenhochlauf im SERCOS-Ring auf einen bestimmten Wert durch. Intern wird eine Instanz des ADSWRITE-Funktionsbausteins aufgerufen.

VAR_INPUT

```
VAR_INPUT
    NETID       : T_AmsNetId;
    DEVICEID    : UDINT;
    PHASE       : BYTE;
    SET         : BOOL;
    TMOUT       : TIME;
END_VAR
```

NETID: Hier kann die AmsNetId des TwinCAT-Rechners angegeben werden, auf dem das ADS-Kommando durchgeführt werden soll. 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 von TwinCAT-System Manager 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 ADS-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.

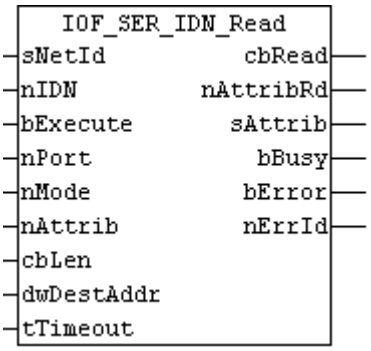
ERR: Sollte ein ADS-Fehler bei der Übertragung des Kommandos erfolgen, dann wird dieser Ausgang gesetzt, nachdem der BUSY-Ausgang zurückgesetzt wurde.

ERRID: Liefert bei einem gesetzten ERR-Ausgang die ADS-Fehlernummer.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO-Hardware	Einzubindende SPS-Bibliotheken
TwinCAT v2.7.0	PC (i386)	Sercans SCS-P ISA; Sercans SCS-P PCI	Standard.Lib; PLCSystem.Lib; TcPLCAds.Lib; TcPlcIoFunctions.Lib
TwinCAT v2.8.0	PC (i386)	Sercans SCS-P ISA; Sercans SCS-P PCI	TcIoFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib werden automatisch eingebunden)

3.16.5 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.

Intern wird eine Instanz des Funktionsbausteins ADSREAD benutzt.

VAR_INPUT

```
VAR_INPUT
    sNetId      : T_AmsNetId;
    nIDN        : UINT;
    bExecute    : BOOL;
    nPort       : UINT;
    nMode       : DINT;
    nAttrib     : DWORD;
    cbLen       : UDINT;
    dwDestAddr  : DWORD;
    tTimeout    : TIME;
END_VAR
```

sNetId: Hier kann die AmsNetId des TwinCAT-Rechners angegeben werden, auf dem das ADS-Kommando durchgeführt werden soll. 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 im TwinCAT System Manager 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 ADS-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 [▶ 126] in einzelne Variablen zerlegt.

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

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

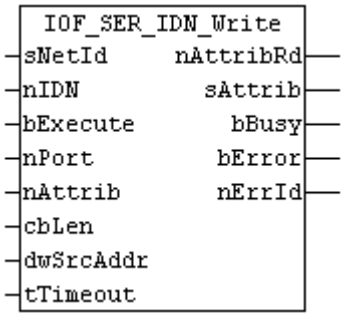
nErrId: Liefert bei einem gesetzten ERR-Ausgang die ADS-Fehlernummer bzw. die spezifische Funktionsbaustein-Fehlernummer.

spezifische Functionsbaustein- Fehlernummer	Beschreibung
0x1003	Falscher Parameter-Mode
0x1004	Falsche Parameterdatengröße

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO-Hardware	Einzubindende SPS-Bibliotheken
TwinCAT v2.8.0 Build > 735	PC (i386)	Sercans SCS-P ISA; Sercans SCS-P PCI; Beckhoff FC750x PCI	TcIoFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib werden automatisch eingebunden)

3.16.6 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.

Intern werden je eine Instanz des Funktionsbausteins ADSREAD und des Funktionsbausteins ADSWRITE benutzt.

VAR_INPUT

```
VAR_INPUT
    sNetId      : T_AmsNetId;
    nIDN        : UINT; (* S: 0******* ********, P: 1******* ******** *)
    bExecute    : BOOL;
    nPort       : UINT;
    nAttrib     : DWORD;
    cbLen       : UDINT;
    dwSrcAddr   : DWORD;
    tTimeout    : TIME;
END_VAR
```

sNetId: Hier kann die AmsNetId des TwinCAT-Rechners angegeben werden, auf dem das ADS-Kommando durchgeführt werden soll. 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 im TwinCAT System Manager 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 ADS-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 [▶ 126] in einzelne Variablen zerlegt.

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

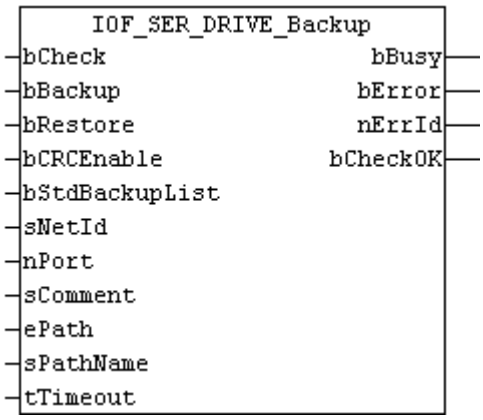
bError: Sollte ein ADS-Fehler bei der Übertragung des Kommandos erfolgen, dann wird dieser Ausgang gesetzt, nachdem der bBusy-Ausgang zurückgesetzt wurde.

nErrId: Liefert bei einem gesetzten ERR-Ausgang die ADS-Fehlernummer.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO-Hardware	Einzubindende SPS-Bibliotheken
TwinCAT v2.8.0 Build > 735	PC (i386)	Sercans SCS-P ISA; Sercans SCS-P PCI; Beckhoff FC750x PCI	TcIoFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib werden automatisch eingebunden)

3.16.7 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.

Intern werden Instanzen der Funktionsbausteine IOF_SER_IDN_Read, IOF_SER_IDN_Write, FB_FileOpen, FB_FileClose, FB_FileRead und FB_FileWrite benutzt.

Das Dateiformat der Backup-Datei ist in Backup-Dateiformat [▶ 126] 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;
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 WAHR gesetzt, anderenfalls wird bCheckOK auf FALSCH 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 = WAHR aktiviert. Die CRC und die Checksumme werden in Parameter IDN142 gespeichert, wenn bCRCEnable = WAHR.

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 AmsNetId des TwinCAT-Rechners angegeben werden, auf dem das ADS-Kommando durchgeführt werden soll. 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 im TwinCAT System Manager während der Hardware-Konfiguration vergeben.

sComment: ist ein Kommentar, der in den Datei-Header der Backup-Datei geschrieben wird.

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.

sPathName: beinhaltet den Dateinamen (bei Verwendung des Boot-Pfades) bzw. den kompletten Pfad und Dateinamen bei Verwendung des generischen Pfades.

tTimeout: Gibt die Timeout-Zeit an, die bei der Ausführung des ADS-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 ADS-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 bzw. die spezifische Funktionsbaustein-Fehlernummer.

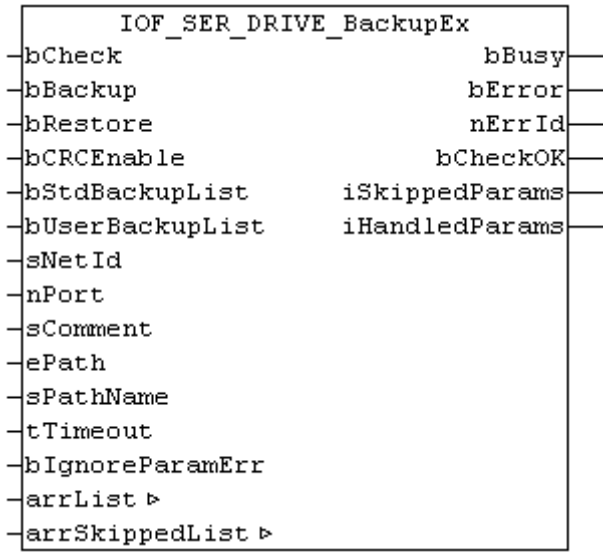
bCheckOk: Ist WAHR, wenn der Checksummen-Test erfolgreich war.

spezifische Functionsbaustein- Fehlernummer	Beschreibung
0x1003	Falscher Parameter-Mode
0x1004	Falsche Parameterdatengröße
0x1005	Falscher Backup Parameter Typ
0x1006	Backup Parameterliste war nicht IDN 192

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO-Hardware	Einzubindende SPS-Bibliotheken
TwinCAT v2.8.0 Build > 735	PC (i386)	Sercans SCS-P ISA; Sercans SCS-P PCI; Beckhoff FC750x PCI	TcIoFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib werden automatisch eingebunden)

3.16.8 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).

Intern werden Instanzen der Funktionsbausteine IOF_SER_IDN_Read, IOF_SER_IDN_Write, FB_FileOpen, FB_FileClose, FB_FileRead und FB_FileWrite benutzt.

Das Dateiformat der Backup-Datei ist in Backup-Dateiformat [▶ 126] beschrieben.

Demoprojekt für Backup/Restore mit IOF_SER_DRIVE_BackupEx in https://infosys.beckhoff.com/content/1031/tcplclibiofunctions/Resources/11843324939.zip.

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;
    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 WAHR gesetzt, anderenfalls wird bCheckOK auf FALSCH 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 (WAHR).

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 (FALSCH) 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 AmsNetId des TwinCAT-Rechners angegeben werden, auf dem das ADS-Kommando durchgeführt werden soll. 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 im TwinCAT System Manager während der Hardware-Konfiguration vergeben.

sComment: Ist ein Kommentar, der in den Datei-Header der Backup-Datei geschrieben wird.

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.

sPathName: Beinhaltet den Dateinamen (bei Verwendung des Boot-Pfades) bzw. den kompletten Pfad und Dateinamen bei Verwendung des generischen Pfades.

tTimeout: Gibt die Timeout-Zeit an, die bei der Ausführung des ADS-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.

arrSkippedList: Enthält eine Liste der übersprungenen Parameter (die Parameternummer und die Fehlernummern).

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 ADS-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 bzw. die spezifische Funktionsbaustein-Fehlernummer.

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.

Strukturdefinitionen

```
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.

```
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.

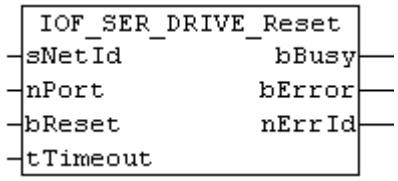
iError: Ist ein Feld von bis zu 2048 Zugriffsfehlernummern.

spezifische Functionsbaustein- Fehler-nummer	Beschreibung
0x1003	Falscher Parameter-Mode
0x1004	Falsche Parameterdatengröße
0x1005	Falscher Backup Parameter Typ
0x1006	Backup Parameterliste war nicht IDN 192 oder benutzerdefiniert

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO-Hardware	Einzubindende SPS-Bibliotheken
TwinCAT v2.8.0 Build > 735	PC (i386)	Sercans SCS-P ISA; Sercans SCS-P PCI; Beckhoff FC750x PCI	TcIoFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib werden automatisch eingebunden)

3.16.9 IOF_SER_DRIVE_Reset



Der Funktionsbaustein "IOF_SER_DRIVE_Reset" führt einen Antriebs-Reset eines Sercos-Antriebes durch. Antriebsfehler werden gelöscht.

Intern wird eine Instanz des Funktionsbausteins ADSWRITE benutzt.

VAR_INPUT

```
VAR_INPUT
    sNetId      : T_AmsNetId;
    nPort       : UINT;
    bReset      : BOOL;
    tTimeout    : TIME;
END_VAR
```

sNetId: Hier kann die AmsNetId des TwinCAT-Rechners angegeben werden, auf dem das ADS-Kommando durchgeführt werden soll. Für den lokalen Rechner kann auch ein Leerstring angegeben werden.

nPort: Die Port-Number nPort wird im TwinCAT System Manager 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 ADS-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 ADS-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.

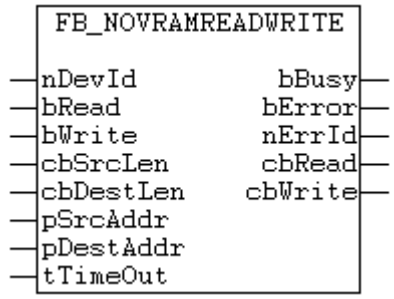
Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO-Hardware	Einzubindende SPS-Bibliotheken
TwinCAT v2.8.0 Build > 743	PC (i386)	Sercans SCS-P ISA; Sercans SCS-P PCI; Beckhoff FC750x PCI	TcIoFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib werden automatisch eingebunden)

3.17 NOV/DP-RAM

3.17.1 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. *Siehe hierzu auch:* Konfigurationsbeispiel (TwinCAT 2.9).



Der Funktionsbaustein *FB_NovRamReadWrite* greift aus einem SPS-Programm auf das NOV-RAM [▶ 48] 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    : UDINT;
    pDestAddr   : UDINT;
    tTimeOut    : TIME;
END_VAR
```

nDevId: Die Geräte-Id [▶ 48] 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 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 *("ADS - Return Codes").*

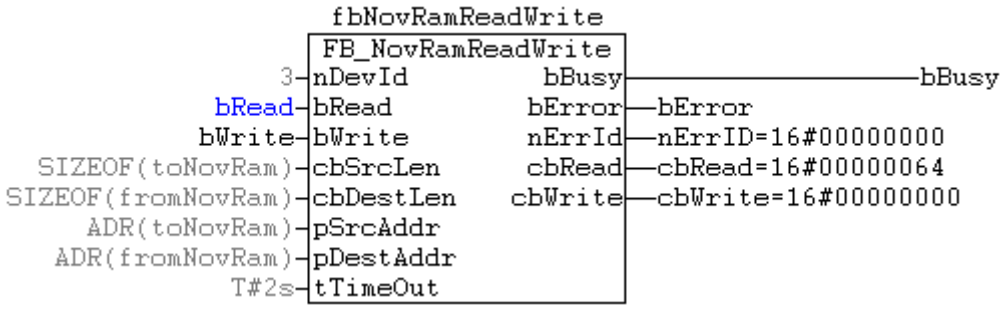
cbRead: Anzahl der erfolgreich gelesenen Datenbytes.

cbWrite: Anzahl der erfolgreich geschriebenen Datenbytes.

Beispiel für einen Aufruf im FUB:

```
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
```

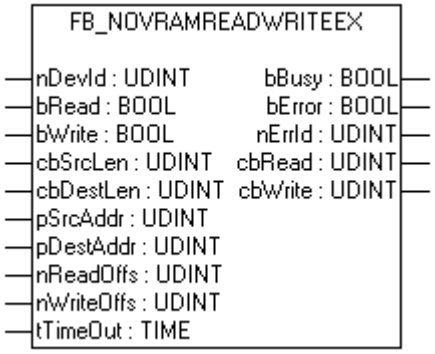


Im Beispiel wurden bei einer steigenden Flanke am *bRead*-Eingang 100 Byte Daten aus dem NOV-RAM ausgelesen und in das Array *fromNovRam* hineinkopiert.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO-Hardware	Einzubindende SPS-Bibliotheken
TwinCAT v2.8.0 Build > 722	PC (x86), (nicht für CX (ARM)!)	FCxxxx cards with NOV-RAM	TcIoFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib werden automatisch eingebunden)

3.17.2 FB_NovRamReadWriteEx



Der Funktionsbaustein *FB_NovRamReadWriteEx* greift aus einem SPS-Programm auf das NOV-RAM [▶ 48] 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        : UDINT;
    pDestAddr       : UDINT;
    nReadOffs       : UDINT;
    nWriteOffs      : UDINT;
    tTimeOut        : TIME;
END_VAR
```

nDevId: Die Geräte-Id [▶ 48] 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..

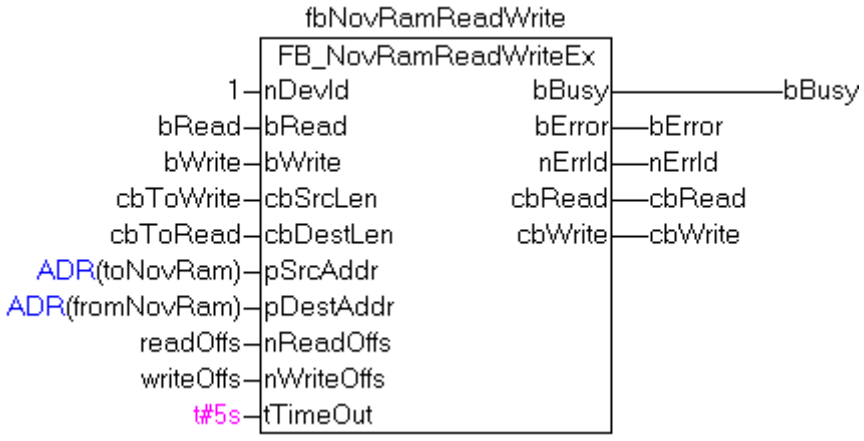
nErrId: Liefert bei einem gesetzten *bError*-Ausgang die ADS-Fehlernummer *("ADS - Return Codes").*

cbRead: Anzahl der erfolgreich gelesenen Datenbytes.

cbWrite: Anzahl der erfolgreich geschriebenen Datenbytes.

Beispiel für einen Aufruf im FUB:

```
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
TwinCAT v2.10.0 Build > 1231	PC (i386)	FCxxxx cards with NOV-RAM	TcIoFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib werden automatisch eingebunden)

3.17.3 FB_GetDPRAMInfo

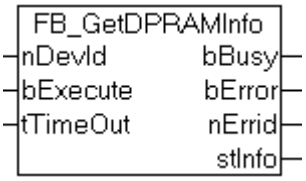


Abb. 1: FB_GetDPRAMInfo

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

(Dritthersteller) schreibend oder lesend zugreifen zu können. Die Karte muss vorher in TwinCAT System Manager als allgemeines NOV/DP-RAM [▶ 48] 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;
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 Manager 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 *("ADS - Return Codes")*.

stInfo: Eine Struktur [▶ 131] mit der Adresse und Größe des NOV/DP-RAM.

Beispiel für einen Aufruf in ST:

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
TwinCAT v2.8.0 Build > 740	PC (i386)	FCxxxx cards with NOV-RAM and all other cards with DPRAM	TcIoFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib werden automatisch eingebunden)

3.18 AX200x Profibus

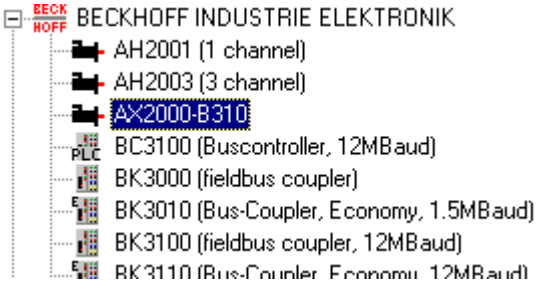
Funktionsbausteine für den Zugriff auf den AX20XX über den Profibus. Voraussetzung für den Betrieb am Profibus ist die Verwendung einer FC310x mit einer Firmwareversion größer 1.20.

Funktionsbausteine

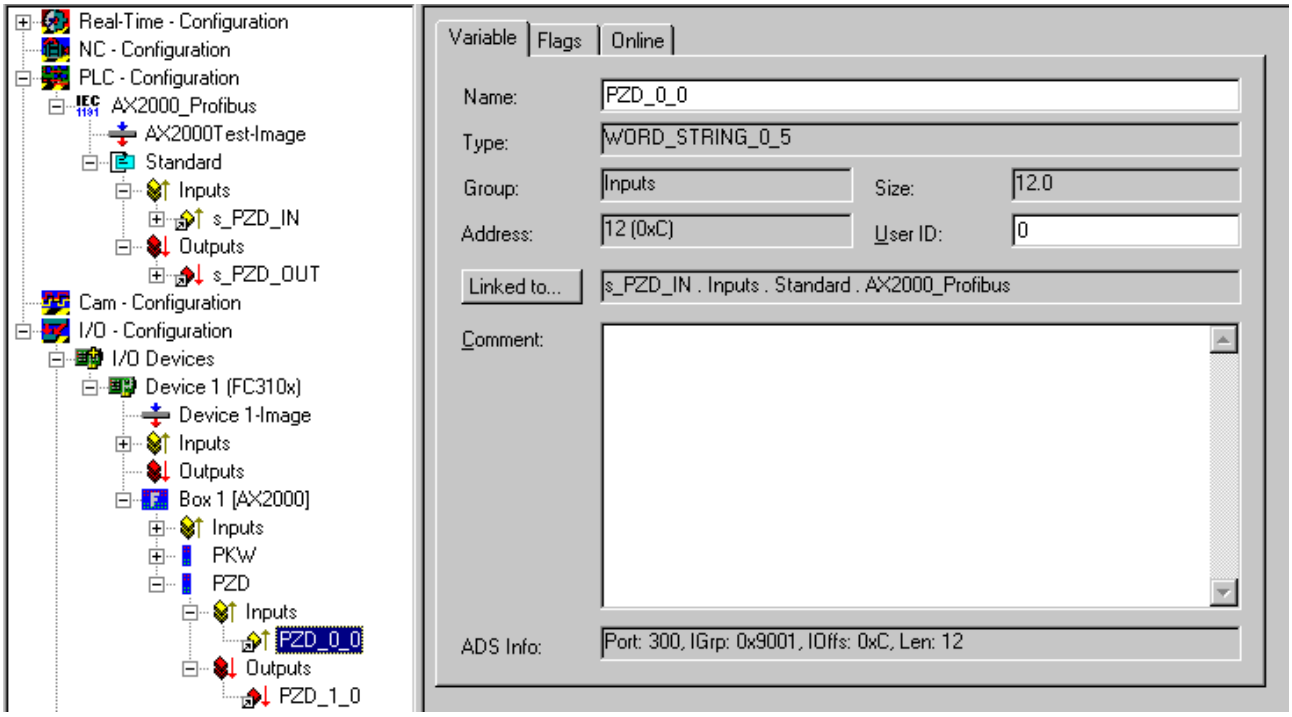
Name	Beschreibung
FB_AX2000_Parameter [▶ 59]	Schreiben/Lesen der Daten zur Parametrierung des Antriebs. **Es muss beachtet werden, dass während des Schreibens eines Parameters zum Wechsel der Betriebsart der Eingang "STOP" des Bausteins AX2000AXACT auf TRUE gehalten werden muss.**
FB_AX2000_AXACT [▶ 56]	Starten von Achsaktionen (muss immer zyklisch aufgerufen werden)
FB_AX2000_Jogmode [▶ 57]	Tippbetrieb
FB_AX2000_Reference [▶ 60]	Setzen des Referenzpunktes bzw. starten einer Referenzfahrt
FB_AX200X_Profibus [▶ 61]	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.

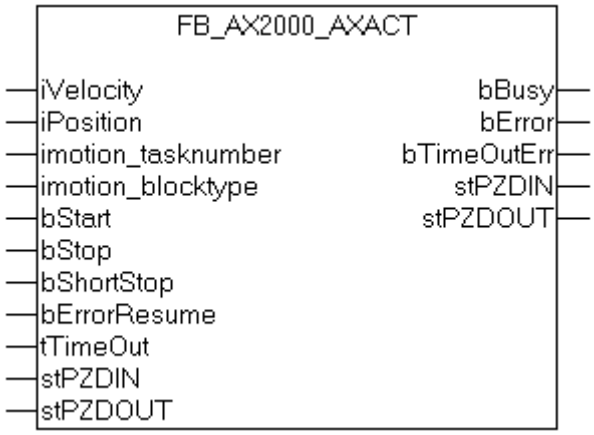


Hinweise zur Benutzung der Funktionsbausteine

- 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.

3.18.1 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 [▶ 128].

stPZDOUT : Datenwörter von der PLC zum Antrieb [▶ 129].

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:=t#5s;
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 Timeout-Zeit.

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.

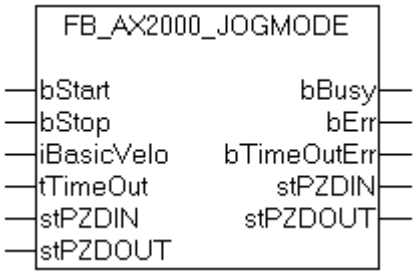
bError : Dieser Ausgang zeigt den Fehlerstatus an.

bTimeOutErr : TimeOut.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO-Hardware	Einzubindende SPS-Bibliotheken
TwinCAT v2.8.0 Build > 737	PC (i386)	AX2000 Profibus box	TcIoFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib werden automatisch eingebunden)

3.18.2 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 [▶ 128].

stPZDOUT : Datenwörter von der PLC zum Antrieb [▶ 129].

VAR_INPUT

```
VAR_INPUT
    bStart      : BOOL;
    bStop       : BOOL;
    iBasicVelo  : INT;      (*BasicVelocity*)
    tTimeOut    : TIME:=t#5s;
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 : TimeOut

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.

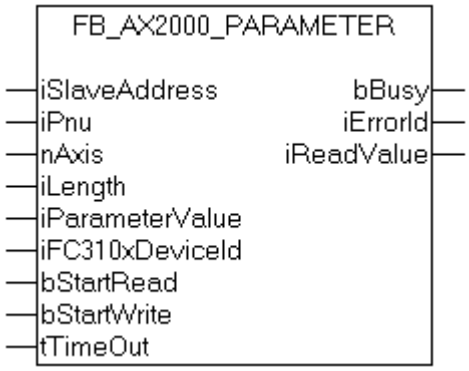
bErr : Dieser Ausgang zeigt den Fehlerstatus an.

bTimeOutErr : TimeOut.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO-Hardware	Einzubindende SPS-Bibliotheken
TwinCAT v2.8.0 Build > 737	PC (i386)	AX2000 Profibus box	TcIoFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib werden automatisch eingebunden)

3.18.3 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:=t#5s;
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 istder Schreibbefehl nur bei Stop=TRUE an dem Baustein** FB_AX2000_AXACT [▶ 56] **wirksam.**

tTimeOut : Maximale TimeOut-Zeit die 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.

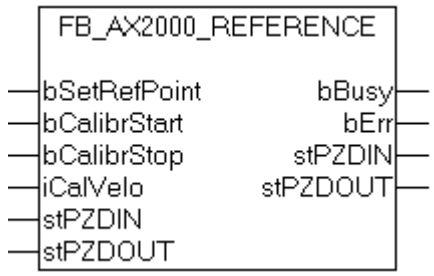
iErrorId : Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Wird durch das Ausführen eines Befehls an den Eingängen auf 0 zurückgesetzt.

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

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO-Hardware	Einzubindende SPS-Bibliotheken
TwinCAT v2.8.0 Build > 737	PC (i386)	AX2000 Profibus box	TcIoFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib werden automatisch eingebunden)

3.18.4 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 [▶ 128].

stPZDOUT : Datenwörter von der PLC zum Antrieb [▶ 129].

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
```

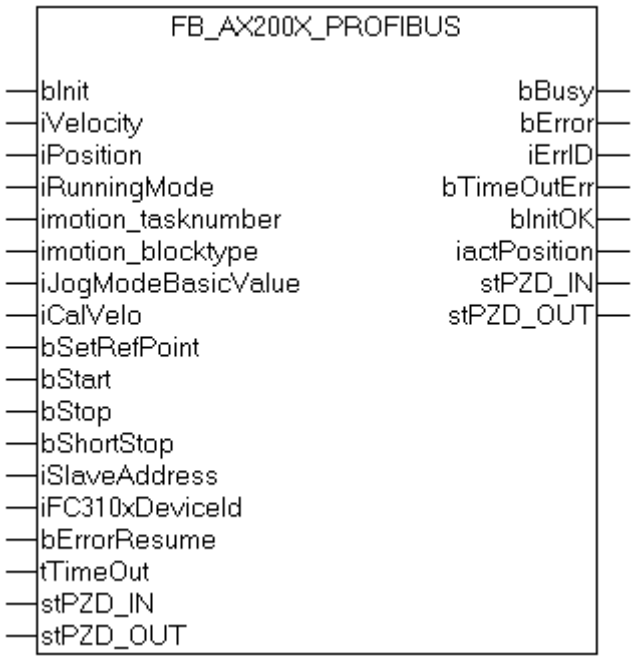
bBusy : Dieser Ausgang bleibt solange auf TRUE, bis der Baustein eine Befehlsanforderung ausführt. Während Busy = TRUE wird an den Eingängen kein neuer Befehl angenommen. Bitte beachten Sie, dass nicht die Ausführung des Dienstes, sondern nur dessen Annahme zeitlich überwacht wird.

bErr : Dieser Ausgang zeigt den Fehlerstatus an.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO-Hardware	Einzubindende SPS-Bibliotheken
TwinCAT v2.8.0 Build > 737	PC (i386)	AX2000 Profibus box	TcIoFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib werden automatisch eingebunden)

3.18.5 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 [▶ 128].

stPZD_OUT : Datenwörter von der PLC zum Antrieb [▶ 129].

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:=t#5s;
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 : TimeOut-Zeit.

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-Error.

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
TwinCAT v2.8.0 Build > 737	PC (i386)	AX2000 Profibus box	TcIoFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib werden automatisch eingebunden)

3.19 ASI-Masterklemme

Funktionsbausteine für den Zugriff auf die ASI-Masterklemme.

Funktionsbausteine

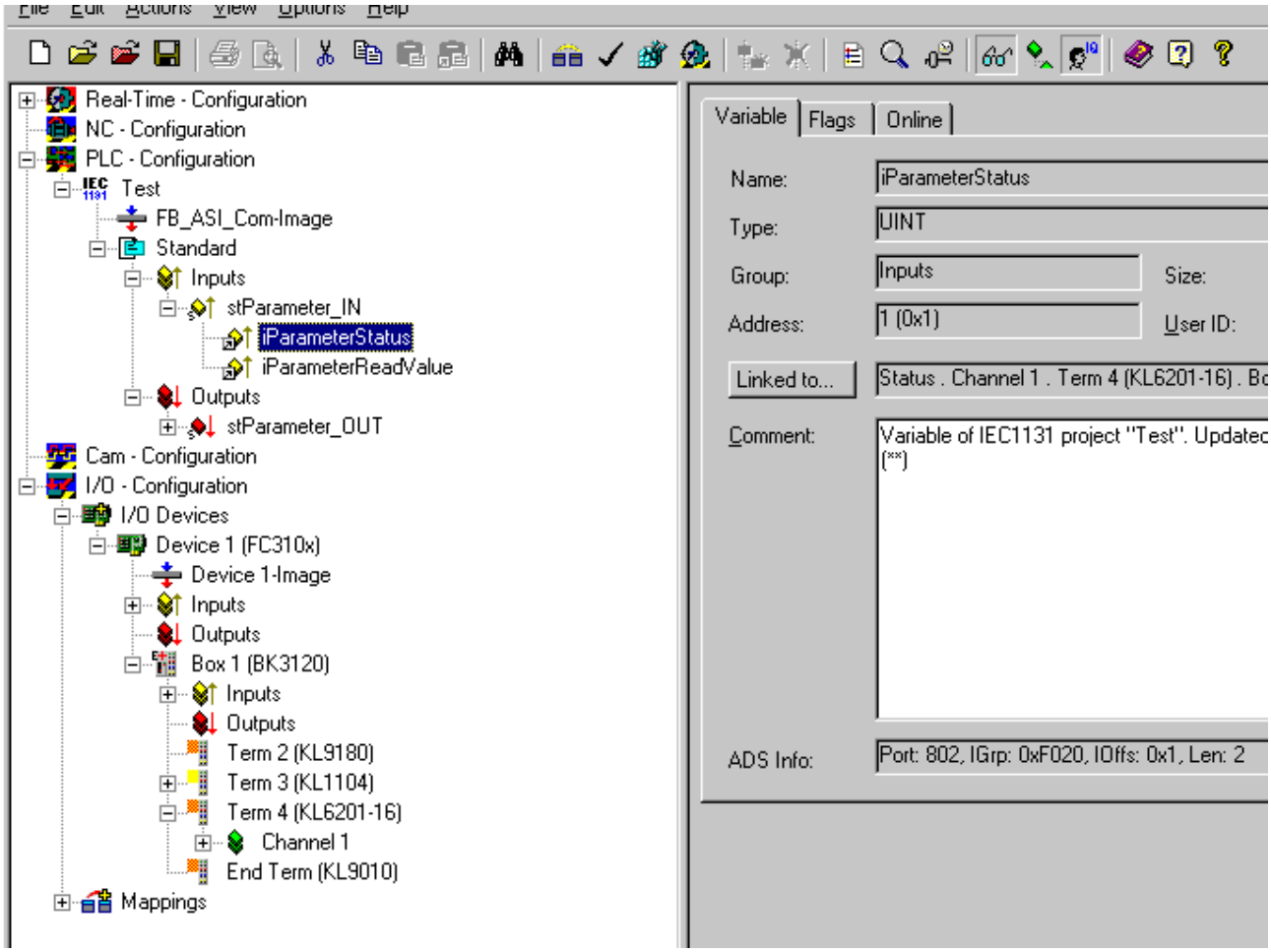
Name	Beschreibung
FB_ASI_Addressing [▶ 64]	Festlegen oder Verändern von Adressen der ASI-Slaves
FB_ASI_SlaveDiag [▶ 65]	Zyklische Slave-Diagnose (z.B. Zählerstände)
FB_ASI_ReadParameter [▶ 67]	Universeller FB zum Auslesen aller Parameter eines ASI-Slaves
FB_ASI_WriteParameter [▶ 68]	Universeller FB zum Setzen aller Parameter eines ASI-Slaves
FB_ReadInput_analog [▶ 71]	Lesen von analogen Werten
FB_WriteOutput_analog [▶ 72]	Schreiben von analogen Werten
FB_ASI_Processdata_digital [▶ 69]	Lesen/Schreiben von digitalen Werten
FB_ASI_ParameterControl [▶ 70]	Hintergrundkommunikation **Dieser Baustein muss immer zyklisch aufgerufen werden!!!**

Fehlercodes:

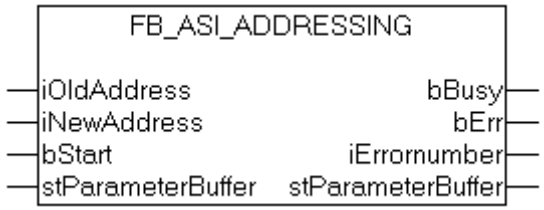
Fehlercode (dezimal):	Beschreibung
1	Kommunikationstimeout
2	ASI-Slaveadresse nicht vorhanden
3 - 10	Reserviert
11	ASI-Slave ist nicht aktiviert (Slave ist nicht in LAS)
12	Bei der Kommunikation ist ein Fehler aufgetreten
13	Datenaustauschbit (CN.4) nicht gesetzt

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.



3.19.1 FB_ASI_Addressing



VAR_IN_OUT

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

stParameterBuffer: Datenpuffer [▶ 131] für die Hintergrundkommunikation

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; (*errorcode 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.

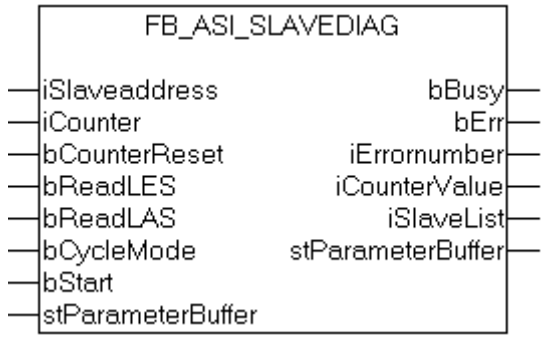
bErr: Dieser Ausgang zeigt den Fehlerstatus an

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

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO-Hardware	Einzubindende SPS-Bibliotheken
TwinCAT Version >= 2.8.0 Build > 739	PC (i386)	ASI-Master-Klemme	TcIoFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib werden automatisch eingebunden)

3.19.2 FB_ASI_SlaveDiag



VAR_IN_OUT

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

stParameterBuffer: Datenpuffer [▶ 131] für die Hintergrundkommunikation

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

VAR_OUTPUT

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

bBUSY: Dieser Ausgang bleibt so lange 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.

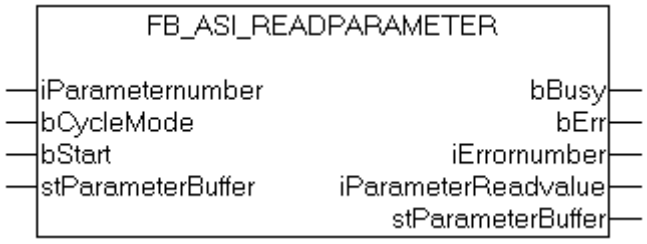
iCountervalue: Zählerstand

iSlaveList: LES bzw. LAS

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO-Hardware	Einzubindende SPS-Bibliotheken
TwinCAT Version >= 2.8.0 Build > 739	PC (i386)	ASI-Master-Klemme	TcIoFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib werden automatisch eingebunden)

3.19.3 FB_ASI_ReadParameter



VAR_IN_OUT

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

stParameterBuffer: Datenpuffer [▶ 131] für die Hintergrundkommunikation

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;     (*errorcode of ASI-Master*)
    iParameterReadvalue :BYTE;
END_VAR
```

bBUSY: Dieser Ausgang bleibt so lange 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

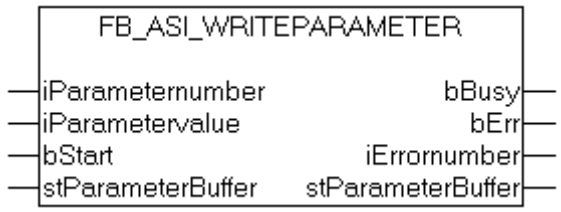
iErrornumber: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Wird durch das Ausführen eines Befehls an den Eingängen auf 0 zurückgesetzt.

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

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO-Hardware	Einzubindende SPS-Bibliotheken
TwinCAT Version >= 2.8.0 Build > 739	PC (i386)	ASI-Master-Klemme	TcIoFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib werden automatisch eingebunden)

3.19.4 FB_ASI_WriteParameter



VAR_IN_OUT

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

stParameterBuffer: Datenpuffer [▶ 131] für die Hintergrundkommunikation

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;     (*errorcode 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

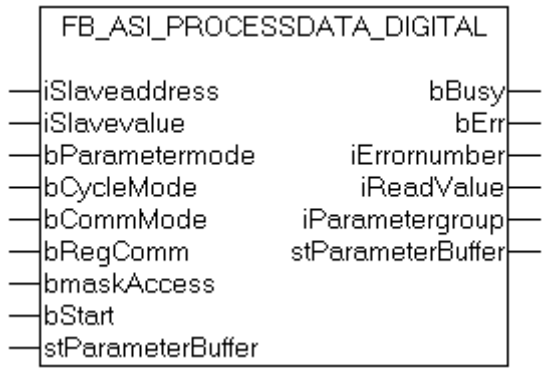
iErrornumber: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Wird durch das Ausführen eines Befehls an den Eingängen auf 0 zurückgesetzt.

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

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO-Hardware	Einzubindende SPS-Bibliotheken
TwinCAT Version >= 2.8.0 Build > 739	PC (i386)	ASI-Master-Klemme	TcIoFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib werden automatisch eingebunden)

3.19.5 FB_ASI_Processdata_digital



VAR_IN_OUT

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

stParameterBuffer: Datenpuffer [▶ 131] für die Hintergrundkommunikation

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: Registerkommuni
kation *)
    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;     (*errorcode 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.

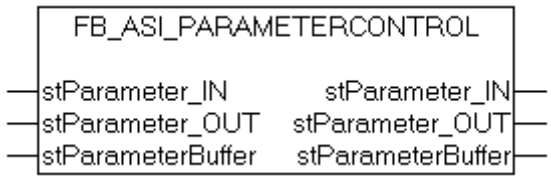
iReadvalue: Prozesswert

iParametergroup: Parametergruppe

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO-Hardware	Einzubindende SPS-Bibliotheken
TwinCAT Version >= 2.8.0 Build > 739	PC (i386)	ASI-Master-Klemme	TcIoFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib werden automatisch eingebunden)

3.19.6 FB_ASI_ParameterControl



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

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 [▶ 131] für die Hintergrundkommunikation

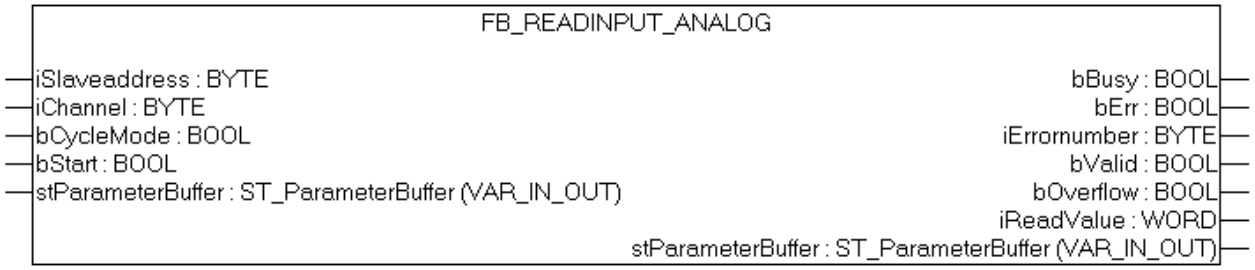
stParameter_IN: Eingangsdaten von der ASI-Klemme [▶ 129]

stParameter_OUT: Ausgangsdaten von der ASI-Klemme [▶ 130]

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO-Hardware	Einzubindende SPS-Bibliotheken
TwinCAT Version >= 2.8.0 Build > 739	PC (i386)	ASI-Master-Klemme	TcIoFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib werden automatisch eingebunden)

3.19.7 FB_ReadInput_analog



VAR_IN_OUT

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

stParameterBuffer:Datenpuffer [▶ 131] für die Hintergrundkommunikation

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;     (*errorcode of ASI-Master*)
    bValid              :BOOL;
    bOverflow           :BOOL;
    iReadValue          :WORD;
END_VAR
```

bBUSY: Dieser Ausgang bleibt so lange 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.

bValid: Gültigkeit der gelesenen Werte

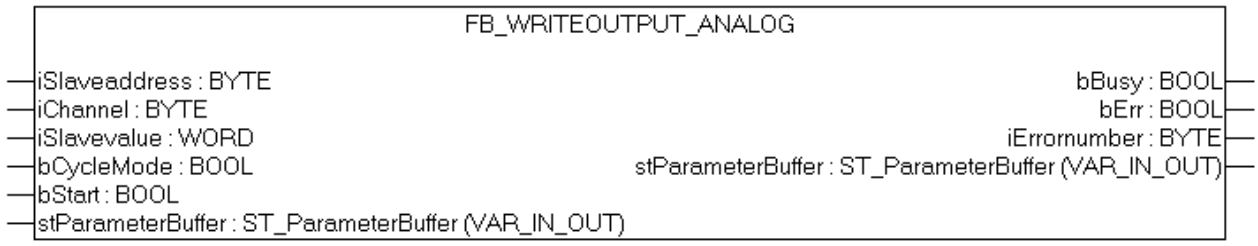
bOverflow: Slave hat einen Wert außerhalb seines Wertebereiches

iReadvalue: Prozesswert

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO-Hardware	Einzubindende SPS-Bibliotheken
TwinCAT Version >= 2.8.0 Build > 739	PC (i386)	ASI-Master-Klemme	TcIoFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib werden automatisch eingebunden)

3.19.8 FB_WriteOutput_analog



VAR_IN_OUT

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

stParameterBuffer: Datenpuffer [▶ 131] für die Hintergrundkommunikation

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;     (*errorcode 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.

bValid: Gültigkeit der gelesenen Werte

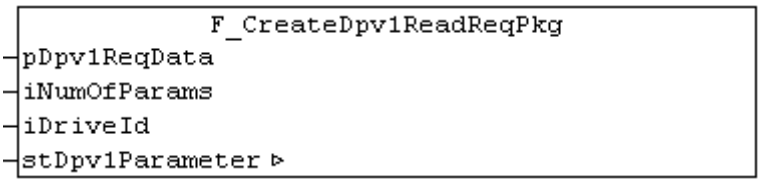
bOverflow: Slave hat einen Wert außerhalb seines Wertebereiches

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO-Hardware	Einzubindende SPS-Bibliotheken
TwinCAT Version >= 2.8.0 Build > 739	PC (i386)	ASI-Master-Klemme	TcIoFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib werden automatisch eingebunden)

3.20 Profibus DPV1 (Sinamics)

3.20.1 F_CreateDpv1ReadReqPkg : USINT



Die Funktion "F_CreateDpv1ReadReqPkg" erzeugt ein DPV1 Telegramm für einen FB_Dpv1Read [▶ 74] 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.

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.

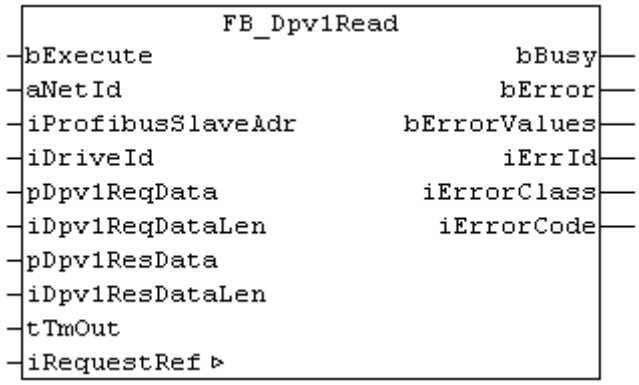
iDriveID: Die ID ist 1 für die ControllerUnit, 2 für das Antriebsobjekt A, 3 für das Antriebsobjekt B eines Doppel/Dreifach-Antriebs. Die Drive ID wird in der Starter Software gesetzt. 1..16 ist möglich.

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 [▶ 139], die zum DPV1 Lesetelegramm zugefügt werden sollen.

3.20.2 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 [▶ 73] erstellt werden, bevor an bExecute eine steigende Flanke ansteht. Das DPV1 Antworttelegramm muss mit F_SplitDpv1ReadResPkg [▶ 76] 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.

Intern werden Instanzen von ADSREAD und ADSWRITE benutzt.

Siehe https://infosys.beckhoff.com/content/1031/tcplclibiofunctions/Resources/11843326347.zip.

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;
END_VAR
```

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

aNetId: Die AmsNetId des Profibus Master Gerätes (siehe ADS-Tab des Profibus Master Gerätes in der I/O-Konfiguration im System Manager).

iProfibusSlaveAdr: Die Profibus slave DP-Adresse des Antriebs. Das ist eine Adresse für mehrere Achsen, spezifiziert im TwinCAT System Manager 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 [▶ 73] 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 [▶ 76] ausgewertet werden nachdem auf bBusy eine negative Flanke erscheint.

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

tTimeout: Bestimmt den Timeout der von den ADS-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 ADS 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 oder funktionsbausteinspezifische Fehlernummern, wenn bError = TRUE.

nErrClass: Profidrive Fehlerklasse

nErrCode: Profidrive Fehlercode

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.

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

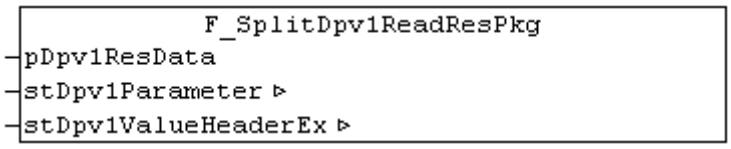
Fehlerklassen	Beschreibung	Fehlercode
0x0 - 0x9	reserviert	-
0xA	Anwendungsfehler	0x0: Lesefehler 0x1: Schreibfehler 0x2: Modulfehler 0x3 - 0x7: reserviert 0x8: Versionskonflikt 0x9: nicht unterstützt 0xA - 0xF: benutzerabhängig
0xB	Zugriffsfehler	0x0: ungültiger Index (kein Datenblock DB47, Parameterzugriff wird nicht unterstützt) 0x1: Schreiblängenfehler 0x2: ungültiger Slot 0x3: Typkonflikt 0x4: ungültiger Bereich 0x5: Zustandskonflikt (Zugriff auf DB47 temporär nicht möglich wegen interner Prozesszustände)

Fehlerklassen	Beschreibung	Fehlercode
		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	-

Voraussetzungen

Entwicklungsumgebung	Zielsystemtyp	IO Hardware	einzufügende PLC-Bibliotheken
TwinCAT v2.10.0 Build > 1307	PC (i386)	Beckhoff FC310x PCI, CX1500-M310, EL6731	TcIoFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib werden automatisch eingefügt)

3.20.3 F_SplitDpv1ReadResPkg : USINT



Die Funktion "F_SplitDpv1ReadResPkg" wertet ein DPV1 Telegramm eines FB_Dpv1Read [▶ 74] 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.

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.

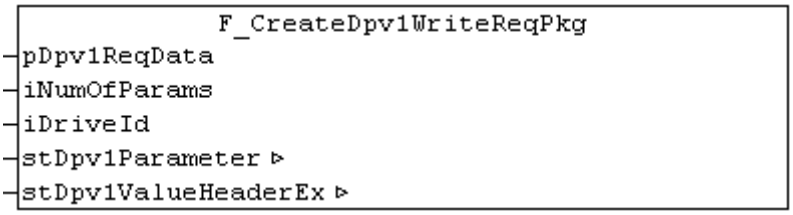
VAR_IN_OUT

```
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 [▶ 139], die zum DPV1 Lesetelegramm zugefügt wurden.

stDpv1ValueHeaderEx: Array von 39 Parameterwerten [▶ 142], die vom Antrieb gelesen wurden.

3.20.4 F_CreateDpv1WriteReqPkg : USINT



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.

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.

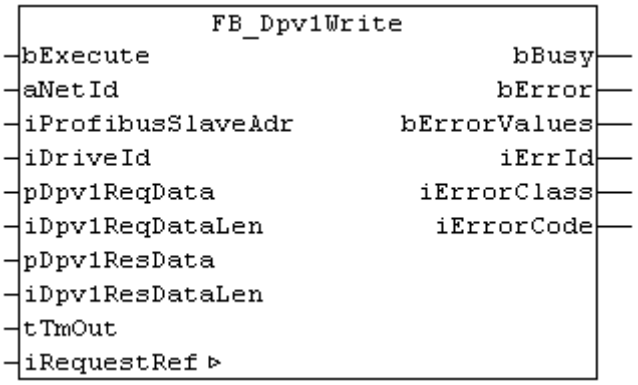
VAR_IN_OUT

```
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 [▶ 139], die zum DPV1 Schreibtelegramm zugefügt werden sollen.

stDpv1ValueHeaderEx: Array of 39 Parameterwerten [▶ 142], die zum DPV1 Schreibtelegramm zugefügt werden sollen.

3.20.5 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 [▶ 77] erstellt werden, bevor an bExecute eine steigende Flanke ansteht. Das DPV1 Antworttelegramm muss mit F_SplitDpv1WriteResPkg [▶ 80] 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.

Intern werden Instanzen von ADSREAD und ADSWRITE benutzt.

Siehe https://infosys.beckhoff.com/content/1031/tcplclibiofunctions/Resources/11843326347.zip.

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;
END_VAR
```

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

aNetId: Die AmsNetId des Profibus Master Gerätes (siehe ADS-Tab des Profibus Master Gerätes in der I/O-Konfiguration im System Manager).

iProfibusSlaveAdr: Die Profibus slave DP-Adresse des Antriebs. Das ist eine Adresse für mehrere Achsen, spezifiziert im TwinCAT System Manager 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 [▶ 77] 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 [▶ 80] ausgewertet werden nachdem auf bBusy eine negative Flanke erscheint.

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

tTimeout: Bestimmt den Timeout der von den ADS-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 ADS 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 oder funktionsbausteinspezifische Fehlernummern, wenn bError = TRUE.

nErrClass: Profidrive Fehlerklasse

nErrCode: Profidrive Fehlercode

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.

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

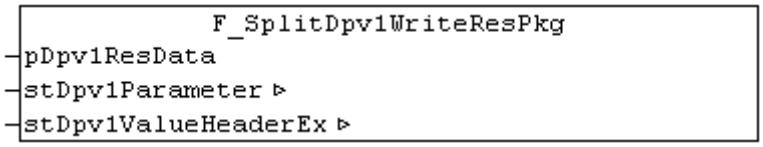
Fehlerklassen	Beschreibung	Fehlercode
0x0 - 0x9	reserviert	-
0xA	Anwendungsfehler	0x0: Lesefehler 0x1: Schreibfehler 0x2: Modulfehler 0x3 - 0x7: reserviert 0x8: Versionskonflikt 0x9: nicht unterstützt 0xA - 0xF: benutzerabhängig
0xB	Zugriffsfehler	0x0: ungültiger Index (kein Datenblock DB47, Parameterzugriff 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)

Fehlerklassen	Beschreibung	Fehlercode
		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	-

Voraussetzungen

Entwicklungsumgebung	Zielsystemtyp	IO Hardware	einzufügende PLC-Bibliotheken
TwinCAT v2.10.0 Build > 1307	PC (i386)	Beckhoff FC310x PCI, CX1500-M310, EL6731	TcIoFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib werden automatisch eingefügt)

3.20.6 F_SplitDpv1WriteResPkg : USINT



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.

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

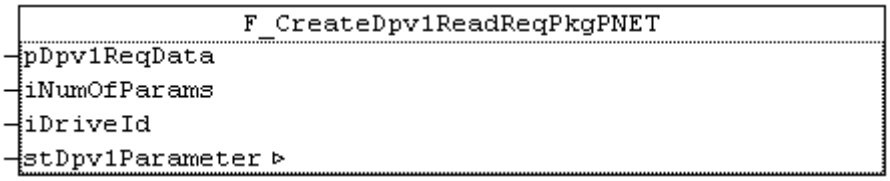
```
VAR_IN_OUT
    stDpv1Parameter     : ARRAY [1..iMAX_DPV1_PARAMS] OF ST_Dpv1ParamAddrEx;  (* list of parameters
*)
    stDpv1ValueHeaderEx : ARRAY [1..iMAX_DPV1_PARAMS] OF ST_Dpv1ValueHeaderEx; (* list of values *)
END_VAR
```

stDpv1Parameter: Array von 39 Parametern [▶ 139], die zum DPV1 Schreibtelegramm zugefügt wurden.

stDpv1ValueHeaderEx: Array von 39 Parameterwerten [▶ 142], die vom Antrieb gelesen wurden.

3.21 Profinet DPV1 (Sinamics)

3.21.1 F_CreateDpv1ReadReqPkgPNET : USINT



Die Funktion "F_CreateDpv1ReadReqPkg" erzeugt ein DPV1 Telegramm für einen FB_Dpv1ReadPNET [▶ 82] 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.

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

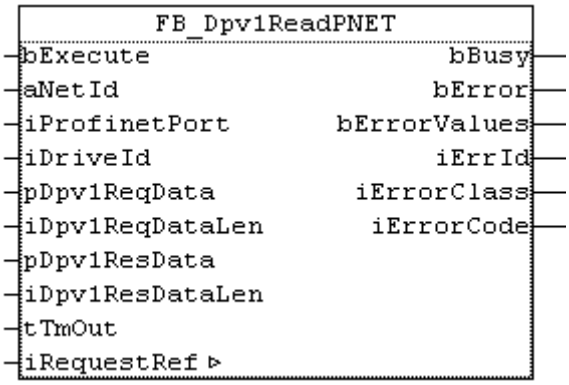
```
VAR_IN_OUT
    stDpv1Parameter : ARRAY [1..iMAX_DPV1_PARAMS] OF ST_Dpv1ParamAddrEx; (* list of parameters *)
END_VAR
```

stDpv1Parameter: Array von 39 Parametern [▶ 139], die zum DPV1 Lesetelegramm zugefügt werden sollen.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO Hardware	Einzubindende SPS-Bibliotheken
TwinCAT v2.11.0, Build > 1553 TwinCAT v2.11.0 R2, Build > 2024	PC (i386)	Beckhoff EL6632	TcIoFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib werden automatisch eingebunden)

3.21.2 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 [▶ 81] erstellt werden, bevor an bExecute eine steigende Flanke ansteht. Das DPV1 Antworttelegramm muss mit F_SplitDpv1ReadResPkgPNET [▶ 84] 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.

Intern werden Instanzen von ADSREAD und ADSWRITE benutzt.

Siehe https://infosys.beckhoff.com/content/1031/tcplclibiofunctions/Resources/11843967755.zip.

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 AmsNetId des Profibus Master Gerätes (siehe ADS-Tab des Profibus Master Gerätes in der I/O-Konfiguration im System Manager).

iProfinetPort: Die Profinet Port-Nummer des Antriebs. Das ist eine Adresse für mehrere Achsen, spezifiziert im TwinCAT System Manager 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 [▶ 73] 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 [▶ 76] ausgewertet werden nachdem auf bBusy eine negative Flanke erscheint.

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

tTmOut: Bestimmt das Timeout der von den ADS-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 ADS 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 oder funktionsbausteinspezifische Fehlernummern, wenn bError = TRUE.

nErrClass: Profidrive Fehlerklasse

nErrCode: Profidrive Fehlercode

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.

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)

Fehlerklassen	Beschreibung	Fehlercode
		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	-

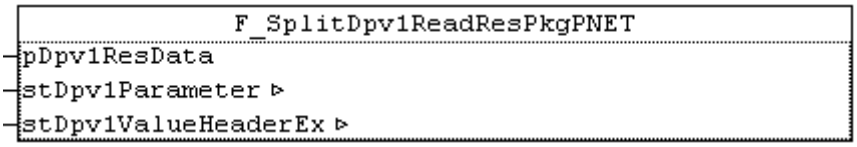
Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO Hardware	Einzubindende SPS-Bibliotheken
TwinCAT v2.11.0, Build > 1553 TwinCAT v2.11.0 R2, Build > 2024	PC (i386)	Beckhoff EL6632	TcIoFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib werden automatisch eingebunden)

Sehen Sie dazu auch

- F_CreateDpv1ReadReqPkgPNET : USINT [▶ 81]
- F_SplitDpv1ReadResPkgPNET : USINT [▶ 84]

3.21.3 F_SplitDpv1ReadResPkgPNET : USINT



Die Funktion "F_SplitDpv1ReadResPkgPNET" wertet ein DPV1 Telegramm eines FB_Dpv1ReadPNET [▶ 82] 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.

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
```

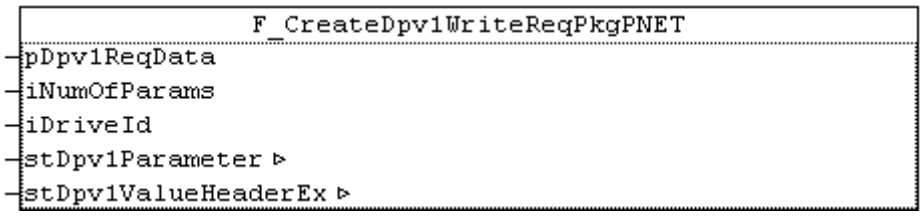
stDpv1Parameter: Array von 39 Parametern [▶ 139], die zum DPV1 Lesetelegramm zugefügt wurden.

stDpv1ValueHeaderEx: Array von 39 Parameterwerten [▶ 142], die vom Antrieb gelesen wurden.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO Hardware	Einzubindende SPS-Bibliotheken
TwinCAT v2.11.0, Build > 1553 TwinCAT v2.11.0 R2, Build > 2024	PC (i386)	Beckhoff EL6632	TcIoFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib werden automatisch eingebunden)

3.21.4 F_CreateDpv1WriteReqPkgPNET : USINT



Die Funktion "F_CreateDpv1WriteReqPkgPNET" erzeugt ein DPV1 Telegramm für einen FB_Dpv1WritePNET [▶ 86] 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.

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
```

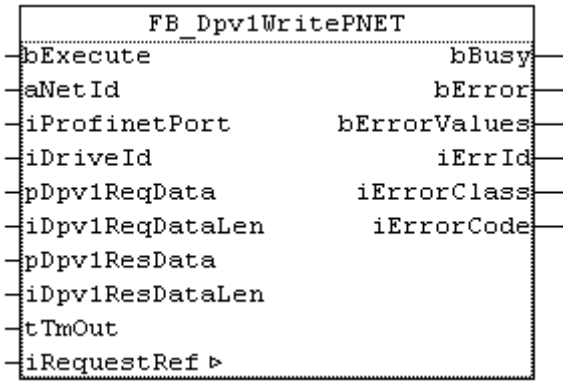
stDpv1Parameter: Array von 39 Parametern [▶ 139], die zum DPV1 Schreibtelegramm zugefügt werden sollen.

stDpv1ValueHeaderEx: Array of 39 Parameterwerten [▶ 142], die zum DPV1 Schreibtelegramm zugefügt werden sollen.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO Hardware	Einzubindende SPS-Bibliotheken
TwinCAT v2.11.0, Build > 1553 TwinCAT v2.11.0 R2, Build > 2024	PC (i386)	Beckhoff EL6632	TcIoFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib werden automatisch eingebunden)

3.21.5 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 [▶ 85] erstellt werden, bevor an bExecute eine steigende Flanke ansteht. Das DPV1 Antworttelegramm muss mit F_SplitDpv1WriteResPkgPNET [▶ 89] 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.

Intern werden Instanzen von ADSREAD und ADSWRITE benutzt.

Siehe https://infosys.beckhoff.com/content/1031/tcplclibiofunctions/Resources/11843967755.zip.

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 AmsNetId des Profibus Master Gerätes (siehe ADS-Tab des Profibus Master Gerätes in der I/O-Konfiguration im System Manager).

iProfinetPort: Die Profinet Port Nummer des Antriebs. Das ist eine Adresse für mehrere Achsen, spezifiziert im TwinCAT System Manager 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 [▶ 85] 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 [▶ 89] ausgewertet werden nachdem auf dem bBusy eine negative Flanke erscheint.

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

tTmOut: Bestimmt den Timeout der von den ADS-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 ADS 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 oder funktionsbausteinspezifische Fehlernummern, wenn bError = TRUE.

nErrClass: Profidrive Fehlerklasse

nErrCode: Profidrive Fehlercode

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.

Funktionbausteinspezifische Fehlercodes	Beschreibung
0x2	falsche Antwortreferenz

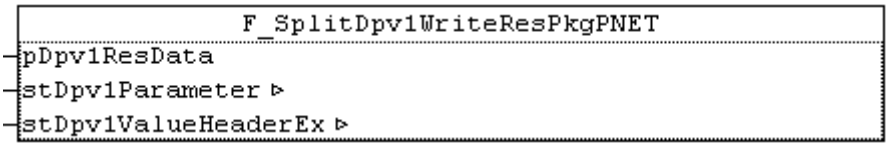
Funktionbausteinspezifische Fehlercodes	Beschreibung
0x3	DPV1 Write fehlerhaft oder teilweise fehlerhaft
0x4	falsche Antwort-ID
other error IDs	siehe ADS Fehlercodes

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	-

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO Hardware	Einzubindende SPS-Bibliotheken
TwinCAT v2.11.0, Build > 1553 TwinCAT v2.11.0 R2, Build > 2024	PC (i386)	Beckhoff EL6632	TcIoFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib werden automatisch eingebunden)

3.21.6 F_SplitDpv1WriteResPkgPNET : USINT



Die Funktion "F_SplitDpv1WriteResPkgPNET" wertet ein DPV1 Telegramm eines FB_Dpv1WritePNET [▶ 86] 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.

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 [▶ 139], die zum DPV1 Schreibtelegramm zugefügt wurden.

stDpv1ValueHeaderEx: Array von 39 Parameterwerten [▶ 142], die vom Antrieb gelesen wurden.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO Hardware	Einzubindende SPS-Bibliotheken
TwinCAT v2.11.0, Build > 1553 TwinCAT v2.11.0 R2, Build > 2024	PC (i386)	Beckhoff EL6632	TcIoFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib werden automatisch eingebunden)

3.22 Beckhoff USV (konfiguriert mit Windows USV service)

3.22.1 FB_GetUPSStatus

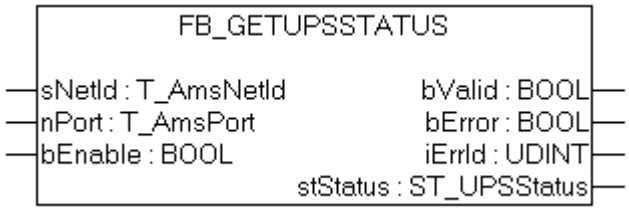


Abb. 2: FB_GetUpsStatus

Voraussetzungen:

a) Die Beckhoff USV Softwarekomponenten wurden installiert.

- Windows 10, 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.

b) 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. Für den lokalen Rechner kann auch ein Leerstring angegeben werden.

nPort: Die ADS-Portnummer. 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_UPS Status-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 oder einen Befehlsspezifischen Fehlercode zurück (Tabelle).

stStatus: Struktur [▶ 132] mit den Statusinformationen der USV.

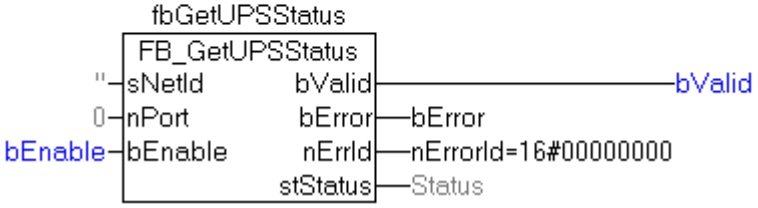
Fehlercodes	Fehlerbeschreibung
0x0000	Kein Fehler
0x8001	USV-Konfigurationsfehler. Möglicherweise ist die USV nicht richtig oder gar keine USV konfiguriert.
0x8002	Kommunikationsfehler. Die Kommunikation zu der USV wurde unterbrochen.
0x8003	Fehler beim Lesen der Statusdaten.

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

Beispiel für einen Aufruf in FUP:

Online-Daten mit Statusinformationen einer USV:

```
⊞ fbGetUPSStatus
⊟ Status
    .Vendor = 'Beckhoff'
    .Model = 'Beckhoff P24V250W'
    .FirmwareRev = '11.7.I'
    .SerialNumber = 'QB0249330541'
    .BatteryLifePercent = 16#00000064
    .BatteryLifeTime = 16#00000123
    .eBatteryStatus = BatteryOk
    .eCommStatus = UpsCommOk
    .ePowerStatus = PowerOnLine
    .dwChargeFlags = 16#00000000
  bError = FALSE
  bValid = TRUE
  nErrorId = 16#00000000
  bEnable = TRUE
```

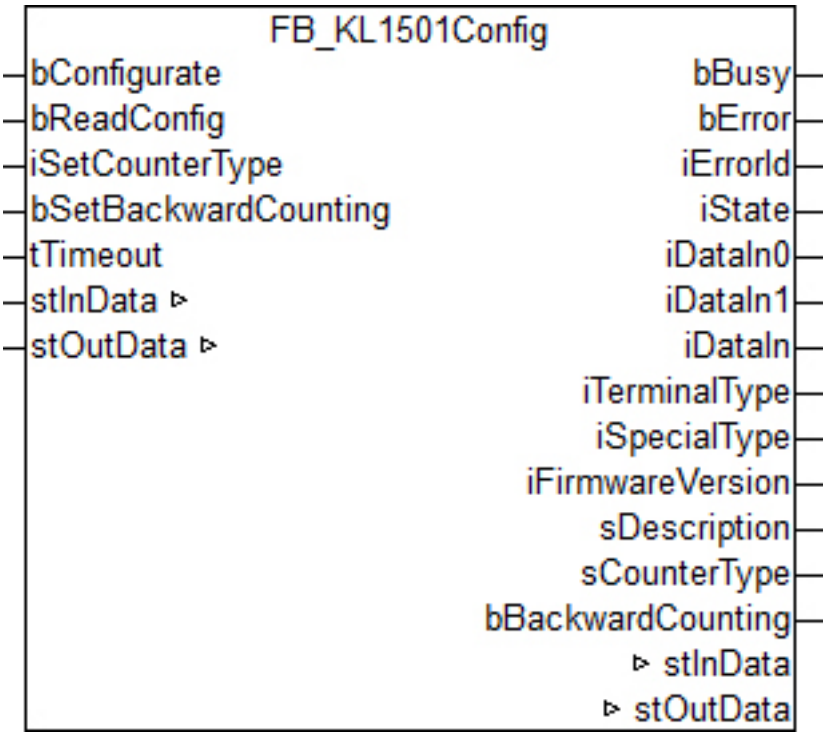


Voraussetzungen

USV-Hardware	Zielplattform	Entwicklungsumgebung	Einzubindende SPS-Bibliotheken
• Beckhoff HID UPS; • 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;	PC oder CX	TwinCAT v2.8.0, Build > 745 TwinCAT v2.9.0, Build > 945	TcIoFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib werden automatisch eingebunden)
		TwinCAT v3.0 und höher	Tc2_IoFunctions

3.23 Busklemmen-Konfiguration

3.23.1 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

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

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,* entgegengenommen.

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,* entgegengenommen.

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

```
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;
```

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 [▶ 121].

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.

iDataIn0: Entspricht der Datenvariable 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 Datenvariable 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 (1501_{dez}) 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

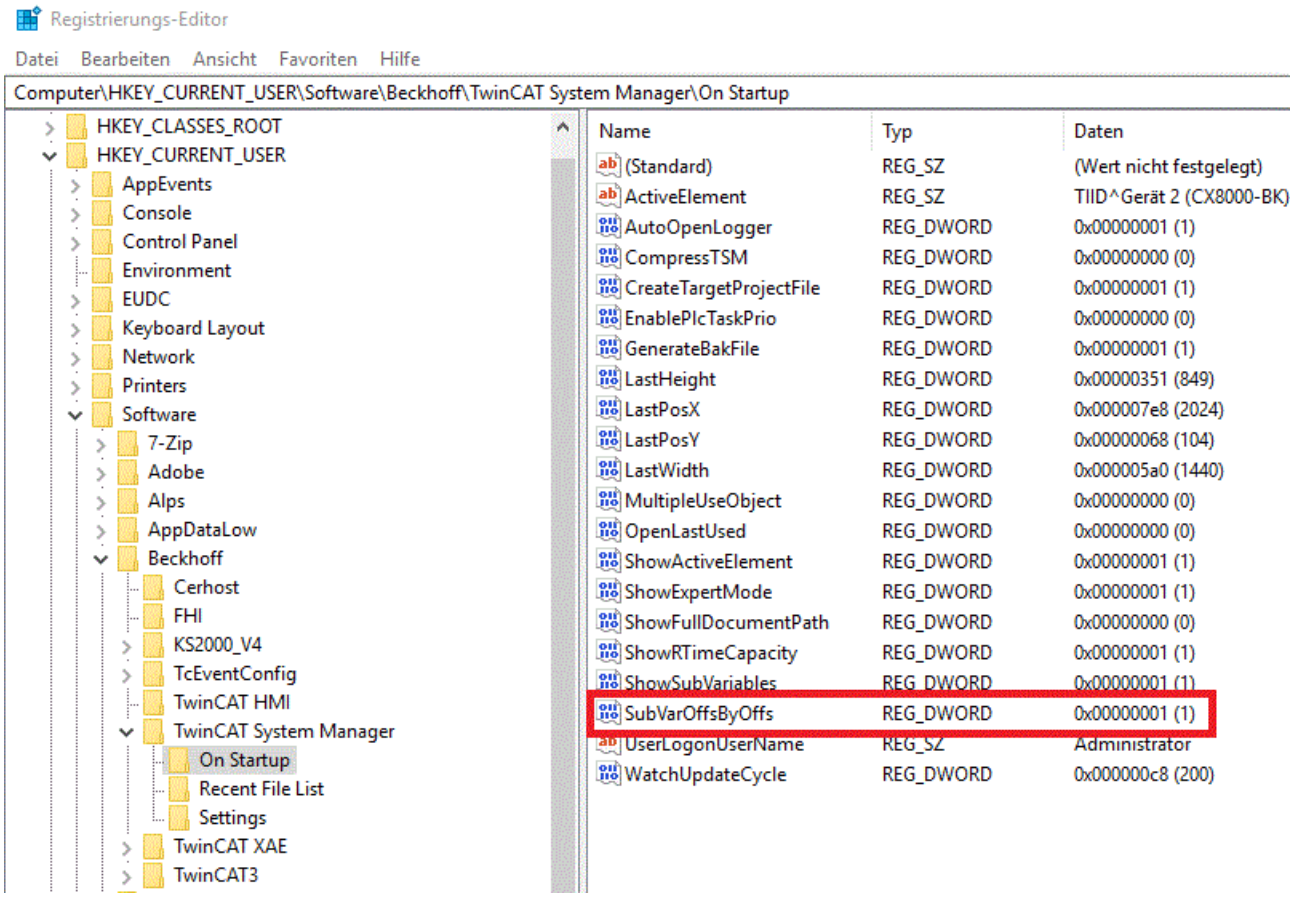
```
stInData   : ST_KL1501InData;
stOutData  : ST_KL1501OutData;
```

stInData: Verweis auf die Struktur des Eingangsprozessabbildes [▶ 143].

stOutData: Verweis auf die Struktur des Ausgangsprozessabbildes [▶ 144].

Hintergrundinformationen

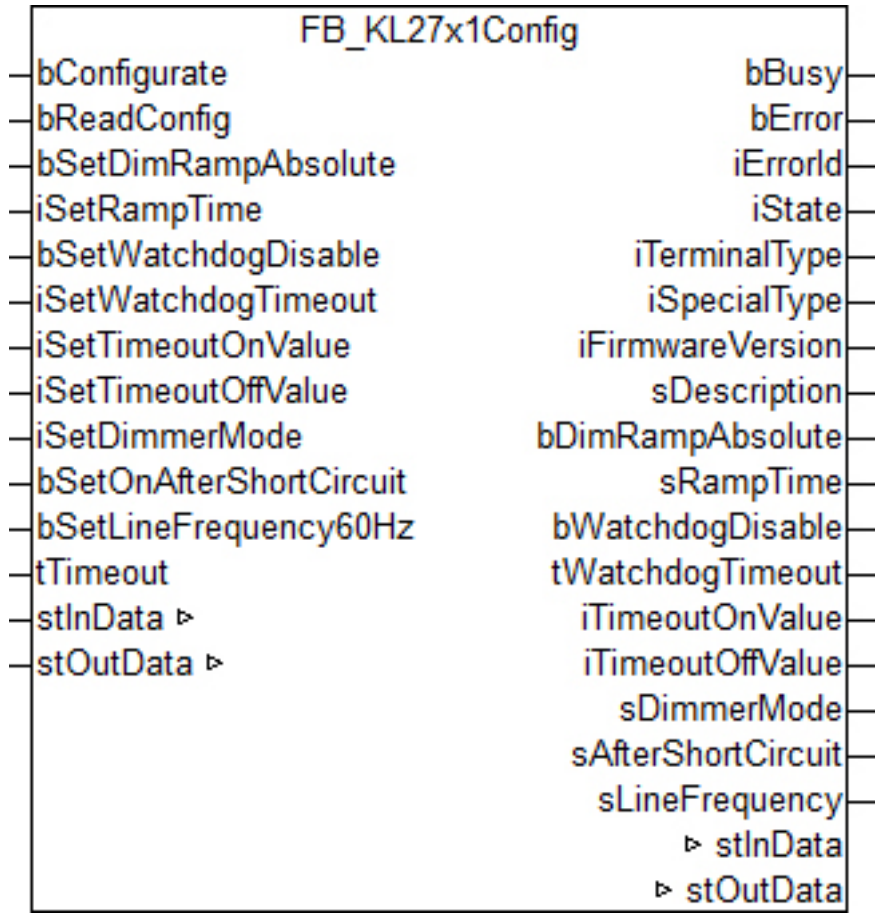
Bei ARM-Systemen kann nicht die Struktur als Ganzes auf das Abbild der Klemme verknüpft werden - die Struktur-Variablen müssen einzeln verknüpft werden.
Zusätzlich muss in der Windows-Registry des Entwicklungsrechners ein neuer DWORD-Key angelegt werden, da es sonst zu Speicherabbildverschiebungen kommt.
Unter "HKEY_CURRENT_USER\Software\Beckhoff\TwinCAT System Manager\On Startup" wird der Key "SubVarOffsByOffs" mit dem Wert "1" eingefügt.
Danach den Entwicklungsrechner einmal neu starten.



Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO-Hardware	Einzubindende SPS-Bibliotheken
TwinCAT v2.11 R3/x64 ab Build 2254	PC/CX	KL1501	TcIoFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib werden automatisch eingebunden)
TwinCAT v2.11 R3/x64 ab Build 2256	BC	KL1501	TcIoFunctions.lb6 (Standard.lb6 wird automatisch eingebunden)
TwinCAT v2.11 R3/x64 ab Build 2256	BX	KL1501	TcIoFunctions.lbx (Standard.lbx wird automatisch eingebunden)

3.23.2 FB_KL27x1Config



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

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;
```

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,* entgegengenommen.

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,* entgegengenommen.

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

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

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 [▶ 121].

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 (2751_{dez}) oder 0x0AC9 (2761_{dez})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

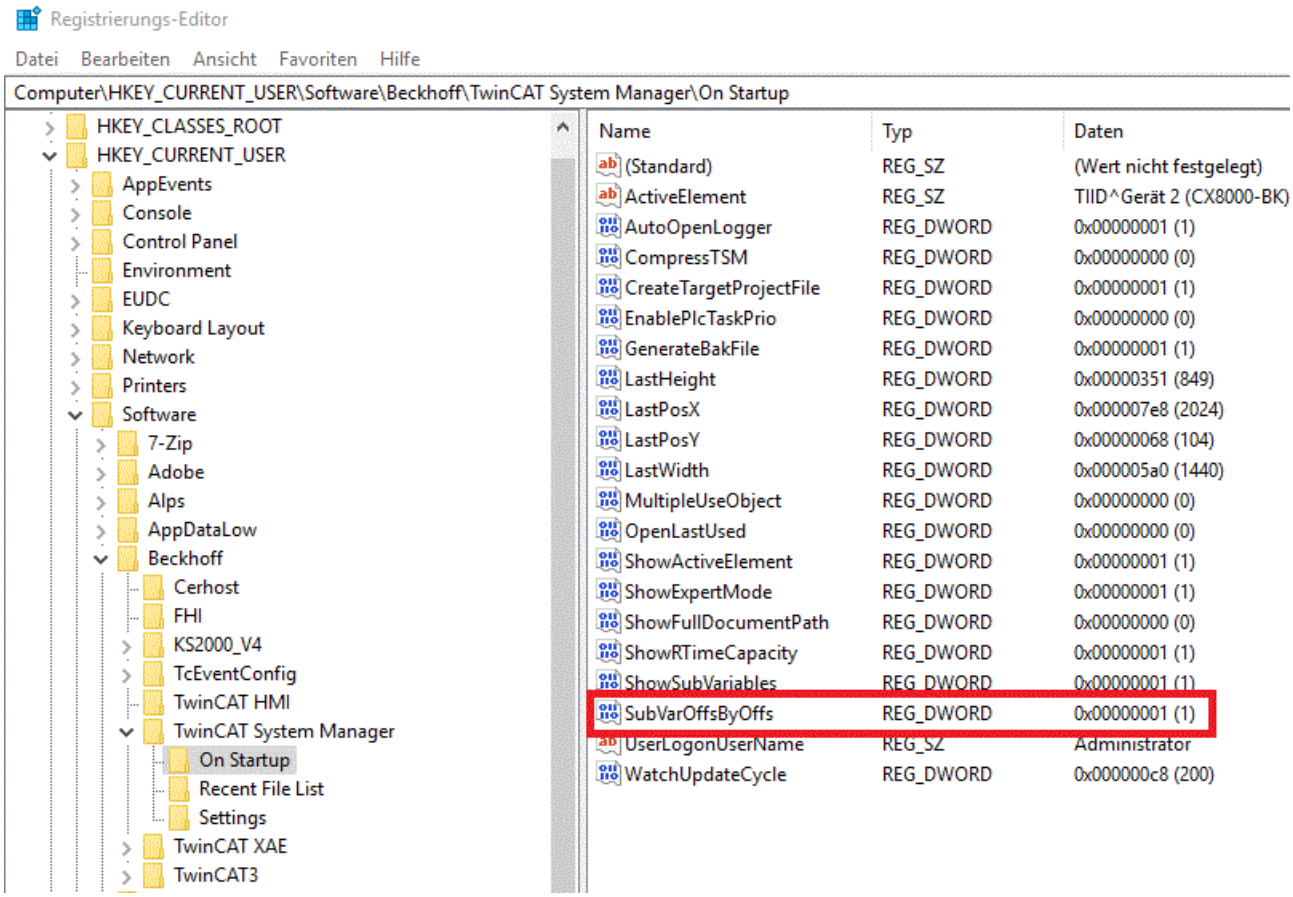
```
stInData   : ST_KL27x1InData;
stOutData  : ST_KL27x1OutData;
```

stInData: Verweis auf die Struktur des Eingangsprozessabbildes [▶ 144].

stOutData: Verweis auf die Struktur des Ausgangsprozessabbildes [▶ 145].

Hintergrundinformationen

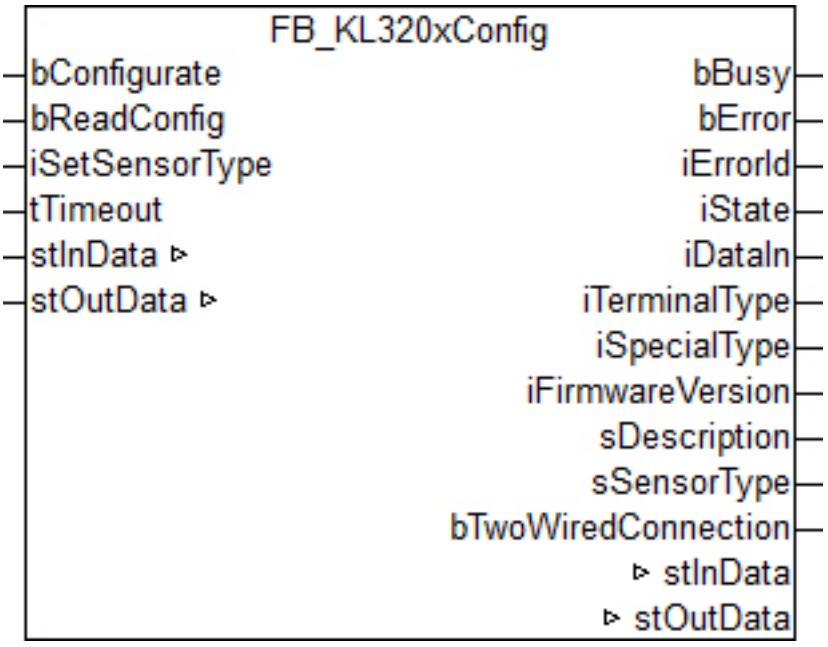
Bei ARM-Systemen kann nicht die Struktur als Ganzes auf das Abbild der Klemme verknüpft werden - die Struktur-Variablen müssen einzeln verknüpft werden.
Zusätzlich muss in der Windows-Registry des Entwicklungsrechners ein neuer DWORD-Key angelegt werden, da es sonst zu Speicherabbildverschiebungen kommt.
Unter "HKEY_CURRENT_USER\Software\Beckhoff\TwinCAT System Manager\On Startup" wird der Key "SubVarOffsByOffs" mit dem Wert "1" eingefügt.
Danach den Entwicklungsrechner einmal neu starten.



Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO-Hardware	Einzubindende SPS-Bibliotheken
TwinCAT v2.11 R3/x64 ab Build 2254	PC/CX	KL2751, KL2761	TcIoFunctions.lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib werden automatisch eingebunden)
TwinCAT v2.11 R3/x64 ab Build 2256	BC	KL2751, KL2761	TcIoFunctions.lb6 (Standard.lb6 wird automatisch eingebunden)
TwinCAT v2.11 R3/x64 ab Build 2256	BX	KL2751, KL2761	TcIoFunctions.lbx (Standard.lbx wird automatisch eingebunden)

3.23.3 FB_KL320xConfig



Funktionsbaustein zur Parametrierung einer KL3201, KL3202 oder KL3204: 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

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

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,* entgegengenommen.

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,* entgegengenommen.

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

iSetSensorType	Element
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

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

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 [▶ 121].

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

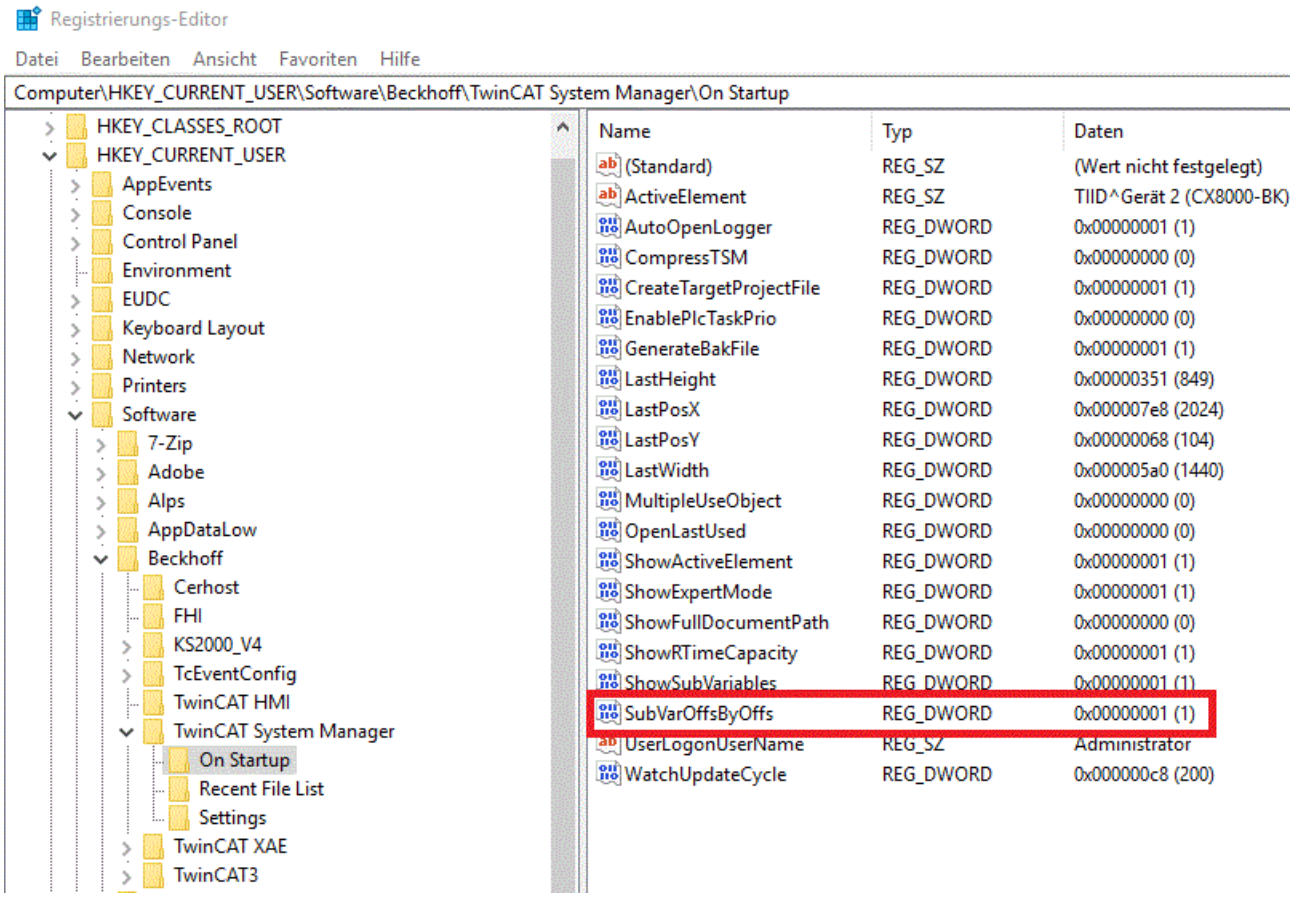
```
stInData   : ST_KL320xInData;
stOutData  : ST_KL320xOutData;
```

stInData: Verweis auf die Struktur des Eingangsprozessabbildes [▶ 146].

stOutData: Verweis auf die Struktur des Ausgangsprozessabbildes [▶ 147].

Hintergrundinformationen

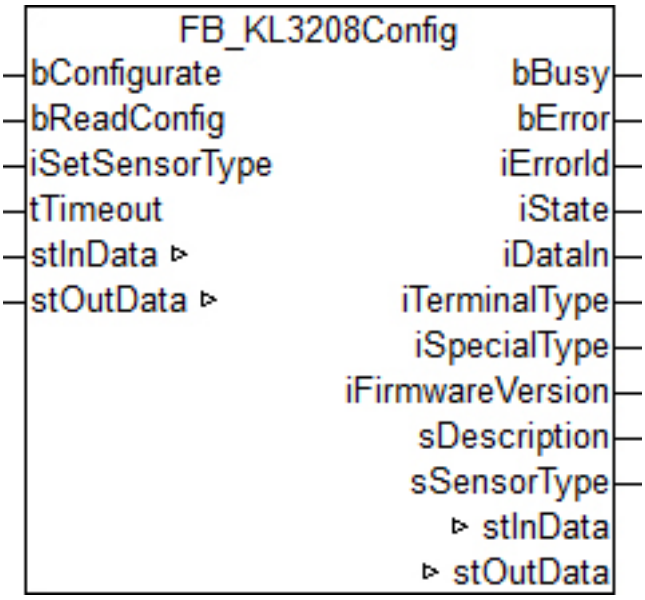
Bei ARM-Systemen kann nicht die Struktur als Ganzes auf das Abbild der Klemme verknüpft werden - die Struktur-Variablen müssen einzeln verknüpft werden.
Zusätzlich muss in der Windows-Registry des Entwicklungsrechners ein neuer DWORD-Key angelegt werden, da es sonst zu Speicherabbildverschiebungen kommt.
Unter "HKEY_CURRENT_USER\Software\Beckhoff\TwinCAT System Manager\On Startup" wird der Key "SubVarOffsByOffs" mit dem Wert "1" eingefügt.
Danach den Entwicklungsrechner einmal neu starten.



Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO-Hardware	Einzubindende SPS-Bibliotheken
TwinCAT v2.11 R3/x64 ab Build 2254	PC/CX	KL3201, KL3202, KL3204	TcIoFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib werden automatisch eingebunden)
TwinCAT v2.11 R3/x64 ab Build 2256	BC	KL3201, KL3202, KL3204	TcIoFunctions.lb6 (Standard.lb6 wird automatisch eingebunden)
TwinCAT v2.11 R3/x64 ab Build 2256	BX	KL3201, KL3202, KL3204	TcIoFunctions.lbx (Standard.lbx wird automatisch eingebunden)

3.23.4 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

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

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,* entgegengenommen.

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,* entgegengenommen.

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

iSetSensorType	Element
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

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

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 [▶ 121].

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 (3208_{dez}) 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

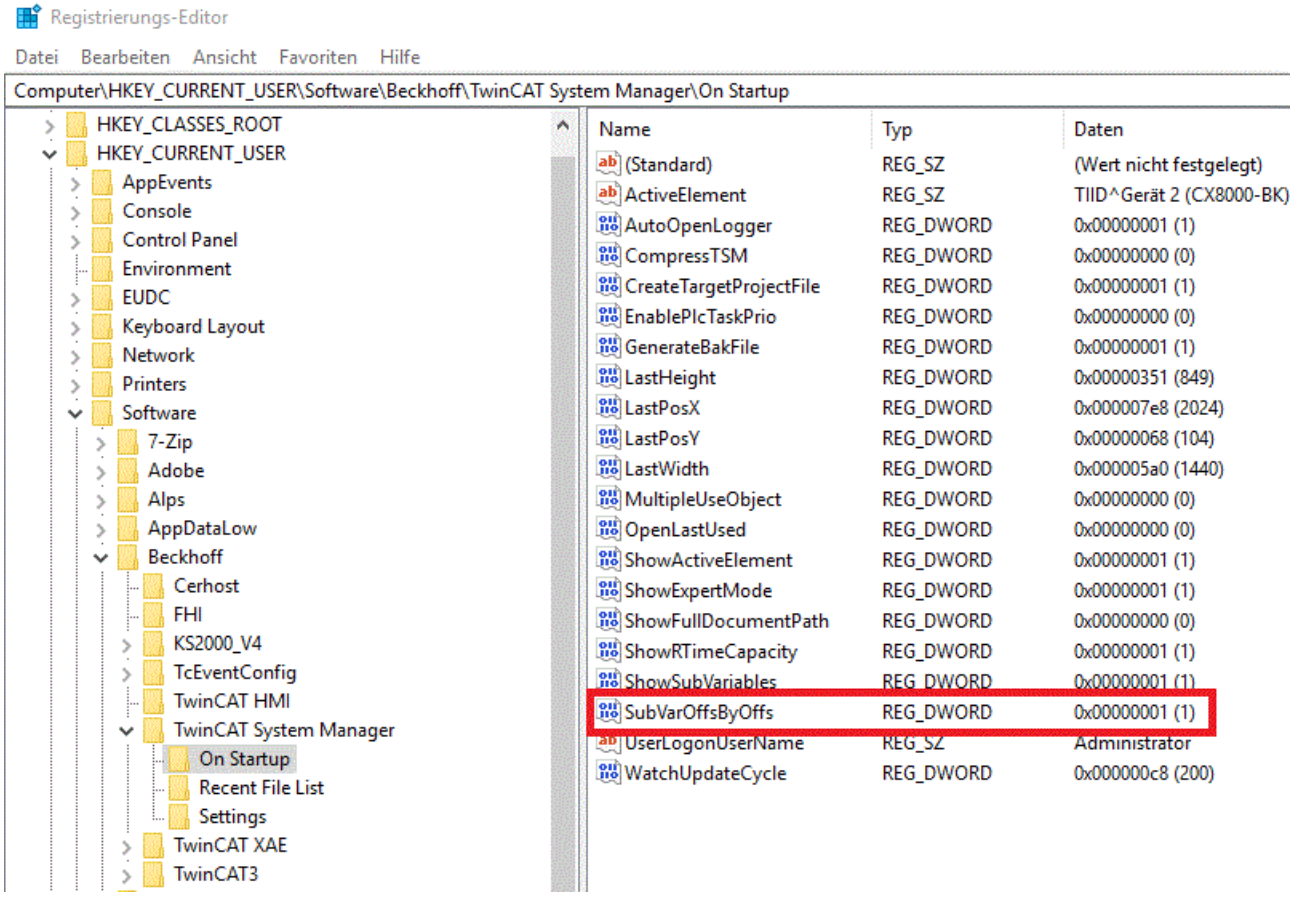
```
stInData    : ST_KL3208InData;
stOutData   : ST_KL3208OutData;
```

stInData: Verweis auf die Struktur des Eingangsprozessabbildes [▶ 145].

stOutData: Verweis auf die Struktur des Ausgangsprozessabbildes [▶ 146].

Hintergrundinformationen

Bei ARM-Systemen kann nicht die Struktur als Ganzes auf das Abbild der Klemme verknüpft werden - die Struktur-Variablen müssen einzeln verknüpft werden.
Zusätzlich muss in der Windows-Registry des Entwicklungsrechners ein neuer DWORD-Key angelegt werden, da es sonst zu Speicherabbildverschiebungen kommt.
Unter "HKEY_CURRENT_USER\Software\Beckhoff\TwinCAT System Manager\On Startup" wird der Key "SubVarOffsByOffs" mit dem Wert "1" eingefügt.
Danach den Entwicklungsrechner einmal neu starten.

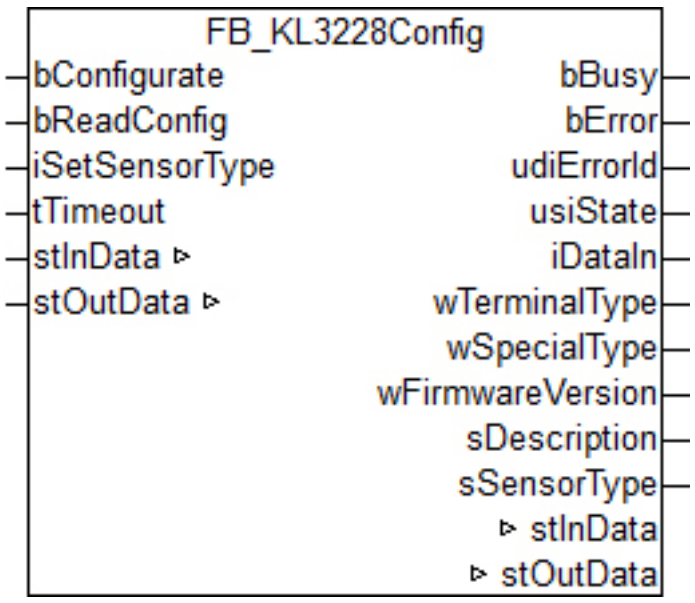


Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO-Hardware	Einzubindende SPS-Bibliotheken
TwinCAT v2.11 R3/x64 ab Build 2254	PC/CX	KL3208	TcIoFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib werden automatisch eingebunden)

Entwicklungsumgebung	Zielplattform	IO-Hardware	Einzubindende SPS-Bibliotheken
TwinCAT v2.11 R3/x64 ab Build 2256	BC	KL3208	TcIoFunctions.lb6 (Standard.lb6 wird automatisch eingebunden)
TwinCAT v2.11 R3/x64 ab Build 2256	BX	KL3208	TcIoFunctions.lbx (Standard.lbx wird automatisch eingebunden)

3.23.5 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

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

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,* entgegengenommen.

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,* entgegengenommen.

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

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

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 [▶ 121].

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 (3228_{dez}) 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

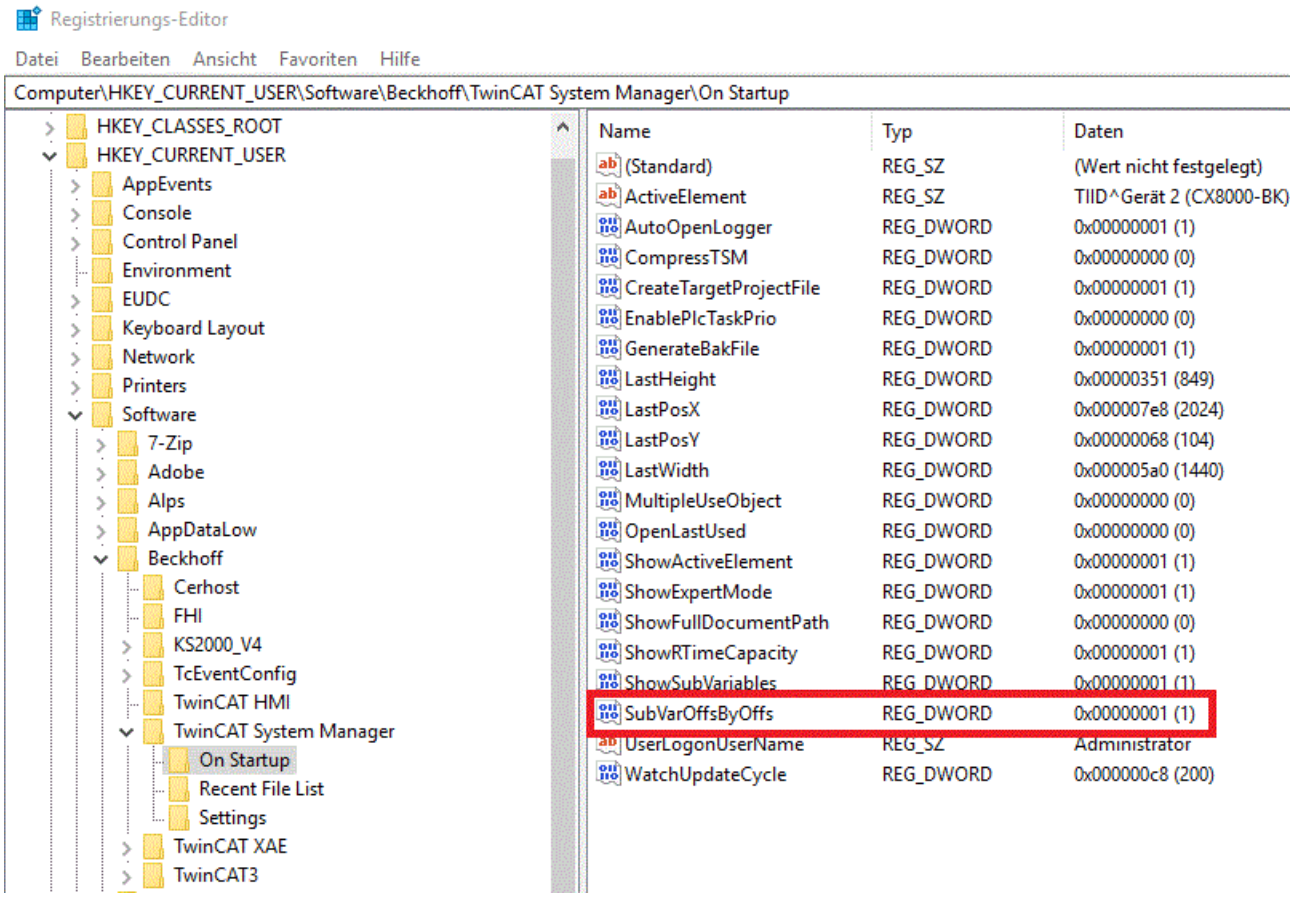
```
stInData   : ST_KL3228InData;
stOutData  : ST_KL3228OutData;
```

stInData: Verweis auf die Struktur des Eingangsprozessabbildes.

stOutData: Verweis auf die Struktur des Ausgangsprozessabbildes.

Hintergrundinformationen

Bei ARM-Systemen kann nicht die Struktur als Ganzes auf das Abbild der Klemme verknüpft werden - die Struktur-Variablen müssen einzeln verknüpft werden.
Zusätzlich muss in der Windows-Registry des Entwicklungsrechners ein neuer DWORD-Key angelegt werden, da es sonst zu Speicherabbildverschiebungen kommt.
Unter "HKEY_CURRENT_USER\Software\Beckhoff\TwinCAT System Manager\On Startup" wird der Key "SubVarOffsByOffs" mit dem Wert "1" eingefügt.
Danach den Entwicklungsrechner einmal neu starten.



Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO-Hardware	Einzubindende SPS-Bibliotheken
TwinCAT v2.11 R3/x64 ab Build 2254	PC/CX	KL3228	TcIoFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib werden automatisch eingebunden)
TwinCAT v2.11 R3/x64 ab Build 2256	BC	KL3228	TcIoFunctions.lb6 (Standard.lb6 wird automatisch eingebunden)
TwinCAT v2.11 R3/x64 ab Build 2256	BX	KL3228	TcIoFunctions.lbx (Standard.lbx wird automatisch eingebunden)

Sehen Sie dazu auch

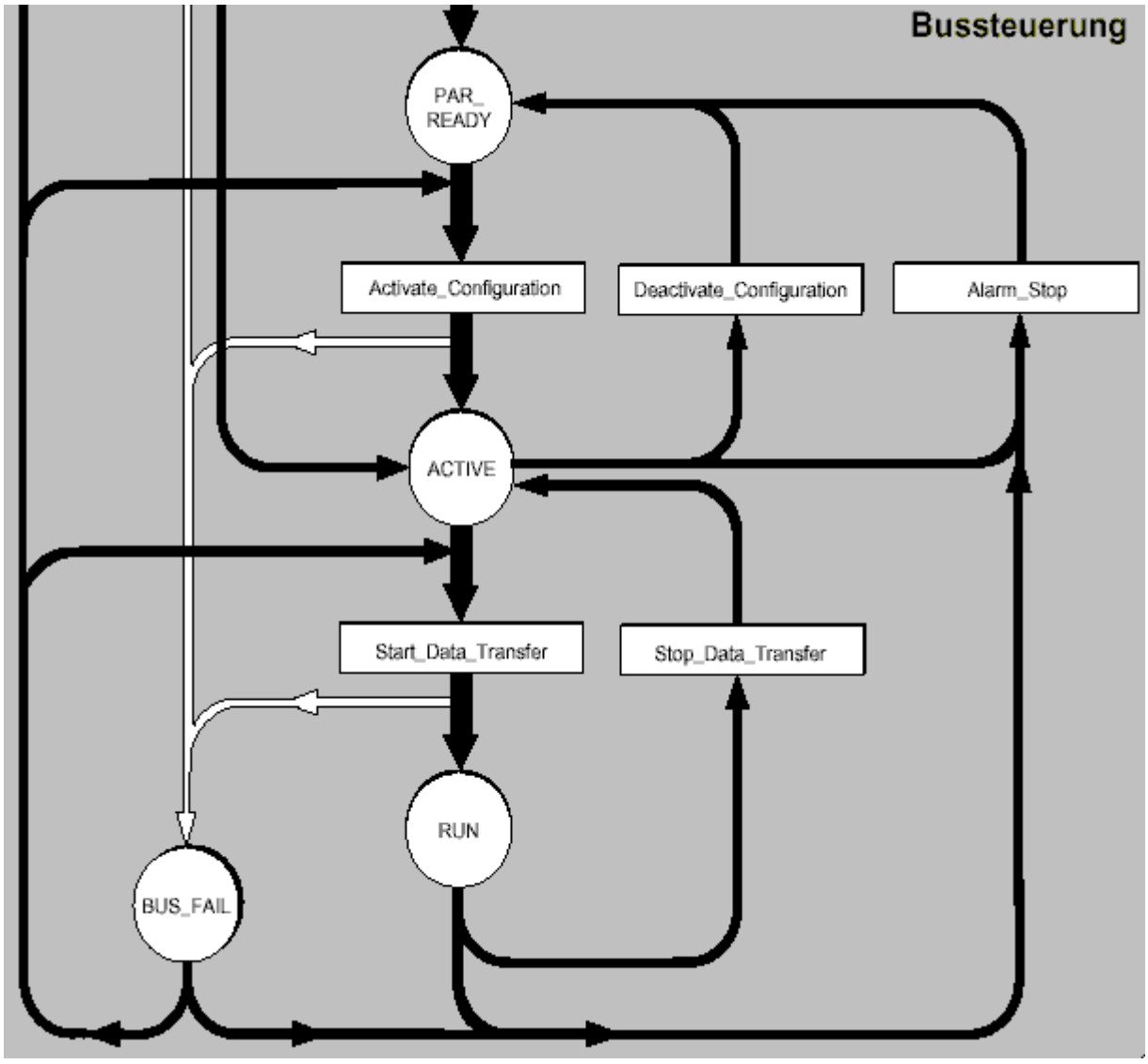
- ST_KL3228InData [▶ 147]
- ST_KL3228OutData [▶ 147]

3.24 Drittherstellergeräte

3.24.1 Phoenix IBS SC/I-T

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 [▶ 110] : Führt den Befehl **Activate_Configuration** aus.

SCIT_DeactivateConfiguration [▶ 111] : Führt den Befehl **Deactivate_Configuration** aus.

SCIT_StartDataTransfer [▶ 112] : Führt den Befehl **Start_Data_Transfer** aus.

SCIT_StopDataTransfer [▶ 113] : Führt den Befehl **Stop_Data_Transfer** aus.

SCIT_AlarmStop [▶ 115] : Führt den Befehl **Alarm_Stop** aus.

Konfiguration

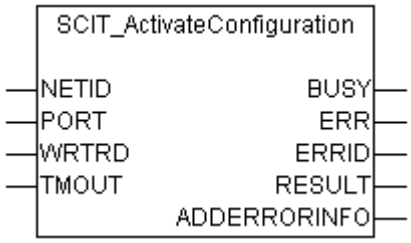
SCIT_ControlActiveConfiguration [▶ 116] : Dient zur Beeinflussung der aktiven Konfiguration der Interbus-Teilnehmer. Dieses Kommando kann sowohl im Zustand *PAR_READY* als auch im Zustand *ACTIVE* und *RUN* ausgeführt werden. Hierüber können einzelne, abhängige und gruppierte Teilnehmer aktiviert und deaktiviert werden.

Fehlerdiagnose

SCIT_GetErrorInfo [▶ 117]: Liefert Fehlerart und Fehlerort eines Interbus-Teilnehmers nach einem Busfehler.

SCIT_ConfDevErrAll [▶ 119]: Peripheriestörungen aller Geräte quittieren.

3.24.1.1 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. Intern wird ein ADSRDWRT Baustein aufgerufen, der mit den in der TwinCAT System Manager Hilfe beschriebenen Parametern versehen wird.

Durch einen **Activate_Configuration** wird die Karte in den Zustand ACTIVE [▶ 109] versetzt.

VAR_INPUT

```
VAR_INPUT
    NETID       : T_AmsNetId;
    PORT        : T_AmsPort;
    WRTRD       : BOOL;
    TMOUT       : TIME;
END_VAR
```

NETID: Hier kann die AmsNetId des Rechners angegeben werden, in dem die Karte eingebaut ist. Befindet sich die Karte auf demselben System kann auch ein Leerstring angegeben werden.

PORT: Beinhaltet die ADS-Portnummer der Karte, die im System Manager vergeben wurde.

WRTRD: Über eine positive Flanke an diesem Wert wird der Baustein aktiviert.

TMOUT: Gibt die Timeout-Zeit an, die an den internen ADSWRTRD Baustein weitergeleitet wird.

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 so lange an, bis eine Rückmeldung erfolgt.

ERR: Sollte ein ADS-Fehler bei der Übertragung des Kommandos erfolgen wird der ERR Ausgang nach Rücknahme des BUSY-Signals gesetzt.

ERRID: Liefert bei positivem ADS-Fehler die ADS-Fehlernummer.

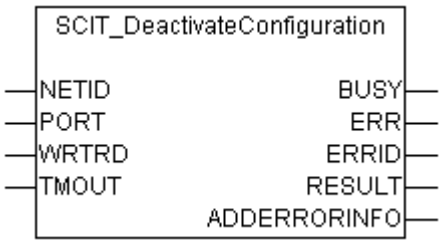
RESULT: Liefert das Ergebnis der Karte zurück (Voraussetzung ist ein fehlerfreier ADS-Transport (ERR = FALSE)). RESULT = 0 kennzeichnet eine erfolgreiche Ausführung des Befehls. Ein Wert ungleich 0 beinhaltet die Fehlernummer der Phoenixkarte.

ADDERRINFO: Enthält bei negativer Rückmeldung der Karte zusätzliche Fehlerinformationen (vgl. Befehlsbeschreibung der Phoenixkarte).

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO-Hardware	Einzubindende SPS-Bibliotheken
TwinCAT v2.7.0	PC (i386)	Phoenix: IBS SC/I-T; IBS SC/RI/RT-LK	Standard.Lib; PLCSystem.Lib; TcPLCAds.Lib; TcPlcIoFunctions.Lib
TwinCAT v2.8.0	PC (i386)	Phoenix: IBS SC/I-T; IBS SC/RI/RT-LK	TcIoFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib werden automatisch eingebunden)

3.24.1.2 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. Intern wird ein ADSRDWRT Baustein aufgerufen, der mit den in der TwinCAT System Manager Hilfe beschriebenen Parametern versehen wird.

Durch einen **Deactivate_Configuration** wird die Karte in den Zustand PAR_READY [▶ 109] versetzt und alle Ausgänge zurückgenommen.

VAR_INPUT

```
VAR_INPUT
    NETID       : T_AmsNetId;
    PORT        : T_AmsPort;
    WRTRD       : BOOL;
    TMOUT       : TIME;
END_VAR
```

NETID: Hier kann die AmsNetId des Rechners angegeben werden, in dem die Karte eingebaut ist. Befindet sich die Karte auf demselben System kann auch ein Leerstring angegeben werden.

PORT: Beinhaltet die ADS-Portnummer der Karte, die im System Manager vergeben wurde.

WRTRD: Über eine positive Flanke an diesem Wert wird der Baustein aktiviert.

TMOUT: Gibt die Timeout-Zeit an, die an den internen ADSWRTRD Baustein weitergeleitet wird.

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 so lange an, bis eine Rückmeldung erfolgt.

ERR: Sollte ein ADS-Fehler bei der Übertragung des Kommandos erfolgen wird der ERR Ausgang nach Rücknahme des BUSY-Signals gesetzt.

ERRID: Liefert bei positivem ADS-Fehler die ADS-Fehlernummer.

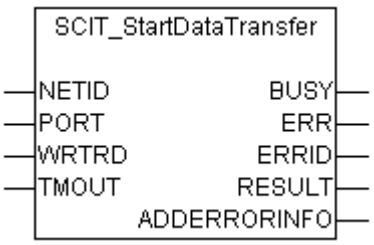
RESULT: Liefert das Ergebnis der Karte zurück (Voraussetzung ist ein fehlerfreier ADS-Transport (ERR = FALSE)). RESULT = 0 kennzeichnet eine erfolgreiche Ausführung des Befehls. Ein Wert ungleich 0 beinhaltet die Fehlernummer der Phoenixkarte.

ADDERRINFO: Enthält bei negativer Rückmeldung der Karte zusätzliche Fehlerinformationen (vgl. Befehlsbeschreibung der Phoenixkarte).

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO-Hardware	Einzubindende SPS-Bibliotheken
TwinCAT v2.7.0	PC (i386)	Phoenix: IBS SC/I-T; IBS SC/RI/RT-LK	Standard.Lib; PLCSystem.Lib; TcPLCAds.Lib; TcPlcIoFunctions.Lib
TwinCAT v2.8.0	PC (i386)	Phoenix: IBS SC/I-T; IBS SC/RI/RT-LK	TcIoFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib werden automatisch eingebunden)

3.24.1.3 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. Intern wird ein ADSRDWRT Baustein aufgerufen, der mit den in der TwinCAT System Manager Hilfe beschriebenen Parametern versehen wird.

Durch einen **Start_Data_Transfer** wird die Karte in den Zustand RUN [▶ 109] versetzt.

VAR_INPUT

```
VAR_INPUT
    NETID       : T_AmsNetId;
    PORT        : T_AmsPort;
```

```
    WRTRD       : BOOL;
    TMOUT       : TIME;
END_VAR
```

NETID: Hier kann die AmsNetId des Rechners angegeben werden, in dem die Karte eingebaut ist. Befindet sich die Karte auf demselben System kann auch ein Leerstring angegeben werden.

PORT: Beinhaltet die ADS-Portnummer der Karte, die im System Manager vergeben wurde.

WRTRD: Über eine positive Flanke an diesem Wert wird der Baustein aktiviert.

TMOUT: Gibt die Timeout-Zeit an, die an den internen ADSWRTRD Baustein weitergeleitet wird.

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 so lange an, bis eine Rückmeldung erfolgt.

ERR: Sollte ein ADS-Fehler bei der Übertragung des Kommandos erfolgen wird der ERR Ausgang nach Rücknahme des BUSY-Signals gesetzt.

ERRID: Liefert bei positivem ADS-Fehler die ADS-Fehlernummer.

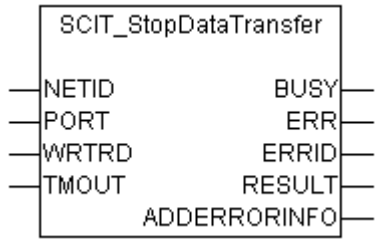
RESULT: Liefert das Ergebnis der Karte zurück (Voraussetzung ist ein fehlerfreier ADS-Transport (ERR = FALSE)). RESULT = 0 kennzeichnet eine erfolgreiche Ausführung des Befehls. Ein Wert ungleich 0 beinhaltet die Fehlernummer der Phoenixkarte.

ADDERRINFO: Enthält bei negativer Rückmeldung der Karte zusätzliche Fehlerinformationen (vgl. Befehlsbeschreibung der Phoenixkarte).

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO-Hardware	Einzubindende SPS-Bibliotheken
TwinCAT v2.7.0	PC (i386)	Phoenix: IBS SC/I-T; IBS SC/RI/RT-LK	Standard.Lib; PLCSystem.Lib; TcPLCAds.Lib; TcPlcIoFunctions.Lib
TwinCAT v2.8.0	PC (i386)	Phoenix: IBS SC/I-T; IBS SC/RI/RT-LK	TcIoFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib werden automatisch eingebunden)

3.24.1.4 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. Intern wird ein ADSRDWRT Baustein aufgerufen, der mit den in der TwinCAT System Manager Hilfe beschriebenen Parametern versehen wird.

Durch einen **Stop_Data_Transfer** wird die Karte in den Zustand ACTIVE [▶ 109] versetzt, die Ausgänge werden *nicht* zurückgenommen.

VAR_INPUT

```
VAR_INPUT
    NETID       : T_AmsNetId;
    PORT        : T_AmsPort;
    WRTRD       : BOOL;
    TMOUT       : TIME;
END_VAR
```

NETID: Hier kann die AmsNetId des Rechners angegeben werden, in dem die Karte eingebaut ist. Befindet sich die Karte auf demselben System kann auch ein Leerstring angegeben werden.

PORT: Beinhaltet die ADS-Portnummer der Karte, die im System Manager vergeben wurde.

WRTRD: Über eine positive Flanke an diesem Wert wird der Baustein aktiviert.

TMOUT: Gibt die Timeout-Zeit an, die an den internen ADSWRTRD Baustein weitergeleitet wird.

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 so lange an, bis eine Rückmeldung erfolgt.

ERR: Sollte ein ADS-Fehler bei der Übertragung des Kommandos erfolgen wird der ERR Ausgang nach Rücknahme des BUSY-Signals gesetzt.

ERRID: Liefert bei positivem ADS-Fehler die ADS-Fehlernummer.

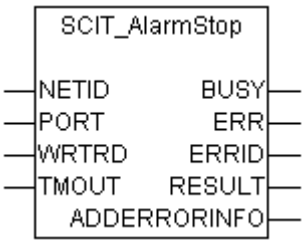
RESULT: Liefert das Ergebnis der Karte zurück (Voraussetzung ist ein fehlerfreier ADS-Transport (ERR = FALSE)). RESULT = 0 kennzeichnet eine erfolgreiche Ausführung des Befehls. Ein Wert ungleich 0 beinhaltet die Fehlernummer der Phoenixkarte.

ADDERRINFO: Enthält bei negativer Rückmeldung der Karte zusätzliche Fehlerinformationen (vgl. Befehlsbeschreibung der Phoenixkarte).

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO-Hardware	Einzubindende SPS-Bibliotheken
TwinCAT v2.7.0	PC (i386)	Phoenix: IBS SC/I-T; IBS SC/RI/RT-LK	Standard.Lib; PLCSystem.Lib; TcPLCAds.Lib; TcPlcIoFunctions.Lib
TwinCAT v2.8.0	PC (i386)	Phoenix: IBS SC/I-T; IBS SC/RI/RT-LK	TcIoFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib werden automatisch eingebunden)

3.24.1.5 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. Intern wird ein ADSRDWRT Baustein aufgerufen, der mit den in der TwinCAT System Manager Hilfe beschriebenen Parametern versehen wird.

Durch einen **Alarm_Stop** wird die Karte in den Zustand PAR_READY [▶ 109] versetzt und alle Ausgänge zurückgenommen.

VAR_INPUT

```
VAR_INPUT
    NETID       : T_AmsNetId;
    PORT        : T_AmsPort;
    WRTRD       : BOOL;
    TMOUT       : TIME;
END_VAR
```

NETID: Hier kann die AmsNetId des Rechners angegeben werden, in dem die Karte eingebaut ist. Befindet sich die Karte auf demselben System kann auch ein Leerstring angegeben werden.

PORT: Beinhaltet die ADS-Portnummer der Karte, die im System Manager vergeben wurde.

WRTRD: Über eine positive Flanke an diesem Wert wird der Baustein aktiviert.

TMOUT: Gibt die Timeout-Zeit an, die an den internen ADSWRTRD Baustein weitergeleitet wird.

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 so lange an, bis eine Rückmeldung erfolgt.

ERR: Sollte ein ADS-Fehler bei der Übertragung des Kommandos erfolgen wird der ERR Ausgang nach Rücknahme des BUSY-Signals gesetzt.

ERRID: Liefert bei positivem ADS-Fehler die ADS-Fehlernummer.

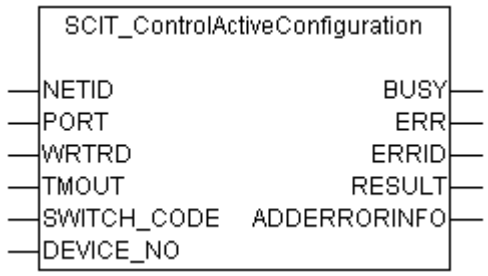
RESULT: Liefert das Ergebnis der Karte zurück (Voraussetzung ist ein fehlerfreier ADS-Transport (ERR = FALSE)). RESULT = 0 kennzeichnet eine erfolgreiche Ausführung des Befehls. Ein Wert ungleich 0 beinhaltet die Fehlernummer der Phoenixkarte.

ADDERRINFO: Enthält bei negativer Rückmeldung der Karte zusätzliche Fehlerinformationen (vgl. Befehlsbeschreibung der Phoenixkarte).

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO-Hardware	Einzubindende SPS-Bibliotheken
TwinCAT v2.7.0	PC (i386)	Phoenix: IBS SC/I-T; IBS SC/RI/RT-LK	Standard.Lib; PLCSystem.Lib; TcPLCAds.Lib; TcPlcIoFunctions.Lib
TwinCAT v2.8.0	PC (i386)	Phoenix: IBS SC/I-T; IBS SC/RI/RT-LK	TcIoFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib werden automatisch eingebunden)

3.24.1.6 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. Intern wird ein ADSRDWRT Baustein aufgerufen, der mit den in der TwinCAT System Manager Hilfe beschriebenen Parametern versehen 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;
    SWITCH_CODE : WORD;
    DEVICE_NO   : WORD;
END_VAR
```

NETID: Hier kann die AmsNetId des Rechners angegeben werden, in dem die Karte eingebaut ist. Befindet sich die Karte auf dem selben System kann auch ein Leerstring angegeben werden.

PORT: Beinhaltet die ADS-Portnummer der Karte, die im System Manager vergeben wurde.

WRTRD: Über eine positive Flanke an diesem Wert wird der Baustein aktiviert.

TMOUT: Gibt die Timeout-Zeit an, die an den internen ADSWRTRD Baustein weitergeleitet wird.

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 ADS-Fehler bei der Übertragung des Kommandos erfolgen wird der ERR Ausgang nach Rücknahme des BUSY-Signals gesetzt.

ERRID: Liefert bei positivem ADS-Fehler die ADS-Fehlernummer.

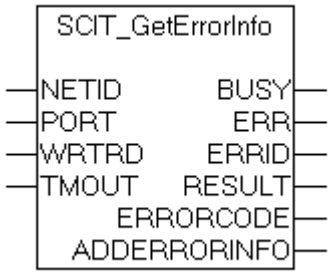
RESULT: Liefert das Ergebnis der Karte zurück (Voraussetzung ist ein fehlerfreier ADS-Transport (ERR = FALSE)). RESULT = 0 kennzeichnet eine erfolgreiche Ausführung des Befehls. Ein Wert ungleich 0 beinhaltet die Fehlernummer der Phoenixkarte.

ADDERRINFO: Enthält bei negativer Rückmeldung der Karte zusätzliche Fehlerinformationen (vgl. Befehlsbeschreibung der Phoenixkarte).

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO-Hardware	Einzubindende SPS-Bibliotheken
TwinCAT v2.7.0	PC (i386)	Phoenix: IBS SC/I-T; IBS SC/RI/RT-LK	Standard.Lib; PLCSystem.Lib; TcPLCAds.Lib; TcPlcIoFunctions.Lib
TwinCAT v2.8.0	PC (i386)	Phoenix: IBS SC/I-T; IBS SC/RI/RT-LK	TcIoFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib werden automatisch eingebunden)

3.24.1.7 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. Intern wird ein ADSRDWRT Baustein aufgerufen, der mit den in der TwinCAT System Manager Hilfe beschriebenen Parametern versehen wird.

VAR_INPUT

```
VAR_INPUT
    NETID       : T_AmsNetId;
    PORT        : T_AmsPort;
    WRTRD       : BOOL;
    TMOUT       : TIME;
END_VAR
```

NETID: Hier kann die AmsNetId des Rechners angegeben werden, in dem die Karte eingebaut ist. Befindet sich die Karte auf demselben System kann auch ein Leerstring angegeben werden.

PORT: Beinhaltet die ADS-Portnummer der Karte, die im System Manager vergeben wurde.

WRTRD: Über eine positive Flanke an diesem Wert wird der Baustein aktiviert.

TMOUT: Gibt die Timeout-Zeit an, die an den internen ADSWRTRD Baustein weitergeleitet wird.

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 so lange an, bis eine Rückmeldung erfolgt.

ERR: Sollte ein ADS-Fehler bei der Übertragung des Kommandos erfolgen wird der ERR Ausgang nach Rücknahme des BUSY-Signals gesetzt.

ERRID: Liefert bei positivem ADS-Fehler die ADS-Fehlernummer.

RESULT: Liefert das Ergebnis 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.

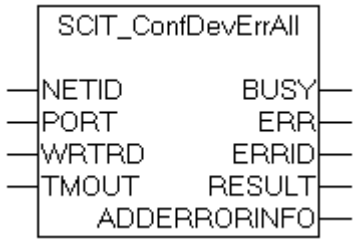
ERRORCODE: Liefert Informationen zur Fehlerart (vgl. Fehlerbeschreibung der Phoenixkarte).

ADDERRORINFO: Enthält bei negativer Rückmeldung der Karte den Fehlerort (vgl. Fehlerbeschreibung der Phoenixkarte).

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO-Hardware	Einzubindende SPS-Bibliotheken
TwinCAT v2.7.0	PC (i386)	Phoenix: IBS SC/I-T; IBS SC/RI/RT-LK	Standard.Lib; PLCSystem.Lib; TcPLCAds.Lib; TcPlcIoFunctions.Lib
TwinCAT v2.8.0	PC (i386)	Phoenix: IBS SC/I-T; IBS SC/RI/RT-LK	TcIoFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib werden automatisch eingebunden)

3.24.1.8 SCIT_ConfDevErrAll



Der Funktionsbaustein "SCIT_ConfDevErrAll" quittiert Peripheriestörungen aller vorhandenen Geräte gleichzeitig. Intern wird von dem ADSRDWRT-Funktionsbaustein 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;
END_VAR
```

NETID: Hier kann die AmsNetId des Rechners angegeben werden, in dem die Karte eingebaut ist. Befindet sich die Karte auf demselben System kann auch ein Leerstring angegeben werden.

PORT: Beinhaltet die ADS-Portnummer der Karte, die im System Manager vergeben wurde.

WRTRD: Über eine positive Flanke an diesem Wert wird der Baustein aktiviert.

TMOUT: Gibt die Timeout-Zeit an, die an den internen ADSWRTRD Baustein weitergeleitet wird.

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 so lange an, bis eine Rückmeldung erfolgt.

ERR: Sollte ein ADS-Fehler bei der Übertragung des Kommandos erfolgen wird der ERR Ausgang nach Rücknahme des BUSY-Signals gesetzt.

ERRID: Liefert bei positivem ADS-Fehler die ADS-Fehlernummer.

RESULT: Liefert das Ergebnis 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).

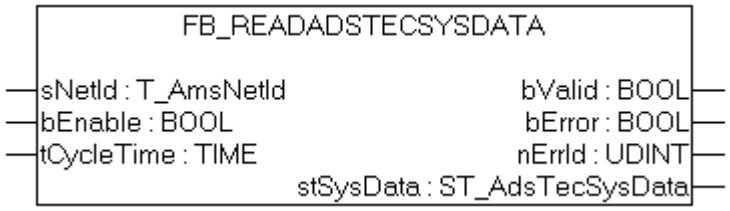
Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO-Hardware	Einzubindende SPS-Bibliotheken
TwinCAT v2.7.0	PC (i386)	Phoenix: IBS SC/I-T; IBS SC/RI/RT-LK	Standard.Lib; PLCSystem.Lib; TcPLCAds.Lib; TcPlcIoFunctions.Lib

Entwicklungsumgebung	Zielplattform	IO-Hardware	Einzubindende SPS-Bibliotheken
TwinCAT v2.8.0	PC (i386)	Phoenix: IBS SC/I-T; IBS SC/RI/RT-LK	TcIoFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib werden automatisch eingebunden)

3.24.2 ads-tec

3.24.2.1 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. 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.

stSysData : Struktur [▶ 138] mit den Systemdaten/Diagnosedaten.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO-Hardware	Einzubindende SPS-Bibliotheken
TwinCAT v2.8.0, Build > 746 TwinCAT v2.9.0, Build > 945	PC (i386)	ads-tec PC	TcIoFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib werden automatisch eingebunden)

3.25 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*.
0c801B	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 gelesen. Siehe Ausgang *sSensorType*.
0x8028	32808	FB_KL320xConfig: Zweileiteranschluß ist für KL3204 nicht zulässig.

ErrId (hex)	ErrId (dez)	Beschreibung
0x8029	32809	FB_KL320xConfig: Ungültiger Sensortyp gewählt. Siehe Eingang *iSetSensorType*.
0x802A	30810	FB_KL320xConfig: Ungültiger Sensortyp gelesen. Siehe Ausgang *sSensorType*.
0x8030	32816	FB_KL3228Config: Ungültiger Sensortyp gewählt. Siehe Eingang *iSetSensorType*.
0x8031	32817	FB_KL3228Config: Ungültiger Sensortyp gelesen. Siehe Ausgang *sSensorType*.

4 Datenstrukturen

4.1 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	Einzubindende SPS-Bibliotheken
TwinCAT v2.7.0	PC (i386)	Standard.Lib; PLCSystem.Lib; TcPLCAds.Lib; TcPlcIoFunctions.Lib
TwinCAT v2.8.0	PC (i386)	TcIoFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib werden automatisch eingebunden)

4.2 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	Einzubindende SPS-Bibliotheken
TwinCAT v2.8.0 Build > 735	PC (i386)	TcIoFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib werden automatisch eingebunden)

4.3 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	Einzubindende SPS-Bibliotheken
TwinCAT v2.8.0 Build > 735	PC (i386)	TcIoFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib werden automatisch eingebunden)

4.4 ST_SercosParamAttrib

ST_SercosParamAttrib beinhaltet das Attribut nAttrib des Sercos-Parameters in einzelne Variablen zerlegt.

```
TYPE ST_SercosParamAttrib :
STRUCT
    nFactor         : UINT;         (* bits  0..15 *)
    eLength         : E_SercosAttribLen;    (* bits 16..18 *)
    bCommand        : BOOL;         (* bit  19     *)
    eType           : E_SercosAttribType;  (* bits 20..22 *)
    bReserved1      : BOOL;         (* bit  23     *)
    nComma          : USINT;            (* bits 24..27 *)
    bWriteProtCP2   : BOOL;         (* bit  28     *)
    bWriteProtCP3   : BOOL;         (* bit  29     *)
    bWriteProtCP4   : BOOL;         (* bit  30     *)
    bReserved2      : BOOL;         (* bit  31     *)
END_STRUCT
END_TYPE
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v2.8.0 Build > 735	PC (i386)	TcIoFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib werden automatisch eingebunden)

Sehen Sie dazu auch

- E_SercosAttribLen [▶ 125]
- E_SercosAttribType [▶ 125]

4.5 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 n = 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
(= 0)
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. = nnnn

cbSize
(    2 Bytes), i.e. = 02
00              (= 2)
nAttrib
  (    4 Bytes), i.e. = xx xx xx xx
arrData     (2 Bytes), i.e. = 12 34

2nd parameter
ST_SercosParamHeader + Data             (16
Bytes)
------------------------------------------------------------------------------

nIDN        (    2 Bytes), i.e. = nnnn

cbSize
(    2 Bytes), i.e. = 08
00              (= 8)
nAttrib
  (    4 Bytes), i.e. = xx xx xx xx
arrData     (    8 Bytes),
i.e. = 12 34 56 78 9a bc de
f0

3rd parameter
ST_SercosParamHeader + Data             (12
Bytes)
------------------------------------------------------------------------------

nIDN        (    2 Bytes), i.e. = nnnn

cbSize
(    2 Bytes), i.e. = 04
00              (= 4)
nAttrib
  (    4 Bytes), i.e. = xx xx xx 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 [▶ 126]), 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	Einzubindende SPS-Bibliotheken
TwinCAT v2.8.0	PC (i386)	TcIoFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib werden automatisch eingebunden)

4.6 ST_PZD_IN

```
TYPE ST_PZD_IN :
STRUCT
    wSTW :WORD;
    wHIW :WORD;
    PZD3 :WORD;
    PZD4 :WORD;
    PZD5 :WORD;
    PZD6 :WORD;
END_STRUCT
END_TYPE
```

Datenwörter vom Antrieb zur PLC.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO-Hardware	Einzubindende SPS-Bibliotheken
TwinCAT v2.8.0 Build > 737	PC (i386)	AX2000 Profibus box	TcIoFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib;

Entwicklungsumgebung	Zielplattform	IO-Hardware	Einzubindende SPS-Bibliotheken
			TcUtilities.Lib werden automatisch eingebunden)

4.7 ST_PZD_OUT

```
TYPE ST_PZD_OUT :
STRUCT
    wCtrlW          :WORD;
    PZD2            :WORD;
    PZD3            :WORD;
    PZD4            :WORD;
    PZD5            :WORD;
    PZD6            :WORD;
END_STRUCT
END_TYPE
```

Datenwörter von der PLC zum Antrieb.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO-Hardware	Einzubindende SPS-Bibliotheken
TwinCAT v2.8.0 Build > 737	PC (i386)	AX2000 Profibus box	TcIoFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib werden automatisch eingebunden)

4.8 ST_Parameter_IN

```
TYPE ST_ParameterBuffer :
STRUCT
    iParameterStatus            :WORD;
    iParameterReadValue         :DWORD;
END_STRUCT
END_TYPE
```

Eingangsdaten von der ASI-Klemme

ParameterStatus

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

Byte-Offset	Bit-Offset	Beschreibung
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
TwinCAT Version >= 2.8.0 Build > 739	PC (i386)	ASI-Master-Klemme	TcIoFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib werden automatisch eingebunden)

4.9 ST_Parameter_OUT

```
TYPE ST_ParameterBuffer :
STRUCT
    ParameterControl        :WORD;
    iParametervalue         :DWORD;
END_STRUCT
END_TYPE
```

Ausgangsdaten zur ASI-Klemme

ParameterStatus

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
TwinCAT Version >= 2.8.0 Build > 739	PC (i386)	ASI-Master-Klemme	TcIoFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib;

Entwicklungsumgebung	Zielplattform	IO-Hardware	Einzubindende SPS-Bibliotheken
			TcUtilities.Lib werden automatisch eingebunden)

4.10 ST_ParameterBuffer

```
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
```

Datenpuffer für die E/A-Daten der ASI-Klemme

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO-Hardware	Einzubindende SPS-Bibliotheken
TwinCAT Version >= 2.8.0 Build > 739	PC (i386)	ASI-Master-Klemme	TcIoFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib werden automatisch eingebunden)

4.11 ST_NovRamAddrInfo

```
TYPE ST_NovRamAddrInfo:
STRUCT
    pCardAddress    :       UDINT;
    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
TwinCAT v2.8.0 Build > 740	PC (i386)	FCxxxx cards with NOV-RAM and all other cards with DPRAM	TcIoFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib werden automatisch eingebunden)

4.12 ST_UPSStatus

```
TYPE ST_UPSStatus
STRUCT
    Vendor          : STRING;(* USV-Herstellername*)
 Model           : STRING;(* USV-Modelstring*)
 FirmwareRev         : STRING;(* Versionsinformationen zur USV-Firmware*)
 SerialNumber        : STRING;(* Seriennummer der USV*)
 BatteryLifePercent          : DWORD;(* Verbliebene Akkulaufzeit in Prozent (0..100%)*)
 BatteryLifeTime     : DWORD;(* Verbliebene Akkulaufzeit in Minuten*)
 eBatteryStatus      : E_BatteryStatus;(* USV-Akkustatus*)
 eCommStatus         : E_UpsCommStatus;(* Status der Kommunikation zur USV*)
 ePowerStatus        : E_UpsPowerStatus;(* Status der externen Spannungsversorgung*)
 nPowerFailCnt       : DWORD;(* Power-Fail-
Zähler. Der Zähler wird inkrementiert wenn ein Spannungsaufall vom USV-Service erkannt wurde*)
 dwChargeFlags       : DWORD;
(* Akku und spezielle Statusinformationen. Es können eins oder mehrere Bits gleichzeitig gesetzt sei
n.
    Bits0..7 := Allgemeine Akku-Statusinformation (wenn alle Bits Null sind => Status unbekannt)
        Bit0 := High (Bit gesetzt => Ladestatus hoch)
        Bit1 := Low (Bit gesetzt => Ladestatus niedrig)
        Bit2 := Critical (Bit gesetzt => Ladestatus kritisch)
        Bit3 := Charging (Bit gesetzt => Akku wird geladen)
        Bits4..6 := zur Zeit reserviert (alle Bits sind 0)
        Bit7 := No Battery (Bit gesetzt => Akku fehlt oder wurde abgeklemmt)
    Bits8..15 := Spezielle Statusinformationen (alle Bits sind 0 => Status in Ordnung oder unbekannt
)
        Bit8 := UPS Fan Error (Bit gesetzt => Lüfterhardware meldet einen Fehler, Bit nicht gesetzt
=> kein Fehler detektiert)
        Bit9 := Over Temperature (Bit gesetzt => Übertemperatur wurde von der Hardware detektiert, B
it nicht gesetzt => keine Übertemperatur detektiert)
        Bit10 := Service Interval Notify (Bit gesetzt => Intervalezeit ist abgelaufen, Bit nicht ges
etzt =>Intervalzeit ist noch nicht abgelaufen )
        Bit11 := Under Temperature (Bit gesetzt => Untertemperatur wurde von der Hardware detektiert
, Bit nicht gesetzt => keine Untertemperatur detektiert)
        Bit12 := Fuse Not Ok (Bit gesetzt => Sicherung defekt oder fehlt, Bit nicht gesetzt => Siche
rung ist in Ordnung)
        Bit13 := Alarm1 (zur Zeit reserviert, Bit ist 0)
        Bit14 := Alarm2 (zur Zeit reserviert, Bit ist 0)
        Bit15 := Alarm3 (zur Zeit reserviert, Bit ist 0)
    Bits16..31 := zur Zeit reserviert, alle Bits sind 0)
     *)
END_STRUCT
END_TYPE
```

Nicht alle USV-Modelle können alle Statusinformationen liefern.

X : Die Statusinformation ist bei diesem Model vorhanden.

*) Nur vorhanden, wenn das Model "Smart Signaling to any APC UPS & TwinCAT" konfiguriert wurde.

Statusinformation	Beckhoff HID UPS, BAPI v1	Beckhoff P24Vxxxx	Beckhoff CP903x ISA/ PCI-Karte	Beckhoff CX2100-09x 4	APC Back-UPS Pro 280, Smart-UPS 420	Beschreibung
Vendor	X	X	X	X	X	Herstellername.
Model	X	X	X	X	X	Modelstring. Leerstring, wenn keine USV konfiguriert wurde.

Statusinformation	Beckhoff HID UPS, BAPI v1	Beckhoff P24Vxxxx	Beckhoff CP903x ISA/ PCI-Karte	Beckhoff CX2100-09x 4	APC Back-UPS Pro 280, Smart-UPS 420	Beschreibung
FirmwareRev	X	X	X	X	X	Versionsinformationen zur USV-Firmware. Leerstring, wenn die USV diesen Parameter nicht unterstützt.
SerialNumber	X	X	Keine	X	X	Seriennummer der USV. Leerstring, wenn die USV diesen Parameter nicht unterstützt.
BatteryLifePercent	X	X	Keine	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	Verbliebene Akkulaufzeit in Minuten. Der Wert ist immer NULL wenn die USV diesen Parameter nicht liefern kann.
eBatteryStatus [▶ 136]	• **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	Akku-Status.
eCommStatus [▶ 137]	X	X	X	X	X	Status der Kommunikation zur USV.

Statusinformation	Beckhoff HID UPS, BAPI v1	Beckhoff P24Vxxxx	Beckhoff CP903x ISA/ PCI-Karte	Beckhoff CX2100-09x 4	APC Back-UPS Pro 280, Smart-UPS 420	Beschreibung
ePowerStatus [▶ 137]	X	X	X	X	X	Status der externen Spannungsversorgung.
nPowerFailCnt	X	X	X	X	*X	Power-Fail-Zähler. Der Zähler wird inkrementiert wenn ein Spannungsausfall vom USV-Service erkannt wurde.

Statusinformation	Beckhoff HID UPS, BAPI v1	Beckhoff P24Vxxxx	Beckhoff CP903x ISA/ PCI-Karte	Beckhoff CX2100-09x4	APC Back-UPS Pro 280, Smart-UPS 420	Beschreibung
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 Interval-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!** • **Service Interval Notify** (Bit 10 gesetzt).Der konfigurierte Akkuwechsel Interval-Service ist abgelaufen. Implementiert in der USV-Softwareversion >= 3.0.0.8.	• **High** (Bit 0 gesetzt) wenn Akku voll geladen. • **Charging** (Bit 3 gesetzt) wenn Akku lädt. • **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. • **Service Interval Notify** (Bit 10 gesetzt) Der konfigurierte Akku-Service-Intervalzeit ist abgelaufen. • **Under Temperature** (Bit 11 gesetzt) wenn Untertemperatur detektiert wurde und	Keine	Akku-Ladestatus-Flags und spezielle Statusinformationen.

Voraussetzungen

USV-Hardware	Zielplattform	Entwicklungsumgebung	Einzubindende SPS-Bibliotheken
• Beckhoff HID UPS; • 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;	PC oder CX	TwinCAT v2.8.0, Build > 745 TwinCAT v2.9.0, Build > 945	TcIoFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib werden automatisch eingebunden)
		TwinCAT v3.0 und höher	Tc2_IoFunctions

4.13 E_BatteryStatus

```
TYPE E_BatteryStatus :
(
    BatteryUnknownStatus,   (*Der Akkustatus ist unbekannt*)
 BatteryOk,           (*Der Akku ist in Ordnung*)
    BatteryReplace      (*Der Akku sollte gewechselt werden*)
);
END_TYPE
```

Voraussetzungen

USV-Hardware	Zielplattform	Entwicklungsumgebung	Einzubindende SPS-Bibliotheken
• Beckhoff HID UPS; • 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	PC oder CX	TwinCAT v2.8.0, Build > 745 TwinCAT v2.9.0, Build > 945	TcIoFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib werden automatisch eingebunden)
		TwinCAT v3.0 und höher	Tc2_IoFunctions

USV-Hardware	Zielplattform	Entwicklungsumgebung	Einzubindende SPS-Bibliotheken
unterstützen und mit dem Windows USV-Dienst konfiguriert werden können;			

4.14 E_UpsCommStatus

```
TYPE E_UpsCommStatus :
(
    UpsCommUnknownStatus,   (*Kommunikationsstatus zur USV-Hardware ist unbekannt*)
 UpsCommOk,          (*Kommunikation zur USV-Hardware ist hergestellt*)
    UpsCommFailed       (*Kommunikation zur USV wurde unterbrochen*)
);
END_TYPE
```

Voraussetzungen

USV-Hardware	Zielplattform	Entwicklungsumgebung	Einzubindende SPS-Bibliotheken
• Beckhoff HID UPS; • 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;	PC oder CX	TwinCAT v2.8.0, Build > 745 TwinCAT v2.9.0, Build > 945	TcIoFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib werden automatisch eingebunden)
		TwinCAT v3.0 und höher	Tc2_IoFunctions

4.15 E_UpsPowerStatus

```
TYPE E_UpsPowerStatus :
(
    PowerUnknownStatus,     (*Der Status der Stromversorgung ist unbekannt*)
 PowerOnLine,        (*Das Gerät befindet sich im Netzbetrieb*)
    PowerOnBattery      (*Stromausfall ist aufgetreten und das Gerät befindet sich im Akkubetrieb*)
);
END_TYPE
```

Voraussetzungen

USV-Hardware	Zielplattform	Entwicklungsumgebung	Einzubindende SPS-Bibliotheken
• Beckhoff HID UPS; • 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;	PC oder CX	TwinCAT v2.8.0, Build > 745 TwinCAT v2.9.0, Build > 945	TcIoFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib werden automatisch eingebunden)
		TwinCAT v3.0 und höher	Tc2_IoFunctions

4.16 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
TwinCAT v2.8.0, Build > 746 TwinCAT v2.9.0, Build > 945	PC (i386)	ads-tec PC	TcIoFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib werden automatisch eingebunden)

4.17 ST_Dpv1ParamAddrEx

TYPE ST_Dpv1ParamAddrEx

ST_Dpv1ParamAddrEx enthält die Daten eines Profidrive-Parameters.

```
TYPE ST_Dpv1ParamAddrEx :
STRUCT
    (* parameter *)
    iAttribute      : USINT;        (* 0x10: Wert; 0x20: Beschreibung; 0x30: Text; 0x80..F0: herstel
lerspezifisch; else: reserviert *)
    iNumOfElements  : USINT;        (* 1..234: Anzahl der Elemente; 0: Spezialfunktionen; else: rese
rviert *)
    iParameterNumber    : UINT;         (* 1..65535: Parameternummer; 0: reserviert *)
    iSubIndex       : UINT;         (* 0..65535: Unterindex *)
    iDataAddr       : UDINT;        (* Adresse des Puffers/Adresse der SPS-Variablen *)
    iDataSize       : UDINT;        (* Größe des Puffers/Größe der SPS-Variablen *)
    eFormat         : E_PD_Datatype;    (* 0x01..0x36: Datentyp; 0x40: ZERO; 0x41: BYTE; 0x42: WORD;
 0x43: DWORD; 0x44: Fehler; else reserviert *)
    iNumOfValues    : UINT;         (* 0..234: Anzahl der WErte; else: reserviert *)
    iErrorValue     : UDINT;        (* DPV1 Fehlerwert *)
    stError         : ST_PD_Dpv1Error;(* DPV1 Fehlerflag, DPV1 Fehleraufzählungstyp *)
END_STRUCT
END_TYPE
```

TYPE 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
```

TYPE ST_PD_Dpv1Error

```
TYPE ST_PD_Dpv1Error :
STRUCT
    bError     : BOOL;
    eErrorId   : E_PD_Dpv1Error;
END_STRUCT
END_TYPE
```

TYPE 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
```

4.18 ST_Dpv1ValueHeaderEx

TYPE ST_Dpv1ValueHeaderEx

ST_Dpv1ValueHeaderEx enthält die Daten eines Parameters im DPV1 Telegramm und seine String-Repräsentation.

```
TYPE ST_Dpv1ValueHeaderEx :
STRUCT
    eFormat         : E_PD_Datatype; (* 0x01..0x36: Datentyp; 0x40: ZERO; 0x41: BYTE; 0x42: WORD; 0x
43: DWORD; 0x44: Fehler; else reserviert *)
    iNumOfValues : USINT;  (* 0..234: Anzahl der Werte; else: reserviert *)
    iOffset      : USINT;  (* Offset im DPV1 Antworttelegramm *)
    iDataLen     : UINT;   (* Datenlänge *)
    strData      : STRING; (* Daten als STRING *)
END_STRUCT
END_TYPE
```

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
```

4.19 ST_PNET_CCDSTS

```
TYPE ST_PNET_CCDSTS :
STRUCT
    iCycleCounter  : UINT;
    iDataState     : USINT;
    iTransferState : USINT;
END_STRUCT
END_TYPE
```

4.20 ST_PNIOConfigRecord

```
TYPE ST_PNIOConfigRecord :
STRUCT
    iRW          : UINT; (* 0: Read, 1: Write *)
    iNumOfAR     : UINT;
    iAPI         : UDINT;
    iSlot        : UINT;
    iSubSlot     : UINT;
    stPNIORecord : ST_PNIORecord;
END_STRUCT
END_TYPE
```

4.21 ST_PNIORecord

```
TYPE ST_PNIORecord :
STRUCT
    iIndex     : UINT;
    iLength    : UINT; (* 0 for READ *)
    iTransfSeq : UINT;
    iAligned   : UINT;
END_STRUCT
END_TYPE
```

4.22 ST_PNIOState

```
TYPE ST_PNIOState :
STRUCT
    bInDataExchange : BOOL; (* bit 0 *)
    bApplRunning    : BOOL; (* bit 2 *)
    bDiagIndicator  : BOOL; (* bit 3 *)
END_STRUCT
END_TYPE
```

4.23 ST_KL1501InData

```
TYPE ST_KL1501InData :
STRUCT
  iStatus     : USINT;
  arrDataIn   : ARRAY[0..1] OF UINT;
END_STRUCT
END_TYPE
```

Struktur zur Verknüpfung im Eingangs-Prozessabbild.

Bei ARM-Systemen kann nicht die Struktur als Ganzes auf das Abbild der Klemme verknüpft werden - die Struktur-Variablen müssen einzeln verknüpft werden.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO-Hardware	Einzubindende SPS-Bibliotheken
TwinCAT v2.11 R3/x64 ab Build 2254	PC/CX	KL1501	TcIoFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib werden automatisch eingebunden)

4.24 ST_KL1501OutData

```
TYPE ST_KL1501OutData :
STRUCT
  iCtrl       : USINT;
  arrDataOut  : ARRAY[0..1] OF UINT;
END_STRUCT
END_TYPE
```

Struktur zur Verknüpfung im Ausgangs-Prozessabbild.

Bei ARM-Systemen kann nicht die Struktur als Ganzes auf das Abbild der Klemme verknüpft werden - die Struktur-Variablen müssen einzeln verknüpft werden.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO-Hardware	Einzubindende SPS-Bibliotheken
TwinCAT v2.11 R3/x64 ab Build 2254	PC/CX	KL1501	TcIoFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib werden automatisch eingebunden)

4.25 ST_KL27x1InData

```
TYPE ST_KL27x1InData :
STRUCT
  iStatus   : USINT;
  iDataIn   : INT;
END_STRUCT
END_TYPE
```

Struktur zur Verknüpfung im Eingangs-Prozessabbild.

Bei ARM-Systemen kann nicht die Struktur als Ganzes auf das Abbild der Klemme verknüpft werden - die Struktur-Variablen müssen einzeln verknüpft werden.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO-Hardware	Einzubindende SPS-Bibliotheken
TwinCAT v2.11 R3/x64 ab Build 2254	PC/CX	KL2751, KL2761	TcIoFunctions.Lib

Entwicklungsumgebung	Zielplattform	IO-Hardware	Einzubindende SPS-Bibliotheken
			(Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib werden automatisch eingebunden)

4.26 ST_KL27x1OutData

```
TYPE ST_KL27x1OutData :
STRUCT
  iCtrl     : USINT;
  iDataOut  : INT;
END_STRUCT
END_TYPE
```

Struktur zur Verknüpfung im Ausgangs-Prozessabbild.

Bei ARM-Systemen kann nicht die Struktur als Ganzes auf das Abbild der Klemme verknüpft werden - die Struktur-Variablen müssen einzeln verknüpft werden.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO-Hardware	Einzubindende SPS-Bibliotheken
TwinCAT v2.11 R3/x64 ab Build 2254	PC/CX	KL2751, KL2761	TcIoFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib werden automatisch eingebunden)

4.27 ST_KL3208InData

```
TYPE ST_KL3208InData :
STRUCT
  iStatus   : USINT;
  iDataIn   : INT;
END_STRUCT
END_TYPE
```

Struktur zur Verknüpfung im Eingangs-Prozessabbild.

Bei ARM-Systemen kann nicht die Struktur als Ganzes auf das Abbild der Klemme verknüpft werden - die Struktur-Variablen müssen einzeln verknüpft werden.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO-Hardware	Einzubindende SPS-Bibliotheken
TwinCAT v2.11 R3/x64 ab Build 2254	PC/CX	KL3208	TcIoFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib;

Entwicklungsumgebung	Zielplattform	IO-Hardware	Einzubindende SPS-Bibliotheken
			TcUtilities.Lib werden automatisch eingebunden)

4.28 ST_KL3208OutData

```
TYPE ST_KL3208OutData :
STRUCT
  iCtrl     : USINT;
  iDataOut  : INT;
END_STRUCT
END_TYPE
```

Struktur zur Verknüpfung im Ausgangs-Prozessabbild.

Bei ARM-Systemen kann nicht die Struktur als Ganzes auf das Abbild der Klemme verknüpft werden - die Struktur-Variablen müssen einzeln verknüpft werden.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO-Hardware	Einzubindende SPS-Bibliotheken
TwinCAT v2.11 R3/x64 ab Build 2254	PC/CX	KL3208	TcIoFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib werden automatisch eingebunden)

4.29 ST_KL320xInData

```
TYPE ST_KL320xInData :
STRUCT
  iStatus   : USINT;
  iDataIn   : INT;
END_STRUCT
END_TYPE
```

Struktur zur Verknüpfung im Eingangs-Prozessabbild.

Bei ARM-Systemen kann nicht die Struktur als Ganzes auf das Abbild der Klemme verknüpft werden - die Struktur-Variablen müssen einzeln verknüpft werden.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO-Hardware	Einzubindende SPS-Bibliotheken
TwinCAT v2.11 R3/x64 ab Build 2254	PC/CX	KL3201, KL3202, KL3204	TcIoFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib werden automatisch eingebunden)

4.30 ST_KL320xOutData

```
TYPE ST_KL320xOutData :
STRUCT
  iCtrl     : USINT;
  iDataOut  : INT;
END_STRUCT
END_TYPE
```

Struktur zur Verknüpfung im Ausgangs-Prozessabbild.

Bei ARM-Systemen kann nicht die Struktur als Ganzes auf das Abbild der Klemme verknüpft werden - die Struktur-Variablen müssen einzeln verknüpft werden.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO-Hardware	Einzubindende SPS-Bibliotheken
TwinCAT v2.11 R3/x64 ab Build 2254	PC/CX	KL3201, KL3202, KL3204	TcIoFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib werden automatisch eingebunden)

4.31 ST_KL3228InData

```
TYPE ST_KL3228InData :
STRUCT
  iStatus   : USINT;
  iDataIn   : INT;
END_STRUCT
END_TYPE
```

Struktur zur Verknüpfung im Eingangs-Prozessabbild.

Bei ARM-Systemen kann nicht die Struktur als Ganzes auf das Abbild der Klemme verknüpft werden - die Struktur-Variablen müssen einzeln verknüpft werden.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO-Hardware	Einzubindende SPS-Bibliotheken
TwinCAT v2.11 R3/x64 ab Build 2254	PC/CX	KL3228	TcIoFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib werden automatisch eingebunden)

4.32 ST_KL3228OutData

```
TYPE ST_KL3228OutData :
STRUCT
  iCtrl     : USINT;
  iDataOut  : INT;
END_STRUCT
END_TYPE
```

Struktur zur Verknüpfung im Ausgangs-Prozessabbild.

Bei ARM-Systemen kann nicht die Struktur als Ganzes auf das Abbild der Klemme verknüpft werden - die Struktur-Variablen müssen einzeln verknüpft werden.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	IO-Hardware	Einzubindende SPS-Bibliotheken
TwinCAT v2.11 R3/x64 ab Build 2254	PC/CX	KL3228	TcIoFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib werden automatisch eingebunden)

5 Anhang

5.1 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

PNU	Datentyp	Zugriff	Kurzbeschrei-bung	SERVOSTAR™ ASCII-Befehl
1208	UINT32	r/w	Zählrichtung	DIR
Lagereglerparameter				
1250	UINT32	r/w	Multiplikator für Geschwindigkeiten Tippen/Ref.	VMUL
1251	UINT32	r/w	Achstyp	POSCNFG
1251	INTEGER32	r/w	In-Position-Fenster	PEINPOS
1253	INTEGER32	r/w	Schleppfehlerfenst er	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	Konfigurationsvaria ble für Softwareschalter	SWCNFG
1267	UINT32	r/w	Konfigurationsvaria ble 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

PNU	Datentyp	Zugriff	Kurzbeschreibung	SERVOSTAR™ ASCII-Befehl
1310	2 * UINT16	wo	Kopieren eines Fahrauftrags	OCOPY
Einrichtbetrieb Lage				
1350	UINT32	r/w	Referenzfahrtart	NREF
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

PNU	Datentyp	Zugriff	Kurzbeschrei-bung	SERVOSTAR™ ASCII-Befehl
1455	INTEGER32	r/w	Hilfsvariable für digitalen Eingang 2	IN2TRIG
1456	INTEGER32	r/w	Hilfsvariable für digitalen Eingang 3	IN3TRIG
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 Eingangsfunktione n	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				

PNU	Datentyp	Zugriff	Kurzbeschrei-bung	SERVOSTAR™ ASCII-Befehl
1550	UINT32	r/w	Konfiguration Bremse	MBRAKE
1551	UINT32	r/w	Motornummer aus Motordatenbank	MNUMBER

Mehr Informationen:
www.beckhoff.de/tx1200

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