

Manual | EN

TX1200

TwinCAT 2 | PLC Library: TcIoFunctions

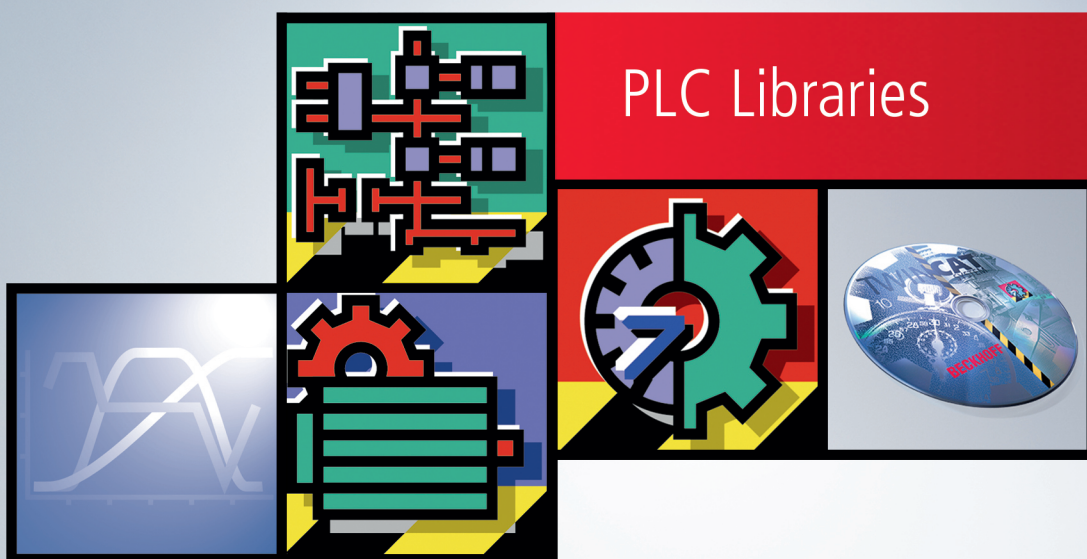


Table of contents

1 Foreword	7
1.1 Notes on the documentation	7
1.2 Safety instructions	8
1.3 Notes on information security.....	9
2 Overview	10
3 I/O functions	13
3.1 IOF_DeviceReset.....	13
3.2 IOF_GetBoxAddrByName	14
3.3 IOF_GetBoxAddrByNameEx.....	15
3.4 IOF_GetBoxCount.....	16
3.5 IOF_GetBoxNameByAddr	17
3.6 IOF_GetBoxNetId.....	18
3.7 IOF_GetDeviceCount.....	19
3.8 IOF_GetDeviceIDByName	20
3.9 IOF_GetDeviceIDs	22
3.10 IOF_GetDeviceName	23
3.11 IOF_GetDeviceNetId	24
3.12 IOF_GetDeviceType	25
3.13 IOF_GetDeviceInfoByName.....	27
3.14 Beckhoff Lightbus.....	28
3.14.1 IOF_LB_BreakLocationTest.....	28
3.14.2 IOF_LB_ParityCheck	29
3.14.3 IOF_LB_ParityCheckWithReset.....	30
3.15 CANopen.....	32
3.15.1 IOF_CAN_Layer2Command	32
3.16 SERCOS	34
3.16.1 IOF_SER_GetPhase.....	34
3.16.2 IOF_SER_SaveFlash.....	35
3.16.3 IOF_SER_ResetErr.....	36
3.16.4 IOF_SER_SetPhase	37
3.16.5 IOF_SER_IDN_Read	38
3.16.6 IOF_SER_IDN_Write	40
3.16.7 IOF_SER_DRIVE_Backup.....	41
3.16.8 IOF_SER_DRIVE_BackupEx.....	43
3.16.9 IOF_SER_DRIVE_Reset	46
3.17 NOV/DP-RAM	47
3.17.1 FB_NovRamReadWrite.....	47
3.17.2 FB_NovRamReadWriteEx	49
3.17.3 FB_GetDPRAMInfo.....	51
3.18 AX200x Profibus	53
3.18.1 FB_AX2000_AXACT.....	54
3.18.2 FB_AX2000_JogMode	56
3.18.3 FB_AX2000_Parameter	57
3.18.4 FB_AX2000_Reference	58

3.18.5	FB_AX200X_Profibus	59
3.19	ASI master terminal.....	61
3.19.1	FB_ASI_Addressing.....	62
3.19.2	FB_ASI_SlaveDiag	63
3.19.3	FB_ASI_ReadParameter	65
3.19.4	FB_ASI_WriteParameter.....	66
3.19.5	FB_ASI_Processdata_digital	67
3.19.6	FB_ASI_ParameterControl	68
3.19.7	FB_ReadInput_analog	69
3.19.8	FB_WriteOutput_analog.....	70
3.20	Profibus DPV1 (Sinamics).....	71
3.20.1	F_CreateDpv1ReadReqPkg : USINT.....	71
3.20.2	FB_Dpv1Read.....	72
3.20.3	F_SplitDpv1ReadResPkg : USINT.....	74
3.20.4	F_CreateDpv1WriteReqPkg : USINT.....	75
3.20.5	FB_Dpv1Write.....	75
3.20.6	F_SplitDpv1WriteResPkg : USINT.....	78
3.21	Profinet DPV1 (Sinamics)	78
3.21.1	F_CreateDpv1ReadReqPkgPNET : USINT	78
3.21.2	FB_Dpv1ReadPNET	79
3.21.3	F_SplitDpv1ReadResPkgPNET : USINT	82
3.21.4	F_CreateDpv1WriteReqPkgPNET : USINT	82
3.21.5	FB_Dpv1WritePNET	83
3.21.6	F_SplitDpv1WriteResPkgPNET : USINT	86
3.22	Beckhoff UPS (configured with Windows UPS Service	87
3.22.1	FB_GetUPSStatus	87
3.23	Busterminal-Configuration.....	89
3.23.1	FB_KL1501Config.....	89
3.23.2	FB_KL27x1Config	93
3.23.3	FB_KL320xConfig	97
3.23.4	FB_KL3208Config.....	100
3.23.5	FB_KL3228Config.....	103
3.24	Third party devices.....	105
3.24.1	Phoenix IBS SC/I-T	105
3.24.2	ads-tec	117
3.25	Errorcodes.....	118
4	Data structures	119
4.1	IODEVICETYPES	119
4.2	E_SercosAttribLen	121
4.3	E_SercosAttribType	121
4.4	ST_SercosParamAttrib	122
4.5	File format of the backup file	122
4.6	ST_PZD_IN.....	125
4.7	ST_PZD_OUT	125
4.8	ST_Parameter_IN	125
4.9	ST_Parameter_OUT	126

4.10	ST_ParameterBuffer	127
4.11	ST_NovRamAddrInfo	127
4.12	ST_UPSStatus	128
4.13	E_BatteryStatus	132
4.14	E_UpsCommStatus.....	133
4.15	E_UpsPowerStatus	133
4.16	ST_AdsTecSysData.....	134
4.17	ST_Dpv1ParamAddrEx.....	135
4.18	ST_Dpv1ValueHeaderEx.....	137
4.19	ST_PNET_CCDSTS	138
4.20	ST_PNIOConfigRecord.....	139
4.21	ST_PNIORecord	139
4.22	ST_PNIOState	139
4.23	ST_KL1501InData.....	139
4.24	ST_KL1501OutData.....	140
4.25	ST_KL27x1InData.....	140
4.26	ST_KL27x1OutData.....	140
4.27	ST_KL3208InData.....	141
4.28	ST_KL3208OutData.....	141
4.29	ST_KL320xInData.....	142
4.30	ST_KL320xOutData.....	142
4.31	ST_KL3228InData.....	143
4.32	ST_KL3228OutData.....	143
5	Appendix.....	144
5.1	AX200x Profibus Parameter Number.....	144

1 Foreword

1.1 Notes on the documentation

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

It is essential that the documentation and the following notes and explanations are followed when installing and commissioning the components.

It is the duty of the technical personnel to use the documentation published at the respective time of each installation and commissioning.

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

Disclaimer

The documentation has been prepared with care. The products described are, however, constantly under development.

We reserve the right to revise and change the documentation at any time and without prior announcement. No claims for the modification of products that have already been supplied may be made on the basis of the data, diagrams and descriptions in this documentation.

Trademarks

Beckhoff®, TwinCAT®, TwinCAT/BSD®, TC/BSD®, EtherCAT®, EtherCAT G®, EtherCAT G10®, EtherCAT P®, Safety over EtherCAT®, TwinSAFE®, XFC®, XTS® and XPlanar® are registered trademarks of and licensed by Beckhoff Automation GmbH.

Other designations used in this publication may be trademarks whose use by third parties for their own purposes could violate the rights of the owners.

Patent Pending

The EtherCAT Technology is covered, including but not limited to the following patent applications and patents:

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

EtherCAT®

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

Copyright

© Beckhoff Automation GmbH & Co. KG, Germany.

The reproduction, distribution and utilization of this document as well as the communication of its contents to others without express authorization are prohibited.

Offenders will be held liable for the payment of damages. All rights reserved in the event of the grant of a patent, utility model or design.

1.2 Safety instructions

Safety regulations

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

Exclusion of liability

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

Personnel qualification

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

Description of symbols

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

DANGER

Serious risk of injury!

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

WARNING

Risk of injury!

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

CAUTION

Personal injuries!

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

NOTE

Damage to the environment or devices

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



Tip or pointer

This symbol indicates information that contributes to better understanding.

1.3 Notes on information security

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

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

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

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

2 Overview

The IO functions library contains function blocks with which services/functions can be carried out on the I/O devices (fieldbus master or slave).

General Device Functions

Name	Description
IOF_DeviceReset [► 13]	Reset an I/O device
IOF_GetBoxAddrByName [► 14]	Find the fieldbus address of the box, knowing the device ID and the box identifier
IOF_GetBoxAddrByNameEx [► 15]	Find the fieldbus address of the box, knowing the device identifier and the box identifier
IOF_GetBoxCount [► 16]	Read the number of boxes
IOF_GetBoxNameByAddr [► 17]	Read the box identifier knowing the fieldbus address of the box and the device ID
IOF_GetBoxNetId [► 18]	Read the box AmsNetId knowing the fieldbus address of the box and the device ID
IOF_GetDeviceCount [► 19]	Read the number of I/O devices.
IOF_GetDeviceIDByName [► 20]	Find the device ID knowing the device identifier
IOF_GetDeviceIDs [► 22]	Read all device IDs
IOF_GetDeviceName [► 23]	Read the device identifier knowing the device ID
IOF_GetDeviceNetId [► 24]	Read the device AmsNetId knowing the device ID
IOF_GetDeviceType [► 25]	Read the device type knowing the device ID
IOF_GetDeviceInfoByName [► 27]	Determine the device ID and the AmsNetId by the device identifier.

Fieldbus-specific and device-specific functions

CANopen

Name	Description
IOF_CAN_Layer2Command [► 32]	Carry out a layer 2 command

Beckhoff Lightbus

Name	Description
IOF_LB_BreakLocationTest [► 28]	Optical fibre ring break location test
IOF_LB_ParityCheck [► 29]	Read parity counter
IOF_LB_ParityCheckWithReset [► 30]	Read and reset parity counter

SERCOS

Name	Description
IOF_SER_GetPhase [► 34]	Read the current phase
IOF_SER_ResetErr [► 36]	Reset the error buffer
IOF_SER_SaveFlash [► 35]	Save parameter in flash
IOF_SER_SetPhase [► 37]	Set the current phase
IOF_SER_IDN_Read [► 38]	Read the Sercos drive parameter
IOF_SER_IDN_Write [► 40]	Write the Sercos drive parameter
IOF_SER_DRIVE_Backup [► 41]	Backup and restore drive parameter to/from file

Name	Description
IOF_SER_DRIVE_BackupEx [▶ 43]	Backup and restore drive parameter to/from file (extended functionality)
IOF_SER_DRIVE_Reset [▶ 46]	Sercos drive reset (parameter S-0-0099 (IDN99))

Profibus DPV1 (Sinamics)

Name	Description
F_CreateDpv1ReadReqPkg [▶ 71]	Creates a DPV1 Read telegram
FB_Dpv1Read [▶ 72]	Sends a DPV1 Read telegram
F_SplitDpv1ReadResPkg [▶ 74]	Splits a DPV1 Read telegram
F_CreateDpv1WriteReqPkg [▶ 75]	Creates a DPV1 Write telegram
FB_Dpv1Write [▶ 75]	Sends a DPV1 Write telegram
F_SplitDpv1WriteResPkg [▶ 78]	Splits a DPV1 Write telegram

Profinet DPV1 (Sinamics)

Name	Description
F_CreateDpv1ReadReqPkgPNET [▶ 78]	Creates a DPV1 Read telegram
FB_Dpv1ReadPNET [▶ 79]	Sends a DPV1 Read telegram
F_SplitDpv1ReadResPkgPNET [▶ 82]	Splits a DPV1 Read telegram
F_CreateDpv1WriteReqPkgPNET [▶ 82]	Creates a DPV1 Write telegram
FB_Dpv1WritePNET [▶ 83]	Sends a DPV1 Write telegram
F_SplitDpv1WriteResPkgPNET [▶ 86]	Splits a DPV1 Write telegram

NOV/DP-RAM

Name	Description
FB_NovRamReadWrite [▶ 47]	Read/Write NOV/DP-RAM
FB_NovRamReadWriteEx [▶ 49]	Read/Write NOV/DP-RAM. Proofs if a special access type to memory is necessary and copies the data in the correct way.(e.g access to CX_9000 NOV-RAM).
FB_GetDPRAMInfo [▶ 51]	Read the NOV/DP-RAM address pointer and configured size.

AX200x Profibus

Function blocks for access to the AX20XX via Profibus: [overview \[▶ 53\]](#).

ASI Master Terminal

Function blocks for access to an ASI Master Terminal: [overview \[▶ 61\]](#).

Beckhoff UPS (under Windows UPS service)

Name	Description
FB_GetUPSStatus [▶ 87]	Read the status of the UPS from the PLC.

Bus Terminal configuration

Name	Description
FB_KL1501Config [▶ 89]	Configuration of a KL1501.

Name	Description
FB_KL27x1Config [► 93]	Configuration of a KL2751 or KL2761.
FB_KL320xConfig [► 97]	Configuration of a KL3201, KL3202 or KL3204.
FB_KL3208Config [► 100]	Configuration of a KL3208.
FB_KL3228Config [► 103]	Configuration of a KL3228.

Third party devices

INTERBUS Phoenix IBS SC/I-T functions

Phoenix IBS SC/I-T functions: [Overview \[► 105\]](#).

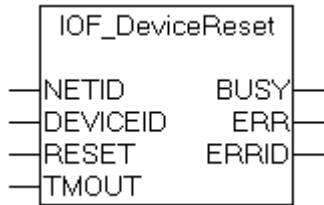
Name	Description
SCIT_ActivateConfiguration [► 107]	Executes the Activate_Configuration command
SCIT_DeactivateConfiguration [► 108]	Executes the Deactivate_Configuration command
SCIT_StartDataTransfer [► 109]	Executes the Start_Data_Transfer command
SCIT_StopDataTransfer [► 110]	Executes the Stop_Data_Transfer command
SCIT_AlarmStop [► 111]	Executes the Alarm_Stop command
SCIT_ControlActiveConfiguration [► 113]	Is used to affect the active configuration of the Interbus devices. This command can be executed in the <i>PAR_READY</i> state as well as when in the <i>ACTIVE</i> or <i>RUN</i> states. Single, dependent and grouped devices can be activated and deactivated in this way
SCIT_GetErrorInfo [► 114]	Returns the error type and error location of an Interbus device after a bus error
SCIT_ConfDevErrAll [► 115]	Confirms peripheral errors of all Interbus devices

Table 1: ads-tec

Name	Description
FB_ReadAdsTecSysData [► 117]	Reads the system and diagnostic data

3 I/O functions

3.1 IOF_DeviceReset



The function block IOF_DeviceReset allows an I/O device (such as a fieldbus card) to be reset. The function corresponds to the I/O reset function in the TwinCAT system menu.°

VAR_INPUT

```

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

NETID: It is possible here to provide the AmsNetId of the TwinCAT computer on which the function to be executed. If it is to be run on the local computer, an empty string can be entered.

DEVICEID: DeviceId specifies the I/O device on which the function is to be executed. The device Ids are specified by the TwinCAT System Manager during the hardware configuration.

RESET: The function block is activated by a positive edge at this input.

TMOUT: States the length of the timeout that may not be exceeded by execution of the ADS command.

VAR_OUTPUT

```

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

BUSY: When the function block is activated this output is set. It remains set until an acknowledgement is received.

ERR: If an ADS error should occur during the transfer of the command, then this output is set once the BUSY output is reset.

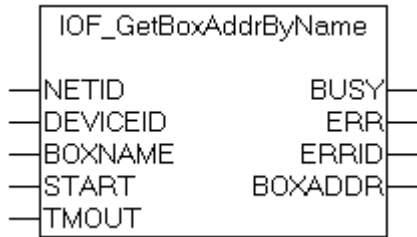
ERRID: Supplies the ADS error number when the ERR output is set.

Requirements

Development environment	Target system type	IO hardware	PLC libraries to include
TwinCAT v2.7.0	PC (i386)	All IO devices	Standard.Lib; PLCSystem.Lib; TcPLCAds.Lib; TcPcIoFunctions.Lib
TwinCAT v2.8.0	PC (i386)	All IO devices	TcIoFunctions.Lib

Development environment	Target system type	IO hardware	PLC libraries to include
			(Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib are included automatically)

3.2 IOF_GetBoxAddrByName



The function block IOF_GetBoxAddrByName permits determination of the fieldbus address of a box (box = slave, module, station) from the box identifier and the device ID. If no fieldbus address is available, the function block returns a physical or a logical address. (In the case of the Beckhoff Lightbus, for instance, it is the physical box number in the optical fibre ring, while in Profibus it is the station address). The box identifier is passed as a string to the function block, and can be specified by the user during the configuration in the TwinCAT System Manager. Internally to the function block an instance of the ADSRDWRT function block is called.

VAR_INPUT

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

NETID: It is possible here to provide the AmsNetId of the TwinCAT computer on which the function to be executed. If it is to be run on the local computer, an empty string can be entered.

DEVICEID: DeviceId specifies the device on which the function is to be executed. The device Ids are specified by the TwinCAT System Manager during the hardware configuration.

BOXNAME: The box identifier as a string.

START: The function block is activated by a positive edge at this input.

TMOUT: States the length of the timeout that may not be exceeded by execution of the ADS command.

VAR_OUTPUT

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

BUSY: When the function block is activated this output is set. It remains set until an acknowledgement is received.

ERR: If an ADS error should occur during the transfer of the command, then this output is set once the BUSY output is reset.

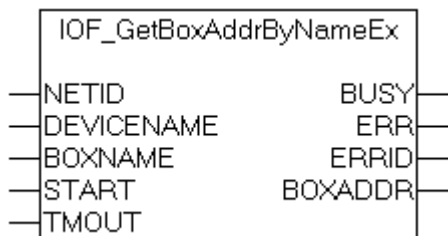
ERRID: Supplies the ADS error number when the ERR output is set.

BOXADDR: The fieldbus address of the box.

Requirements

Development environment	Target system type	IO hardware	PLC libraries to include
TwinCAT v2.7.0	PC (i386)	All IO devices	Standard.Lib; PLCSystem.Lib; TcPLCAds.Lib; TcPcIoFunctions.Lib
TwinCAT v2.8.0	PC (i386)	All IO devices	TcIoFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib are included automatically)

3.3 IOF_GetBoxAddrByNameEx



The function block IOF_GetBoxAddrByNameEx permits determination of the fieldbus address of a box (box = slave, module, station) from the box identifier and the device identifier. If no fieldbus address is available, the function block returns a physical or a logical address. (In the case of the Beckhoff Lightbus, for instance, it is the physical box number in the optical fibre ring, while in Profibus it is the station address). The box identifier and the device identifier are passed as strings to the function block, and can be specified by the user during the configuration in the TwinCAT System Manager. Internally to the function block an instance of the ADSRDWRT function block is called.

VAR_INPUT

```

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

NETID: It is possible here to provide the AmsNetId of the TwinCAT computer on which the function to be executed. If it is to be run on the local computer, an empty string can be entered.

DEVICENAME: The device identifier of an I/O device as a string.

BOXNAME: The box identifier as a string.

START: The function block is activated by a positive edge at this input.

TMOUT: States the length of the timeout that may not be exceeded by execution of the ADS command.

VAR_OUTPUT

```

VAR_OUTPUT
  BUSY : BOOL;
  ERR  : BOOL;
    
```

```

    ERRID      : UDINT;
    BOXADDR    : UINT;
END_VAR

```

BUSY: When the function block is activated this output is set. It remains set until an acknowledgement is received.

ERR: If an ADS error should occur during the transfer of the command, then this output is set once the BUSY output is reset.

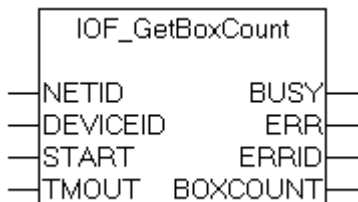
ERRID: Supplies the ADS error number when the ERR output is set.

BOXADDR: The fieldbus address of the box.

Requirements

Development environment	Target system type	IO hardware	PLC libraries to include
TwinCAT v2.7.0	PC (i386)	All IO devices	Standard.Lib; PLCSystem.Lib; TcPLCAds.Lib; TcPcIoFunctions.Lib
TwinCAT v2.8.0	PC (i386)	All IO devices	TcIoFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib are included automatically)

3.4 IOF_GetBoxCount



The function block IOF_GetBoxCount can be used to read the number of an I/O device's active and configured boxes (box = slave, module, station). Internally to the function block an instance of the ADSREAD function block is called.

VAR_INPUT

```

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

```

NETID: It is possible here to provide the AmsNetId of the TwinCAT computer on which the function is to be executed. If it is to be run on the local computer, an empty string can be entered.

DEVICEID: DeviceId specifies the device on which the function is to be executed. The device Ids are specified by the TwinCAT System Manager during the hardware configuration.

START: The function block is activated by a positive edge at this input.

TMOUT: States the length of the timeout that may not be exceeded by execution of the ADS command.

VAR_OUTPUT

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

BUSY: When the function block is activated this output is set. It remains set until an acknowledgement is received.

ERR: If an ADS error should occur during the transfer of the command, then this output is set once the BUSY output is reset.

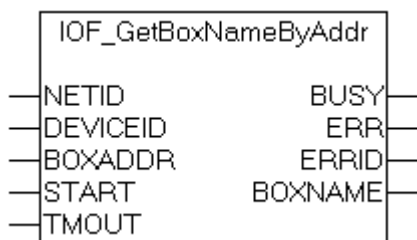
ERRID: Supplies the ADS error number when the ERR output is set.

BOXCOUNT: The number of boxes.

Requirements

Development environment	Target system type	IO hardware	PLC libraries to include
TwinCAT v2.7.0	PC (i386)	All IO devices	Standard.Lib; PLCSystem.Lib; TcPLCAds.Lib; TcPcIoFunctions.Lib
TwinCAT v2.8.0	PC (i386)	All IO devices	TcIoFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib are included automatically)

3.5 IOF_GetBoxNameByAddr



The function block IOF_GetBoxNameByAddr permits determination of the box identifier from the device ID and the fieldbus address of a box (box = slave, module, station). If no fieldbus address is available, a physical or a logical address can be supplied as a fieldbus address to the function block (In the case of the Beckhoff Lightbus, for instance, it is the physical box number in the optical fibre ring). When successful the function block returns the box identifier configured in the TwinCAT System Manager as a string. Internally to the function block an instance of the ADSRDWRT function block is called.

VAR_INPUT

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

NETID: It is possible here to provide the AmsNetId of the TwinCAT computer on which the function is to be executed. If it is to be run on the local computer, an empty string can be entered.

DEVICEID: DeviceId specifies the device on which the function is to be executed. The device Ids are specified by the TwinCAT System Manager during the hardware configuration.

BOXADDR: The fieldbus address of the box.

START: The function block is activated by a positive edge at this input.

TMOUT: States the length of the timeout that may not be exceeded by execution of the ADS command.

VAR_OUTPUT

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

BUSY: When the function block is activated this output is set. It remains set until an acknowledgement is received.

ERR: If an ADS error should occur during the transfer of the command, then this output is set once the BUSY output is reset.

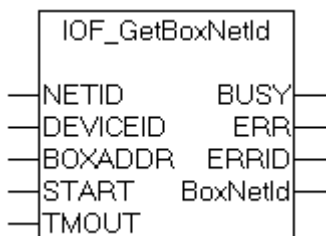
ERRID: Supplies the ADS error number when the ERR output is set.

BOXNAME: The box identifier as a string.

Requirements

Development environment	Target system type	IO hardware	PLC libraries to include
TwinCAT v2.7.0	PC (i386)	All IO devices	Standard.Lib; PLCSystem.Lib; TcPLCAds.Lib; TcPclIoFunctions.Lib
TwinCAT v2.8.0	PC (i386)	All IO devices	TcIoFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib are included automatically)

3.6 IOF_GetBoxNetId



Some boxes (slave modules) can be assigned an AmsNetId during the configuration in the TwinCAT System Manager. The AmsNetId can then be used to execute firmware functions on the box. The function block IOF_GetBoxNetId can be used to determine the AmsNetId using the master's device ID and either the fieldbus address or the logical address in the fieldbus. The device IDs are specified during configuration by the TwinCAT System Manager, and can not be configured by the user. Internally to the function block an instance of the ADSRDWRT function block is called.

VAR_INPUT

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

NETID: It is possible here to provide the AmsNetId of the TwinCAT computer on which the function to be executed. If it is to be run on the local computer, an empty string can be entered.

DEVICEID: The master's device ID.

BOXADDR: The fieldbus address, or the logical address of the box (slave module) whose AmsNetId is to be read.

START: The function block is activated by a positive edge at this input.

TMOUT: States the length of the timeout that may not be exceeded by execution of the ADS command.

VAR_OUTPUT

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

BUSY: When the function block is activated this output is set. It remains set until an acknowledgement is received.

ERR: If an ADS error should occur during the transfer of the command, then this output is set once the BUSY output is reset.

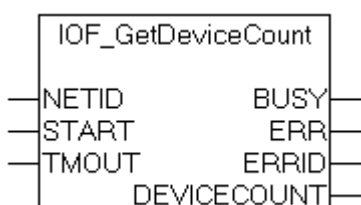
ERRID: Supplies the ADS error number when the ERR output is set.

BoxNetId: The AmsNetId of the box as a string

Requirements

Development environment	Target system type	IO hardware	PLC libraries to include
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 are included automatically)

3.7 IOF_GetDeviceCount



The function block IOF_GetDeviceCount allows the number of configured and active I/O devices to be read. Internally to the function block an instance of the ADSREAD function block is called.

VAR_INPUT

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

NETID: It is possible here to provide the AmsNetId of the TwinCAT computer on which the function to be executed. If it is to be run on the local computer, an empty string can be entered.

START: The function block is activated by a positive edge at this input.

TMOUT: States the length of the timeout that may not be exceeded by execution of the ADS command.

VAR_OUTPUT

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

BUSY: When the function block is activated this output is set. It remains set until an acknowledgement is received.

ERR: If an ADS error should occur during the transfer of the command, then this output is set once the BUSY output is reset.

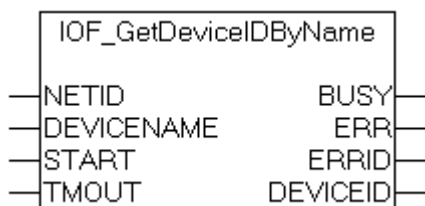
ERRID: Supplies the ADS error number when the ERR output is set.

DEVICECOUNT: The number of I/O devices.

Requirements

Development environment	Target system type	IO hardware	PLC libraries to include
TwinCAT v2.7.0	PC (i386)	All IO devices	Standard.Lib; PLCSystem.Lib; TcPLCAds.Lib; TcPcIoFunctions.Lib
TwinCAT v2.8.0	PC (i386)	All IO devices	TcIoFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib are included automatically)

3.8 IOF_GetDeviceIDByName



The function block IOF_GetDeviceIDByName allows the device ID of an I/O device to be determined from the device identifier. When successful, the function block returns the device ID specified by the TwinCAT System Manager during the configuration. The device IDs can not be configured by the user. Internally to the function block an instance of the ADSRDWRT function block is called.

VAR_INPUT

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

NETID: It is possible here to provide the AmsNetId of the TwinCAT computer on which the function to be executed. If it is to be run on the local computer, an empty string can be entered.

DEVICENAME: The device identification of an I/O device.

START: The function block is activated by a positive edge at this input.

TMOUT: States the length of the timeout that may not be exceeded by execution of the ADS command.

VAR_OUTPUT

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

BUSY: When the function block is activated this output is set. It remains set until an acknowledgement is received.

ERR: If an ADS error should occur during the transfer of the command, then this output is set once the BUSY output is reset.

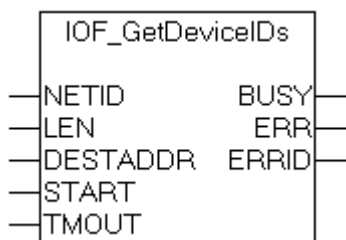
ERRID: Supplies the ADS error number when the ERR output is set.

DEVICEID: The device ID of an I/O device.

Requirements

Development environment	Target system type	IO hardware	PLC libraries to include
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 are included automatically)

3.9 IOF_GetDeviceIDs



The function block IOF_GetDeviceIDs allows the device IDs of all the configured and active I/O devices to be read into a data buffer. The data buffer can be defined as an array of word variables. When successful, the function block returns the total number of the device IDs that exist in the first data word, while the remaining data words contain the corresponding device IDs of the individual I/O devices. The device IDs are specified during configuration by the TwinCAT System Manager, and can not be configured by the user. Internally to the function block an instance of the ADSREAD function block is called.

VAR_INPUT

```

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

```

NETID: It is possible here to provide the AmsNetId of the TwinCAT computer on which the function to be executed. If it is to be run on the local computer, an empty string can be entered.

LEN: The length in bytes of the data buffer into which the device IDs are to be read.

DESTADDR: Address of the data buffer into which the device IDs are to be read.

START: The function block is activated by a positive edge at this input.

TMOUT: States the length of the timeout that may not be exceeded by execution of the ADS command.

VAR_OUTPUT

```

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

```

BUSY: When the function block is activated this output is set. It remains set until an acknowledgement is received.

ERR: If an ADS error should occur during the transfer of the command, then this output is set once the BUSY output is reset.

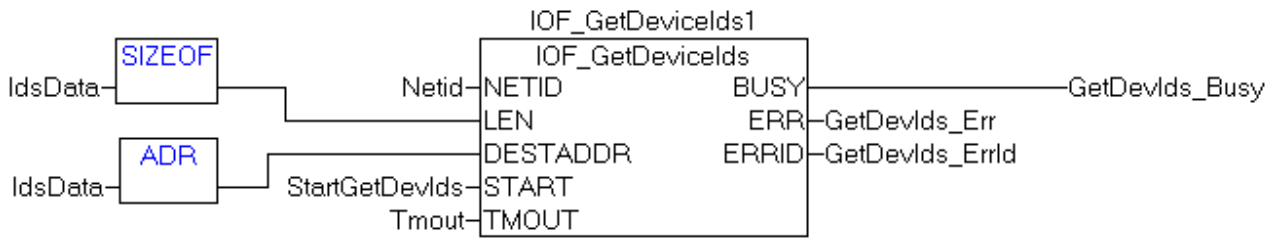
ERRID: Supplies the ADS error number when the ERR output is set.

Example of a call in FBD:

```

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

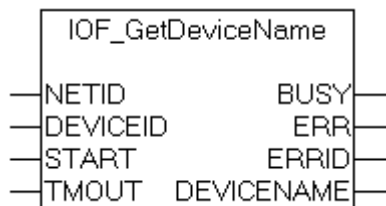
```



Requirements

Development environment	Target system type	IO hardware	PLC libraries to include
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 are included automatically)

3.10 IOF_GetDeviceName



The function block IOF_GetDeviceName allows the device identifier of an I/O device to be read. Internally to this function block an instance of the ADSREAD function block is called. The device identifier can be specified by the user during the configuration in the TwinCAT System Manager. When the system starts up it is then sent as a string to the I/O drivers, and can be read by the ADS commands. The I/O device whose device identifier is to be read is specified by the input variable **DEVICEID**.

VAR_INPUT

```

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

NETID: It is possible here to provide the AmsNetId of the TwinCAT computer on which the function to be executed. If it is to be run on the local computer, an empty string can be entered.

DEVICEID: DeviceId specifies the device on which the function is to be executed. The device Ids are specified by the TwinCAT System Manager during the hardware configuration.

START: The function block is activated by a positive edge at this input.

TMOUT: States the length of the timeout that may not be exceeded by execution of the ADS command.

VAR_OUTPUT

```

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

```

BUSY: When the function block is activated this output is set. It remains set until an acknowledgement is received.

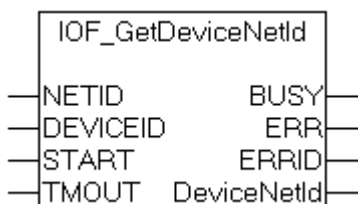
ERR: If an ADS error should occur during the transfer of the command, then this output is set once the BUSY output is reset.

ERRID: Supplies the ADS error number when the ERR output is set.

DEVICENAME: The device identifier of the I/O device.

Requirements

Development environment	Target system type	IO hardware	PLC libraries to include
TwinCAT v2.7.0	PC (i386)	All IO devices	Standard.Lib; PLCSystem.Lib; TcPLCAds.Lib; TcPcIoFunctions.Lib
TwinCAT v2.8.0	PC (i386)	All IO devices	TcIoFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib are included automatically)

3.11 IOF_GetDeviceNetId

Some I/O devices can be assigned an AmsNetId during the configuration in the TwinCAT System Manager (e.g. the FC310x Profibus card or the CP9030 card). The AmsNetId can then be used to execute firmware functions on the device. The function block IOF_GetDeviceNetId can be used to determine the AmsNetId by means of the device ID (DEVICEID). The device IDs are specified during configuration by the TwinCAT System Manager, and can not be configured by the user. Internally to the function block an instance of the ADSREAD function block is called.

VAR_INPUT

```

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

```

NETID: It is possible here to provide the AmsNetId of the TwinCAT computer on which the function is to be executed. If it is to be run on the local computer, an empty string can be entered.

DEVICEID: The ID of the device whose AmsNetId is to be read

START: The function block is activated by a positive edge at this input.

TMOUT: States the length of the timeout that may not be exceeded by execution of the ADS command.

VAR_OUTPUT

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

BUSY: When the function block is activated this output is set. It remains set until an acknowledgement is received.

ERR: If an ADS error should occur during the transfer of the command, then this output is set once the BUSY output is reset.

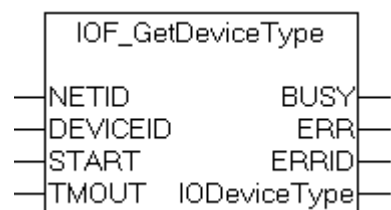
ERRID: Supplies the ADS error number when the ERR output is set.

DeviceNetId: The AmsNetId of the device as a string

Requirements

Development environment	Target system type	IO hardware	PLC libraries to include
TwinCAT v2.7.0	PC (i386)	All IO devices with NetID	Standard.Lib; PLCSystem.Lib; TcPLCAds.Lib; TcPcIoFunctions.Lib
TwinCAT v2.8.0	PC (i386)	All IO devices with NetID	TcIoFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib are included automatically)

3.12 IOF_GetDeviceType



The function block "IOF_GetDeviceType" can be used to obtain the device type from the device ID. The device IDs are specified during configuration by the TwinCAT System Manager, and can not be configured by the user. Internally to the function block an instance of the ADSREAD function block is called.

VAR_INPUT

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

NETID: It is possible here to provide the AmsNetId of the TwinCAT computer on which the function is to be executed. If it is to be run on the local computer, an empty string can be entered.

DEVICEID: The ID of the device whose device type is to be read

START: The function block is activated by a positive edge at this input.

TMOUT: States the length of the timeout that may not be exceeded by execution of the ADS command.

VAR_OUTPUT

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

BUSY: When the function block is activated this output is set. It remains set until an acknowledgement is received.

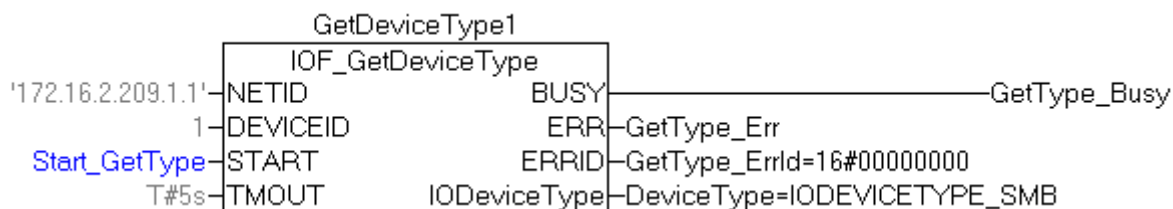
ERR: If an ADS error should occur during the transfer of the command, then this output is set once the BUSY output is reset.

ERRID: Supplies the ADS error number when the ERR output is set.

IODeviceType: The [device type constant](#) [► 119].

Example of a call in FBD:

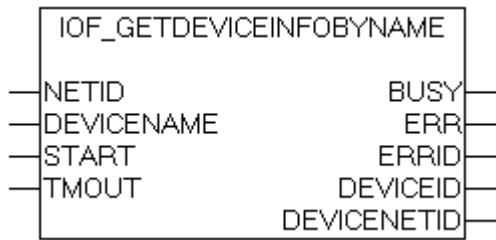
```
PROGRAM MAIN
VAR
  GetDeviceType1 : IOF_GetDeviceType;
  Start_GetType  : BOOL;
  GetType_Busy   : BOOL;
  GetType_Err    : BOOL;
  GetType_ErrId  : UDINT;
  DeviceType     : IODEVICETYPES;
END_VAR
```



Requirements

Development environment	Target system type	IO hardware	PLC libraries to include
TwinCAT v2.7.0	PC (i386)	All IO devices	Standard.Lib; PLCSystem.Lib; TcPLCAds.Lib; TcPcIoFunctions.Lib
TwinCAT v2.8.0	PC (i386)	All IO devices	TcIoFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib are included automatically)

3.13 IOF_GetDeviceInfoByName



The function block IOF_GetDeviceInfoByName allows the device ID and AmsNetId (network address) of an I/O device to be determined from the device identifier. The device IDs can not be configured by the user.

VAR_INPUT

```

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

NETID: It is possible here to provide the AmsNetId of the TwinCAT computer on which the function to be executed. If it is to be run on the local computer, an empty string can be entered.

DEVICENAME: The device identification of an I/O device.

START: The function block is activated by a positive edge at this input.

TMOUT: States the length of the timeout that may not be exceeded by execution of the ADS command.

VAR_OUTPUT

```

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

BUSY: When the function block is activated this output is set. It remains set until an acknowledgement is received.

ERR: If an ADS error should occur during the transfer of the command, then this output is set once the BUSY output is reset.

ERRID: Supplies the ADS error number when the ERR output is set.

DEVICEID: The device ID of an I/O device.

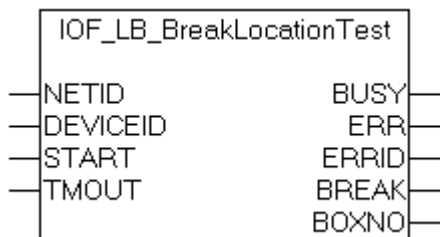
DEVICENETID: The AmsNetid (network address) of an I/O device.

Requirements

Development environment	Target system type	IO hardware	PLC libraries to include
TwinCAT v2.8.0 Builds > 737	PC (i386)	All IO devices	TcIoFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib are included automatically)

3.14 Beckhoff Lightbus

3.14.1 IOF_LB_BreakLocationTest



The function block IOF_LB_BreakLocationTest can be used to carry out a test for a break location in a Beckhoff Lightbus optical fibre ring, and to locate any break that may be found. Internally, an instance of the ADSREAD function block is called. If no break location is detected during the test, the output variable **BOXNO** returns the current number of Lightbus modules in the ring. If a break location is detected before the Nth module in front of the receiver input, the **BREAK** flag is set and the module number is provided via the output variable **BOXNO**. If the **BOXNO** variable returns a value of **0xFF** the break location is situated immediately in front of the receiver input, and can not be located.

VAR_INPUT

```

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

NETID: It is possible here to provide the AmsNetId of the TwinCAT computer on which the function to be executed. If it is to be run on the local computer, an empty string can be entered.

DEVICEID: DeviceId specifies the device on which the function is to be executed. The device Ids are specified by the TwinCAT System Manager during the hardware configuration.

START: The function block is activated by a positive edge at this input.

TMOUT: States the length of the timeout that may not be exceeded by execution of the ADS command.

VAR_OUTPUT

```

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

BUSY: When the function block is activated this output is set. It remains set until an acknowledgement is received.

ERR: If an ADS error should occur during the transfer of the command, then this output is set once the BUSY output is reset.

ERRID: Supplies the ADS error number when the ERR output is set.

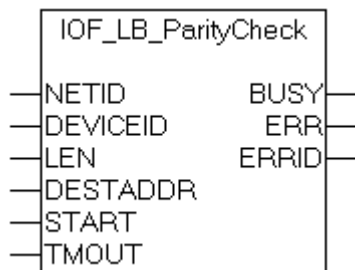
BREAK: This flag is set if a break location is detected in the optical fibre ring.

BOXNO: The module number before the receiver input in front of which the break location has been detected.

Requirements

Development environment	Target system type	IO hardware	PLC libraries to include
TwinCAT v2.7.0	PC (i386)	Beckhoff Lightbus Master: C1220 ISA; FC200x PCI	Standard.Lib; PLCSystem.Lib; TcPLCAds.Lib; TcPloFunctions.Lib
TwinCAT v2.8.0	PC (i386)	Beckhoff Lightbus Master: C1220 ISA; FC200x PCI	TcloFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib are included automatically)

3.14.2 IOF_LB_ParityCheck



The function block IOF_LB_ParityCheck allows the parity error counter of Beckhoff Lightbus modules (e.g. the BK2000) to be read. In contrast to the function block IOF_LB_ParityCheckWithReset [▶ 30], the counter states are not reset. Internally, an instance of the ADSREAD function block is called. The master maintains an 8-bit error counter for each module. These counters work without overflow. A maximum of **256** bytes of data, and therefore **256** counters, can be read. The number of error counters to be read is specified by the input variables **LEN** and **DESTADDR**. If, for instance, there are only 5 modules in the ring, then the **DESTADDR** parameter can be supplied with the address of a data buffer of 5 bytes, and the **LEN** parameter can be supplied with the value 5.

VAR_INPUT

```

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

NETID: It is possible here to provide the AmsNetId of the TwinCAT computer on which the function to be executed. If it is to be run on the local computer, an empty string can be entered.

DEVICEID: DeviceId specifies the device on which the function is to be executed. The device Ids are specified by the TwinCAT System Manager during the hardware configuration.

LEN: The length in bytes of the data to be read.

DESTADDR: The address of the data buffer into which the parity data is to be written.

START: The function block is activated by a positive edge at this input.

TMOUT: States the length of the timeout that may not be exceeded by execution of the ADS command.

VAR_OUTPUT

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

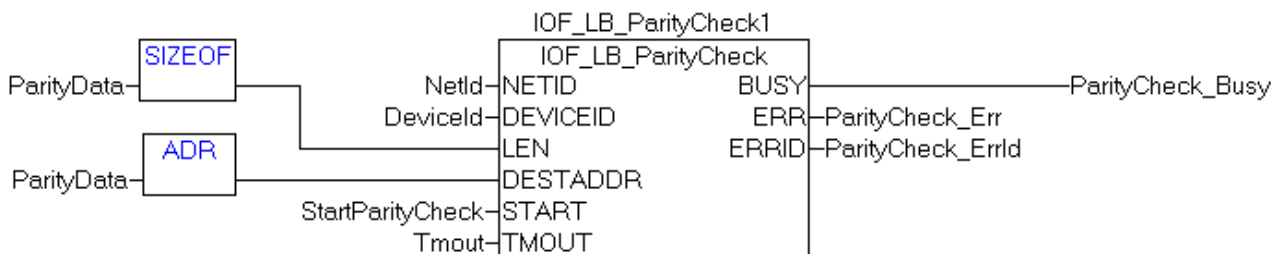
BUSY: When the function block is activated this output is set. It remains set until an acknowledgement is received.

ERR: If an ADS error should occur during the transfer of the command, then this output is set once the BUSY output is reset.

ERRID: Supplies the ADS error number when the ERR output is set.

Example of a call in FBD:

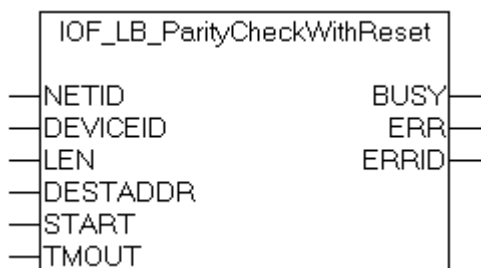
```
IOF_LB_ParityCheck1 : IOF_LB_ParityCheck;
ParityData           : ARRAY[1..256] OF BYTE;
StartParityCheck     : BOOL;
ParityCheck_Busy     : BOOL;
ParityCheck_Err      : BOOL;
ParityCheck_ErrId    : UDINT;
```



Requirements

Development environment	Target system type	IO hardware	PLC libraries to include
TwinCAT v2.7.0	PC (i386)	Beckhoff Lightbus Master: C1220 ISA; FC200x PCI	Standard.Lib; PLCSystem.Lib; TcPLCAds.Lib; TcPcIoFunctions.Lib
TwinCAT v2.8.0	PC (i386)	Beckhoff Lightbus Master: C1220 ISA; FC200x PCI	TcIoFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib are included automatically)

3.14.3 IOF_LB_ParityCheckWithReset



The function block IOF_LB_ParityCheckWithReset allows the parity error counter of Beckhoff Lightbus modules (e.g. the BK2000) to be read. The counters are then reset. Internally, an instance of the ADSREAD function block is called. The master maintains an 8-bit error counter for each module. These counters work without overflow. A maximum of **256** bytes of data, and therefore **256** counters, can be read. The number of error counters to be read is specified by the input variables **LEN** and **DESTADDR**. If, for instance, there are only 5 modules in the ring, then the **DESTADDR** parameter can be supplied with the address of a data buffer of 5 bytes, and the **LEN** parameter can be supplied with the value 5.

VAR_INPUT

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

NETID: It is possible here to provide the AmsNetId of the TwinCAT computer on which the function to be executed. If it is to be run on the local computer, an empty string can be entered.

DEVICEID: DeviceId specifies the device on which the function is to be executed. The device Ids are specified by the TwinCAT System Manager during the hardware configuration.

LEN: The length in bytes of the data to be read.

DESTADDR: The address of the data buffer into which the parity data is to be written.

START: The function block is activated by a positive edge at this input.

TMOUT: States the length of the timeout that may not be exceeded by execution of the ADS command.

VAR_OUTPUT

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

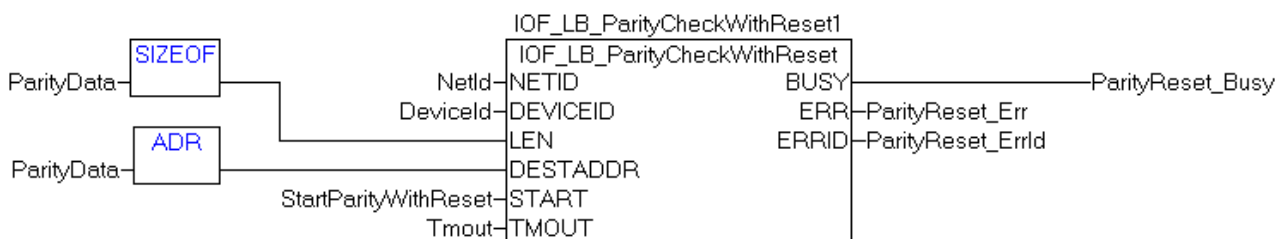
BUSY: When the function block is activated this output is set. It remains set until an acknowledgement is received.

ERR: If an ADS error should occur during the transfer of the command, then this output is set once the BUSY output is reset.

ERRID: Supplies the ADS error number when the ERR output is set.

Example of a call in FBD:

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

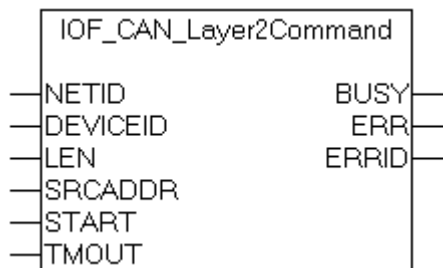


Requirements

Development environment	Target system type	IO hardware	PLC libraries to include
TwinCAT v2.7.0	PC (i386)	Beckhoff Lightbus Master: C1220 ISA; FC200x PCI	Standard.Lib; PLCSystem.Lib; TcPLCAds.Lib; TcPloFunctions.Lib
TwinCAT v2.8.0	PC (i386)	Beckhoff Lightbus Master: C1220 ISA; FC200x PCI	TcloFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib are included automatically)

3.15 CANopen

3.15.1 IOF_CAN_Layer2Command



The function block IOF_CAN_Layer2Command permits a command 10 bytes long to be sent to layer 2 of a CAN master. Internally, an instance of the ADSWRITE function block is called.

VAR_INPUT

```

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

```

NETID: It is possible here to provide the AmsNetId of the TwinCAT computer on which the function to be executed. If it is to be run on the local computer, an empty string can be entered.

DEVICEID: DeviceId specifies the device (CAN master) on which the function is to be executed. The device Ids are specified by the TwinCAT System Manager during the hardware configuration.

LEN: The length of the layer 2 command in bytes.

SRCADDR: The address of the first data word in the CAN layer 2 command.°

START: The function block is activated by a positive edge at this input.

TMOUT: States the length of the timeout that may not be exceeded by execution of the ADS command.

VAR_OUTPUT

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

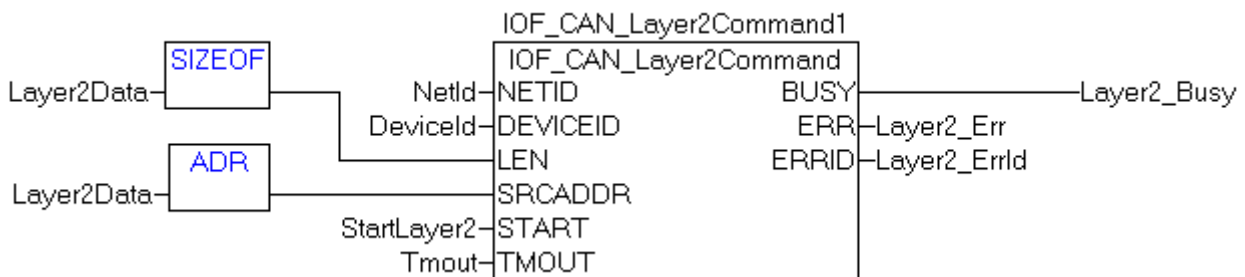
BUSY: When the function block is activated this output is set. It remains set until an acknowledgement is received.

ERR: If an ADS error should occur during the transfer of the command, then this output is set once the BUSY output is reset.

ERRID: Supplies the ADS error number when the ERR output is set.

Example of a call in FBD:

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

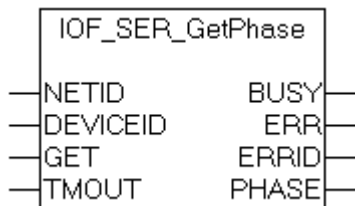


Requirements

Development environment	Target system type	IO-Hardware	PLC libraries to include
TwinCAT v2.7.0	PC (i386)	HILSCHER CIF3xx COM master card	Standard.Lib; PLCSystem.Lib; TcPLCAds.Lib; TcPcIoFunctions.Lib
TwinCAT v2.8.0	PC (i386)	HILSCHER CIF3xx COM master card	TcIoFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib are included automatically)

3.16 SERCOS

3.16.1 IOF_SER_GetPhase



The function block "IOF_SER_GetPhase" allows the current communication phase of the SERCOS ring to be determined. The communication phase can have a value from 0 to 4. Internally, an instance of the ADSREAD function block is called.

VAR_INPUT

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

NETID: It is possible here to provide the AmsNetId of the TwinCAT computer on which the ADS command is to be executed. If it is to be run on the local computer, an empty string can be entered.

DEVICEID: The DeviceId is used to specify the SERCOS master whose communication phase is to be determined. The device Ids are specified by the TwinCAT System Manager during the hardware configuration.

GET: The function block is activated by a positive edge at this input.

TMOUT: States the length of the timeout that may not be exceeded by execution of the ADS command.

VAR_OUTPUT

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

BUSY: When the function block is activated this output is set. It remains set until an acknowledgement is received.

ERR: If an ADS error should occur during the transfer of the command, then this output is set once the BUSY output is reset.

ERRID: Supplies the ADS error number when the ERR output is set.

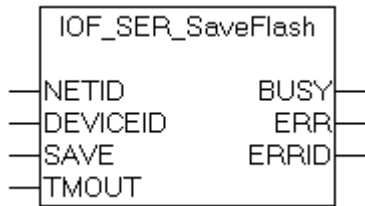
PHASE: The current communication phase in the SERCOS ring.

Requirements

Development environment	Target system type	IO hardware	PLC libraries to include
TwinCAT v2.7.0	PC (i386)	Sercans SCS-P ISA; Sercans SCS-P PCI	Standard.Lib; PLCSystem.Lib; TcPLCAds.Lib; TcPcIoFunctions.Lib

Development environment	Target system type	IO hardware	PLC libraries to include
TwinCAT v2.8.0	PC (i386)	Sercans SCS-P ISA; Sercans SCS-P PCI	TcIoFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib are included automatically)

3.16.2 IOF_SER_SaveFlash



The "IOF_SER_SaveFlash" function block checks the system parameters located in the DPRAM memory. If no error is present it activates them and saves them in the EEPROM. The function block can adjust system parameters in the EEPROM of the controller to suit the application.



The EEPROM can be re-written up to 100,000 times. The PLC should not automatically activate this function block, but it should be called by the user for specific purposes.

VAR_INPUT

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

NETID: It is possible here to provide the AmsNetId of the TwinCAT computer on which the ADS command is to be executed. If it is to be run on the local computer, an empty string can be entered.

DEVICEID: The DeviceId is used to specify the SERCOS master whose system parameters are to be stored. The device Ids are specified by the TwinCAT System Manager during the hardware configuration.

SAVE: The function block is activated by a positive edge at this input.

TMOUT: States the length of the timeout that may not be exceeded by execution of the ADS command.

VAR_OUTPUT

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

BUSY: When the function block is activated this output is set. It remains set until an acknowledgement is received.

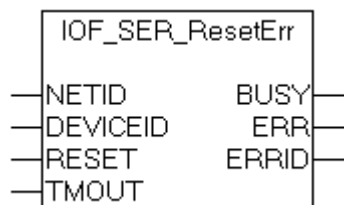
ERR: If an ADS error should occur during the transfer of the command, then this output is set once the BUSY output is reset.

ERRID: Supplies the ADS error number when the ERR output is set.

Requirements

Development environment	Target platform	IO Hardware	PLC libraries to include
TwinCAT v2.7.0	PC (i386)	Sercans SCS-P ISA; Sercans SCS-P PCI	Standard.Lib; PLCSystem.Lib; TcPLCAds.Lib; TcPloFunctions.Lib
TwinCAT v2.8.0	PC (i386)	Sercans SCS-P ISA; Sercans SCS-P PCI	TcloFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib are included automatically)

3.16.3 IOF_SER_ResetErr



The function block "IOF_SER_ResetErr" clears the following errors from a SERCOS master:

- the errors in the existing drives,
- the diagnostic status in the diagnostics channel of the existing drives, and
- the system errors.

Internally, an instance of the ADSWRITE function block is called.

VAR_INPUT

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

NETID: It is possible here to provide the AmsNetId of the TwinCAT computer on which the ADS command is to be executed. If it is to be run on the local computer, an empty string can be entered.

DEVICEID: The DeviceId is used to specify the SERCOS master whose errors are to be cleared. The device Ids are specified by the TwinCAT System Manager during the hardware configuration.

RESET: The function block is activated by a positive edge at this input.

TMOUT: States the length of the timeout that may not be exceeded by execution of the ADS command.

VAR_OUTPUT

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

BUSY: When the function block is activated this output is set. It remains set until an acknowledgement is received.

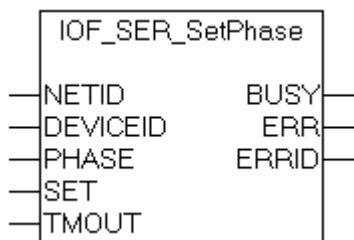
ERR: If an ADS error should occur during the transfer of the command, then this output is set once the BUSY output is reset.

ERRID: Supplies the ADS error number when the ERR output is set.

Requirements

Development environment	Target system type	IO hardware	PLC libraries to include
TwinCAT v2.7.0	PC (i386)	Sercans SCS-P ISA; Sercans SCS-P PCI	Standard.Lib; PLCSystem.Lib; TcPLCAds.Lib; TcPcIoFunctions.Lib
TwinCAT v2.8.0	PC (i386)	Sercans SCS-P ISA; Sercans SCS-P PCI	TcloFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib are included automatically)

3.16.4 IOF_SER_SetPhase



The function block "IOF_SER_SetPhase" allows the phase start-up in the SERCOS ring to be carried out up to a specific value. Internally, an instance of the ADSWRITE function block is called.

VAR_INPUT

```

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

NETID: It is possible here to provide the AmsNetId of the TwinCAT computer on which the ADS command is to be executed. If it is to be run on the local computer, an empty string can be entered.

DEVICEID: The DeviceId is used to specify the SERCOS master whose communication phase is to be set. The device Ids are specified by the TwinCAT System Manager during the hardware configuration.

PHASE: The communication phase in the SERCOS ring that is to be set.

SET: The function block is activated by a positive edge at this input.

TMOUT: States the length of the timeout that may not be exceeded by execution of the ADS command.

VAR_OUTPUT

```

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

BUSY: When the function block is activated this output is set. It remains set until an acknowledgement is received.

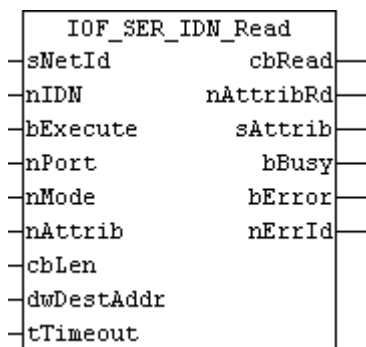
ERR: If an ADS error should occur during the transfer of the command, then this output is set once the BUSY output is reset.

ERRID: Supplies the ADS error number when the ERR output is set.

Requirements

Development environment	Target system type	IO hardware	PLC libraries to include
TwinCAT v2.7.0	PC (i386)	Sercans SCS-P ISA; Sercans SCS-P PCI	Standard.Lib; PLCSystem.Lib; TcPLCAds.Lib; TcPcIoFunctions.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 are included automatically)

3.16.5 IOF_SER_IDN_Read



The function block "IOF_SER_IDN_Read" allows to read an S- or P-Parameter from a SERCOS drive. The data type and size are determined automatically by reading the parameter attribute. Internally, an instance of the ADSREAD function block is called.

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: It is possible here to provide the AmsNetId of the TwinCAT computer on which the ADS command is to be executed. If it is to be run on the local computer, an empty string can be entered.

nIDN: The nIDN variable is used to specify the SERCOS parameter which has to be read. For standard S-Parameters nIDN has to be between 0 and 32767, for manufacturer specific P-Parameters nIDN has to be between 32768 and 65535.

bExecute: The function block is activated by a positive edge at this input.

nPort: The port number nPort is specified by the TwinCAT System Manager during the hardware configuration.

nAttrib: Attribute of the parameter, if known. If nAttrib = 0 then IOF_SER_IDN_Write reads the parameter attribute from the drive, before it writes the parameter value to the drive.

cbLen: Maximum length of data buffer.

dwDestAddr: Address of data buffer.

nMode: The mode nMode determines which data of the parameter nIDN should be read.

nMode = 0: Value

nMode = 2: Name

nMode = 3: Attribute (is always read to determine data type and size, unless nAttrib is <> 0)

nMode = 4: Unit (not available for all parameters)

nMode = 5: Minimum (not available for all parameters)

nMode = 6: Maximum (not available for all parameters)

tTimeout: States the length of the timeout that may not be exceeded by execution of the ADS command.

VAR_OUTPUT

```
VAR_OUTPUT
  cbRead      : UDINT;
  nAttribRd   : DWORD;
  sAttrib     : ST_SercosParamAttrib;
  bBusy      : BOOL;
  bError     : BOOL;
  nErrId     : UDINT;
END_VAR
```

cbRead: Number of bytes read and copied to dwDestAddr.

nAttribRd: contains the attribute of the parameter and can be used if saved as attribute (nAttrib) for the next parameter access with read or write

sAttrib: contains the attribute of the parameter [►_122] split to separate bits in to the structure ST_SercosParamAttrib

bBusy: When the function block is activated this output is set. It remains set until and acknowledgement is received.

bError: If an ADS error should occur during the transfer of the command, then this output is set once the bBusy output is reset.

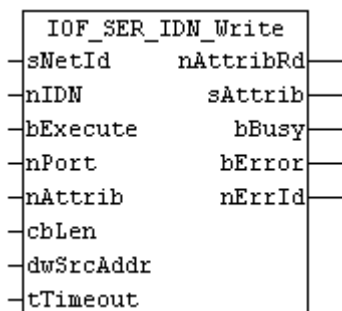
nErrId: Supplies the ADS error number or function block specific error code when the bError output is set.

Function block specific error codes	Description
0x1003	Wrong parameter mode
0x1004	Data parameter wrong value size

Requirements

Development environment	Target system type	IO hardware	PLC libraries to include
TwinCAT v2.8.0 Build > 735	PC (i386)	Sercans SCS-P ISA; Sercans SCS-P PCI; Beckhoff FC750x PCI	TcloFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib are included automatically)

3.16.6 IOF_SER_IDN_Write



The function block "IOF_SER_IDN_Write" allows to WRITE a value to an S- or P-Parameter of a SERCOS drive. The data type and size are determined automatically from the drive.

Internally, an instance of the ADSREAD function block and an instance of the ADSWRITE function block are called.

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: It is possible here to provide the AmsNetId of the TwinCAT computer on which the ADS command is to be executed. If it is to be run on the local computer, an empty string can be entered.

nIDN: The nIDN is used to specify the SERCOS parameter which has to be written. For standard S-Parameters nIDN has to be between 0 and 32767, for manufacturer specific P-Parameters nIDN has to be between 32768 and 65535.

bExecute: The function block is activated by a positive edge at this input.

nPort: The port number nPort is specified by the TwinCAT System Manager during the hardware configuration.

nAttrib: Attribute of the parameter, if known. If nAttrib = 0 then IOF_SER_IDN_Write reads the parameter attribute from the drive, before it writes the parameter value to the drive.

cbLen: Length of data buffer, that contains the value.

dwSrcAddr: Address of data buffer, that contains the value.

tTimeout: States the length of the timeout that may not be exceeded by execution of the ADS command.

VAR_OUTPUT

```
VAR_OUTPUT
  nAttribRd  : DWORD;
  sAttrib    : ST_SercosParamAttrib;
  bBusy      : BOOL;
  bError     : BOOL;
  nErrId     : UDINT;
END_VAR
```

nAttribRd: contains the attribute of the parameter and can be used if saved as attribute (nAttrib) for the next parameter access with read or write

sAttrib: contains the actual attribute of the parameter [►_122] split to separate bits in to the structure ST_SercosParamAttrib

bBusy: When the function block is activated this output is set. It remains set until an acknowledgement is received.

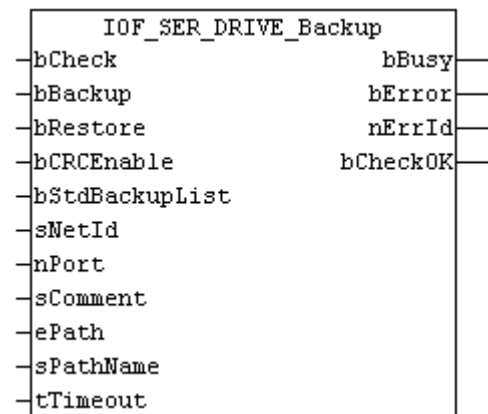
bError: If an ADS error should occur during the transfer of the command, then this output is set once the bBusy output is reset.

nErrId: Supplies the ADS error number when the bError output is set.

Requirements

Development environment	Target system type	IO hardware	PLC libraries to include
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 are included automatically)

3.16.7 IOF_SER_DRIVE_Backup



The function block "IOF_SER_DRIVE_Backup" allows to create and restore a binary backup file [▶ 122] from within the PLC containing all S- and P-Parameters of the SERCOS drive specified in the parameter list IDN192. The parameter list IDN192 contains a list of all backup parameters. Backup and restore require SERCOS to be in parameter mode (phase 2).

If bStdBackupList = TRUE (default setting) then parameter list IDN192 is used for the backup, otherwise the parameter list IDN17, the list of all parameters. Restore requires a backup file that was created with the standard backup list (IDN192), since some parameters of the parameter list IDN17 are write protected.

Backup and restore are creating a CRC16-CCITT and a 16 bit check sum and store this per default in IDN142, if available.

Internally, instances of the function blocks IOF_SER_IDN_Read, IOF_SER_IDN_Write, FB_FileOpen, FB_FileClose, FB_FileRead, FB_FileWrite are called.

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: The check of the CRC and the checksum is activated by a positive edge at this input. The CRC and the checksum are stored in IDN142 and persistent in the IOF_SER_DRIVE_Backup. If the value of IDN142 and the persistent saved value are the same, then the output bCheckOK is set to TRUE, otherwise bCheckOK is set to FALSE.

bBackup: The backup is activated by a positive edge at this input.

bRestore: The restore is activated by a positive edge at this input.

bCRCEnable: The CRC16-CCITT and the 16 bit checksum are enabled with a steady TRUE at this input. The CRC and the checksum are stored in IDN142, if bCRCEnable = TRUE.

bStdBackupList: determines if the backup list IDN192 (bStdBackupList = TRUE) is used for the backup or if the parameter list of all parameters IDN017 (bStdBackupList = FALSE) is used for the backup. Restore requires a backup file created with list IDN192.

sNetId: It is possible here to provide the AmsNetId of the TwinCAT computer on which the ADS command is to be executed. If it is to be run on the local computer, an empty string can be entered.

nPort: The port number nPort is specified by the TwinCAT System Manager during the hardware configuration.

sComment: sComment is a comment that is written to/read from the file header of the backup file.

ePath: determines the path of the backup and restore file. If ePath = PATH_BOOTPATH then the file is located in the TwinCAT boot folder, if ePath = PATH_GENERIC then the file is located in the folder specified in sPathName.

sPathName: Contains the path name or the file name of the backup and restore file, depending on ePath. If the boot path is used, then only the file name has to be set in sPathName. If the generic path is used, then sPathName has to contain the complete path and the file name.

tTimeout: States the length of the timeout that may not be exceeded by execution of the ADS command.

VAR_OUTPUT

```

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

```

bBusy: When the function block is activated this output is set. It remains set until an acknowledgement is received.

bError: If an ADS error should occur during the transfer of the command, then this output is set once the bBusy output is reset.

nErrId: Supplies the ADS error number or function block specific error code when the bError output is set.

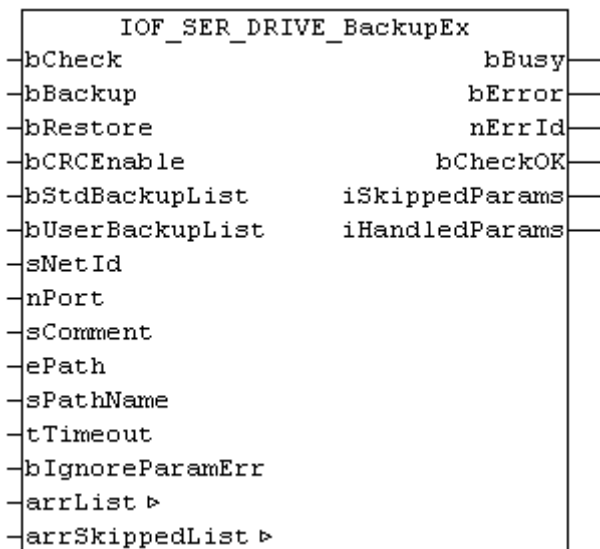
bCheckOk: Is TRUE, if bCheck was successful.

Function block specific error codes	Description
0x1003	Wrong parameter mode
0x1004	Data parameter wrong value size
0x1005	False backup parameter type
0x1006	Backup list was not IDN 192

Requirements

Development environment	Target system type	IO hardware	PLC libraries to include
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 are included automatically)

3.16.8 IOF_SER_DRIVE_BackupEx



The function block "IOF_SER_DRIVE_Backup" allows to create and restore a binary backup file from within the PLC containing all S- and P-Parameters of the SERCOS. The parameter list IDN192 contains a list of all backup parameters of the drive and is used by default. Backup and restore require SERCOS to be in parameter mode (phase 2).

If bStdBackupList = TRUE (default setting) then parameter list IDN192 is used for the backup.
 If bUserBackupList = TRUE then userdefined parameter list arrList is used for the backup.
 Otherwise the parameter list IDN17, the list of all parameters, is used for the backup.

Restore requires a backup file that was created with the standard backup list (IDN192) or with a userdefined list. Some parameters of the parameter list IDN17 are write protected.

Backup and restore can create a CRC16-CCITT and a 16 bit check sum and store this in IDN142, if available. The option bCRCEnable is per default FALSE.

Internally, instances of the function blocks IOF_SER_IDN_Read, IOF_SER_IDN_Write, FB_FileOpen, FB_FileClose, FB_FileRead, FB_FileWrite are called.

The file format of the backup file is described in [Backup-Fileformat \[▶ 122\]](#).

A demo project for Backup/Restore with IOF_SER_DRIVE_BackupEx is in <https://infosys.beckhoff.com/content/1033/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: The check of the CRC and the checksum is activated by a positive edge at this input. The CRC and the checksum are stored in IDN142 and persistent in the IOF_SER_DRIVE_Backup. If the value of IDN142 and the persistent saved value are the same, then the output bCheckOK is set to TRUE, otherwise bCheckOK is set to FALSE.

bBackup: The backup is activated by a positive edge at this input.

bRestore: The restore is activated by a positive edge at this input.

bCRCEnable: The CRC16-CCITT and the 16 bit checksum are enabled with a steady TRUE at this input. The CRC and the checksum are stored in IDN142, if bCRCEnable = TRUE.

bStdBackupList: Determines if the backup list IDN192 (bStdBackupList = TRUE) is used for the backup or if the parameter list of all parameters IDN017(bStdBackupList = FALSE) is used for the backup. Restore requires a backup file created with list IDN192.

bUserBackupList: determines if the userdefined backup list arrList is used for the backup. By default the IDN192 is used for the backup. If bStdBackupList = FALSE and bUserBackupList = TRUE then the userdefined list arrList is used for the backup. Restore requires a backup file created with list IDN192 or with a user defined parameter list.

sNetId: It is possible here to provide the AmsNetId of the TwinCAT computer on which the ADS command is to be executed. If it is to be run on the local computer, an empty string can be entered.

nPort: The port number nPort is specified by the TwinCAT System Manager during the hardware configuration.

sComment: sComment is a comment that is written to/read from the file header of the backup file.

ePath: Determines the path of the backup and restore file. If ePath = PATH_BOOTPATH then the file is located in the TwinCAT boot folder, if ePath = PATH_GENERIC then the file is located in the folder specified in sPathName.

sPathName: Contains the path name or the file name of the backup and restore file, depending on ePath. If the boot path is used, then only the file name has to be set in sPathName. If the generic path is used, then sPathName has to contain the complete path and the file name.

tTimeout: States the length of the timeout that may not be exceeded by execution of the ADS command.

blgnoreParamErr: Determines if Backup/Restore aborts or continues in case of parameter read/write errors. By default it is aborted (blgnoreParamErr = FALSE). If ignore of errors is activated (blgnoreParamErr = TRUE), then the skipped parameters are stored in the list arrSkippedList (parameter numbers and error numbers).

VAR_IN_OUT

```

VAR_IN_OUT
arrList          : ST_SercosParamList;
arrSkippedList  : ST_SercosParamErrList;
END_VAR

```

arrList: In case of backup via IDN192 (bStdBackupList = TRUE) the list arrList contains after the backup the list of the backup parameters from IDN192.

In case of userdefined backup (bUserBackupList = TRUE and bStdBackupList = FALSE) the list arrList has to contain the list of parameters before the backup is started.

In case of backup via IDN17 (bStdBackupList = FALSE and bUserBackupList = FALSE) the list arrList contains after the backup the list all parameters from IDN17.

arrSkippedList: Contains the list of skipped parameters (the parameter numbers and the error numbers).

See structure definitions below.

VAR_OUTPUT

```
VAR_OUTPUT
  bBusy      : BOOL;
  bError     : BOOL;
  nErrId     : UDINT;
  bCheckOK   : BOOL;
  iSkippedParams : UINT;
  iHandledParams : UINT;
END_VAR
```

bBusy: When the function block is activated this output is set. It remains set until an acknowledgement is received.

bError: If an ADS error should occur during the transfer of the command, then this output is set once the bBusy output is reset.

nErrId: Supplies the ADS error number or function block specific error code when the bError output is set.

bCheckOk: Is TRUE, if bCheck was successful.

iSkippedParams: Contains the number of skipped parameters (see arrSkippedList), if ignoring of parameter faults is active (blgnoreParamErr = TRUE).

iHandledParams: Contains the number of successfully backed up/restored parameters.

Structure definitions

```
TYPE ST_SercosParamList :
STRUCT
  iActCount : UINT;
  iMaxCount : UINT;
  iList     : ARRAY [0..2047] OF UINT;
END_STRUCT
END_TYPE
```

iActCount: is the current number of parameters of a list * 2. Sercos stores the number of bytes here and a parameter number needs 2 bytes, f.i. the value 6 means 3 parameters.

iMaxCount: is the max. number of parameters of a list * 2. Sercos stores the number of bytes here and a parameter number needs 2 bytes, f.i. the value 6 means 3 parameters.

iList: is an array of up to 2048 parameter numbers.

```
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: is the number of skipped parameters (here 3 means 3 parameter faults)

iMaxCount: is the number of skipped parameters (here 3 means 3 parameter faults)

iList: is an array of up to 2048 parameter numbers, which were skipped because of parameter faults during access.

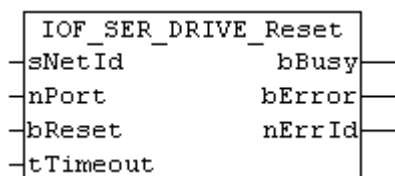
iError: is an array of up to 2048 error numbers, which occurred during parameter access.

Function block specific error codes	Description
0x1003	Wrong parameter mode
0x1004	Data parameter wrong value size
0x1005	False backup parameter type
0x1006	Backup list was not IDN 192

Requirements

Development environment	Target system type	IO hardware	PLC libraries to include
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 are included automatically)

3.16.9 IOF_SER_DRIVE_Reset



The function block "IOF_SER_DRIVE_Reset" performs a drive reset of a SERCOS drive. The drive errors are cleared.

Internally, an instance of the ADSWRITE function block is called.

VAR_INPUT

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

sNetId: It is possible here to provide the AmsNetId of the TwinCAT computer on which the ADS command is to be executed. If it is to be run on the local computer, an empty string can be entered.

nPort: The port number nPort is specified by the TwinCAT System Manager during the hardware configuration.

bReset: The function block is activated by a positive edge at this input.

tTimeout: States the length of the timeout that may not be exceeded by execution of the ADS command.

VAR_OUTPUT

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

bBusy: When the function block is activated this output is set. It remains set until an acknowledgement is received.

bError: If an ADS error should occur during the transfer of the command, then this output is set once the bBusy output is reset.

nErrId: Supplies the ADS error number when the bError output is set.

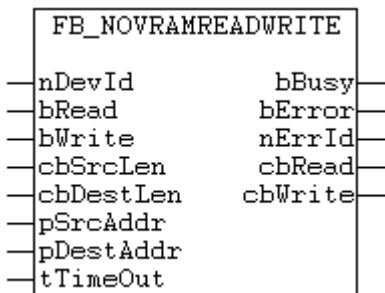
Requirements

Development environment	Target system type	IO hardware	PLC libraries to include
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 are included automatically)

3.17 NOV/DP-RAM

3.17.1 FB_NovRamReadWrite

The following description is referenced to TwinCAT v2.8. From TwinCAT v2.9 [Build 927] on, no function block for read and write of PLC data to NOV-RAM is necessary anymore. *Please see also: [Configuration example \(TwinCAT 2.9\)](#).*



With FUNCTION_BLOCK "FB_NovRamReadWrite" an access to the NOV-RAM [▶ 47] IC on the Beckhoff FCxxx-0002 Fieldbus cards can be achieved through the PLC program. With a rising edge at the *bRead* or *bWrite* input, the FB instance is activated, and a various number of data bytes can be read from the NOV-RAM or written into the NOV-RAM with the given *nDevId*. If both inputs (*bRead* and *bWrite*) are set at the same time, the passed data is going to be written first and read back after that procedure again.

Note:

To detect the address pointer of the NOV-RAM, an instance of the ADSREAD FUNCTION_BLOCK is used inside the *FB_NovRamReadWrite* function block. This address pointer is detected only during the initial call of the *FB_NovRamReadWrite* function block or after a change of the *nDevId* input. Multiple PLC cycles are needed for getting this address pointer. To write the data into the NOV-RAM, the FUNCTION MEMCPY is used. Therefore, data can be written or read during the same PLC cycle. Internally, the maximum byte length of the NOV-RAM IC is detected also and the number of bytes to be written or read is limited to this value.

VAR_INPUT

```

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

nDevId: The appropriate [device Id \[▶ 47\]](#) of the NOV-RAM device. Via this **Id**, the NOV-RAM IC which is implemented on the FCxxx-0002 card is specified. The device ID's are specified during the I/O Configuration within the TwinCAT System Manager.

bRead: With a rising edge on this input, the FB is activated and *cbDestLen* data is copied from NOV-RAM into the buffer with the *pDestAddr* address (starting with address offset NULL).

bWrite: With a positive edge on this input, the FB is activated and *cbSrcLen* data is copied out of the buffer with the address of *pSrcAddr* into the NOV-RAM (starting with address offset NULL).

cbSrcLen: : The byte length of data , to be written into the NOV-RAM.

cbDestLen: The byte length of data, to be read from the NOV-RAM.

pSrcAddr: The address pointer to the buffer for the data, to be written into the NOV-RAM. The address pointer can be determined with the **ADR()** -Operator.

pDestAddr: The address pointer to the buffer for the data, where the read values are going to be copied into.

tTimeout: Names the timeout delay, which is not to be exceeded with this command.

VAR_OUTPUT

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

bBusy: With activation of the FB, this output is set until the execution of the write or read command is finished.

bError: If an error occurred during the execution of the FB, this output is set.

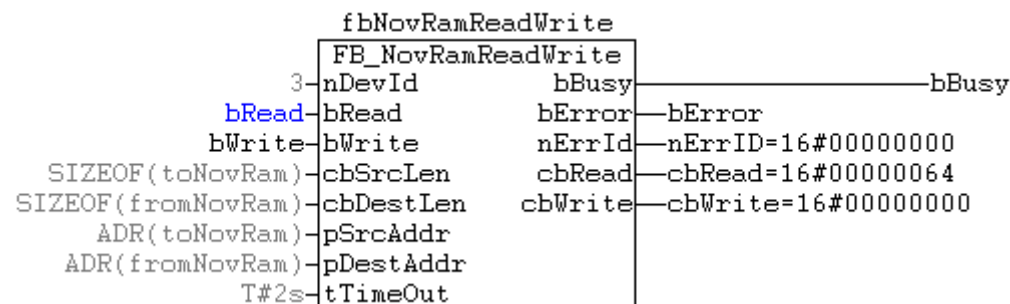
nErrId: Shows with a TRUE on the *bError* output the appropriate ADS error ID ("ADS - Return Codes").

cbRead: Number of successfully read data bytes.

cbWrite: Number of successfully written data bytes..

Example for the FB call in FBD:

```
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
```

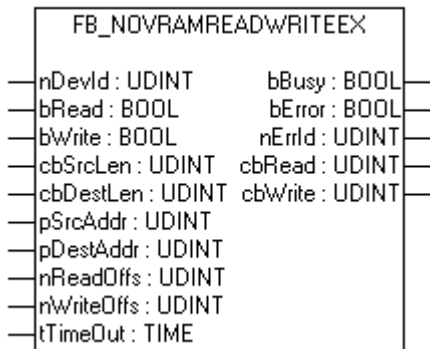


In the above example, with a rising edge at *bRead* 100 byte data were read from the NOV-RAM and copied into the *fromNovRam* array.

Requirements

Development environment	Target System	IO-Hardware	PLC libraries to include
TwinCAT v2.8.0 Build > 722	PC (x86), (not for CX (ARM)!)	FCxxxx cards with NOV-RAM (FCxxxx-0002)	TcIoFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib are included automatically)

3.17.2 FB_NovRamReadWriteEx



The function block *FB_NovRamReadWriteEx* provides access to [NOV-RAM \[► 47\]](#) (e.g. for **FCxxxx-0002** fieldbus cards, **CX9000** NOV-RAM etc.) from a PLC program. Activation of the function block is triggered by a rising edge at the *bRead* or *bWrite* input. A certain number of data bytes is read from or written to the NOV-RAM. If both inputs, i.e. *bRead* and *bWrite* are set at the same time, data are first written to the NOV-RAM and then read back. Unlike with the *FB_NovRamReadWrite* block, the address offset for write and read access can be specified in the NOV-RAM. The block also checks the permitted NOV-RAM memory access mode and copies data byte by byte into the NOV-RAM memory if required, instead of using MEMCPY. The CX9000 NOV-RAM, for example, only allows byte access, and the *FB_NovRamReadWrite* block would return an error in this case.

Comments:

In order to determine the NOV-RAM address pointer, the *FB_NovRamReadWriteEx* function block internally uses an instance of the ADSREAD function block. This address pointer is only determined when the *FB_NovRamReadWriteEx* function block is called for the first time or in the event of a change in *nDevId*. This task requires several PLC cycles. The NOV-RAM memory is accessed directly for writing data to or reading data from the NOV-RAM. This enables data to be written or read in the same PLC cycle. Internally the maximum byte length of the NOV-RAM is also determined, and the maximum data length that can be read or written is limited to this length.

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: The device ID [▶ 47] of a NOV-RAM card. Via the **Id** the NOV-RAM of an FCxxxx-0002 card is specified for which write or read access is required via the function block. The device IDs are specified by the TwinCAT System Manager during hardware configuration.

bRead: The block is activated by a rising edge at this input, and *cbDestLen* data are copied from the NOV-RAM (from address offset *nReadOffs*) into the buffer with address *pDestAddr*.

bWrite: The block is activated by a rising at this input, and *cbSrcLen* data are copied from the buffer with address *pSrcAddr* into the NOV-RAM (from address offset *nWriteOffs*).

cbSrcLen: : The byte length of the data to be written into the NOV-RAM.

cbDestLen: The byte length of the data to be read from the NOV-RAM.

pSrcAddr: The address pointer to a data buffer containing the data to be written into the NOV-RAM. The address pointer can be determined via the **ADR()** operator.

pDestAddr: The address pointer to a data buffer into which the read NOV-RAM data are to be copied.

nReadOffs: The address offset in the NOV-RAM from which data are to be read.

nWriteOffs: The address offset in the NOV-RAM from which data are to be written.

tTimeout: States the length of the timeout that may not be exceeded during execution of the command/function.

VAR_OUTPUT

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

bBusy: This output is set when the function block is activated, and remains set until execution of the function has been completed.

bError: This output is set if an error occurs during execution.

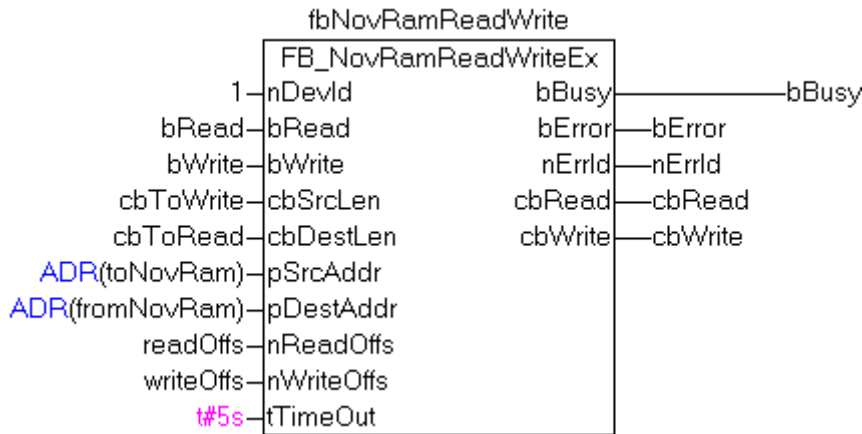
nErrId: If the *bError* output is set, this parameter supplies the ADS error number ("ADS Return Codes").

cbRead: Number of successfully read data bytes.

cbWrite: Number of successfully written data bytes.

Example of a call in the FBD:

```
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
```



In the example a rising edge at the *bRead* input led to 100 bytes of data being read from the NOV-RAM and copied into the *fromNovRam* array.

Requirements

Development environment	Target system type	IO Hardware	PLC libraries to include
TwinCAT v2.10.0 Build > 1231	PC (i386)	FCxxxx cards with NOV-RAM	TcloFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib are included automatically)

3.17.3 FB_GetDPRAMInfo

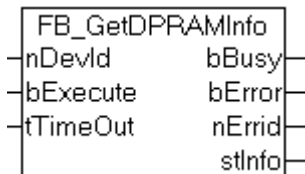


Fig. 1: FB_GetDPRAMInfo

The function block *FB_GetDPRAMInfo* can be used to determine the address pointer and the configured size of the NOV/DP-RAM of a fieldbus card. The address pointer can be used, for example, for direct write or read access of the NOV-RAM of the FCxxx-0002 (Beckhoff) cards or the DPRAM of cards that are not supported by TwinCAT (third-party manufacturers). First, the card has to be configured as general NOV/DP-RAM [▶ 47] within the TwinCAT System Manager. The MEMCPY, MEMSET or MEMCMP functions of the PLC program can then be used for write/read access to any memory offset.

VAR_INPUT

```

VAR_INPUT
    nDevId      : UDINT;
    bExecute    : BOOL;
    tTimeout    : TIME;
END_VAR
    
```

nDevId: The device **ID** of a NOV/DP-RAM card. The **ID** is used to specify the card from which information is to be read. The device IDs are specified by the TwinCAT System Manager during hardware configuration.

bExecute: The function block is activated via a rising edge at this input.

tTimeout: States the length of the timeout that may not be exceeded during execution of the command/function.

VAR_OUTPUT

```

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

```

bBusy: This output is set when the function block is activated and remains set until execution of the function has been completed.

bError: This output is set if an error occurs during execution.

nErrId : If the bError output is set, this parameter supplies the ADS error number ("ADS Return Codes").

stInfo: A structure | [127](#) with the address and size of the NOV/DP-RAM.

Example in ST:

```

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

```

Requirements

Development environment	Target system type	IO Hardware	PLC libraries to include
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 are included automatically)

3.18 AX200x Profibus

Function blocks for the access to the AX20XX via Profibus (FC310x with firmware > 1.20).

Function blocks

Name	Description
FB_AX2000_Parameter [▶ 57]	Write /read data for parameterization of the drive. It must be remembered that the "STOP" input of the AX2000AXACT block must be held TRUE while a parameter that will change the operating mode is being written.
FB_AX2000_AXACT [▶ 54]	Start Axis actions (must be called cyclic)
FB_AX2000_Jogmode [▶ 56]	Jog Mode
FB_AX2000_Reference [▶ 58]	Set the reference point, resp. start calibration
FB_AX200X_Profibus [▶ 59]	This function block combines the three previous function blocks. It offers the complete interface to the AX2000 with access to all functions(excepting parameters).

Linking into the System Manager

The "AX2000" box is inserted in the TwinCAT System Manager into the I/O configuration under the corresponding Profibus card.

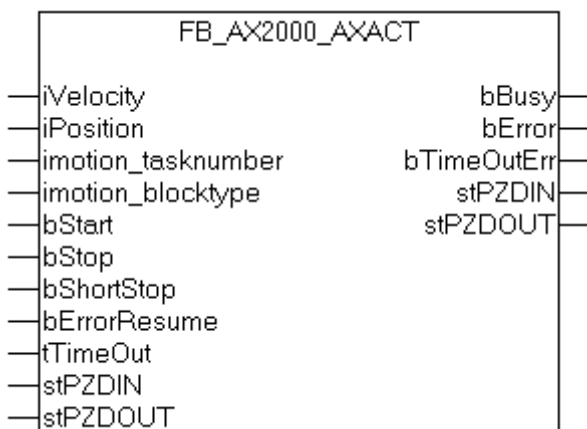


The I/O variables can now be linked directly to the corresponding I/O variables in the PLC application in the "PZD" (process data) module for the AX2000 box. The "PKW" module is not linked, because the PKW data is transmitted by ADS.

Notes on use of the function blocks

- Instances of the I/O structures stPZD_IN and stPZD_OUT must be created and addressed to be able to link them with the axis in the System Manager.
- After first being switched on, the drive is in a safe operating mode, which means that the "positioning" mode must be set before the first axis actions can be executed. This is done by setting the "bInit" input at the AX200X_Profibus block.
- The direction of travel in jog mode is specified by the arithmetic sign of "JogModeBasicVelo".
- Every reference travel, and every setting of the reference point, **must** be ended by setting bStop = TRUE.

3.18.1 FB_AX2000_AXACT



VAR_IN_OUT

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

stPZDIN : [Data words from drive to PLC \[► 125\]](#)

stPZDOUT : [Data words from PLC to drive \[► 125\]](#)

VAR_INPUT

```
VAR_INPUT
  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
```

iVelocity : The parameter contains the required transport speed for a following transport instruction, e.g. µm/s.

iPosition: Target position in physical magnitudes, e.g. µm, degrees

imotion_tasknumber : Travel block number. This input can be used to select a travel block that has previously been stored in the drive's memory.

imotion_blocktype: Travel block type (optional). This input can be used to modify properties of a direct travel command.

bStart: A rising edge at this boolean input sends a start command to the axis.

bStop : A rising edge at this boolean input sends a stop command to the axis. The axis stops and enters the "disabled" state.

bShortStop : A rising edge at this boolean input sends a stop command to the axis. The axis stops but remains in the "enabled" state.

bErrorResume : A rising edge at this boolean input resets an "AX200X error" (not a time-out error).

tTimeout: Maximum time out period.

VAR_OUTPUT

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

bBusy : This output remains TRUE until the block has executed a command. While Busy = TRUE, no new command will be accepted at the inputs. Please note that it is not the execution of the service but its acceptance whose time is monitored.

bErr : This output displays the error state.

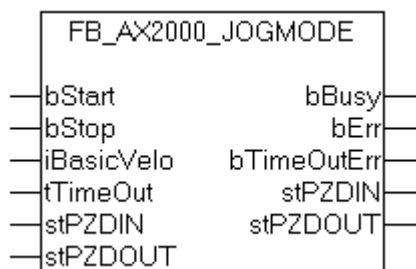
bTimeoutErr : TimeOut.

Requirements

Development Environment	Target System	IO Hardware	PLC Libraries to include
TwinCAT v2.8.0 Build > 737	PC (i386)	AX2000 Profibus box	TcloFunctions.Lib

Development Environment	Target System	IO Hardware	PLC Libraries to include
			(Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib are included automatically)

3.18.2 FB_AX2000_JogMode



Jogging

VAR_IN_OUT

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

stPZDIN : [Data words from drive to PLC \[► 125\]](#)

stPZDOUT : [Data words from PLC to drive \[► 125\]](#)

VAR_INPUT

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

bStart : Start the jogmode

bStop : Stop the jog mode

iBasicVelo : Basic velocity, the real velocity consists of basic velocity and the factor "v-jogmode" of the drive.

tTimeOut : TimeOut

VAR_OUTPUT

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

bBusy : This output remains TRUE until the block has executed a command. While Busy = TRUE, no new command will be accepted at the inputs. Please note that it is not the execution of the service but its acceptance whose time is monitored.

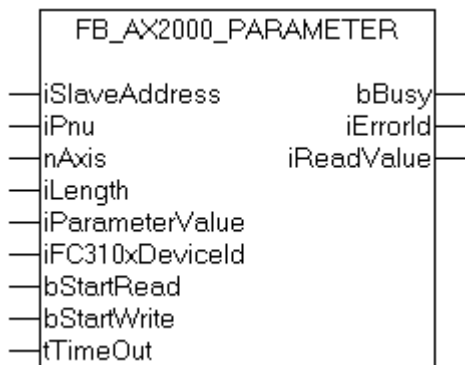
bErr : This output displays the error state.

bTimeOutErr : TimeOut.

Requirements

Development Environment	Target System	IO Hardware	PLC Libraries to include
TwinCAT v2.8.0 Build > 737	PC (i386)	AX2000 Profibus box	TcIoFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib are included automatically)

3.18.3 FB_AX2000_Parameter



Reading/writing the parameters via the parameter channel.

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: Station address.

iPnu: Selection of the parameter to be read or written. List of the available [parameter numbers](#) [▶ 144].

nAxis: Axis ID.

iLength: Length of the parameter (2 or 4).

iParameterValue : Value of the parameter to be read or written.

iFC310xDeviceId : Device-Id

bStartRead : A rising edge at this boolean input sends a command to the axis to start reading the parameters selected with "Pnu".

bStartWrite : A rising edge at this boolean input sends a command to the axis to start writing the parameters selected with "Pnu". **When changing operating mode, the write command is only effective if Stop = TRUE at the [FB_AX2000_AXACT](#) [▶ 54] block.**

tTimeOut: The maximum timeout - this time must not be exceeded.

VAR_OUTPUT

```

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

```

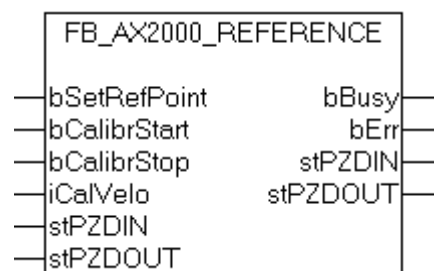
bBusy : This output remains TRUE until the block has executed a command. While Busy = TRUE, no new command will be accepted at the inputs. Please note that it is not the execution of the service but its acceptance whose time is monitored.

iErrorId : Contains the command-specific error code of the most recently executed command. Is reset to 0 by the execution of an instruction at the inputs.

iReadValue : Parameter value as a response to the "StartRead" command.

Requirements

Development Environment	Target System	IO Hardware	PLC Libraries to include
TwinCAT v2.8.0 Build > 737	PC (i386)	AX2000 Profibus box	TcIoFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib are included automatically)

3.18.4 FB_AX2000_Reference**Calibration****VAR_IN_OUT**

```

VAR_IN_OUT
  stPZDIN      : ST_PZD_IN;
  stPZDOUT     : ST_PZD_OUT;
END_VAR

```

stPZDIN : [Data words from drive to PLC \[► 125\]](#)

stPZDOUT : [Data words from PLC to drive \[► 125\]](#)

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: Set reference Point

bCalibrStart: Start home running.

bCalibrStop: Stop home running

iCalVelo: Basic velocity of Calibration. The end velocity contains of the basic velocity and the factor "v-jogmode" of the drive.

VAR_OUTPUT

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

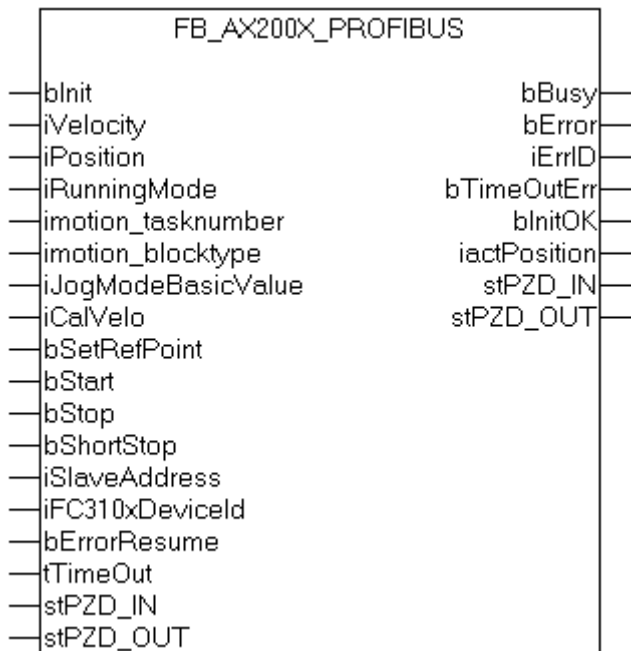
bBusy : This output remains TRUE until the block has executed a command. While Busy = TRUE, no new command will be accepted at the inputs. Please note that it is not the execution of the service but its acceptance whose time is monitored.

bErr : This output displays the error state.

Requirements

Development Environment	Target System	IO Hardware	PLC Libraries to include
TwinCAT v2.8.0 Build > 737	PC (i386)	AX2000 Profibus box	TcloFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib are included automatically)

3.18.5 FB_AX200X_Profibus



VAR_IN_OUT

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

stPZDIN : [Data words from drive to PLC |► 125|](#)

stPZDOUT : Data words from PLC to drive [► 125]

VAR_INPUT

```

VAR_INPUT
  bInit          : BOOL;      (*Initialization*)
  iVelocity      : DWORD;     (*Velocity*)
  iPosition      : DINT;      (*Position*)
  iRunningMode   : BYTE;      (*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: Initialisation of the drive. If bInit is TRUE then operating mode 2, "positioning", is set in the drive

iVelocity: The parameter contains the required transport speed for a following transport instruction, e.g. $\mu\text{m/s}$.

iPosition: Target position.

iRunningMode: 1: motiontask, 2: JogMode, 3: Calibration.

imotion_tasknumber : Travel block number. This input can be used to select a travel block that has previously been stored in the drive's memory.

imotion_blocktype: Travel block type (optional). This input can be used to modify properties of a direct travel command.

iJogModeBasicValue : Basic speed for jogging mode; the actual speed is derived from the basic speed and the "v-jogging mode" factor for the drive.

iCalVelo : Basic speed for the reference travel. The final speed is composed of the basic speed and the "v-jogging mode" factor for the drive.

bSetRefPoint : Setting the reference point.

bStart: Starting the action, depending on the state of iRunningMode.

bStop : Stopping the action, depending on the state of iRunningMode.

bShortStop :

iSlaveAddress : Station address.

iFC310xDeviceId : Device-Id.

bErrorResume : A rising edge at this boolean input resets an "AX200X error" (not a time-out error).

tTimeOut: Time out period.

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: This output remains TRUE until the block has executed a command. While Busy = TRUE, no new command will be accepted at the inputs. Please note that it is not the execution of the service but its acceptance whose time is monitored.

bError : This output displays the error state.

iErrID : Contains the command-specific error code of the most recently executed command. Is reset to 0 by the execution of an instruction at the inputs.

bTimeOutErr : TimeOut-Error.

blnitOK : Initialisation state of the drive, blnit:= TRUE: The drive is initialised, and is in operating mode 2, "positioning".

iactPosition : Display of current position in running mode 1: Motiontask .

Requirements

Development Environment	Target System	IO Hardware	PLC Libraries to include
TwinCAT v2.8.0 Build > 737	PC (i386)	AX2000 Profibus box	TcIcFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib are included automatically)

3.19 ASI master terminal

Function blocks for accessing the ASI master terminal.

Function blocks

Name	Description
FB_ASI_Addressing [▶ 62]	Specify or modify addresses of the ASI slaves
FB_ASI_SlaveDiag [▶ 63]	Cyclic slave diagnostics (e.g. counter states)
FB_ASI_ReadParameter [▶ 65]	Universal function block for reading all the parameters of an ASI slave
FB_ASI_WriteParameter [▶ 66]	Universal function block for setting all the parameters of an ASI slave
FB_ReadInput_analog [▶ 69]	Read analog values
FB_WriteOutput_analog [▶ 70]	Write analog values
FB_ASI_Processdata_digital [▶ 67]	Read/write digital values
FB_ASI_ParameterControl [▶ 68]	Background communication. This block must always be called cyclically!!!

Errorcodes:

Errorcode (dec):	Description
1	Communication Timeout
2	ASI-Slaveaddress does not exist
3 - 10	reserved
11	ASI-Slave not activated (Slave is not in LAS)
12	Communication Failure
13	Dataexchange not enabled (CN.4 is not set)

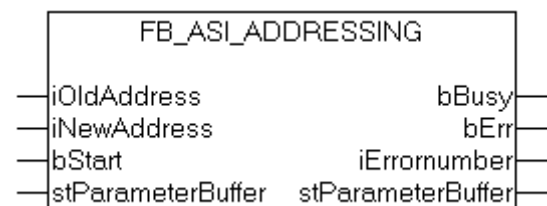
Linking into the System Manager

The library has one input structure: ST_Parameter_IN, and one output structure: ST_Parameter_OUT. These must be instantiated and addressed, so that on the one hand they can be passed to the FB_ParameterControl function block as VAR_IN_OUT, and on the other hand can be linked into the System Manager. The terminals' process data contains 6 or 16 bytes, depending on which ASI module is linked into the System Manager. These can be linked directly.

The screenshot shows the Beckhoff System Manager interface. On the left, a tree view displays the project structure under 'Test' > 'Inputs', with 'iParameterStatus' highlighted. On the right, the 'Variable' tab shows details for 'iParameterStatus':

- Name: iParameterStatus
- Type: UINT
- Group: Inputs
- Address: 1 (0x1)
- Linked to: Status . Channel 1 . Term 4 (KL6201-16) . Bc
- Comment: Variable of IEC1131 project "Test". Updated (**)
- ADS Info: Port: 802, IGrp: 0xF020, IOffs: 0x1, Len: 2

3.19.1 FB_ASI_Addresssing



VAR_IN_OUT

```

VAR_IN_OUT
  stParameterBuffer:    ST_ParameterBuffer;
END_VAR
  
```

stParameterBuffer: Data buffer [► 127] for background communication

VAR_INPUT

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

iOldAddress: old address of the slave to be addressed (new slaves have address 0)

iNewAddress: new address of the slave to be addressed

bStart : Addressing is carried out via a rising edge of this boolean input

VAR_OUTPUT

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

bBUSY: This output remains TRUE until the block has executed a command. While Busy = TRUE, no new command will be accepted at the inputs. Please note that it is not the execution of the service but its acceptance whose time is monitored.

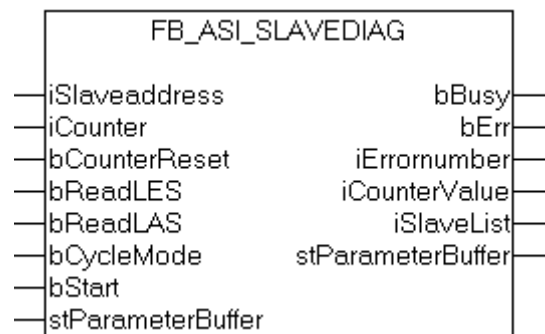
bErr: This output shows the error status

iErrornumber: Contains the command-specific error code of the most recently executed command. Is reset to 0 by the execution of an instruction at the inputs.

Requirements

Development environment	Target system type	IO Hardware	PLC libraries to include
TwinCAT Version >= 2.8.0 Build > 739	PC (i386)	ASI master terminal	TcIoFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib are included automatically)

3.19.2 FB_ASI_SlaveDiag



VAR_IN_OUT

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

stParameterBuffer: [Data buffer \[► 127\]](#) for background communication

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

```

iSlaveaddress: slave address

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

bCounterReset: resetting of the current counter

bReadLES: list of recognised ASI slaves (LES)

bReadWrite: 0=READ, 1=WRITE

bReadLAS: list of activated ASI slaves (LAS)

bStart: The respective task is carried out via a rising edge at this boolean input

bCycleMode: 0=continuous reading 1= reading once

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: This output remains TRUE until the block has executed a command. While Busy = TRUE, no new command will be accepted at the inputs. Please note that it is not the execution of the service but its acceptance whose time is monitored.

bErr: This output shows the error status

iErrornumber: Contains the command-specific error code of the most recently executed command. Is reset to 0 by the execution of an instruction at the inputs.

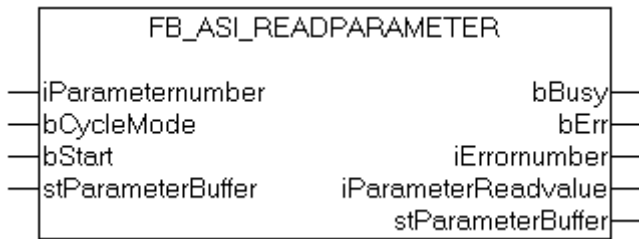
iCountervalue: counter reading

iSlaveList: LES or LAS

Requirements

Development environment	Target system type	IO Hardware	PLC libraries to include
TwinCAT Version >= 2.8.0 Build > 739	PC (i386)	ASI master terminal	TcIoFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib are included automatically)

3.19.3 FB_ASI_ReadParameter



VAR_IN_OUT

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

stParameterBuffer: [Data buffer \[► 127\]](#) for background communication

VAR_INPUT

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

iParameterNumber: parameter number

bCycleMode: 0=continuous reading 1= reading once

bStart: The respective task is carried out via a rising edge at this boolean input

VAR_OUTPUT

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

bBUSY: This output remains TRUE until the block has executed a command. While Busy = TRUE, no new command will be accepted at the inputs. Please note that it is not the execution of the service but its acceptance whose time is monitored.

bErr: This output shows the error status

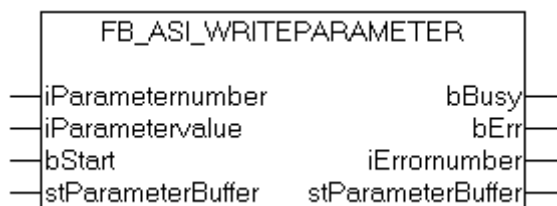
iErrornumber: Contains the command-specific error code of the most recently executed command. Is reset to 0 by the execution of an instruction at the inputs.

iParameterReadvalue: I/O ID or ID code of the addressed slave

Requirements

Development environment	Target system type	IO Hardware	PLC libraries to include
TwinCAT Version >= 2.8.0 Build > 739	PC (i386)	ASI master terminal	TcloFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib are included automatically)

3.19.4 FB_ASI_WriteParameter



VAR_IN_OUT

```

VAR_IN_OUT
    stParameterBuffer:    ST_ParameterBuffer;
END_VAR
  
```

stParameterBuffer: [Data buffer](#) [► 127] for background communication

VAR_INPUT

```

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

iParameterNumber: parameter number

iParametervalue: Parameter value

bStart: The respective task is carried out via a rising edge at this boolean input

VAR_OUTPUT

```

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

bBUSY: This output remains TRUE until the block has executed a command. While Busy = TRUE, no new command will be accepted at the inputs. Please note that it is not the execution of the service but its acceptance whose time is monitored.

bErr: This output shows the error status

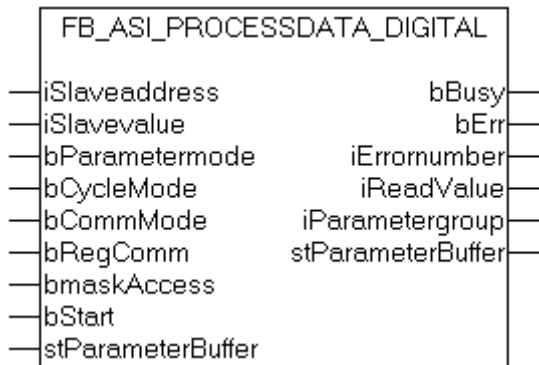
iErrornumber: Contains the command-specific error code of the most recently executed command. Is reset to 0 by the execution of an instruction at the inputs.

iParameterReadvalue: I/O ID or ID code of the addressed slave

Requirements

Development environment	Target system type	IO Hardware	PLC libraries to include
TwinCAT Version >= 2.8.0 Build > 739	PC (i386)	ASI master terminal	TcIoFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib are included automatically)

3.19.5 FB_ASI_Processdata_digital



VAR_IN_OUT

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

stParameterBuffer: [Data buffer \[► 127\]](#) for background communication

VAR_INPUT

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

iSlaveaddress: slave address

iSlavevalue: process value

bParametermode: 0: Read, 1: Write

bCycleMode: 0: Cyclic (permanent Read/Write), 1: Acyclic

bCommMode: 0: parameter access, 1: ADS (currently always 0)

bRegComm: Register communication: 0: parameter access, 1: register communication (currently always 0)

bmaskAccess: 0:normal access, 1:masked access

bStart: The respective task is carried out via a rising edge at this boolean input

VAR_OUTPUT

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

bBUSY: This output remains TRUE until the block has executed a command. While Busy = TRUE, no new command will be accepted at the inputs. Please note that it is not the execution of the service but its acceptance whose time is monitored.

bErr: This output shows the error status

iErrornumber: Contains the command-specific error code of the most recently executed command. Is reset to 0 by the execution of an instruction at the inputs.

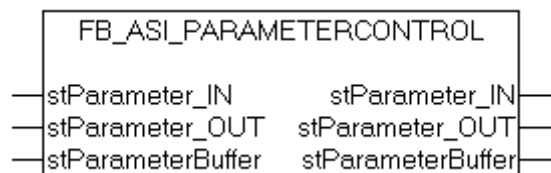
iReadvalue: process value

iParametergroup: parameter group

Requirements

Development environment	Target system type	IO Hardware	PLC libraries to include
TwinCAT Version >= 2.8.0 Build > 739	PC (i386)	ASI master terminal	TcIoFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib are included automatically)

3.19.6 FB_ASI_ParameterControl



The FB_ASI_ParameterControl realises the background communication between the ASI master terminal and the individual blocks of the Lib.

This block always has to be called cyclically!

```

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

stParameterBuffer: [Data buffer \[► 127\]](#) for background communication

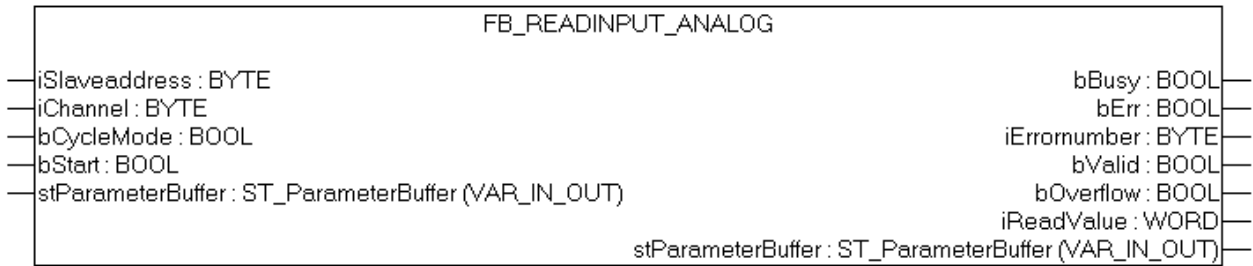
stParameter_IN : [Input data of the ASI terminal \[► 125\]](#)

stParameter_OUT : [Output data of the ASI terminal \[► 126\]](#)

Requirements

Development environment	Target system type	IO Hardware	PLC libraries to include
TwinCAT Version >= 2.8.0 Build > 739	PC (i386)	ASI master terminal	TcIoFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib are included automatically)

3.19.7 FB_ReadInput_analog



VAR_IN_OUT

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

stParameterBuffer: [Data buffer \[► 127\]](#) for background communication

VAR_INPUT

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

iSlaveaddress: Slave address

iChannel: Slave channel

bCycleMode: 0: Acyclic , 1:Cyclic (permanent Read/Write) If this bit is set, the output bBusy will only be reset if the input bStart will become FALSE. If the input bStart becomes to early FALSE, no current value exists at the output.

bStart : Addressing is carried out via a rising edge of this boolean input

VAR_OUTPUT

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

bBUSY: This output remains TRUE until the block has executed a command. While Busy = TRUE, no new command will be accepted at the inputs. Please note that it is not the execution of the service but its acceptance whose time is monitored.

bErr: This output shows the error status

iErrornumber: Contains the command-specific error code of the most recently executed command. Is reset to 0 by the execution of an instruction at the inputs.

bValid: If this output is set the read data are valid.

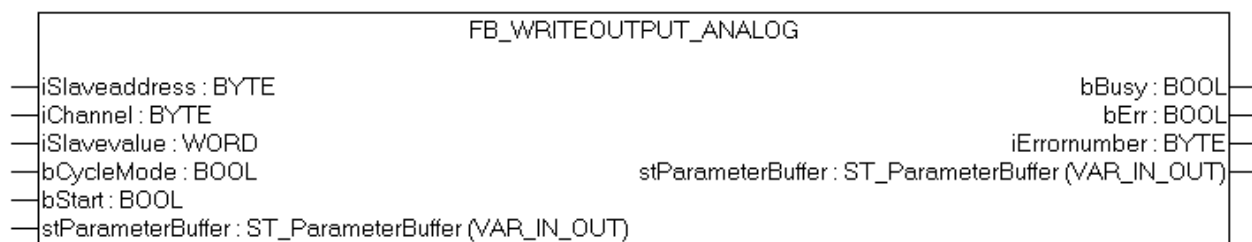
bOverflow: The value of the slave is out of range.

iReadvalue: Process value

Requirements

Development Environment	Target System	IO-Hardware	PLC Libraries to include
TwinCAT Version >= 2.8.0 Build > 739	PC (i386)	ASI-Master-Terminal	TcIoFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib are included automatically)

3.19.8 FB_WriteOutput_analog



VAR_IN_OUT

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

stParameterBuffer: [Data buffer](#) [▶ 127] for background communication

VAR_INPUT

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

END_VAR

iSlaveaddress: Slave address

iChannel: Slave channel

iSlavevalue: Data to be written

bCycleMode: 0: Acyclic , 1:Cyclic (permanent Read/Write) If this bit is set, the output bBusy will only be reset if the input bStart will become FALSE. If the input bStart becomes to early FALSE, no current value exists at the output.

bStart : Addressing is carried out via a rising edge of this boolean input

VAR_OUTPUT

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

END_VAR

bBUSY: This output remains TRUE until the block has executed a command. While Busy = TRUE, no new command will be accepted at the inputs. Please note that it is not the execution of the service but its acceptance whose time is monitored.

bErr: This output shows the error status

iErrornumber: Contains the command-specific error code of the most recently executed command. Is reset to 0 by the execution of an instruction at the inputs.

bValid: If this output is set the read data are valid.

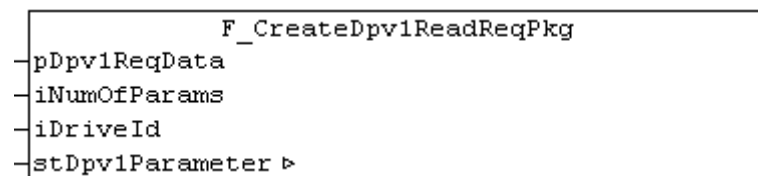
bOverflow: The value of the slave is out of range.

Requirements

Development Environment	Target System	IO-Hardware	PLC Libraries to include
TwinCAT Version >= 2.8.0 Build > 739	PC (i386)	ASI-Master-Terminal	TcIoFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib are included automatically)

3.20 Profibus DPV1 (Sinamics)

3.20.1 F_CreateDpv1ReadReqPkg : USINT



The function "F_CreateDpv1ReadReqPkg" creates a DPV1 telegram for a [FB_Dpv1Read \[▶ 72\]](#) of one or multiple parameters of a drive or the controller unit of a Sinamics Profidrive (Profidrive specification 3.1). Since the profidrive uses the Motorola format and the IPC uses the Intel format, the function automatically swaps the bytes in the DPV1 telegram of parameters with data types bigger than one byte.

The function returns the current length of the DPV1 telegram in bytes (max. 240 bytes).

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: Pointer to an array of 240 bytes, which contains the DPV1 read request telegram. This telegram is being created in this function.

iNumOfParams: Number of parameters to be read (1 to 39). Another limitation is the telegram size to 240 bytes.

iDriveID: The drive ID is 1 for the Controller Unit, 2 for the A drive, 3 for the B drive of double/triple drive. Drive ID is set in the Starter software. 1..16 is possible.

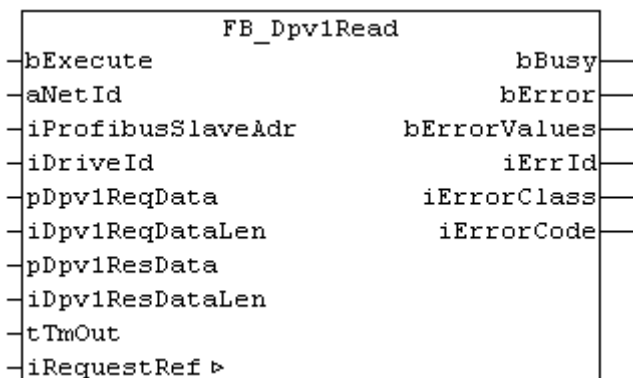
VAR_IN_OUT

```

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

stDpv1Parameter: Array of 39 [parameters \[▶ 135\]](#), which have to be added to the DPV1 read request telegram.

3.20.2 FB_Dpv1Read



The function block "FB_Dpv1Read" reads one or multiple parameter from a Sinamics Profidrive via DPV1 (Profidrive specification 3.1). The DPV1 request telegram needs to be created with [F_CreateDpv1ReadReqPkg \[▶ 71\]](#), before the bExecute sees a rising edge. The DPV1 response telegram needs to be split with [F_SplitDpv1ReadResPkg \[▶ 74\]](#) after the bBusy sends a falling edge.

The execution of this function block takes some time, depending on the number of parameters read. The function block sends the request telegram to the drive and then polls the drive for a corresponding response telegram.

Internally instances of the ADSREAD and ADSWRITE function blocks are called.

See sample <https://infosys.beckhoff.com/content/1033/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: The function block is activated by a positive edge at this input.

aNetId: Provide the AmsNetId of the Profibus Master device (see ADS-Tab of the Profibus Master device in the I/O configuration in the System Manager).

iProfibusSlaveAdr: The profibus slave DP address of the drive. This is one address for a multiple axes drive specified in the TwinCAT System Manager in the hardware configuration.

iDriveID: The drive ID is 1 for the ControllerUnit, 2 for the A drive, 3 for the B drive of double/triple drive. Drive ID is set in the Starter software. 1..16 is possible.

pDpv1ReqData: Pointer to an array of 240 bytes, which contains the DPV1 request telegram. This telegram needs to be created with [F_CreateDpv1ReadReqPkg \[▶ 71\]](#) before the executing the DPV1 read.

iDv1ReqDataLen: Maximum length of the DPV1 request data buffer (240 bytes).

pDpv1ResData: Pointer to an array of 240 bytes, which contains the DPV1 response telegram after the DPV1 read. This DPV1 response telegram needs to be split with [F_SplitDpv1ReadResPkg \[▶ 74\]](#) after the bBusy sends a falling edge.

iDv1ResDataLen: Maximum length of the DPV1 response data buffer (240 bytes).

tTimeout: States the length of the timeout that may not be exceeded by execution of the ADS command.

VAR_OUTPUT

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

bBusy: When the function block is activated this output is set. It remains set until an acknowledgement is received.

bError: If an ADS error should occur during the transfer of the command, then this output is set once the bBusy output is reset.

bErrorValues: Is TRUE if the DPV1 Read was not successful or partly not successful. The response telegram contains error values.

nErrId: Supplies the ADS error number or function block specific error code when the bError output is set.

nErrClass: Profidrive error class

nErrCode: Profidrive error code

VAR_IN_OUT

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

iRefRequest: Reference, which is increased automatically with each telegram. The reference is needed to assign the response telegram to the corresponding read/write request telegram.

Function block specific error codes	Description
0x2	wrong reference response
0x3	DPV1 Read not successful or partly not successful
0x4	wrong response ID
other error IDs	see ADS return codes

Error classes	Description	Error code
0x0 - 0x9	reserved	-
0xA	Application error	0x0: read error 0x1: write error 0x2: module failure 0x3 - 0x7: reserved 0x8: version conflict 0x9: feature not supported 0xA - 0xF: user specific
0xB	Access error	0x0: invalid index (no datablock DB47, parameter requests are not supported) 0x1: write length error 0x2: invalid slot 0x3: type conflict 0x4: invalid area 0x5: state conflict (access to DB47 temporarily not possible due to internal processing status) 0x6: access denied 0x7: invalid range (write DB47 error in the DB47 header) 0x8: invalid parameter 0x9: invalid type 0xA - 0xF: user specific

Error classes	Description	Error code
0xC	Resource error	0x0: read constraint conflict 0x1: write constraint conflict 0x2: resource busy 0x3: resource unavailable 0x4 - 0x7: reserved 0x8 - 0xF: user specific
0xD - 0xF	User defined error	-

Requirements

Development environment	Target system type	IO hardware	PLC libraries to include
TwinCAT v2.10.0 Build > 1307	PC (i386)	Beckhoff FC310x PCI, CX1500-M310, EL6731	TcIoFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib are included automatically)

3.20.3 F_SplitDpv1ReadResPkg : USINT

F_SplitDpv1ReadResPkg
<p>pDpv1ResData</p> <p>stDpv1Parameter ▶</p> <p>stDpv1ValueHeaderEx ▶</p>

The function "F_SplitDpv1ReadResPkg" parses a DPV1 telegram of a FB [Dpv1Read](#) [▶ 72] for one or multiple parameters of a drive or the controller unit of a Sinamics Profidrive (Profidrive specification 3.1). Since the profidrive uses the Motorola format and the IPC uses the Intel format, the function automatically swaps the bytes in the DPV1 telegram of parameters with data types bigger than one byte.

The function returns the current length of the DPV1 telegram in bytes (max. 240 bytes).

VAR_INPUT

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

pDpv1ResData: Pointer to an array of 240 bytes, which contains the DPV1 read response telegram. This telegram is being split in this function.

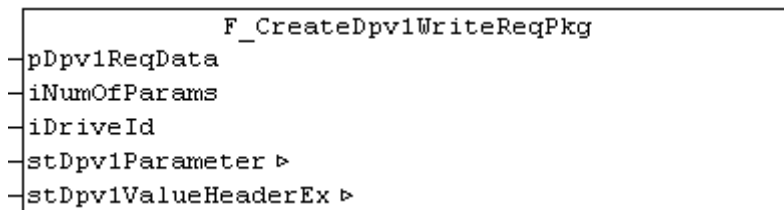
VAR_IN_OUT

```
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 of 39 [parameters](#) [▶ 135], which have been added to the DPV1 read request telegram.

stDpv1ValueHeaderEx: Array of 39 [parameter](#) [▶ 137] values, which have to be read from the drive.

3.20.4 F_CreateDpv1WriteReqPkg : USINT



The function "F_CreateDpv1WriteReqPkg" creates a DPV1 telegram for a [FB_Dpv1Write](#) [▶ 75] of one or multiple parameters of a drive or the controller unit of a Sinamics Profidrive (Profidrive specification 3.1). Since the profidrive uses the Motorola format and the IPC uses the Intel format, the function automatically swaps the bytes in the DPV1 telegram of parameters with data types bigger than one byte.

The function returns the current length of the DPV1 telegram in bytes (max. 240 bytes).

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: Pointer to an array of 240 bytes, which contains the DPV1 write request telegram. This telegram is being created in this function.

iNumOfParams: Number of parameters to be written (1 to 39). Another limitation is the telegram size to 240 bytes.

iDriveID: The drive ID is 1 for the Controller Unit, 2 for the A drive, 3 for the B drive of double/triple drive. Drive ID is set in the Starter software. 1..16 is possible.

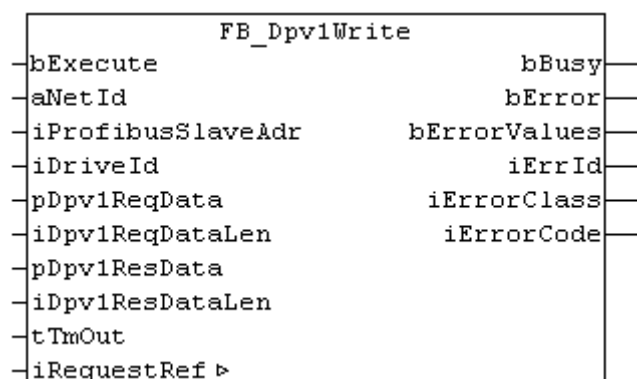
VAR_IN_OUT

```
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 of 39 [parameters](#) [▶ 135], which must be added to the DPV1 write request telegram.

stDpv1ValueHeaderEx: Array of 39 [parameter](#) [▶ 137] values, which must be added to the DPV1 write request telegram.

3.20.5 FB_Dpv1Write



The function block "FB_Dpv1Write" writes one or multiple parameters to a Sinamics Profidrive via DPV1 (Profidrive specification 3.1). The DPV1 request telegram needs to be created with [F_CreateDpv1WriteReqPkg \[► 75\]](#), before the bExecute sees a rising edge. The DPV1 response telegram needs to be split with [F_SplitDpv1WriteResPkg \[► 78\]](#) after the bBusy sends a falling edge.

The execution of this function block takes some time, depending on the amount of parameters written. The function block sends the request telegram to the drive and then polls the drive for a corresponding response telegram.

Internally instances of the ADSREAD and ADSWRITE function blocks are called.

See sample <https://infosys.beckhoff.com/content/1033/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: The function block is activated by a positive edge at this input.

aNetId: Provide the AmsNetId of the Profibus Master device (see ADS-Tab of the Profibus Master device in the I/O configuration in the System Manager).

iProfibusSlaveAdr: The profibus slave DP address of the drive. This is one address for a multiple axes drive specified in the TwinCAT System Manager in the hardware configuration.

iDriveID: The drive ID is 1 for the ControllerUnit, 2 for the A drive, 3 for the B drive of double/triple drive. Drive ID is set in the Starter software. 1..16 is possible.

pDpv1ReqData: Pointer to an array of 240 bytes, which contains the DPV1 request telegram. This telegram needs to be created with [F_CreateDpv1WriteReqPkg \[► 75\]](#) before the executing the DPV1 write.

iDv1ReqDataLen: Maximum length of the DPV1 request data buffer (240 bytes).

pDpv1ResData: Pointer to an array of 240 bytes, which contains the DPV1 response telegram after the DPV1 read. This DPV1 response telegram needs to be split with [F_SplitDpv1WriteResPkg \[► 78\]](#) after the bBusy sends a falling edge.

iDv1ResDataLen: Maximum length of the DPV1 response data buffer (240 bytes).

tTimeout: States the length of the timeout that may not be exceeded by execution of the ADS command.

VAR_OUTPUT

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

bBusy: When the function block is activated this output is set. It remains set until an acknowledgement is received.

bError: If an ADS error should occur during the transfer of the command, then this output is set once the bBusy output is reset.

bErrorValues: Is TRUE if the DPV1 Read was not successful or partly not successful. The response telegram contains error values.

nErrId: Supplies the ADS error number or function block specific error code when the bError output is set.

nErrClass: Profidrive error class

nErrCode: Profidrive error code

VAR_IN_OUT

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

iRefRequest: Reference, which is increased automatically with each telegram. The reference is needed to assign the response telegram to the corresponding read/write request telegram.

Function block specific error codes	Description
0x2	wrong reference response
0x3	DPV1 Read not successful or partly not successful
0x4	wrong response ID
other error IDs	see ADS return codes

Error classes	Description	Error code
0x0 - 0x9	reserved	-
0xA	Application error	0x0: read error 0x1: write error 0x2: module failure 0x3 - 0x7: reserved 0x8: version conflict 0x9: feature not supported 0xA - 0xF: user specific
0xB	Access error	0x0: invalid index (no datablock DB47, parameter requests are not supported) 0x1: write length error 0x2: invalid slot 0x3: type conflict 0x4: invalid area 0x5: state conflict (access to DB47 temporarily not possible due to internal processing status) 0x6: access denied 0x7: invalid range (write DB47 error in the DB47 header) 0x8: invalid parameter 0x9: invalid type 0xA - 0xF: user specific
0xC	Resource error	0x0: read constraint conflict 0x1: write constraint conflict 0x2: resource busy 0x3: resource unavailable 0x4 - 0x7: reserved 0x8 - 0xF: user specific
0xD - 0xF	User defined error	-

Requirements

Development environment	Target system type	IO hardware	PLC libraries to include
TwinCAT v2.10.0 Build > 1307	PC (i386)	Beckhoff FC310x PCI, CX1500-M310, EL6731	TcloFunctions.Lib

Development environment	Target system type	IO hardware	PLC libraries to include
			(Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib are included automatically)

3.20.6 F_SplitDpv1WriteResPkg : USINT

```

F_SplitDpv1WriteResPkg
- pDpv1ResData
- stDpv1Parameter ▷
- stDpv1ValueHeaderEx ▷

```

The function "F_SplitDpv1WriteResPkg" parses a DPV1 telegram of a [FB_Dpv1Write](#) [▶ 75] for one or multiple parameters of a drive or the controller unit of a Sinamics Profidrive (Profidrive specification 3.1). Since the profidrive uses the Motorola format and the IPC uses the Intel format, the function automatically swaps the bytes in the DPV1 telegram of parameters with data types bigger than one byte.

The function returns the current length of the DPV1 telegram in bytes (max. 240 bytes).

VAR_INPUT

```

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

```

pDpv1ResData: Pointer to an array of 240 bytes, which contains the DPV1 read response telegram. This telegram is being split in this function.

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 of 39 [parameters](#) [▶ 135], which have been added to the DPV1 write request telegram.

stDpv1ValueHeaderEx: Array of 39 [parameter](#) [▶ 137] values, which have to be written to the drive.

3.21 Profinet DPV1 (Sinamics)

3.21.1 F_CreateDpv1ReadReqPkgPNET : USINT

```

F_CreateDpv1ReadReqPkgPNET
- pDpv1ReqData
- iNumOfParams
- iDriveId
- stDpv1Parameter ▷

```

The function "F_CreateDpv1ReadReqPkgPNET" creates a DPV1 telegram for a [FB_Dpv1Read](#) [▶ 79] of one or multiple parameters of a drive or the controller unit of a Sinamics Profidrive (Profidrive specification 3.1) via Profinet. Since the profidrive uses the Motorola format and the IPC uses the Intel format, the function automatically swaps the bytes in the DPV1 telegram of parameters with data types bigger than one byte.

The function returns the current length of the DPV1 telegram in bytes (max. 240 bytes).

VAR_INPUT

```
VAR_INPUT
  pDpvlReqData      : POINTER TO ARRAY [1..iMAX_DPv1_SIZE] OF BYTE; (* DPV1 read request *)
  iNumOfParams      : USINT; (* 1..39; else: reserved *)
  iDriveId          : USINT;
END_VAR
```

pDpvlReqData: Pointer to an array of 240 bytes, which contains the DPV1 read request telegram. This telegram is being created in this function.

iNumOfParams: Number of parameters to be read (1 to 39). Another limitation is the telegram size to 240 bytes.

iDriveID: The drive ID is 1 for the Controller Unit, 2 for the A drive, 3 for the B drive of double/triple drive. Drive ID is set in the Starter software. 1..16 is possible.

VAR_IN_OUT

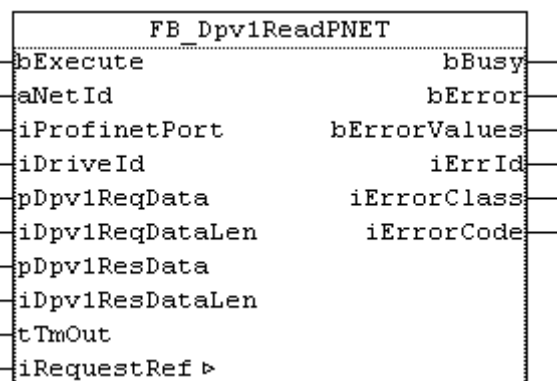
```
VAR_IN_OUT
  stDpvlParameter : ARRAY [1..iMAX_DPv1_PARAMS] OF ST_DpvlParamAddrEx; (* list of parameters *)
END_VAR
```

stDpvlParameter: Array of 39 parameters [▶ 135], which have to be added to the DPV1 read request telegram.

Requirements

Development environment	Target platform	IO hardware	PLC libraries to include
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 linked automatically)

3.21.2 FB_Dpv1ReadPNET



The function block "FB_Dpv1ReadPNET" reads one or multiple parameter from a Sinamics Profidrive via DPV1 (Profidrive specification 3.1) via Profinet. The DPV1 request telegram needs to be created with [F_CreateDpvlReadReqPkgPNET \[▶ 78\]](#), before the bExecute sees a rising edge. The DPV1 response telegram needs to be split with [F_SplitDpvlReadResPkgPNET \[▶ 82\]](#) after the bBusy sends a falling edge.

The execution of this function block takes some time, depending on the amount of parameters read. The function block sends the request telegram to the drive and then polls the drive for a corresponding response telegram.

Internally instances of the ADSREAD and ADSWRITE function blocks are called.

See sample <https://infosys.beckhoff.com/content/1033/tcplclbibofunctions/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: The function block is activated by a positive edge at this input.

aNetId: Provide the AmsNetId of the Profibus Master device (see ADS-Tab of the Profibus Master device in the I/O configuration in the System Manager).

iProfinetPort: The profinet port number of the drive. This is one address for a multiple axes drive specified in the TwinCAT System Manager in the hardware configuration.

iDriveID: The drive ID is 1 for the ControllerUnit, 2 for the A drive, 3 for the B drive of double/triple drive. Drive ID is set in the Starter software. 1..16 is possible.

pDpV1ReqData: Pointer to an array of 240 bytes, which contains the DPV1 request telegram. This telegram needs to be created with [F_CreateDpV1ReadReqPkgPNET \[► 78\]](#) before the executing the DPV1 read.

iDv1ReqDataLen: Maximum length of the DPV1 request data buffer (240 bytes).

pDpV1ResData: Pointer to an array of 240 bytes, which contains the DPV1 response telegram after the DPV1 read. This DPV1 response telegram needs to be split with [F_SplitDpV1ReadResPkgPNET \[► 82\]](#) after the bBusy sends a falling edge.

iDv1ResDataLen: Maximum length of the DPV1 response data buffer (240 bytes).

tTimeout: States the length of the timeout that may not be exceeded by execution of the ADS command.

VAR_OUTPUT

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

bBusy: When the function block is activated this output is set. It remains set until and acknowledgement is received.

bError: If an ADS error should occur during the transfer of the command, then this output is set once the bBusy output is reset.

bErrorValues: Is TRUE if the DPV1 Read was not successful or partly not successful. The response telegram contains error values.

nErrId: Supplies the ADS error number or function block specific error code when the bError output is set.

nErrClass: Profidrive error class

nErrCode: Profidrive error code

VAR_IN_OUT

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

iRefRequest: Reference, which is increased automatically with each telegram. The reference is needed to assign the response telegram to the corresponding read/write request telegram.

Function block specific error codes	Description
0x2	wrong reference response
0x3	DPV1 Read not successful or partly not successful
0x4	wrong response ID
other error IDs	see ADS return codes

Error classes	Description	Error code
0x0 - 0x9	reserved	-
0xA	Application error	0x0: read error 0x1: write error 0x2: module failure 0x3 - 0x7: reserved 0x8: version conflict 0x9: feature not supported 0xA - 0xF: user specific
0xB	Access error	0x0: invalid index (no datablock DB47, parameter requests are not supported) 0x1: write length error 0x2: invalid slot 0x3: type conflict 0x4: invalid area 0x5: state conflict (access to DB47 temporarily not possible due to internal processing status) 0x6: access denied 0x7: invalid range (write DB47 error in the DB47 header) 0x8: invalid parameter 0x9: invalid type 0xA - 0xF: user specific
0xC	Resource error	0x0: read constraint conflict 0x1: write constraint conflict 0x2: resource busy 0x3: resource unavailable 0x4 - 0x7: reserved 0x8 - 0xF: user specific
0xD - 0xF	User defined error	-

Requirements

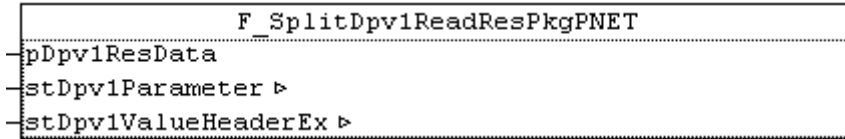
Development environment	Target system type	IO hardware	PLC libraries to include
TwinCAT v2.11.0 Build > 1554 TwinCAT v2.11.0 R2 Build > 2024	PC (i386)	Beckhoff EL6632	TcIoFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib are included automatically)

Also see about this

▣ F_CreateDpv1ReadReqPkg : USINT [▶ 71]

▣ F_SplitDpv1ReadResPkg : USINT [▶ 74]

3.21.3 F_SplitDpv1ReadResPkgPNET : USINT



The function "F_SplitDpv1ReadResPkg" parses a DPV1 telegram of a FB_Dpv1Read [▶ 79] for one or multiple parameters of a drive or the controller unit of a Sinamics Profidrive (Profidrive specification 3.1) via Profinet. Since the profidrive uses the Motorola format and the IPC uses the Intel format, the function automatically swaps the bytes in the DPV1 telegram of parameters with data types bigger than one byte.

The function returns the current length of the DPV1 telegram in bytes (max. 240 bytes).

VAR_INPUT

```

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

pDpv1ResData: Pointer to an array of 240 bytes, which contains the DPV1 read response telegram. This telegram is being split in this function.

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
  
```

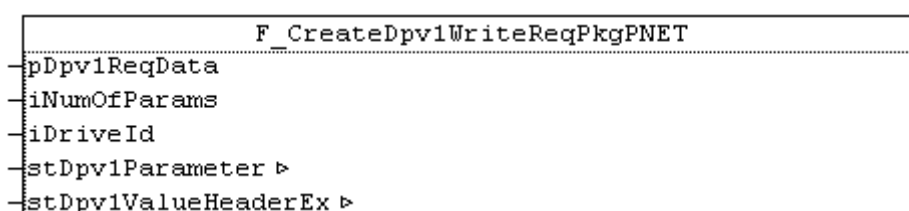
stDpv1Parameter: Array of 39 [parameters](#) [▶ 135], which have been added to the DPV1 read request telegram.

stDpv1ValueHeaderEx: Array of 39 [parameter](#) [▶ 137] values, which have to be read from the drive.

Requirements

Development environment	Target platform	IO hardware	PLC libraries to include
TwinCAT v2.11.0, Build > 1553 TwinCAT v2.11.0 R2, Build > 2024	PC (i386)	Beckhoff EL6632	TcloFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib linked automatically)

3.21.4 F_CreateDpv1WriteReqPkgPNET : USINT



The function "F_CreateDpv1WriteReqPkg" creates a DPV1 telegram for a [FB_Dpv1Write](#) [▶ 83] of one or multiple parameters of a drive or the controller unit of a Sinamics Profidrive (Profidrive specification 3.1) via Profinet. Since the profidrive uses the Motorola format and the IPC uses the Intel format, the function automatically swaps the bytes in the DPV1 telegram of parameters with data types bigger than one byte.

The function returns the current length of the DPV1 telegram in bytes (max. 240 bytes).

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: Pointer to an array of 240 bytes, which contains the DPV1 write request telegram. This telegram is being created in this function.

iNumOfParams: Number of parameters to be written (1 to 39). Another limitation is the telegram size to 240 bytes.

iDriveID: The drive ID is 1 for the Controller Unit, 2 for the A drive, 3 for the B drive of double/triple drive. Drive ID is set in the Starter software. 1..16 is possible.

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
```

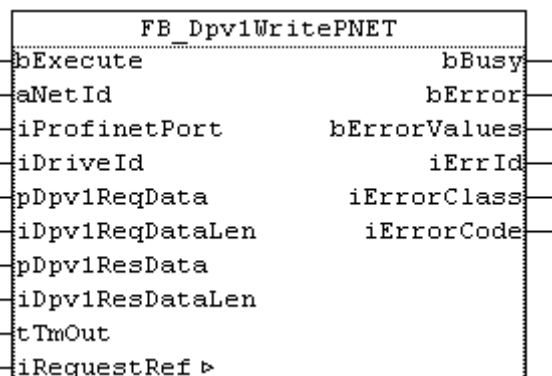
stDpv1Parameter: Array of 39 [parameters](#) [▶ 135], which have to be added to the DPV1 write request telegram.

stDpv1ValueHeaderEx: Array of 39 [parameter](#) [▶ 137] values, which have to be added to the DPV1 write request telegram.

Requirements

Development environment	Target platform	IO hardware	PLC libraries to include
TwinCAT v2.11.0, Build > 1553 TwinCAT v2.11.0 R2, Build > 2024	PC (i386)	Beckhoff EL6632	TcloFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib linked automatically)

3.21.5 FB_Dpv1WritePNET



The function block "FB_Dpv1WritePNET" writes one or multiple parameters to a Sinamics Profidrive via DPV1 (Profidrive specification 3.1) via Profinet. The DPV1 request telegram needs to be created with [F_CreateDpv1WriteReqPkgPNET \[► 82\]](#), before the bExecute sees a rising edge. The DPV1 response telegram needs to be split with [F_SplitDpv1WriteResPkgPNET \[► 86\]](#) after the bBusy sends a falling edge.

The execution of this function block takes some time, depending on the amount of parameters written. The function block sends the request telegram to the drive and then polls the drive for a corresponding response telegram.

Internally instances of the ADSREAD and ADSWRITE function blocks are called.

See sample <https://infosys.beckhoff.com/content/1033/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: The function block is activated by a positive edge at this input.

aNetId: Provide the AmsNetId of the Profibus Master device (see ADS-Tab of the Profibus Master device in the I/O configuration in the System Manager).

iProfinetPort: The profinet port number of the drive. This is one address for a multiple axes drive specified in the TwinCAT System Manager in the hardware configuration.

iDriveID: The drive ID is 1 for the ControllerUnit, 2 for the A drive, 3 for the B drive of double/triple drive. Drive ID is set in the Starter software. 1..16 is possible.

pDpv1ReqData: Pointer to an array of 240 bytes, which contains the DPV1 request telegram. This telegram needs to be created with [F_CreateDpv1WriteReqPkgPNET \[► 82\]](#) before the executing the DPV1 write.

iDv1ReqDataLen: Maximum length of the DPV1 request data buffer (240 bytes).

pDpv1ResData: Pointer to an array of 240 bytes, which contains the DPV1 response telegram after the DPV1 read. This DPV1 response telegram needs to be split with [F_SplitDpv1WriteResPkgPNET \[► 86\]](#) after the bBusy sends a falling edge.

iDv1ResDataLen: Maximum length of the DPV1 response data buffer (240 bytes).

tTimeout: States the length of the timeout that may not be exceeded by execution of the ADS command.

VAR_OUTPUT

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

bBusy: When the function block is activated this output is set. It remains set until an acknowledgement is received.

bError: If an ADS error should occur during the transfer of the command, then this output is set once the bBusy output is reset.

bErrorValues: Is TRUE if the DPV1 Read was not successful or partly not successful. The response telegram contains error values.

nErrId: Supplies the ADS error number or function block specific error code when the bError output is set.

nErrClass: Profidrive error class

nErrCode: Profidrive error code

VAR_IN_OUT

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

iRefRequest: Reference, which is increased automatically with each telegram. The reference is needed to assign the response telegram to the corresponding read/write request telegram.

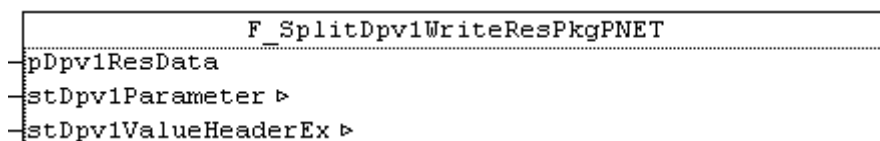
Function block specific error codes	Description
0x2	wrong reference response
0x3	DPV1 Read not successful or partly not successful
0x4	wrong response ID
other error IDs	see ADS return codes

Error classes	Description	Error code
0x0 - 0x9	reserved	-
0xA	Application error	0x0: read error 0x1: write error 0x2: module failure 0x3 - 0x7: reserved 0x8: version conflict 0x9: feature not supported 0xA - 0xF: user specific
0xB	Access error	0x0: invalid index (no datablock DB47, parameter requests are not supported) 0x1: write length error 0x2: invalid slot 0x3: type conflict 0x4: invalid area 0x5: state conflict (access to DB47 temporarily not possible due to internal processing status) 0x6: access denied 0x7: invalid range (write DB47 error in the DB47 header) 0x8: invalid parameter 0x9: invalid type 0xA - 0xF: user specific
0xC	Resource error	0x0: read constraint conflict 0x1: write constraint conflict 0x2: resource busy 0x3: resource unavailable 0x4 - 0x7: reserved 0x8 - 0xF: user specific
0xD - 0xF	User defined error	-

Requirements

Development environment	Target system type	IO hardware	PLC libraries to include
TwinCAT v2.11.0 Build > 1554 TwinCAT v2.11.0 R2 Build > 2024	PC (i386)	Beckhoff EL6632	TcIoFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib are included automatically)

3.21.6 F_SplitDpv1WriteResPkgPNET : USINT



The function "F_SplitDpv1WriteResPkg" parses a DPV1 telegram of a [FB_Dpv1Write](#) [▶ 83] for one or multiple parameters of a drive or the controller unit of a Sinamics Profidrive (Profidrive specification 3.1) via Profinet. Since the profidrive uses the Motorola format and the IPC uses the Intel format, the function automatically swaps the bytes in the DPV1 telegram of parameters with data types bigger than one byte.

The function returns the current length of the DPV1 telegram in bytes (max. 240 bytes).

VAR_INPUT

```

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

```

pDpv1ResData: Pointer to an array of 240 bytes, which contains the DPV1 read response telegram. This telegram is being split in this function.

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 of 39 [parameters](#) [▶ 135], which have been added to the DPV1 write request telegram.

stDpv1ValueHeaderEx: Array of 39 [parameter](#) [▶ 137] values, which have to be written to the drive.

Requirements

Development environment	Target platform	IO hardware	PLC libraries to include
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 linked automatically)

3.22 Beckhoff UPS (configured with Windows UPS Service)

3.22.1 FB_GetUPSStatus



Requirements:

a) The Beckhoff UPS software components are installed.

- Windows 10, Windows 7, Windows Embedded Standard 7 und höher: Configuration dialog under "Start->Programs->Beckhoff->UPS Software Components";
- NT4, Win2K, WinXP, WinXP embedded: Additional tab under "Control Panel->Energy Options->Beckhoff UPS Configuration" or under "Control Panel->Energy Options->UPS";
- Beckhoff CE devices with 24V UPS service are delivered with a special Beckhoff Battery Driver for Windows CE. The driver is included in the standard CE Image;

b) The UPS has been activated and configured. Further information about the UPS activation and configuration can be found in the considering device and UPS software documentation.

- Windows 7, Windows Embedded Standard 7 : Configuration dialog under "Start->Programs->Beckhoff->UPS Software Components";
- NT4, Win2K, WinXP, WinXP embedded: Configuration dialog under "Control Panel->Energy Options->Beckhoff UPS Configuration" or "Control Panel->Energy Options->UPS";
- Windows CE: At delivery time the UPS function is deactivated. The activation is done by execution of RegFiles. Newer images include configuration dialog under "Start->Control Panel->BECKHOFF UPS Configuration".

With the FB_GetUPSStatus function block the PLC can read the status of a UPS hardware. The block is level triggered, which means that the status information is only cyclically read while the *bEnable* input is set. To maintain system loading at a low level, the status information is only read approximately every 4.5 s. When the *bValid* output is set, the most recently read data is valid. The most recent read cycle was, in other words, executed without error. If an error occurs, the read cycle is repeated, and the error signal is automatically reset as soon as the cause of the error (e.g. no communication with the UPS) has been corrected.

VAR_INPUT

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

sNetId: It is possible here to provide the AmsNetId of the TwinCAT computer on which the function to be executed. If it is to be run on the local computer, an empty string can be entered.

nPort: ADS port number. Set this value to zero. Other port numbers are reserved for applications in future.

bEnable: If the input is set, the UPS status is read cyclically.

VAR_OUTPUT

```
VAR_OUTPUT
  bValid      :BOOL;
  bError      :BOOL;
```

```

    nErrId      :UDINT;
    stStatus    :ST_UPSStatus;
END_VAR

```

bValid: if this output is set, the data in the ST_UPS status structure are valid (no error occurred during the last reading cycle).

bError: this output is set if an error occurred when executing the function.

nErrId: supplies the ADS error number or the command-specific error code (table) when the *bError* output is set.

stStatus: [structure](#) [► 128] with the status information of the UPS.

Error Codes	Error description
0x0000	No error
0x8001	UPS configuration error. It is possible that the UPS is not configured correctly or that no UPS is configured at all.
0x8002	Communication error. Communication with the UPS was interrupted.
0x8003	Error during the reading of the status data.



Not every UPS device can supply all the status information. Some devices, for example, cannot supply the *BatteryLifeTime* or *BatteryReplace* status.

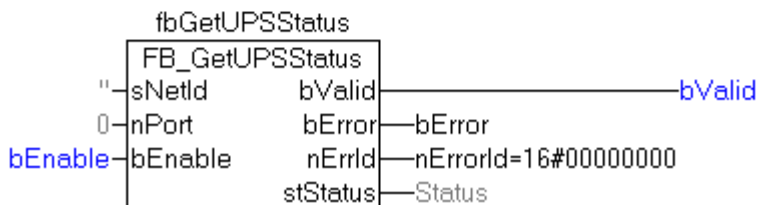
Example for a call in FBD:

Online data with status information of UPS:

```

▣ fbGetUPSStatus
├── Status
│   ├── Vendor = 'Beckhoff'
│   ├── Model = 'Beckhoff P24V250W'
│   ├── FirmwareRev = '11.7.1'
│   ├── 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

```

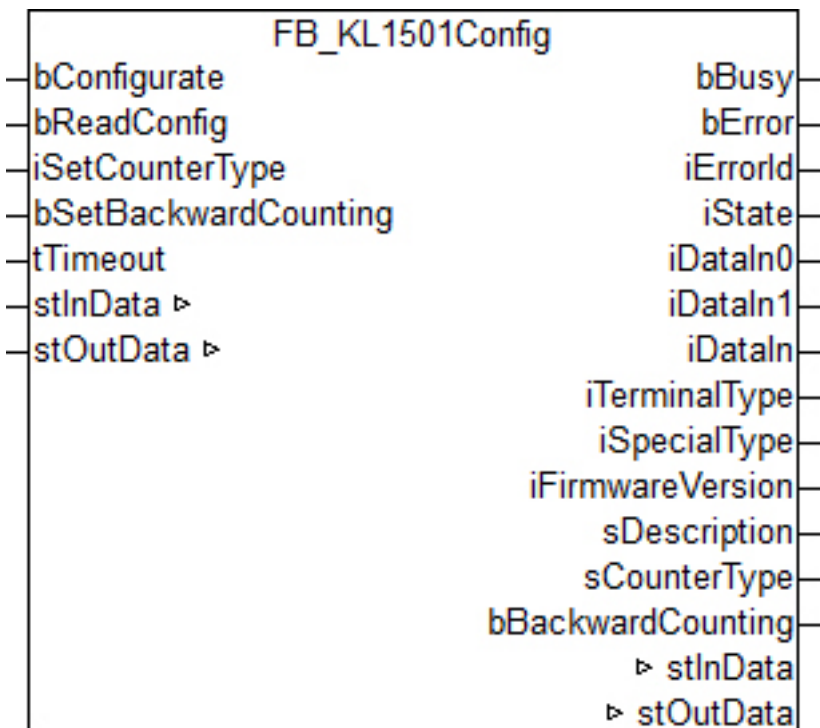


Requirements

UPS hardware	Target system type	Development environment	PLC libraries to include
<ul style="list-style-type: none"> • Beckhoff HID UPS; • Beckhoff BAPI v1; • Beckhoff P24Vxxxx; • Beckhoff CP903x card (PCI/ISA); • Beckhoff CX2100-09x4 models (e.g. CX2100-0904 or CX2100-0914 + "Smart Battery" CX2900-0192); • With Beckhoff Industrial PC's delivered APC UPS models supporting smart protocol and configured with Windows UPS Service; 	PC or CX	TwinCAT v2.8.0, Build > 745	TcIoFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib are included automatically)
		TwinCAT v2.9.0, Build > 945	
		TwinCAT v3.0 and above	Tc2_IoFunctions

3.23 Busterminal-Configuration

3.23.1 FB_KL1501Config



Function block for parameterizing a KL1501: 1-channel counter terminal.



This function block does **not** follow the alternative output format, since the process image shifts during conversion to this format.

VAR_INPUT

```

bConfigure      : BOOL;
bReadConfig    : BOOL;
iSetCounterType : INT;
bSetBackwardCounting : BOOL;
tTimeout       : TIME;

```

bConfigure: A rising edge at this input starts a configuration-sequence. At first the common terminal data like "terminal type", "special type" and "firmware version" will be read. Afterwards the internal registers will be set according to the configurations at the inputs of this function-block. To make sure, the configuration has been successful, the registers will be read and the information will be shown at the outputs of the FB, together with the common terminal-data. As long as this sequence is running, the output *bBusy* is set to TRUE and no other command, like *bReadConfig*, will be executed.

bReadConfig: A rising edge at this input starts the reading-sequence only. The common terminal data like "terminal type", "special type" and "firmware version" will be read together with the internal configuration-parameters. The read information will be shown at the FB-outputs afterwards. As long as this sequence is running, the output *bBusy* is set to TRUE and no other command, like *bConfigure*, will be executed.

iSetCounterType: At this input, the dimmer-mode is to be set according to the table below.

bSetBackwardCounting: A TRUE-signal at this input inverts the counting direction.

tTimeout: The commands *bConfigure* and *bReadConfig* have to be successfully executed within this time.

iSetCounterType	Counter type
0	32-bit up/down counter
1	2 x 16-bit up counter
2	32-bit gated counter, gate input low: counter is disabled
3	32-bit gated counter, gate input high: counter is disabled

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: If a reading- or configuration-sequence is being processed, this output will be set to TRUE.

bError: This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in *udiErrorId*.

iErrorId: Contains the command-specific error code of the most recently executed command. It is set back to 0 by the reactivation of the function block via the *bConfigure* or the *bReadConfig* input. See [Error codes](#) [► 118].

iState: This value corresponds to the state-variable of the process-data *stInData.iState*, see VAR_IN_OUT. During command-execution (*bBusy* = TRUE) this output is set to 0 and remains unchanged. It is thus suitable to evaluate the state of the terminal under normal conditions: disturbing values during configuration and reading due to register-communication will not be shown.

iDataIn0: This value corresponds to the process data *stInData.iDataIn0*, see VAR_IN_OUT. During command-execution (*bBusy* = TRUE) this output is set to the value previous to the execution and remains unchanged. It is thus suitable for direct process control: disturbing values during configuration and reading due to register-communication will not be shown.

iDataIn1: This value corresponds to the process data *stInData.iDataIn1*, see VAR_IN_OUT. During command-execution (*bBusy* = TRUE) this output is set to the value previous to the execution and remains unchanged. It is thus suitable for direct process control: disturbing values during configuration and reading due to register-communication will not be shown.

iDataIn: This UDINT-variable can be used, if a 32-bit-counter is selected. It consists of the variables *iDataIn0* and *iDataIn1* (both UINT), whereas *iDataIn0* is the less-counting part.

iTerminalType: Contents of register 8. If the right terminal is selected, this value should be 0x05DD (1501_{dec}).

iSpecialType: Contents of register 29: special-type.

iFirmwareVersion: Contents of register 9: firmware-version.

sDescription: Terminal-type, special-version and firmware-version put together to a readable string (e.g. 'Terminal KL1501-0000 / Firmware 1C').

sCounterType: Entered counter type as readable text.

bBackwardCounting: TRUE: The counting direction has been inverted.

VAR_IN_OUT

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

stInData: Reference to the [structure \[► 139\]](#) containing the input process image.

stOutData: Reference to the [structure \[► 140\]](#) containing the output process image.

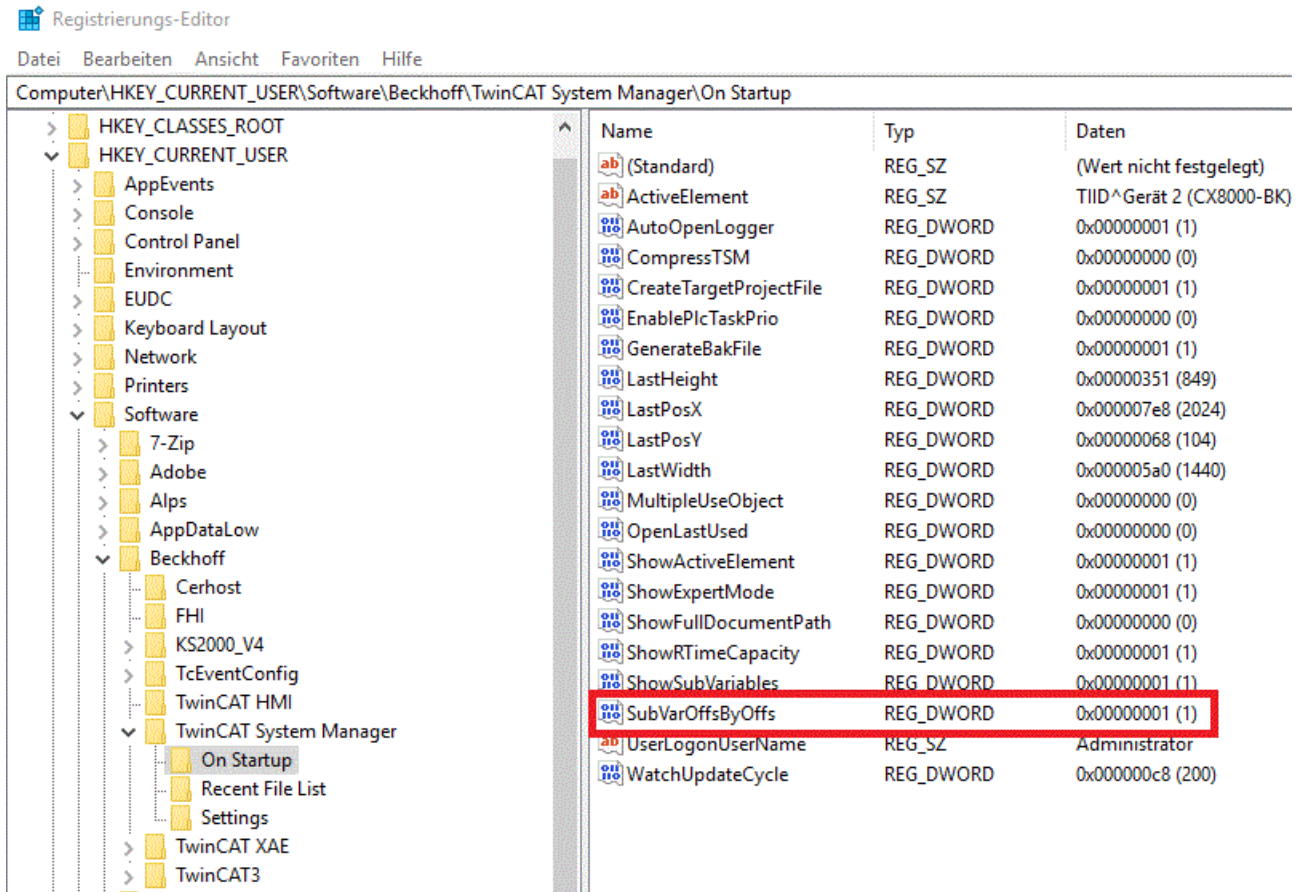
Background information

On ARM systems, the structure as a whole cannot be linked to the image of the terminal - the structure variables must be linked individually.

In addition, a new DWORD key must be created in the Windows registry of the development computer, otherwise memory image shifts will occur.

Under "HKEY_CURRENT_USER\Software\Beckhoff\TwinCAT System Manager\On Startup" the key "SubVarOffsByOffs" with the value "1" is inserted.

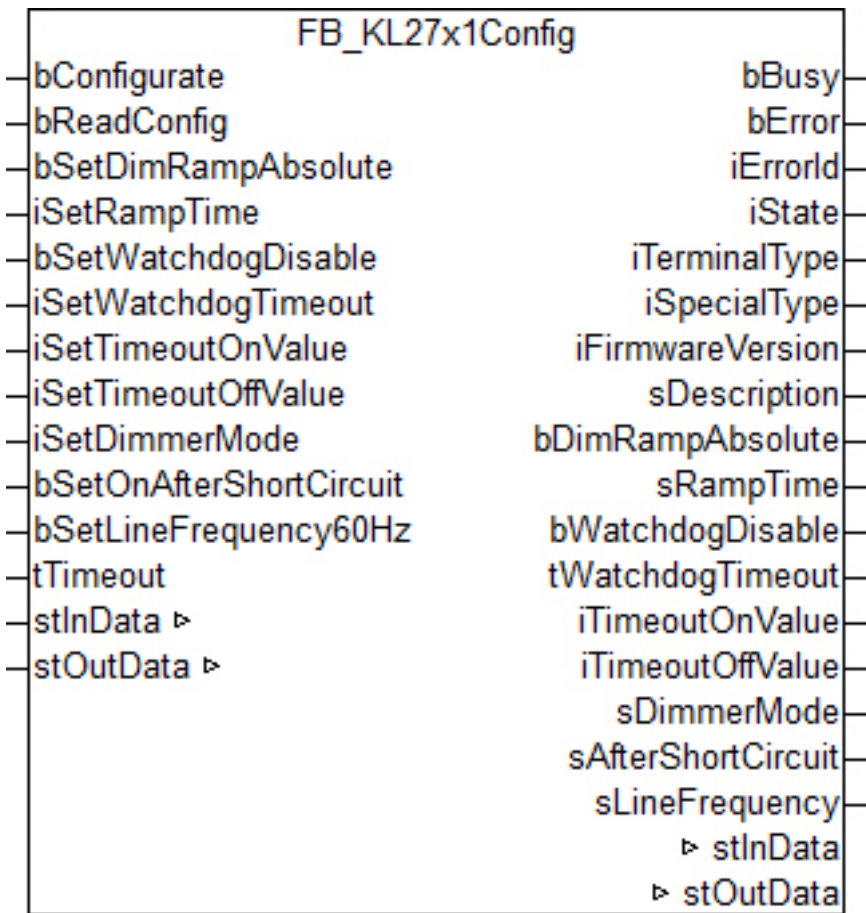
Afterwards restart the development computer once.



Requirements

Development Environment	Target System	IO Hardware	PLC Libraries to include
TwinCAT v2.11 R3/x64 from Build 2254	PC/CX	KL1501	TcloFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib are included automatically)
TwinCAT v2.11 R3/x64 from Build 2256	BC	KL1501	TcloFunctions.lb6 (Standard.lb6 will be included automatically)
TwinCAT v2.11 R3/x64 from Build 2256	BX	KL1501	TcloFunctions.lbx (Standard.lbx will be included automatically)

3.23.2 FB_KL27x1Config



Function block to parameterize a [KL2751](#) or a [KL2761](#): 1-channel dimming-terminal.

VAR_INPUT

```

bConfigure           : 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;
    
```

bConfigure: A rising edge at this input starts a configuration-sequence. At first the common terminal data like "terminal type", "special type" and "firmware version" will be read. Afterwards the internal registers will be set according to the configurations at the inputs of this function-block. To make sure, the configuration has been successful, the registers will be read and the information will be shown at the outputs of the FB, together with the common terminal-data. As long as this sequence is running, the output *bBusy* is set to TRUE and no other command, like *bReadConfig*, will be executed.

bReadConfig: A rising edge at this input starts the reading-sequence only. The common terminal data like "terminal type", "special type" and "firmware version" will be read together with the internal configuration-parameters. The read information will be shown at the FB-outputs afterwards. As long as this sequence is running, the output *bBusy* is set to TRUE and no other command, like *bConfigure*, will be executed.

bSetDimRampAbsolute: FALSE: the entered ramp-time *iSetRampTime* refers to the whole process-data-range (0 - 32767). Smaller jumps will take less time. TRUE: each jump, regardless of size, will take the same time: *iSetRampTime*.

iSetRampTime: Entry of the ramp-time with discrete values as listed in the table below.

bSetWatchdogDisable: The internal watchdog will be disabled.

iSetWatchdogTimeout: Setting the watchdog-time in multiples of 10ms.

iSetTimeoutOnValue: This entry specifies the light value that is issued for a fieldbus error and current process data > 0.

iSetTimeoutOffValue: This entry specifies the light value that is issued for a fieldbus error and current process data = 0.

iSetDimmerMode: At this input, the dimmer mode is to be set according to the table below.

bSetOnAfterShortCircuit: Behavior after a shortcut. FALSE: The light remains switched off. TRUE: The light will be turned on again.

bSetLineFrequency60Hz: This entry specifies the line frequency. FALSE: 50 Hz. TRUE: 60 Hz.

tTimeout: The commands *bConfigure* and *bReadConfig* have to be successfully executed within this time.

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	Automatic detection
1	Trailing edge control
2	Leading edge control
3	Rectifier mode, positive (positive half-wave with leading edge control)
4	Rectifier mode, negative (positive half-wave with leading edge control)

VAR_OUTPUT

```

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

```

bBusy: If a reading- or configuration-sequence is being processed, this output will be set to TRUE.

bError: This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in *udiErrorId*.

iErrorId: Contains the command-specific error code of the most recently executed command. It is set back to 0 by the reactivation of the function block via the *bConfigure* or the *bReadConfig* input. See [Error codes \[► 118\]](#).

iState: This value corresponds to the state-variable of the process-data *stInData.iState*, see VAR_IN_OUT. During command-execution (*bBusy* = TRUE) this output is set to 0 and remains unchanged. It is thus suitable to evaluate the state of the terminal under normal conditions: disturbing values during configuration and reading due to register-communication will not be shown.

iDataIn: This value corresponds to the process data *stInData.iDataIn*, see VAR_IN_OUT. During command-execution (*bBusy* = TRUE) this output is set to the value previous to the execution and remains unchanged. It is thus suitable for direct process control: disturbing values during configuration and reading due to register-communication will not be shown.

iTerminalType: Contents of register 8. If the right terminal is selected, this value should be either 0x0ABF (2751_{dec}) or 0x0AC9 (2761_{dec}).

iSpecialType: Contents of register 29: special-type.

iFirmwareVersion: Contents of register 9: firmware-version.

sDescription: Terminal-type, special-version and firmware-version put together to a readable string (e.g. 'Terminal KL27x1-0000 / Firmware 1C').

bDimRampAbsolute: TRUE: Dim ramp is set to absolute: every step, regardless of size will take the same time, which is entered under *iSetRampTime*.

sRampTime: Entered ramp-time.

bWatchdogDisable: TRUE: Watchdog is disabled.

tWatchdogTimeout: Entered watchdog-time.

iTimeoutOnValue: Entered light value that is issued for a fieldbus error and current process data > 0.

iTimeoutOffValue: Entered light value that is issued for a fieldbus error and current process data = 0.

sDimmerMode: Entered dimmer mode as readable text.

sAfterShortCircuit: Entered behavior after shortcut.

sLineFrequency: Entered line-frequency.

VAR_IN_OUT

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

stInData: Reference to the [structure \[► 140\]](#) containing the input process image.

stOutData: Reference to the [structure \[► 140\]](#) containing the output process image.

Background information

On ARM systems, the structure as a whole cannot be linked to the image of the terminal - the structure variables must be linked individually.

In addition, a new DWORD key must be created in the Windows registry of the development computer, otherwise memory image shifts will occur.

Under "HKEY_CURRENT_USER\Software\Beckhoff\TwinCAT System Manager\On Startup" the key "SubVarOffsByOffs" with the value "1" is inserted.

Afterwards restart the development computer once.

Registrierungs-Editor

Datei Bearbeiten Ansicht Favoriten Hilfe

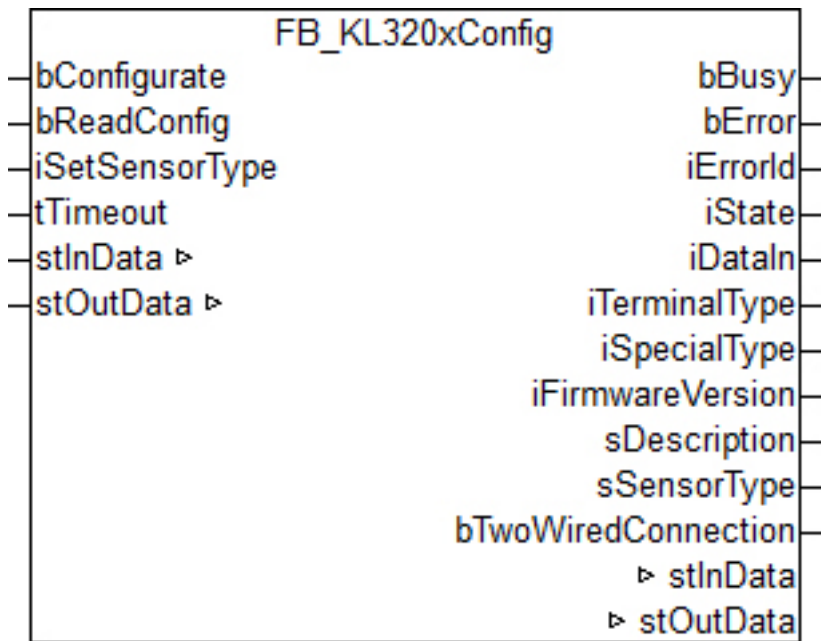
Computer\HKEY_CURRENT_USER\Software\Beckhoff\TwinCAT System Manager\On Startup

Name	Typ	Daten
(Standard)	REG_SZ	(Wert nicht festgelegt)
ActiveElement	REG_SZ	TIID^Gerät 2 (CX8000-BK)
AutoOpenLogger	REG_DWORD	0x00000001 (1)
CompressTSM	REG_DWORD	0x00000000 (0)
CreateTargetProjectFile	REG_DWORD	0x00000001 (1)
EnablePlcTaskPrio	REG_DWORD	0x00000000 (0)
GenerateBakFile	REG_DWORD	0x00000001 (1)
LastHeight	REG_DWORD	0x00000351 (849)
LastPosX	REG_DWORD	0x000007e8 (2024)
LastPosY	REG_DWORD	0x00000068 (104)
LastWidth	REG_DWORD	0x000005a0 (1440)
MultipleUseObject	REG_DWORD	0x00000000 (0)
OpenLastUsed	REG_DWORD	0x00000000 (0)
ShowActiveElement	REG_DWORD	0x00000001 (1)
ShowExpertMode	REG_DWORD	0x00000001 (1)
ShowFullDocumentPath	REG_DWORD	0x00000000 (0)
ShowRTTimeCapacity	REG_DWORD	0x00000001 (1)
ShowSubVariables	REG_DWORD	0x00000001 (1)
SubVarOffsByOffs	REG_DWORD	0x00000001 (1)
UserLogonUserName	REG_SZ	Administrator
WatchUpdateCycle	REG_DWORD	0x000000c8 (200)

Requirements

Development Environment	Target System	IO Hardware	PLC Libraries to include
TwinCAT v2.11 R3/x64 from Build 2254	PC/CX	KL2751, KL2761	TcloFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib are included automatically)
TwinCAT v2.11 R3/x64 from Build 2256	BC	KL2751, KL2761	TcloFunctions.lb6 (Standard.lb6 will be included automatically)
TwinCAT v2.11 R3/x64 from Build 2256	BX	KL2751, KL2761	TcloFunctions.lbx (Standard.lbx will be included automatically)

3.23.3 FB_KL320xConfig



Function block for parameterization of a KL3201, KL3202 or KL3204: input terminal for resistance sensors.

i The function block only parameterizes one terminal channel. For parameterizing all channels, the corresponding number of function blocks has to be instantiated. A mixed configuration (e.g. different sensor types) is possible.

VAR_INPUT

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

bConfigure: A rising edge at this input starts a configuration-sequence. At first the common terminal data like "terminal type", "special type" and "firmware version" will be read. Afterwards the internal registers will be set according to the configurations at the inputs of this function-block. To make sure, the configuration has been successful, the registers will be read and the information will be shown at the outputs of the FB, together with the common terminal-data. As long as this sequence is running, the output *bBusy* is set to TRUE and no other command, like *bReadConfig*, will be executed.

bReadConfig: A rising edge at this input starts the reading-sequence only. The common terminal data like "terminal type", "special type" and "firmware version" will be read together with the internal configuration-parameters. The read information will be shown at the FB-outputs afterwards. As long as this sequence is running, the output *bBusy* is set to TRUE and no other command, like *bConfigure*, will be executed.

iSetSensorType: At this input, the sensor type is to be set according to the table below.

tTimeout: The commands *bConfigure* and *bReadConfig* have to be successfully executed within this time.

iSetSensorType	Element
0	PT100
1	NI100
2	PT1000
3	PT500
4	PT200
5	NI1000
6	NI120
7	Output 10,0 Ω - 5000,0 Ω

iSetSensorType	Element
8	Output 10,0 Ω - 1200,0 Ω
9	PT1000 - two wired connection - not allowed, if a KL3204 is in use.

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: As long as a reading- or configuration-sequence is being processed, this output will be set to TRUE.

bError: This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in *udiErrorId*.

iErrorId: Contains the command-specific error code of the most recently executed command. It is set back to 0 by the reactivation of the function block via the *bConfigure* or the *bReadConfig* input. See [Error codes \[► 118\]](#).

iState: This value corresponds to the state-variable of the process-data *stInData.iState*, see VAR_IN_OUT. During command-execution (*bBusy* = TRUE) this output is set to 0 and remains unchanged. It is thus suitable to evaluate the state of the terminal under normal conditions: disturbing values during configuration and reading due to register-communication will not be shown.

iDataIn: This value corresponds to the process data *stInData.iDataIn*, see VAR_IN_OUT. During command-execution (*bBusy* = TRUE) this output is set to the value previous to the execution and remains unchanged. It is thus suitable for direct process control: disturbing values during configuration and reading due to register-communication will not be shown.

iTerminalType: Contents of register 8. If the right terminal is selected, this value should be 0xC81 for KL3201, 0xC82 for KL3202 and 0xC84 for KL3204.

iSpecialType: Contents of register 29: special-type.

iFirmwareVersion: Contents of register 9: firmware-version.

sDescription: Terminal-type, special-version and firmware-version put together to a readable string (e.g. 'Terminal KL27x1-0000 / Firmware 1C').

sSensorType: Entered sensor type as readable text.

bTwoWiredConnection: Two-wired connection was parameterized.

VAR_IN_OUT

```

stInData       : ST_KL320xInData;
stOutData      : ST_KL320xOutData;

```

stInData: Reference to the [structure \[► 142\]](#) containing the input process image.

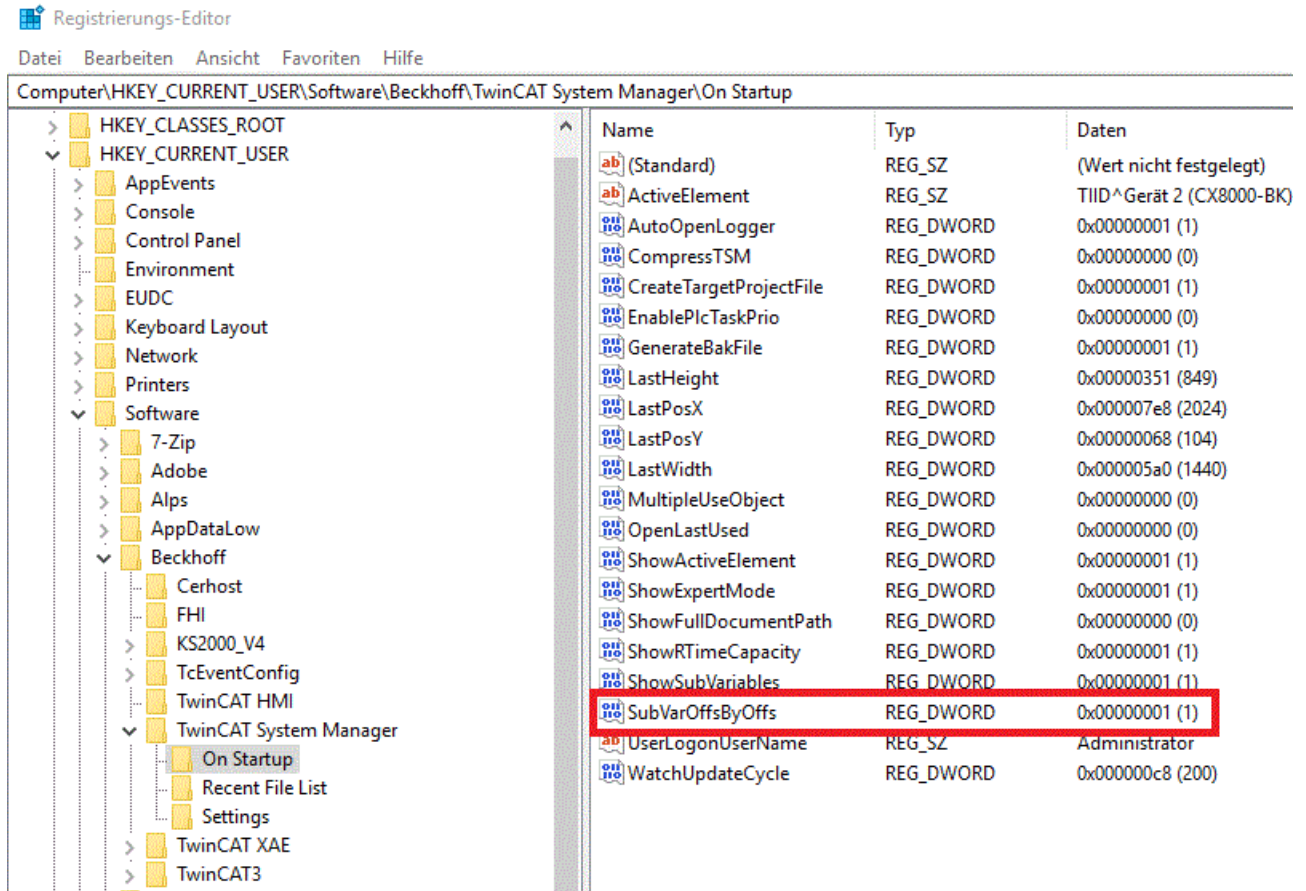
stOutData: Reference to the [structure \[► 142\]](#) containing the output process image.

Background information

On ARM systems, the structure as a whole cannot be linked to the image of the terminal - the structure variables must be linked individually.

In addition, a new DWORD key must be created in the Windows registry of the development computer, otherwise memory image shifts will occur.

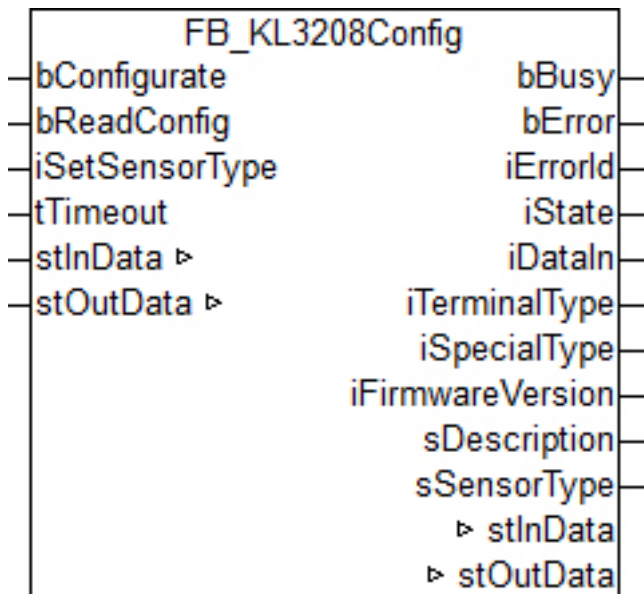
Under "HKEY_CURRENT_USER\Software\Beckhoff\TwinCAT System Manager\On Startup" the key "SubVarOffsByOffs" with the value "1" is inserted. Afterwards restart the development computer once.



Requirements

Development Environment	Target System	IO Hardware	PLC Libraries to include
TwinCAT v2.11 R3/x64 from Build 2254	PC/CX	KL3201, KL3202, KL3204	TcloFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib are included automatically)
TwinCAT v2.11 R3/x64 from Build 2256	BC	KL3201, KL3202, KL3204	TcloFunctions.lb6 (Standard.lb6 will be included automatically)
TwinCAT v2.11 R3/x64 from Build 2256	BX	KL3201, KL3202, KL3204	TcloFunctions.lbx (Standard.lbx will be included automatically)

3.23.4 FB_KL3208Config



Function block for parameterization of a [KL3208-0010](#): 8-channel input terminal for resistance sensors.



The function block only parameterizes one terminal channel. For parameterizing all channels, the corresponding number of function blocks has to be instantiated. A mixed configuration (e.g. different sensor types) is possible.

VAR_INPUT

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

bConfigure: A rising edge at this input starts a configuration-sequence. At first the common terminal data like "terminal type", "special type" and "firmware version" will be read. Afterwards the internal registers will be set according to the configurations at the inputs of this function-block. To make sure, the configuration has been successful, the registers will be read and the information will be shown at the outputs of the FB, together with the common terminal-data. As long as this sequence is running, the output *bBusy* is set to TRUE and no other command, like *bReadConfig*, will be executed.

bReadConfig: A rising edge at this input starts the reading-sequence only. The common terminal data like "terminal type", "special type" and "firmware version" will be read together with the internal configuration-parameters. The read information will be shown at the FB-outputs afterwards. As long as this sequence is running, the output *bBusy* is set to TRUE and no other command, like *bConfigure*, will be executed.

iSetSensorType: At this input, the sensor type is to be set according to the table below.

tTimeout: The commands *bConfigure* and *bReadConfig* have to be successfully executed within this time.

iSetSensorType	Element
0	PT1000
1	NI1000
2	RSNI1000 (NI1000 with Landis&Staefa characteristic: 1000 Ω at 0 °C and 1500 Ω at 100 °C)
3	NCT1K8
4	NCT1K8_TK
5	NCT2K2
6	NCT3K
7	NCT5K

iSetSensorType	Element
8	NTC10K
9	NTC10KPRE
10	NTC10K_3204
11	NTC10KTYP2
12	NTC10KTYP3
13	NTC10KDALE
14	NTC10K3A221
15	NTC20K
16	Poti, resolution 0,1 Ω
17	Poti, resolution 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: If a reading- or configuration-sequence is being processed, this output will be set to TRUE.

bError: This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in *udiErrorId*.

iErrorId: Contains the command-specific error code of the most recently executed command. It is set back to 0 by the reactivation of the function block via the *bConfigure* or the *bReadConfig* input. See [Error codes \[▶ 118\]](#).

iState: This value corresponds to the state-variable of the process-data *stInData.iState*, see VAR_IN_OUT. During command-execution (*bBusy* = TRUE) this output is set to 0 and remains unchanged. It is thus suitable to evaluate the state of the terminal under normal conditions: disturbing values during configuration and reading due to register-communication will not be shown.

iDataIn: This value corresponds to the process data *stInData.iDataIn*, see VAR_IN_OUT. During command-execution (*bBusy* = TRUE) this output is set to the value previous to the execution and remains unchanged. It is thus suitable for direct process control: disturbing values during configuration and reading due to register-communication will not be shown.

iTerminalType: Contents of register 8. If the right terminal is selected, this value should be 0x0C9C (3208_{dec}).

iSpecialType: Contents of register 29: special-type.

iFirmwareVersion: Contents of register 9: firmware-version.

sDescription: Terminal-type, special-version and firmware-version put together to a readable string (e.g. 'Terminal KL3208-0010 / Firmware 1C').

sSensorType: Entered sensor type as readable text.

VAR_IN_OUT

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

stInData: Reference to the [structure \[▶ 141\]](#) containing the input process image.

stOutData: Reference to the [structure \[▶ 141\]](#) containing the output process image.

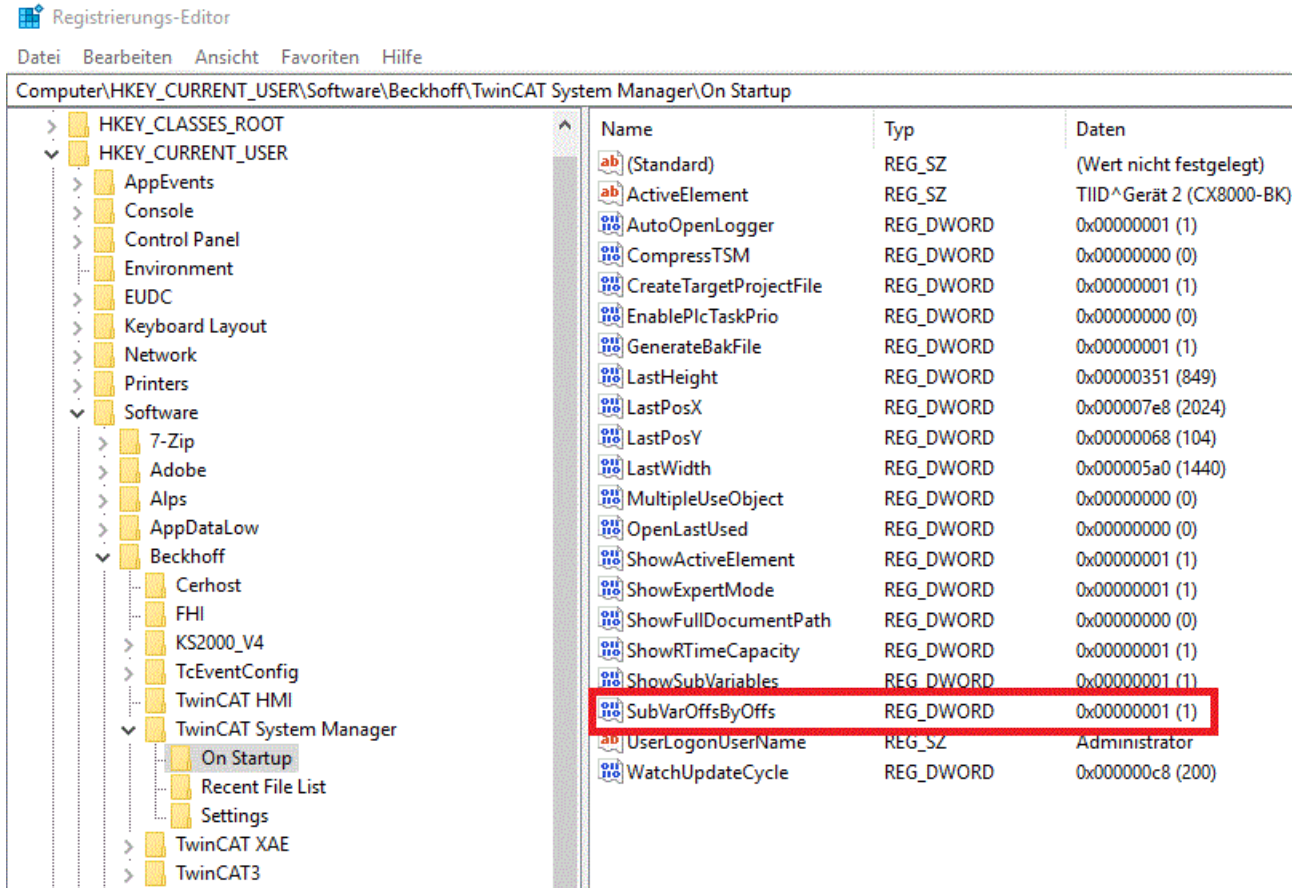
Background information

On ARM systems, the structure as a whole cannot be linked to the image of the terminal - the structure variables must be linked individually.

In addition, a new DWORD key must be created in the Windows registry of the development computer, otherwise memory image shifts will occur.

Under "HKEY_CURRENT_USER\Software\Beckhoff\TwinCAT System Manager\On Startup" the key "SubVarOffsByOffs" with the value "1" is inserted.

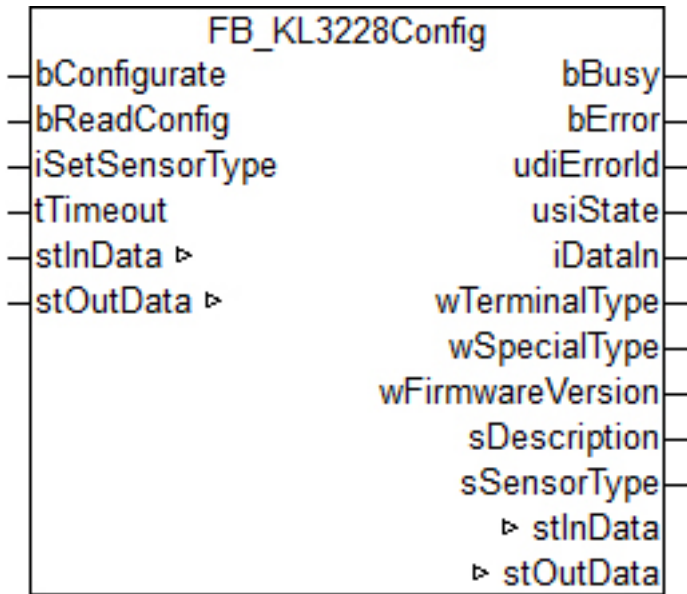
Afterwards restart the development computer once.



Requirements

Development Environment	Target System	IO Hardware	PLC Libraries to include
TwinCAT v2.11 R3/x64 from Build 2254	PC/CX	KL3208	TcloFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib are included automatically)
TwinCAT v2.11 R3/x64 from Build 2256	BC	KL3208	TcloFunctions.lb6 (Standard.lb6 will be included automatically)
TwinCAT v2.11 R3/x64 from Build 2256	BX	KL3208	TcloFunctions.lbx (Standard.lbx will be included automatically)

3.23.5 FB_KL3228Config



Function block for parameterization of a [KL3228](#): 8-channel input terminal for resistance sensors.

i The function block only parameterizes one terminal channel. For parameterizing all channels, the corresponding number of function blocks has to be instantiated. A mixed configuration (e.g. different sensor types) is possible.

VAR_INPUT

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

bConfigure: A rising edge at this input starts a configuration-sequence. At first the common terminal data like "terminal type", "special type" and "firmware version" will be read. Afterwards the internal registers will be set according to the configurations at the inputs of this function-block. To make sure, the configuration has been successful, the registers will be read and the information will be shown at the outputs of the FB, together with the common terminal-data. As long as this sequence is running, the output *bBusy* is set to TRUE and no other command, like *bReadConfig*, will be executed.

bReadConfig: A rising edge at this input starts the reading-sequence only. The common terminal data like "terminal type", "special type" and "firmware version" will be read together with the internal configuration-parameters. The read information will be shown at the FB-outputs afterwards. If this sequence is running, the output *bBusy* is set to TRUE and no other command, like *bConfigure*, will be executed.

iSetSensorType: At this input, the sensor type is to be set according to the table below.

tTimeout: The commands *bConfigure* and *bReadConfig* must be successfully executed within this time.

iSetSensorType	Element
0	PT1000
1	NI1000
2	RSNI1000 (NI1000 with Landis&Staefa characteristic: 1000 Ω at 0 °C and 1500 Ω at 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: If a reading- or configuration-sequence is being processed, this output will be set to TRUE.

bError: This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in *udiErrorId*.

iErrorId: Contains the command-specific error code of the most recently executed command. It is set back to 0 by the reactivation of the function block via the *bConfigure* or the *bReadConfig* input. See [Error codes](#) [[▶ 118](#)].

iState: This value corresponds to the state-variable of the process-data *stInData.iState*, see VAR_IN_OUT. During command-execution (*bBusy* = TRUE) this output is set to 0 and remains unchanged. It is thus suitable to evaluate the state of the terminal under normal conditions: disturbing values during configuration and reading due to register-communication will not be shown.

iDataIn: This value corresponds to the process data *stInData.iDataIn*, see VAR_IN_OUT. During command-execution (*bBusy* = TRUE) this output is set to the value previous to the execution and remains unchanged. It is thus suitable for direct process control: disturbing values during configuration and reading due to register-communication will not be shown.

iTerminalType: Contents of register 8. If the right terminal is selected, this value should be 0x0C88 (3228_{dec}).

iSpecialType: Contents of register 29: special-type.

iFirmwareVersion: Contents of register 9: firmware-version.

sDescription: Terminal-type, special-version and firmware-version put together to a readable string (e.g. 'Terminal KL3228-0000 / Firmware 1C').

sSensorType: Entered sensor type as readable text.

VAR_IN_OUT

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

stInData: Reference to the [structure](#) [[▶ 143](#)] containing the input process image.

stOutData: Reference to the [structure](#) [[▶ 143](#)] containing the output process image.

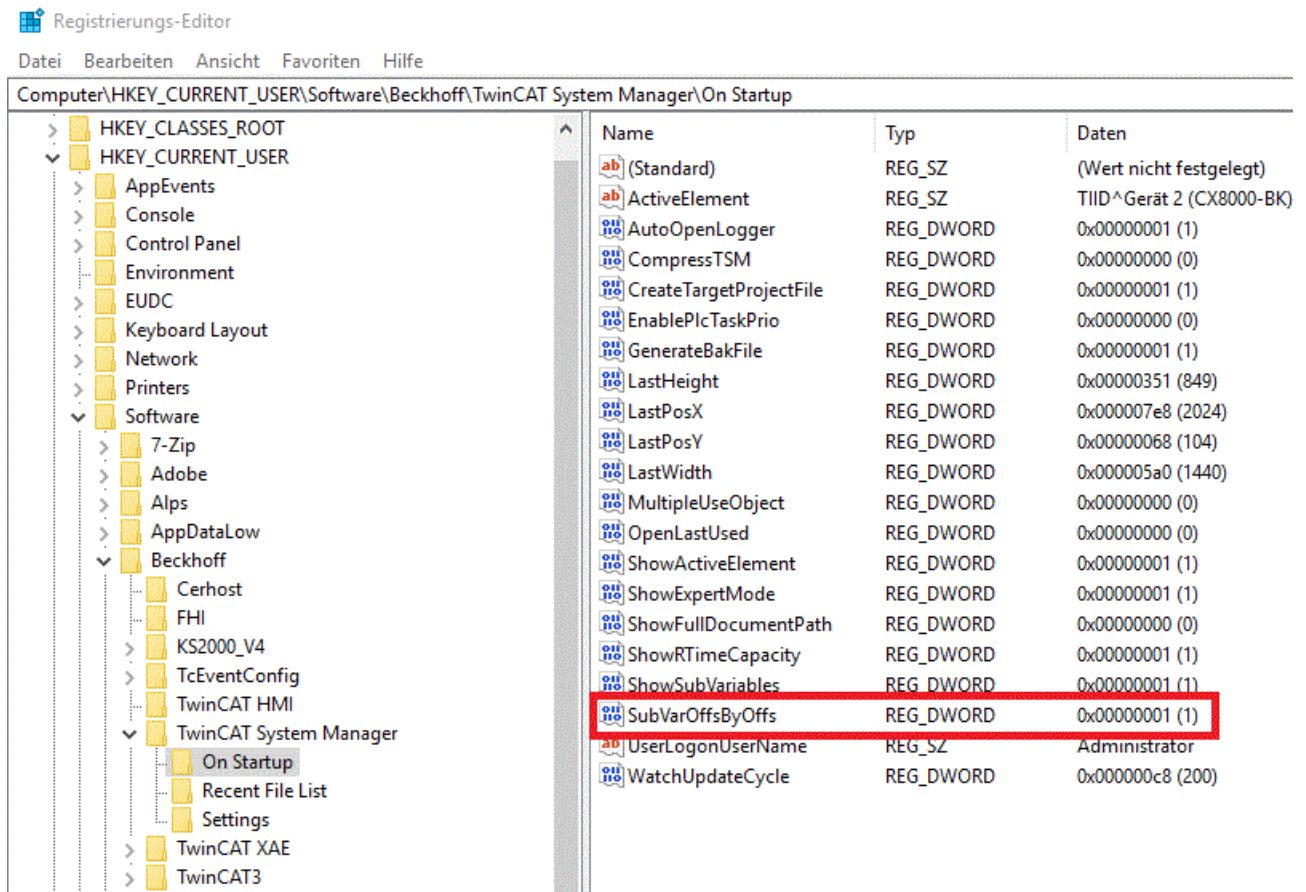
Background information

On ARM systems, the structure as a whole cannot be linked to the image of the terminal - the structure variables must be linked individually.

In addition, a new DWORD key must be created in the Windows registry of the development computer, otherwise memory image shifts will occur.

Under "HKEY_CURRENT_USER\Software\Beckhoff\TwinCAT System Manager\On Startup" the key "SubVarOffsByOffs" with the value "1" is inserted.

Afterwards restart the development computer once.



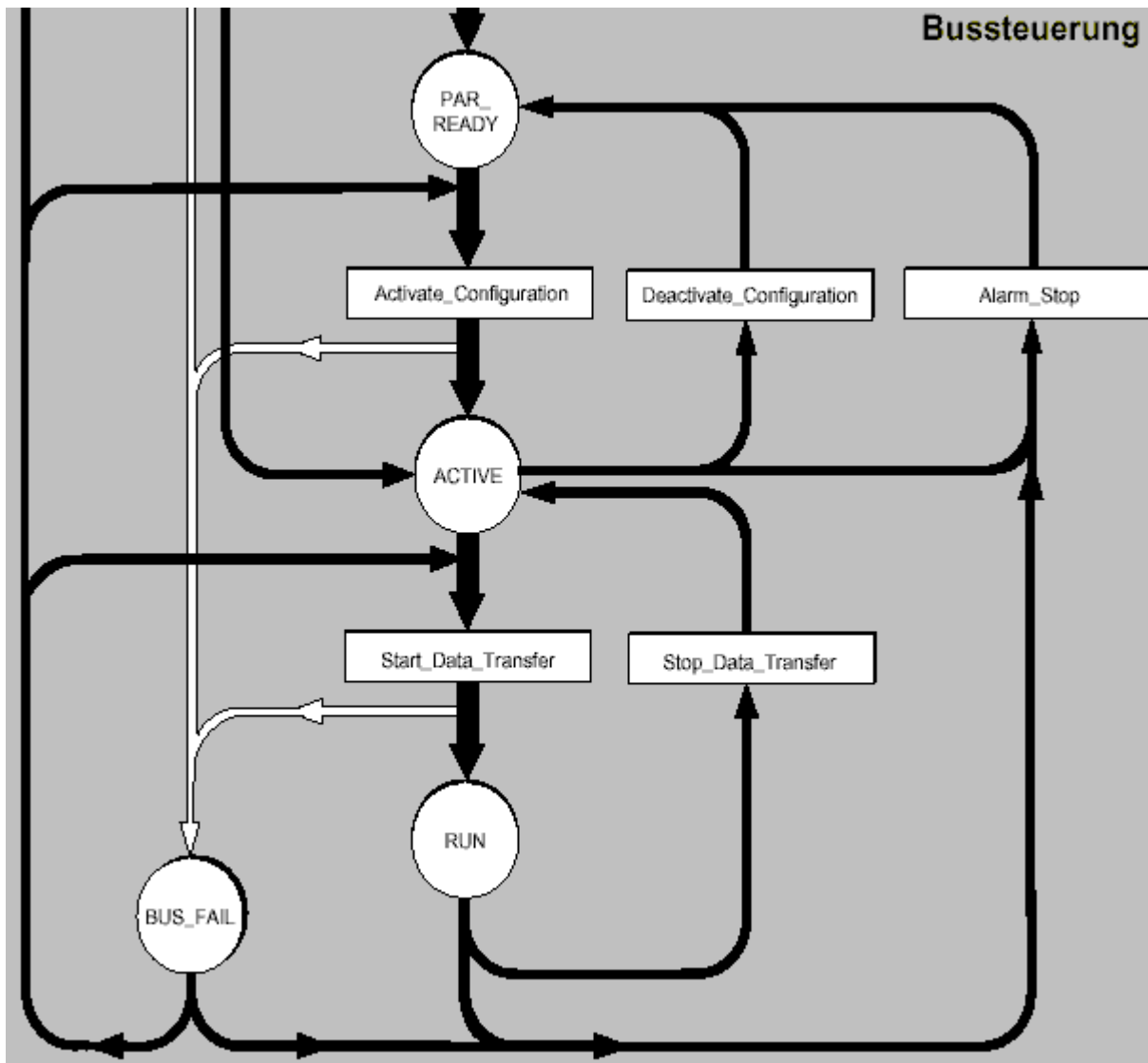
Requirements

Development Environment	Target System	IO Hardware	PLC Libraries to include
TwinCAT v2.11 R3/x64 from Build 2254	PC/CX	KL3228	TcloFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib are included automatically)
TwinCAT v2.11 R3/x64 from Build 2256	BC	KL3228	TcloFunctions.lb6 (Standard.lb6 will be included automatically)
TwinCAT v2.11 R3/x64 from Build 2256	BX	KL3228	TcloFunctions.lbx (Standard.lbx will be included automatically)

3.24 Third party devices

3.24.1 Phoenix IBS SC/I-T

The library offers a convenient way of executing the most important firmware services on the Phoenix Interbus IBS SC/I-T card for bus control from the TwinCAT PLC. The following diagram illustrates the states and the transition conditions of the bus control.

Bus Control

SCIT_ActivateConfiguration [▶ 107] : Executes the **Activate_Configuration** command.

SCIT_DeactivateConfiguration [▶ 108] : Executes the **Deactivate_Configuration** command.

SCIT_StartDataTransfer [▶ 109] : Executes the **Start_Data_Transfer** command.

SCIT_StopDataTransfer [▶ 110] : Executes the **Stop_Data_Transfer** command.

SCIT_AlarmStop [▶ 111] : Executes the **Alarm_Stop** command.

Configuration

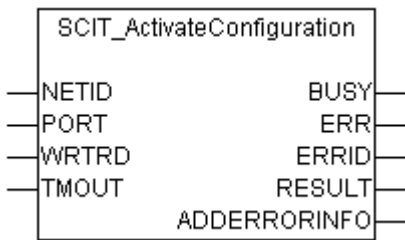
SCIT_ControlActiveConfiguration [▶ 113] : Is used to affect the active configuration of the Interbus devices. This command can be executed in the *PAR_READY* state as well as when in the *ACTIVE* or *RUN* states. Single, dependent and grouped devices can be activated and deactivated in this way.

Error Diagnosis

SCIT_GetErrorInfo [▶ 114]: Returns the error type and error location of an Interbus device after a bus error.

SCIT_ConfDevErrAll [▶ 115]: Confirms peripheral errors of all Interbus devices.

3.24.1.1 SCIT_ActivateConfiguration



The "SCIT_ActivateConfiguration" function block serves as an auxiliary block in order to carry out an **Activate Configuration** on the Interbus card that is addressed by the NETID and the PORT. Internally, an ADSRDWRT function block is called, which has the parameters described in the TwinCAT System Manager Help.

An **Activate Configuration** sets the card in the ACTIVE [► 106] state.

VAR_INPUT

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

NETID: The AmsNetId of the computer into which the card is fitted can be given here. If the card is located within the same system, it is also possible to give an empty string.

PORT: This contains the ADS port number of the card. The number is assigned in the System Manager.

WRTRD: The function block is activated by a positive edge at this value.

TMOUT: This gives the timeout duration that is passed on to the internal ADSWRTRD block.

VAR_OUTPUT

```
VAR_OUTPUT
    BUSY       : BOOL;
    ERR        : BOOL;
    ERRID      : UDINT;
    RESULT     : WORD;
    ADDERRINFO : WORD;
END_VAR
```

BUSY: After activation of the function block the busy signal remains asserted until a feedback is received.

ERR: If an ADS error occurs while the command is being transferred, the ERR output is set once the BUSY signal is cancelled.

ERRID: Supplies the ADS error number when ADS error is true.

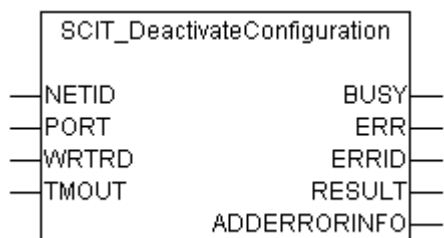
RESULT: Returns the result from the card (provided that fault-free ADS transport has occurred (ERR = FALSE)). RESULT = 0 identifies successful execution of the command. A value other than 0 contains the error number from the Phoenix card.

ADDERRINFO: If feedback from the card is negative, this contains additional error information (cf. the Phoenix card error descriptions).

Requirements

Development environment	Target system type	IO-Hardware	PLC libraries to include
TwinCAT v2.7.0	PC (i386)	Phoenix: IBS SC/I-T; IBS SC/RI/RT-LK	Standard.Lib; PLCSystem.Lib; TcPLCAds.Lib; TcPloFunctions.Lib
TwinCAT v2.8.0	PC (i386)	Phoenix: IBS SC/I-T; IBS SC/RI/RT-LK	TcloFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib are included automatically)

3.24.1.2 SCIT_DeactivateConfiguration



The "SCIT_DeactivateConfiguration" function block serves as an auxiliary block in order to carry out a **Deactivate_Configuration** on the Interbus card that is addressed by the NETID and the PORT. Internally, an ADSRDWRT function block is called, which has the parameters described in the TwinCAT System Manager Help.

An **Deactivate_Configuration** places the card in the PAR_READY [▶ 106] state and resets all the outputs.

VAR_INPUT

```

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

NETID: The AmsNetId of the computer into which the card is fitted can be given here. If the card is located within the same system, it is also possible to give an empty string.

PORT: This contains the ADS port number of the card. The number is assigned in the System Manager.

WRTRD: The function block is activated by a positive edge at this value.

TMOUT: This gives the timeout duration that is passed on to the internal ADSWRTRD block.

VAR_OUTPUT

```

VAR_OUTPUT
  BUSY       : BOOL;
  ERR        : BOOL;
  ERRID      : UDINT;
  RESULT     : WORD;
  ADDERRINFO : WORD;
END_VAR
  
```

BUSY: After activation of the function block the busy signal remains asserted until an feedback is received.

ERR: If an ADS error occurs while the command is being transferred, the ERR output is set once the BUSY signal is cancelled.

ERRID: Supplies the ADS error number when ADS error is true.

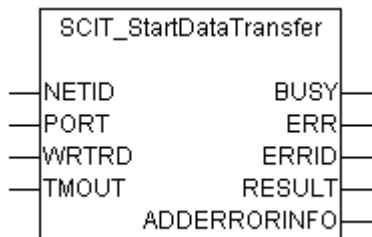
RESULT: Returns the result from the card (provided that fault-free ADS transport has occurred (ERR = FALSE)). RESULT = 0 identifies successful execution of the command. A value other than 0 contains the error number from the Phoenix card.

ADDERRINFO: If feedback from the card is negative, this contains additional error information (cf. the Phoenix card error descriptions).

Requirements

Development environment	Target system type	IO-Hardware	PLC libraries to include
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 are included automatically)

3.24.1.3 SCIT_StartDataTransfer



The "SCIT_StartDataTransfer" function block serves as an auxiliary block in order to carry out a **Start_Data_Transfer** on the Interbus card that is addressed by the NETID and the PORT. Internally, an ADSRDWRT function block is called, which has the parameters described in the TwinCAT System Manager Help.

A **Start_Data_Transfer** places the card into the RUN [▶_106] state.

VAR_INPUT

```

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

NETID: The AmsNetId of the computer into which the card is fitted can be given here. If the card is located within the same system, it is also possible to give an empty string.

PORT: This contains the ADS port number of the card. The number is assigned in the System Manager.

WRTRD: The function block is activated by a positive edge at this value.

TMOUT: This gives the timeout duration that is passed on to the internal ADSWRTRD block.

VAR_OUTPUT

```

VAR_OUTPUT
  BUSY      : BOOL;
  ERR       : BOOL;
  ERRID     : UDINT;
  RESULT    : WORD;
  ADDERRINFO : WORD;
END_VAR

```

BUSY: After activation of the function block the busy signal remains asserted until a feedback is received.

ERR: If an ADS error occurs while the command is being transferred, the ERR output is set once the BUSY signal is cancelled.

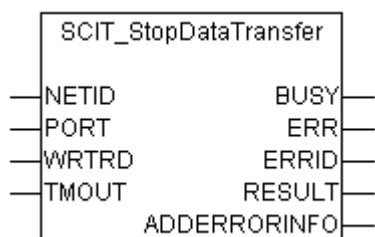
ERRID: Supplies the ADS error number when ADS error is true.

RESULT: Returns the result from the card (provided that fault-free ADS transport has occurred (ERR = FALSE)). RESULT = 0 identifies successful execution of the command. A value other than 0 contains the error number from the Phoenix card.

ADDERRINFO: If feedback from the card is negative, this contains additional error information (cf. the Phoenix card error descriptions).

Requirements

Development environment	Target system type	IO-Hardware	PLC libraries to include
TwinCAT v2.7.0	PC (i386)	Phoenix: IBS SC/I-T; IBS SC/RI/RT-LK	Standard.Lib; PLCSystem.Lib; TcPLCAds.Lib; TcPcIoFunctions.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 are included automatically)

3.24.1.4 SCIT_StopDataTransfer

The "SCIT_StopDataTransfer" function block serves as an auxiliary block in order to carry out a **Stop_Data_Transfer** on the Interbus card that is addressed by the NETID and the PORT. Internally, an ADSRDWRT function block is called, which has the parameters described in the TwinCAT System Manager Help.

A **Stop_Data_Transfer** places the card into the ACTIVE state [[▶ 106](#)], but the outputs are *not* reset.

VAR_INPUT

```

VAR_INPUT
  NETID     : T_AmsNetId;
  PORT      : T_AmsPort;

```

```

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

NETID: The AmsNetId of the computer into which the card is fitted can be given here. If the card is located within the same system, it is also possible to give an empty string.

PORT: This contains the ADS port number of the card. The number is assigned in the System Manager.

WRTRD: The function block is activated by a positive edge at this value.

TMOUT: This gives the timeout duration that is passed on to the internal ADSWRTRD block.

VAR_OUTPUT

```

VAR_OUTPUT
  BUSY      : BOOL;
  ERR       : BOOL;
  ERRID     : UDINT;
  RESULT    : WORD;
  ADDERRINFO : WORD;
END_VAR
    
```

BUSY: After activation of the function block the busy signal remains asserted until an feedback is received.

ERR: If an ADS error occurs while the command is being transferred, the ERR output is set once the BUSY signal is cancelled.

ERRID: Supplies the ADS error number when ADS error is true.

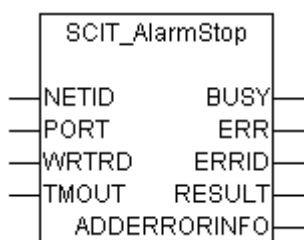
RESULT: Returns the result from the card (provided that fault-free ADS transport has occurred (ERR = FALSE)). RESULT = 0 identifies successful execution of the command. A value other than 0 contains the error number from the Phoenix card.

ADDERRINFO: If feedback from the card is negative, this contains additional error information (cf. the Phoenix card error descriptions).

Requirements

Development environment	Target system type	IO-Hardware	PLC libraries to include
TwinCAT v2.7.0	PC (i386)	Phoenix: IBS SC/I-T; IBS SC/RI/RT-LK	Standard.Lib; PLCSystem.Lib; TcPLCAds.Lib; TcPcIoFunctions.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 are included automatically)

3.24.1.5 SCIT_AlarmStop



The "SCIT_AlarmStop" function block serves as an auxiliary block in order to carry out an **Alarm_Stop** on the Interbus card that is addressed by the NETID and the PORT. Internally, an ADSRDWRT function block is called, which has the parameters described in the TwinCAT System Manager Help.

An **Alarm_Stop** places the card in the **PAR_READY** [▶ 106] state and resets all the outputs.

VAR_INPUT

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

NETID: The AmsNetId of the computer into which the card is fitted can be given here. If the card is located within the same system, it is also possible to give an empty string.

PORT: This contains the ADS port number of the card. The number is assigned in the System Manager.

WRTRD: The function block is activated by a positive edge at this value.

TMOUT: This gives the timeout duration that is passed on to the internal ADSWRTRD block.

VAR_OUTPUT

```
VAR_OUTPUT
  BUSY       : BOOL;
  ERR        : BOOL;
  ERRID      : UDINT;
  RESULT     : WORD;
  ADDERRINFO : WORD;
END_VAR
```

BUSY: After activation of the function block the busy signal remains asserted until a feedback is received.

ERR: If an ADS error occurs while the command is being transferred, the ERR output is set once the BUSY signal is cancelled.

ERRID: Supplies the ADS error number when ADS error is true.

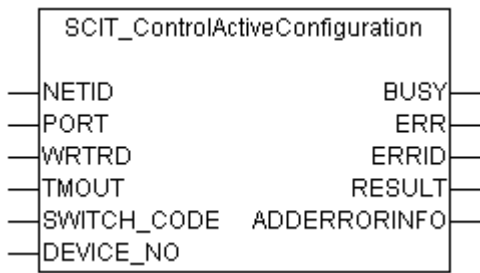
RESULT: Returns the result from the card (provided that fault-free ADS transport has occurred (ERR = FALSE)). RESULT = 0 identifies successful execution of the command. A value other than 0 contains the error number from the Phoenix card.

ADDERRINFO: If feedback from the card is negative, this contains additional error information (cf. the Phoenix card error descriptions).

Requirements

Development environment	Target system type	IO-Hardware	PLC libraries to include
TwinCAT v2.7.0	PC (i386)	Phoenix: IBS SC/I-T; IBS SC/RI/RT-LK	Standard.Lib; PLCSystem.Lib; TcPLCAds.Lib; TcPcIoFunctions.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 are included automatically)

3.24.1.6 SCIT_ControlActiveConfiguration



The "SCIT_ControlActiveConfiguration" function block serves as an auxiliary block in order to carry out a **Control_Active_Configuration** on the Interbus card that is addressed by the NETID and the PORT. Internally, an ADSRDWRT function block is called, which has the parameters described in the TwinCAT System Manager Help.

A **Control_Active_Configuration** can alter the state of a device (or of a number of devices, if the given device is a member of a group).

VAR_INPUT

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

NETID: The AmsNetId of the computer into which the card is fitted can be given here. If the card is located within the same system, it is also possible to give an empty string.

PORT: This contains the ADS port number of the card. The number is assigned in the System Manager.

WRTRD: The function block is activated by a positive edge at this value.

TMOUT: This gives the timeout duration that is passed on to the internal ADSWRTRD block.

SWITCH_CODE: Indicates the action that is to be executed with the device:

0 = Segment Off

1 = Segment On

2 = Device_Off

3 = Device_On

4 = Device_Disable

5 = Device_Enable

DEVICE_NO: Gives the device number of the addressed device. For example, device 3.1 requires a value of 16#0301 to be given.

VAR_OUTPUT

```
VAR_OUTPUT
  BUSY       : BOOL;
  ERR        : BOOL;
  ERRID      : UDINT;
  RESULT     : WORD;
  ADDERRINFO : WORD;
END_VAR
```

BUSY: After activation of the function block the busy signal remains asserted until an feedback is received.

ERR: If an ADS error occurs while the command is being transferred, the ERR output is set once the BUSY signal is cancelled.

ERRID: Supplies the ADS error number when ADS error is true.

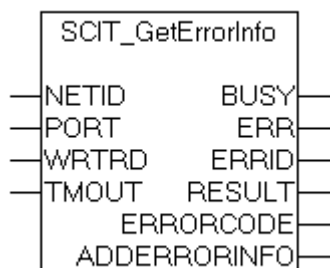
RESULT: Returns the result from the card (provided that fault-free ADS transport has occurred (ERR = FALSE)). RESULT = 0 identifies successful execution of the command. A value other than 0 contains the error number from the Phoenix card.

ADDERRINFO: If feedback from the card is negative, this contains additional error information (cf. the Phoenix card error descriptions).

Requirements

Development environment	Target system type	IO-Hardware	PLC libraries to include
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 are included automatically)

3.24.1.7 SCIT_GetErrorInfo



The function block "SCIT_GetErrorInfo" reads the exact error cause and the precise location of a bus error that has occurred from the Interbus card addressed by the NETID and the PORT. Internally, an ADSRDWRT function block is called, which has the parameters described in the TwinCAT System Manager Help.

VAR_INPUT

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

NETID: The AmsNetId of the computer into which the card is fitted can be given here. If the card is located within the same system, it is also possible to give an empty string.

PORT: This contains the ADS port number of the card. The number is assigned in the System Manager.

WRTRD: The function block is activated by a positive edge at this value.

TMOUT: This gives the timeout duration that is passed on to the internal ADSWRTRD block.

VAR_OUTPUT

```
VAR_OUTPUT
  BUSY      : BOOL;
  ERR       : BOOL;
  ERRID     : UDINT;
  RESULT    : WORD;
  ERRORCODE : WORD;
  ADDERRORINFO: WORD;
END_VAR
```

BUSY: After activation of the function block the busy signal remains asserted until an feedback is received.

ERR: If an ADS error occurs while the command is being transferred, the ERR output is set once the BUSY signal is cancelled.

ERRID: Supplies the ADS error number when ADS error is true.

RESULT: Returns the result from the card (provided that fault-free ADS transport has occurred (ERR = FALSE)). RESULT = 0 identifies successful execution of the command. A value other than 0 contains the error number from the Phoenix card.

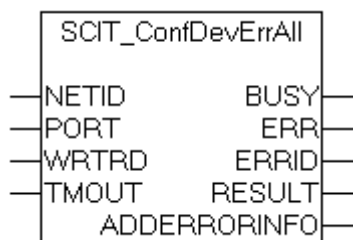
ERRORCODE: Provides information related to the error type (cf. the Phoenix card error descriptions).

ADDERRORINFO: Contains the error location if feedback from the card is negative (cf. the Phoenix card error descriptions).

Requirements

Development environment	Target system type	IO-Hardware	PLC libraries to include
TwinCAT v2.7.0	PC (i386)	Phoenix: IBS SC/I-T; IBS SC/RI/RT-LK	Standard.Lib; PLCSystem.Lib; TcPLCAds.Lib; TcPcIoFunctions.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 are included automatically)

3.24.1.8 SCIT_ConfDevErrAll



The "SCIT_ConfDevErrAll" function block confirms peripheral errors of all interbus devices. Internally, an ADSRDWRT function block is called, which has the parameters described in the TwinCAT System Manager Help.

A **Control_Active_Configuration** can alter the state of a device (or of several devices, if the given device is a member of a group).

VAR_INPUT

```

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

```

NETID: The AmsNetId of the computer into which the card is fitted can be given here. If the card is located within the same system, it is also possible to give an empty string.

PORT: This contains the ADS port number of the card. The number is assigned in the System Manager.

WRTRD: The function block is activated by a positive edge at this value.

TMOUT: This gives the timeout duration that is passed on to the internal ADSWRTRD block.

VAR_OUTPUT

```

VAR_OUTPUT
  BUSY       : BOOL;
  ERR        : BOOL;
  ERRID      : UDINT;
  RESULT     : WORD;
  ADDERRORINFO : WORD;
END_VAR

```

BUSY: After activation of the function block the busy signal remains asserted until a feedback is received.

ERR: If an ADS error occurs while the command is being transferred, the ERR output is set once the BUSY signal is cancelled.

ERRID: Supplies the ADS error number when ADS error is true.

RESULT: Returns the result from the card (provided that fault-free ADS transport has occurred (ERR = FALSE)). RESULT = 0 identifies successful execution of the command. A value other than 0 contains the error number from the Phoenix card.

ADDERRORINFO: If feedback from the card is negative, this contains additional error information (cf. the Phoenix card error descriptions).

Requirements

Development environment	Target system type	IO-Hardware	PLC libraries to include
TwinCAT v2.7.0	PC (i386)	Phoenix: IBS SC/I-T; IBS SC/RI/RT-LK	Standard.Lib; PLCSystem.Lib; TcPLCAds.Lib; TcPcIoFunctions.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 are included automatically)

3.24.2 ads-tec

3.24.2.1 FB_ReadAdsTecSysData

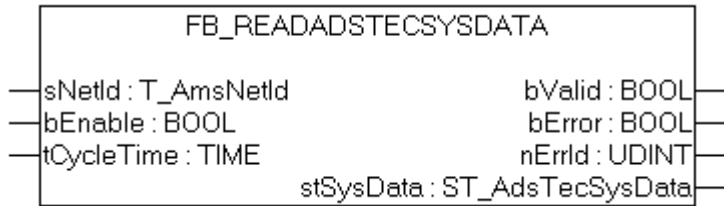


Fig. 2: FB_ReadAdsTecSysData

The *FB_ReadAdsTecSysData* function block allows the system and/or diagnostic data of an ads-tec Industrial PC to be read. The block is level triggered, which means that cyclic reading of the system data only occurs while the *bEnable* input is set. So that this only results in a low level of system loading, the read cycle is automatically repeated about every 100 ms (default value). When the *bValid* output is set, the most recently read data is valid (i.e. the last read cycle was carried out without error). If an error occurs, the *bError* output is set, and cyclic reading is halted. A new rising edge at the *bEnable* input can clear existing errors and restarts cyclic reading.

VAR_INPUT

```
VAR_INPUT
    sNetId      :T_AmsNetId;
    bEnable     :BOOL;
    tCycleTime  :TIME := T#100ms;
END_VAR
```

sNetId: It is possible here to provide the AmsNetId of the TwinCAT computer on whose function data should be read. If it is to be run on the local computer, an empty string can be entered.

bEnable: The block is reset with a rising edge (prior errors at the output *bError* and *nErrId* are deleted). If the input is set, the system data are read cyclically.

tCycleTime: Cyclic read interval.

VAR_OUTPUT

```
VAR_OUTPUT
    bValid      :BOOL;
    bError      :BOOL;
    nErrId      :UDINT;
    stSysData   :ST_AdsTecSysData;
END_VAR
```

bValid: If this output is set the data in the *ST_AdsTecSysData* structure are valid (no error occurs at the last read cycle).

bError: if an error occurs during the execution of a command, this output is set. The error is reset with a rising edge at the *bEnable* input.

nErrId: Supplies the Ads error number when *bError* output is set.

stSysData : [Structure \[► 134\]](#) with system data/ diagnosis data.

Requirements

Development environment	Target System	IO Hardware	PLC Libraries to include
TwinCAT v2.8.0, Build > 746	PC (i386)	ads-tec PC	TcloFunctions.Lib
TwinCAT v2.9.0, Build > 945			

Development environment	Target System	IO Hardware	PLC Libraries to include
			(Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib are included automatically)

3.25 Errorcodes

ErrId (hex)	ErrId (dec)	Description
0x0	0	no error.
0x8001	32769	Timeout-error during configuration.
0x8002	32770	The configuration-FB and the terminal, which it is linked to, do not match.
0x8010	32784	FB_KL1501Config: Invalid counter type selected. See input <i>iSetCounterType</i> .
0x8011	32785	FB_KL1501Config: Invalid counter type readouted. See output <i>sCounterType</i> .
0x8018	32792	FB_KL27x1Config: Invalid ramp time selected. See input <i>iSetRampTime</i> .
0x8019	32793	FB_KL27x1Config: Invalid dimmer mode selected. See input <i>iSetDimmerMode</i> .
0x801A	32794	FB_KL27x1Config: Invalid ramp time readouted. See output <i>sRampTime</i> .
0c801B	32795	FB_KL27x1Config: Invalid dimmer mode readouted. See output <i>sDimmerMode</i> .
0x801C	32796	FB_KL27x1Config: Invalid after-shortcut-behaviour readouted. See output <i>sAfterShortCircuit</i> .
0x801D	32797	FB_KL27x1Config: Invalid line frequency readouted. See output <i>sLineFrequency</i> .
0x8020	32800	FB_KL3208Config: Invalid sensor type selected. See input <i>iSetSensorType</i> .
0x8021	32801	FB_KL3208Config: Invalid sensor type readouted. See output <i>sSensorType</i> .
0x8028	32808	FB_KL320xConfig: Two-wired connection is not applicable for a KL3204.
0x8029	32809	FB_KL320xConfig: Invalid sensor type selected. See input <i>iSetSensorType</i> .
0x802A	30810	FB_KL320xConfig: Invalid sensor type readouted. See output <i>sSensorType</i> .
0x8030	32816	FB_KL3228Config: Invalid sensor type selected. See input <i>iSetSensorType</i> .
0x8031	32817	FB_KL3228Config: Invalid sensor type readouted. See output <i>sSensorType</i> .

4 Data structures

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_COMPOR := 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-
Module1 LWL Basis (Phoenix) *)
IODEVICETYPE_CX1100_BK := 65, (* Beckhoff-CX1100 terminal bus power supply *)
IODEVICETYPE_ENETRTP := 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_IBSSCRIPPCI := 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_ETHERNETPNMPPROT := 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 Ethernet 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

```

Requirements

Development environment	Target system type	PLC libraries to include
TwinCAT v2.7.0	PC (i386)	Standard.Lib; PLCSystem.Lib; TcPLCAds.Lib; TcPlcloFunctions.Lib
TwinCAT v2.8.0	PC (i386)	TcloFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib are included automatically)

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

```

Requirements

Development environment	Target system type	PLC libraries to include
TwinCAT v2.8.0 Build > 735	PC (i386)	TcloFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib are included automatically)

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

```

Requirements

Development environment	Target system type	PLC libraries to include
TwinCAT v2.8.0 Build > 735	PC (i386)	TcloFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib are included automatically)

4.4 ST_SercosParamAttrib

ST_SercosParamAttrib contains the data of the bit string nAttrib split into separate variables.

```



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

```

Requirements

Development environment	Target system type	PLC libraries to include
TwinCAT v2.8.0 Build > 735	PC (i386)	TcloFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib are included automatically)

Also see about this

-  [E_SercosAttribLen \[▶ 121\]](#)
-  [E_SercosAttribType \[▶ 121\]](#)

4.5 File format of the backup file

1. File header of type ST_SercosFileHeader
2. n * Data
 - a) Parameter header of Type ST_SercosParamHeader
 - b) Parameter data as bytes

Sample for n parameters

```

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), can be different for each parameter depending on the type or listlength

Sample for n = 3 parameters

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)

```
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
```

The variable of this type is a structure containing the file header for the backup and restore file.

nVersion: contains the file version, currently 1

nListType: contains the IDN list that was used to create the backup file and is either 192 (list of backup parameters) or 17 (list of all parameters) or 0 (user defined list, only for the FB_IOF_SER_DRIVE_BackupEx). The restore requires nListType to be 192 or 0 (only for the FB_IOF_SER_DRIVE_BackupEx).

cbCommentLen: contains the length of the user comment for the backup file

sComment: contains the user comment for the backup file. The string with all 256 bytes is written.

TYPE ST_SercosParamHeader (8 Bytes)

Following to the file header for each parameter a parameter header (ST_SercosParamHeader) is saved in the file.

```
TYPE ST_SercosParamHeader :
STRUCT
  nIDN          : UINT;      (* 2 Bytes *)
  cbSize        : UINT;      (* 2 Bytes *)
  nAttrib       : DWORD;     (* 4 Bytes *)
END_STRUCT
END_TYPE
```

nIDN: contains the parameter ID number

cbSize: contains the length of data bytes for this parameter in the backup file, following this parameter header

nAttrib: contains the attribute of the parameter (see [ST_SercosParamData \[► 122\]](#)) for the determination of the data type and data size.

Parameter Data (cbSize Bytes)

The parameter header is immediately followed by the actual parameter value. The amount of bytes of the parameter value is stored in cbSize of ST_SercosParamHeader of the parameter.

Requirements

Development environment	Target system type	PLC libraries to include
TwinCAT v2.8.0	PC (i386)	TcloFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib are included automatically)

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
    
```

Data words from drive to PLC.

Requirements

Development Environment	Target System	IO Hardware	PLC Libraries to include
TwinCAT v2.8.0 Build > 737	PC (i386)	AX2000 Profibus box	TcloFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib are included automatically)

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
    
```

Data words from PLC to drive.

Requirements

Development Environment	Target System	IO Hardware	PLC Libraries to include
TwinCAT v2.8.0 Build > 737	PC (i386)	AX2000 Profibus box	TcloFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib are included automatically)

4.8 ST_Parameter_IN

```

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

Input data of the ASI Bus Terminal.

ParameterStatus

Byte Offset	Bit Offset	Description
0	0-5	status bits
0	6	0: no diagnosis, 1: diagnosis
0	7	always 0: no register communication
1	0-3	0-3 reserved for extensions
1	4	Toggle-Bit to acknowledge task (if Cyclic the bit 6 is copied from bit 0)
1	5	Acknowledgement (0: noError, 1: error)
1	6	0: cyclic, 1: acyclic
1	7	0: parameter accesss, 1: ADS
2		input data (cyclic), parameter value (acyclic) or error number bit 0-7
3		input data (cyclic), parameter value (acyclic) or error number bit 8-15
4		input data (cyclic), parameter value (acyclic) or error number bit 16-23
5		input data (cyclic), parameter value (acyclic) or error number bit 24-31

Requirements

Development Environment	Target System	IO hardware	PLC Libraries to include
TwinCAT Version >= 2.8.0 Build > 739	PC (i386)	ASI Master Bus Terminal	TcIoFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib are included automatically)

4.9 ST_Parameter_OUT

```

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

```

Output data for ASI Bus Terminal.

ParameterStatus

Byte Offset	Bit Offset	Description
0	0-5	parameter number bit 0-5 (or parameter offset)
0	6	at acyclic: 0: read, 1:write, at cyclic (always read/write) the bit is copied in the input data to have a direct assignment (then, the cyclic data can be changed)
0	7	0: parameter access, 1: registercommunication
1	0-5	parameter number bit 6-11 (or parameter page)
1	6	0: Cyclic, 1: Acyclic
1	7	0: parameter access, 1: ADS
2		output data (cyclic), parameter value (acyclic) bit 0-7
3		output data (cyclic), parameter value (acyclic) bit 8-15
4		output data (cyclic), parameter value (acyclic) bit 16-23
5		output data (cyclic), parameter value (acyclic) bit 24-32

Requirements

Development Environment	Target System	IO hardware	PLC Libraries to include
TwinCAT Version >= 2.8.0 Build > 739	PC (i386)	ASI master Bus Terminal	TcIoFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib are included automatically)

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
    
```

Data buffer for I/O data of the ASI bus terminal.

Requirements

Development environment	Target system	IO hardware	PLC libraries to include
TwinCAT Version >= 2.8.0 Build > 739	PC (i386)	ASI Master Terminal	TcIoFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib are included automatically)

4.11 ST_NovRamAddrInfo

```

TYPE ST_NovRamAddrInfo:
STRUCT
    pCardAddress      : UDINT;
    iCardMemSize      : UDINT;
END_STRUCT
END_TYPE
    
```

pCardAddress : the address pointer of the NOV/DP-RAM;

iCardMemSize : the configured NOV/DP-RAM size in bytes;

Requirements

Development environment	Target system type	IO Hardware	PLC libraries to include
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 are included automatically)

4.12 ST_UPSStatus

```

TYPE ST_UPSStatus
STRUCT
    Vendor          : STRING; (* Reports the UPS vendor name as displayable string.*)
    Model           : STRING; (* Reports the UPS model name as displayable string.*)
    FirmwareRev    : STRING; (* Reports the UPS firmware revision as a displayable string.*)
    SerialNumber   : STRING; (* Reports the UPS serial number as a displayable string.*)
    BatteryLifePercent : DWORD;
    (* The percent of battery capacity remaining in the UPS, represented as a value in the range of 0..100%.*)
    BatteryLifeTime : DWORD; (* The amount of remaining UPS run time, in minutes.*)
    eBatteryStatus  : E_BatteryStatus; (* The current status of the UPS batteries.*)
    eCommStatus     : E_UpsCommStatus; (* The status of the communication path to the UPS.*)
    ePowerStatus    : E_UpsPowerStatus; (* Reports the status of utility-supplied power into the UPS.*)
    nPowerFailCnt   : DWORD;
    (* Power fail counter. Increments every time the UPS service detects power fail.*)
    dwChargeFlags   : DWORD;
    (* Battery charge status flags. This member can be one or more of the following values.
    Bits0..7 := General battery status flags (if all bits are set to 0 => unknown status)
        Bit0 := High (bit set => high battery charge)
        Bit1 := Low (bit set => low battery charge)
        Bit2 := Critical (bit set => battery is near empty)
        Bit3 := Charging (bit set => battery is charging)
        Bits4..6 := reserved (all bits are 0)
        Bit7 := No Battery (bit set => battery not found or not connected, bit not set => battery is existing or unknown state)
    Bits8..15 := Special status information (if all bits are set to 0 => state ok or unknown state)
        Bit8 := UPS Fan Error (bit set => fan hardware is reporting an error, bit not set => fan is ok)
        Bit9 := Overt Temperature (bit set => over temperature detected, bit not set => temperature is ok)
        Bit10 := Service Interval Notify (bit set => service interval time expired, bit not set => service interval time not expired)
        Bit11 := Under Temperature (bit set => under temperature detected, bit not set => temperature is ok)
        Bit12 := Fuse Not Ok (bit set => fuse broken or missed, bit not set => fuse ok)
        Bit13 := Alarm1 (reserved for later use, bit is 0)
        Bit14 := Alarm2 (reserved for later use, bit is 0)
        Bit15 := Alarm3 (reserved for later use, bit is 0)
    Bits16..31 := reserved for later use, all bits are 0)
    *)
END_STRUCT
END_TYPE

```

Not all UPS devices are able to deliver status information.

X : The status info is present.

*) Only available if model "Smart Signaling to any APC UPS & TwinCAT" configured!

Status info	Beckhoff HID UPS, BAPI v1	Beckhoff P24Vxxxx	Beckhoff CP903x ISA or PCI card	Beckhoff CX2100-09x 4	APC Back-UPS Pro 280, Smart-UPS 420	Description
Vendor	X	X	X	X	X	Reports the UPS vendor name as displayable string.
Model	X	X	X	X	X	Reports the UPS model name as displayable string.

Status info	Beckhoff HID UPS, BAPI v1	Beckhoff P24Vxxxx	Beckhoff CP903x ISA or PCI card	Beckhoff CX2100-09x 4	APC Back-UPS Pro 280, Smart-UPS 420	Description
FirmwareRev	X	X	X	X	X	Reports the UPS firmware revision as a displayable string. Empty string if not available.
SerialNumber	X	X	None	X	X	Reports the UPS serial number as a displayable string. Empty string if not available.
BatteryLifePercent	X	X	None	X	X	The percent of battery capacity remaining in the UPS, represented as a value in the range of 0 through 100. Zero if not available.
BatteryLifeTime	X	X	None	X	X	The amount of remaining UPS run time, in minutes. Zero if not available.
eBatteryStatus [▶ 132]	<ul style="list-style-type: none"> • BatteryOk 	<ul style="list-style-type: none"> • BatteryUnknownStatus if no battery found. Only with UPS software version $\geq 2.0.0.6$ and UPS firmware $\geq 25.1.1$. • BatteryOk 	<ul style="list-style-type: none"> • BatteryUnknownStatus if no battery found. • BatteryOk 	<ul style="list-style-type: none"> • BatteryUnknownStatus if no battery found (only "Smart Battery" and not capacitive model). • BatteryOk 	X	The current status of the UPS batteries.
eCommStatus [▶ 133]	X	X	X	X	X	The status of the communication path to the UPS.

Status info	Beckhoff HID UPS, BAPI v1	Beckhoff P24Vxxxx	Beckhoff CP903x ISA or PCI card	Beckhoff CX2100-09x 4	APC Back-UPS Pro 280, Smart-UPS 420	Description
<u>ePowerStatus</u> [▶ 133]	X	X	X	X	X	Reports the status of utility-supplied power into the UPS.
nPowerFailCount	X	X	X	X	*X	Power fail counter. Increments every time the UPS service detects power fail.

Status info	Beckhoff HID UPS, BAPI v1	Beckhoff P24Vxxxx	Beckhoff CP903x ISA or PCI card	Beckhoff CX2100-09x 4	APC Back-UPS Pro 280, Smart-UPS 420	Description
dwChargeFlags	<ul style="list-style-type: none"> • No Battery (bit 7 set) only with UPS firmware >= 33.12-0 if battery not found or not connected. • Service Interval Notify (bit 10 is set) The configured battery replace service intervall has expired. 	<ul style="list-style-type: none"> • No Battery (bit 7 set) only with UPS software version >=2.0.0.6 and UPS firmware >= 25.1.1. The battery existence is checked every minute. • UPS Fan Error (bit 8 set) only with UPS software version >=2.0.0.7 and UPS firmware >=40.1.1 The fan status is checked every minute. New (second) UPS hardware revision required! • Service Interval Notify (bit 10 is set) The configured battery replace service intervall has expired. Implemented in UPS Software version >= 3.0.0.8. 	<ul style="list-style-type: none"> • High (bit 0 set) if battery full loaded. • Charging (bit 3 set) • No Battery (bit 7 set) if no battery found. 	<ul style="list-style-type: none"> • No Battery (bit 7 set). No communication to battery or battery not found (only "Smart Battery" and not capacitive model). • Over Temperature (bit 9 set) Over temperature detected and charging of battery is stopped. New (second) UPS hardware revision required. Implemented in UPS software version >= 3.0.0.18. • Service Interval Notify (bit 10 set) The configured battery replace service intervall is expired. • Under Temperature (bit 11 set) Under temperature detected and charging of battery is stopped. New (second) UPS hardware revision required. 	None	Battery charge status flags and special status information.

Requirements

UPS hardware	Target system type	Development environment	PLC libraries to include
<ul style="list-style-type: none"> • Beckhoff HID UPS; • Beckhoff BAPI v1; • Beckhoff P24Vxxxx; • Beckhoff CP903x card (PCI/ISA); • Beckhoff CX2100-09x4 models (e.g. CX2100-0904 or CX2100-0914 + "Smart Battery" CX2900-0192); • With Beckhoff Industrial PC's delivered APC UPS models supporting smart protocol and configured with Windows UPS Service; 	PC or CX	TwinCAT v2.8.0, Build > 745	TcIoFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib are included automatically)
		TwinCAT v2.9.0, Build > 945	
		TwinCAT v3.0 and above	Tc2_IoFunctions

4.13 E_BatteryStatus

```

TYPE E_BatteryStatus :
(
  BatteryUnknownStatus, (*The battery status is unknown.*)
  BatteryOk,             (*The batteries are OK.*)
  BatteryReplace        (*The batteries need to be replaced.*)
);
END_TYPE

```

Requirements

UPS hardware	Target system type	Development environment	PLC libraries to include
<ul style="list-style-type: none"> • Beckhoff HID UPS; • Beckhoff BAPI v1; • Beckhoff P24Vxxxx; • Beckhoff CP903x card (PCI/ISA); • Beckhoff CX2100-09x4 models (e.g. CX2100-0904 or CX2100-0914 + "Smart Battery" CX2900-0192); • With Beckhoff Industrial PC's delivered APC UPS models supporting smart protocol and configured with Windows UPS Service; 	PC or CX	TwinCAT v2.8.0, Build > 745	TcIoFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib are included automatically)
		TwinCAT v2.9.0, Build > 945	
		TwinCAT v3.0 and above	Tc2_IoFunctions

4.14 E_UpsCommStatus

```

TYPE E_UpsCommStatus :
(
  UpsCommUnknownStatus, (* The communication path to the UPS is unknown. *)
  UpsCommOk,             (* The communication path to the UPS is OK. *)
  UpsCommFailed          (* The communication path to the UPS has failed. *)
);
END_TYPE

```

Requirements

UPS hardware	Target system type	Development environment	PLC libraries to include
<ul style="list-style-type: none"> • Beckhoff HID UPS; • Beckhoff BAPI v1; • Beckhoff P24Vxxxx; • Beckhoff CP903x card (PCI/ISA); • Beckhoff CX2100-09x4 models (e.g. CX2100-0904 or CX2100-0914 + "Smart Battery" CX2900-0192); • With Beckhoff Industrial PC's delivered APC UPS models supporting smart protocol and configured with Windows UPS Service; 	PC or CX	TwinCAT v2.8.0, Build > 745	TcloFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib are included automatically)
		TwinCAT v2.9.0, Build > 945	
		TwinCAT v3.0 and above	Tc2_IoFunctions

4.15 E_UpsPowerStatus

```

TYPE E_UpsPowerStatus :
(
  PowerUnknownStatus, (* The status of power is unknown. *)
  PowerOnLine,        (* Power is OK. *)
  PowerOnBattery       (* A power failure has occurred. *)
);
END_TYPE

```

Requirements

UPS hardware	Target system type	Development environment	PLC libraries to include
<ul style="list-style-type: none"> • Beckhoff HID UPS; • Beckhoff BAPI v1; • Beckhoff P24Vxxxx; • Beckhoff CP903x card (PCI/ISA); • Beckhoff CX2100-09x4 models (e.g. CX2100-0904 or CX2100-0914 + "Smart Battery" CX2900-0192); 	PC or CX	TwinCAT v2.8.0, Build > 745	TcloFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib are included automatically)
		TwinCAT v2.9.0, Build > 945	
		TwinCAT v3.0 and above	Tc2_IoFunctions

UPS hardware	Target system type	Development environment	PLC libraries to include
<ul style="list-style-type: none"> With Beckhoff Industrial PC's delivered APC UPS models supporting smart protocol and configured with Windows UPS Service; 			

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" key pressed (key at the right);

bRMouseKey : "Right mouse" key pressed;

bHotKey : "Hotkey" pressed;

bTaskChaKey : "Task change" key pressed;

bABCKey : "ABC soft keyboard" key pressed;

bMainFanErr : Main fan error;

bCpuFanErr : CPU fan error;

bTempErr : Internal temperature error (Internal temperature > 50°C);

bBatteryErr : Battery error (at the time reserved)

nMainNtcTemp : *Temperature value 1 (soldered NTC-127°C .. + 127°C);*

nExtNtcTemp : *Temperature value 2 (external NTC, not available with each device);*

bRsrv1 - bRsrv7 : Reserved;

nRsrv8 : Reserved;

Requirements

Development environment	Target system	IO Hardware	PLC Libraries to include
TwinCAT v2.8.0, Build > 746 TwinCAT v2.9.0, Build > 945	PC (i386)	ads-tec PC	TcIcFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib are included automatically)

4.17 ST_Dpv1ParamAddrEx

TYPE ST_Dpv1ParamAddrEx

ST_Dpv1ParamAddrEx contains the data of one Profidrive parameter.

```

TYPE ST_Dpv1ParamAddrEx :
STRUCT
  (* parameter *)
  iAttribute      : USINT;          (* 0x10: Value; 0x20: Description; 0x30: Text; 0x80..F0: manufacturer specific; else: reserved *)
  iNumOfElements : USINT;          (* 1..234: number of elements; 0: Special Function; else: reserved *)
  iParameterNumber : UINT;          (* 1..65535: parameter number; 0: reserved *)
  iSubIndex       : UINT;          (* 0..65535: subindex *)
  iDataAddr       : UDINT;         (* address of buffer/address of plc variable *)
  iDataSize       : UDINT;         (* size of buffer/size of plc variable *)
  eFormat         : E_PD_Datatype; (* 0x01..0x36: datatype; 0x40: ZERO; 0x41: BYTE; 0x42: WORD; 0x43: DWORD; 0x44: Error; else reserved *)
  iNumOfValues    : UINT;          (* 0..234: number of values; else: reserved *)
  iErrorValue     : UDINT;         (* DPV1 error value *)
  stError         : ST_PD_Dpv1Error; (* DPV1 error flag, DPV1 error enum *)
END_STRUCT
END_TYPE
    
```

TYPE E_PD_Datatype

E_PD_Datatype contains the different datatypes of 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 contains the different Profidrive DPV1 access errors.

```

TYPE E_PD_Dpv1Error : (
    ePD_Err_ParamNumber      := 0,    (* Invalid
parameter number *)
    ePD_Err_ParamReadOnly    := 1,    (* Write to
parameter not permitted *)
    ePD_Err_ValueOutOfRange  := 2,    (* Value out of
range *)
    ePD_Err_InvalidSubIndex := 3,    (* Invalid
subindex *)
    ePD_Err_NoArray          := 4,    (* Parameter is
not an array parameter *)
    ePD_Err_WrongDataType    := 5,    (* Wrong data type
*)
    ePD_Err_OnlyResetPermitted := 6,  (* Setting not
permitted, only resetting *)
    ePD_Err_DescNotChangable := 7,    (* Description
element not changeable *)
    ePD_Err_DescNotFound     := 9,    (* Description
data not available *)
    ePD_Err_NoPermissionToChange := 11, (* No permission
to change the value *)
    ePD_Err_NoTextArray      := 15,    (* No text array
available *)
    ePD_Err_JobNotExecutable := 17,    (* Job not
executable *)
    ePD_Err_ValueInvalid     := 20,    (* Value invalid
*)
    ePD_Err_ResponseToLong   := 21,    (* Response is too
long *)
    ePD_Err_ParamAddrInvalid := 22,    (* Parameter
address is invalid *)
    ePD_Err_FormatInvalid    := 23,    (* Format is
invalid *)
    ePD_Err_NumOfValuesInvalid := 24,  (* Number of
values is not consistent *)
    ePD_Err_DriveObjNotExisting := 25,  (* Drive object
not existing *)
    ePD_Err_ParamDeactivated  := 101,  (* Parameter at
the moment deactivated *)

```



```

    ePD_Err_ParamNoWrIfEnabled := 107, (* No write access
to parameter while control is enabled *)
    ePD_Err_ParamInvalidUnit := 108, (* Invalid unit of
parameter *)
    ePD_Err_ParamNoWrIfNotInitFbk := 109, (* Write access to
parameter only during feedback configuration *)
    ePD_Err_ParamWrIfInitMtr := 110, (* Write access to
parameter only during motor configuration *)
    ePD_Err_ParamWrIfInitDrv := 111, (* Write access to
parameter only during drive configuration *)
    ePD_Err_ParamWrIfFastInit := 112, (* Write access to
parameter only during fast configuration *)
    ePD_Err_ParamWrIfReady := 113, (* Write access to
parameter only during ready *)
    ePD_Err_ParamWrIfInitParamReset := 114, (* Write access to
parameter only during parameter reset *)
    ePD_Err_ParamWrIfInitSafety := 115, (* Write access to
parameter only during safety configuration *)
    ePD_Err_ParamWrIfInitTechApp := 116, (* Write access to
parameter only during tech. application/Units *)
    ePD_Err_ParamWrIfInit := 117, (* Write access to
parameter only during configuration *)
    ePD_Err_ParamWrIfInitDwnLd := 118, (* Write access to
parameter only during download *)
    ePD_Err_ParamNoWrtIfDwnLd := 119, (* No write access
to parameter during download *)
    ePD_Err_ParamWrIfInitDrvCfg := 120, (* Write access to
parameter only during drive configuration *)
    ePD_Err_ParamWrIfInitSetDrvType := 121, (* Write access to
parameter only during drive type configuration *)
    ePD_Err_ParamWrIfInitDatasetCfg := 122, (* Write access to
parameter only during data set base configuration *)
    ePD_Err_ParamWrIfInitDevCfg := 123, (* Write access to
parameter only during device configuration *)
    ePD_Err_ParamWrIfInitDevDwnLd := 124, (* Write access to
parameter only during device download *)
    ePD_Err_ParamWrIfInitDevPrmReset := 125, (* Write access to
parameter only during device parameter reset *)
    ePD_Err_ParamWrIfInitDevReady := 126, (* Write access to
parameter only during device ready *)
    ePD_Err_ParamWrIfInitDevice := 127, (* Write access to
parameter only during device configuration *)
    ePD_Err_ParamNoWriteIfDwnLd := 129, (* No write access
to parameter during download *)
    ePD_Err_CtrlTakeOverBlocked := 130, (* Control take
over blocked *)
    ePD_Err_ParamBicoSetInvalid := 131, (* Desired BICO
setup not possible *)
    ePD_Err_ParamChangeBlocked := 132, (* Parameter
change blocked *)
    ePD_Err_ParamNoAccessDefined := 133, (* No access
method defined *)
    ePD_Err_BelowDefinedMinimum := 200, (* Below defined
minimum *)
    ePD_Err_AboveDefinedMaximum := 201, (* Above defined
maximum *)
    ePD_Err_WriteNotPermitted := 204 (* Write not
permitted *)
);
END_TYPE

```

4.18 ST_Dpv1ValueHeaderEx

TYPE ST_Dpv1ValueHeaderEx

ST_Dpv1ValueHeaderEx contains the data of one parameter value in the DPV1 telegram and its string representation.

```

TYPE ST_Dpv1ValueHeaderEx :
STRUCT
    eFormat : E_PD_Datatype; (* 0x01..0x36: datatype; 0x40: ZERO; 0x41: BYTE; 0x42: WORD; 0x
43: DWORD; 0x44: Error; else reserved *)
    iNumOfValues : USINT; (* 0..234: number of values; else: reserved *)
    iOffset : USINT; (* Offset in the DPV1 response telegram *)
    iDataLen : UINT; (* data length *)

```

```

    strData      : STRING; (* printed data *)
END_STRUCT
END_TYPE

```

TYPE E_PD_Datatype

E_PD_Datatype contains the different datatypes of 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
    
```

Structure for linking in the input process image.



In ARM systems, the structure as a whole cannot be linked to the image of the terminal - the structure variables must be linked individually.

Requirements

Development Environment	Target System	IO Hardware	PLC Libraries to include
TwinCAT v2.11 R3/x64 from Build 2254	PC/CX	KL1501	TcloFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib are included automatically)

4.24 ST_KL1501OutData

```

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

```

Structure to be linked in the output process image.



In ARM systems, the structure as a whole cannot be linked to the image of the terminal - the structure variables must be linked individually.

Requirements

Development Environment	Target System	IO Hardware	PLC Libraries to include
TwinCAT v2.11 R3/x64 from Build 2254	PC/CX	KL1501	TcloFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib are included automatically)

4.25 ST_KL27x1InData

```

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

```

Structure to be linked in the input process image.



In ARM systems, the structure as a whole cannot be linked to the image of the terminal - the structure variables must be linked individually.

Requirements

Development Environment	Target System	IO Hardware	PLC Libraries to include
TwinCAT v2.11 R3/x64 from Build 2254	PC/CX	KL2751, KL2761	TcloFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib are included automatically)

4.26 ST_KL27x1OutData

```

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

```

Structure to be linked in the output process image.



In ARM systems, the structure as a whole cannot be linked to the image of the terminal - the structure variables must be linked individually.

Requirements

Development Environment	Target System	IO Hardware	PLC Libraries to include
TwinCAT v2.11 R3/x64 from Build 2254	PC/CX	KL2751, KL2761	TcloFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib are included automatically)

4.27 ST_KL3208InData

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

Structure to be linked in the input process image.



In ARM systems, the structure as a whole cannot be linked to the image of the terminal - the structure variables must be linked individually.

Requirements

Development Environment	Target System	IO Hardware	PLC Libraries to include
TwinCAT v2.11 R3/x64 from Build 2254	PC/CX	KL3208	TcloFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib are included automatically)

4.28 ST_KL3208OutData

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

Structure to be linked in the output process image.



In ARM systems, the structure as a whole cannot be linked to the image of the terminal - the structure variables must be linked individually.

Requirements

Development Environment	Target System	IO Hardware	PLC Libraries to include
TwinCAT v2.11 R3/x64 from Build 2254	PC/CX	KL3208	TcloFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib are included automatically)

4.29 ST_KL320xInData

```

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

```

Structure to be linked in the input process image.



In ARM systems, the structure as a whole cannot be linked to the image of the terminal - the structure variables must be linked individually.

Requirements

Development Environment	Target System	IO Hardware	PLC Libraries to include
TwinCAT v2.11 R3/x64 from Build 2254	PC/CX	KL3201, KL3202, KL3204	TcloFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib are included automatically)

4.30 ST_KL320xOutData

```

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

```

Structure to be linked in the output process image.



In ARM systems, the structure as a whole cannot be linked to the image of the terminal - the structure variables must be linked individually.

Requirements

Development Environment	Target System	IO Hardware	PLC Libraries to include
TwinCAT v2.11 R3/x64 from Build 2254	PC/CX	KL3201, KL3202, KL3204	TcloFunctions.Lib

Development Environment	Target System	IO Hardware	PLC Libraries to include
			(Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib are included automatically)

4.31 ST_KL3228InData

```

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

Structure to be linked in the input process image.



In ARM systems, the structure as a whole cannot be linked to the image of the terminal - the structure variables must be linked individually.

Requirements

Development Environment	Target System	IO Hardware	PLC Libraries to include
TwinCAT v2.11 R3/x64 from Build 2254	PC/CX	KL3228	TcloFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib are included automatically)

4.32 ST_KL3228OutData

```

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

Structure to be linked in the output process image.



In ARM systems, the structure as a whole cannot be linked to the image of the terminal - the structure variables must be linked individually.

Requirements

Development Environment	Target System	IO Hardware	PLC Libraries to include
TwinCAT v2.11 R3/x64 from Build 2254	PC/CX	KL3228	TcloFunctions.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib; TcUtilities.Lib are included automatically)

5 Appendix

5.1 AX200x Profibus Parameter Number

PNU	Data type	Access	Short description	SERVOSTAR™ ASCII command
Profile parameter				
904	UINT32	ro	Number of the supported PPO write, always 2	-
911	UINT32	ro	Number of the supported PPO read, always 2	-
918	UINT32	ro	Device address on the PROFIBUS	ADDR
930	UINT32	r/w	Selection switch for operating mode	-
963	UINT32	ro	PROFIBUS baud rate	-
965	Octet-String2	ro	Number of the PROFIDRIVE profile (0302H)	-
970	UINT32	wo	Load default parameter set	RSTVAR
971	UINT32	wo	Store parameter in non-volatile memory	SAVE
SERVOSTAR™ manufacturer-specific parameters				
General parameters				
1000	Visible String4	ro	Device identification	-
1001	UINT32	ro	Manufacturer-specific error register	ERRCODE
1002	UINT32	ro	Manufacturer-specific status register	-
Speed controller parameters				
1200	UINT32	r/w	Kp - amplification factor of the speed controller	GV
1201	UINT32	r/w	Tn – integral-action time of the speed controller	GVTN
1202	UINT32	r/w	PID – T2 – time constant for the speed controller	GVT2
1203	UINT32	r/w	Set value ramp+, speed controller	ACC
1204	UINT32	r/w	Set value ramp-, speed controller	DEC
1205	UINT32	r/w	Emergency ramp, speed controller	DECSTOP

PNU	Data type	Access	Short description	SERVOSTAR™ ASCII command
1206	UINT32	r/w	Maximum rotation speed	VLIM
1207	UINT32	r/w	Overspeed	VOSPD
1208	UINT32	r/w	Counting direction	DIR
Position controller parameters				
1250	UINT32	r/w	Multiplier for speeds jogging/ref.	VMUL
1251	UINT32	r/w	Axis type	POSCNFG
1251	INTEGER32	r/w	In-position window	PEINPOS
1253	INTEGER32	r/w	Following error window	PEMAX
1254	INTEGER32	r/w	Position register 1	SWE1
1255	INTEGER32	r/w	Position register 2	SWE2
1256	INTEGER32	r/w	Position register 3	SWE3
1257	INTEGER32	r/w	Position register 4	SWE4
1258	UINT32	r/w	Resolution denominator	PGEARO
1259	UINT32	r/w	Resolution numerator	PGEARI
1260	UINT32	r/w	Minimum acceleration or braking time	PTMIN
1261	UINT32	r/w	Position controller feed-forward factor	GPFFV
1262	UINT32	r/w	Position controller KV factor	GP
1263	UINT32	r/w	Position controller KP factor	GPV
1264	UINT32	r/w	Tn - position controller integral-action time	GPTN
1265	UINT32	r/w	Maximum speed for positioning operation	PVMAX
1266	UINT32	r/w	Configuration variable for software switch	SWCNFG
1267	UINT32	r/w	Configuration variable 2 for software switch	SWCNFG2
Positioning data for position controller mode				
1300	INTEGER32	r/w	Position	O_P
1301	INTEGER16	r/w	Speed	O_V
1302	UINT32	r/w	Transport instruction type	O_C
1304	UINT32	r/w	Starting up time (acceleration)	O_ACC1
1305	UINT32	r/w	Braking time (deceleration)	O_DEC1
1306	UINT32	r/w	Jerk limitation (acceleration)	O_ACC2

PNU	Data type	Access	Short description	SERVOSTAR™ ASCII command
1307	UINT32	r/w	Jerk limitation (deceleration)	O_DEC2
1308	UINT32	r/w	Number of the subsequent transport instruction	O_FN
1309	UINT32	r/w	Start delay for subsequent transport instruction	O_FT
1310	2 * UINT16	wo	Copy a transport instruction	OCOPY
Setting-up mode, position				
1350	UINT32	r/w	Reference travel type	NREF
1351	UINT32	r/w	Reference travel direction	DREF
1352	UINT32	r/w	Acceleration ramp (jogging/referencing)	ACCR
1353	UINT32	r/w	Braking ramp	DECR
1354	UINT32	r/w	Reference offset	ROFFS
1355	UINT32	ro	Reference travel speed	VREF
1356	UINT32	ro	Jogging speed	VJOG
Actual values				
1400	INTEGER32	ro	Actual position represented with 20 bits/rotation	PRD
1401	INTEGER32	ro	Rotation speed	-
1402	INTEGER32	ro	Incremental actual position value	-
1403	INTEGER32	ro	SI - actual position value	PFB
1404	INTEGER32	ro	SI - actual speed value	PV
1405	INTEGER32	ro	SI - following error	PE
1406	INTEGER32	ro	Effective current	I
1407	INTEGER32	ro	SI - actual rotation speed	V
1408	INTEGER32	ro	Heatsink temperature	TEMPH
1409	INTEGER32	ro	Internal temperature	TEMPE
1410	INTEGER32	ro	Intermediate circuit voltage	VBUS
1411	INTEGER32	ro	Ballast power	PBAL
1412	INTEGER32	ro	I ² t - loading	I2T
1413	INTEGER32	ro	Period of operation	TRUN
Digital I/O-configuration				
1450	UINT32	r/w	Function of digital input 1	IN1MODE

PNU	Data type	Access	Short description	SERVOSTAR™ ASCII command
1451	UINT32	r/w	Function of digital input 2	IN2MODE
1452	UINT32	r/w	Function of digital input 3	IN3MODE
1453	UINT32	r/w	Function of digital input 4	IN4MODE
1454	INTEGER32	r/w	Auxiliary variable for digital input 1	IN1TRIG
1455	INTEGER32	r/w	Auxiliary variable for digital input 2	IN2TRIG
1456	INTEGER32	r/w	Auxiliary variable for digital input 3	IN3TRIG
1457	INTEGER32	r/w	Auxiliary variable for digital input 4	IN4TRIG
1458	INTEGER32	r/w	Function of digital output 1	O1MODE
1459	INTEGER32	r/w	Function of digital output 2	O2MODE
1460	UINT32	r/w	Auxiliary variable for digital output 1	O1TRIG
1461	UINT32	r/w	Auxiliary variable for digital output 2	O2TRIG
1462	UINT32	r/w	State of 4 digital inputs, enable, 2 digital outputs	STATIO
Analog configuration				
1500	UINT32	r/w	Configuration of the analog input functions	ANCNFG
1501	UINT32	r/w	Configuration of monitor function for analog output 1	ANOUT1
1502	UINT32	r/w	Offset voltage for analog input 1	ANOFF1
1503	UINT32	r/w	Filter time constant for analog input 1	AVZ1
1504	UINT32	r/w	Speed scaling factor for analog input 1	VSCALE1
1505	UINT32	r/w	Current scaling factor for analog input 1	ISCALE1
1506	UINT32	r/w	Configuration of monitor function for analog output 2	ANOUT2
1507	UINT32	r/w	Offset voltage for analog input 2	ANOFF2
1508	UINT32	r/w	Speed scaling factor for analog input 2	VSCALE2
1509	UINT32	r/w	Current scaling factor for analog input 2	ISCALE2

PNU	Data type	Access	Short description	SERVOSTAR™ ASCII command
Motor parameters				
1550	UINT32	r/w	Brake configuration	MBRAKE
1551	UINT32	r/w	Motor number from the motor database	MNUMBER

More Information:
www.beckhoff.com/tx1200

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

