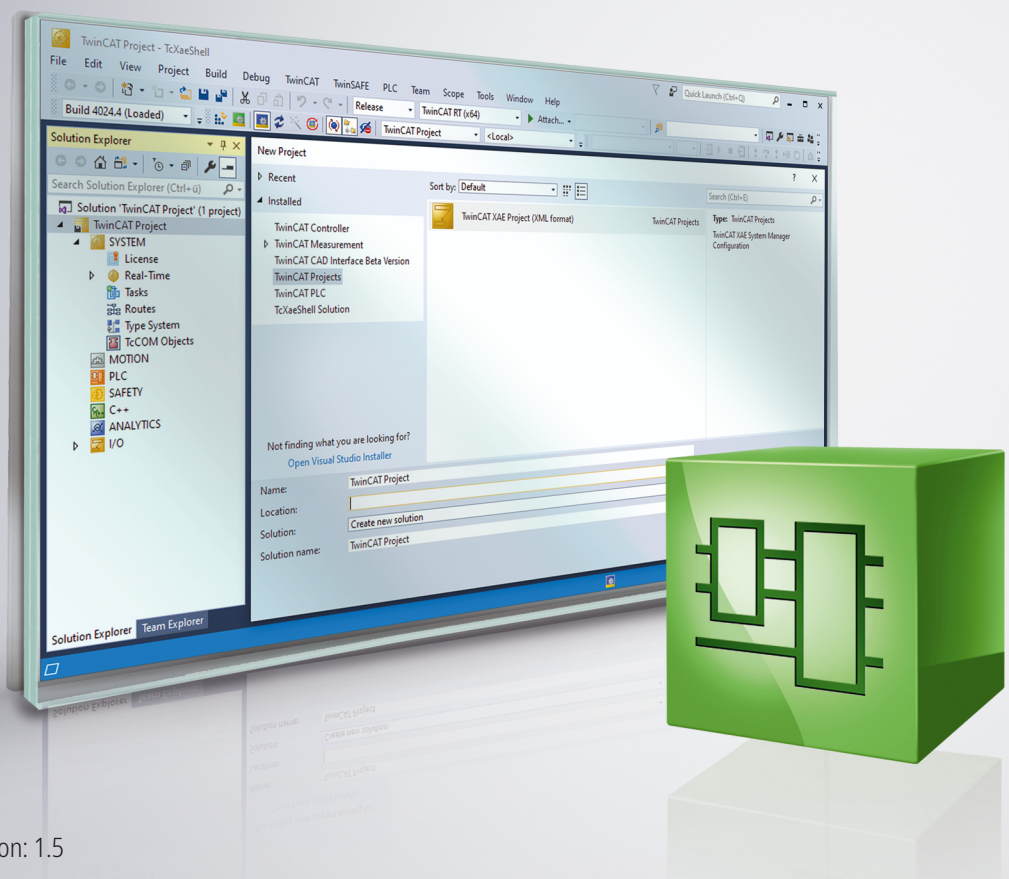
BECKHOFF New Automation Technology

Manual | EN

TE1000

TwinCAT 3 | PLC Library: Tc2_IoFunctions



2021-04-26 | Version: 1.5

Table of contents

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

Copyright

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.

2 Overview

The IO Functions library contains function blocks with which services/functions can be executed on the IO devices (fieldbus masters or slaves).

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 AmsNetId of a box, knowing the fieldbus address of the box and the device ID
IOF_GetDeviceCount [▶ 19]	Read the number of I/O devices.
IOF_GetDeviceIDByName [▶ 19]	Find the device ID knowing the device identifier
IOF_GetDeviceIDs [▶ 20]	Read all device IDs
IOF_GetDeviceName [▶ 23]	Read the device identifier knowing the device ID
IOF_GetDeviceNetId [▶ 24]	Read the AmsNetId, knowing the device ID
IOF_GetDeviceType [▶ 25]	Read the device type, knowing the device ID
IOF_GetDeviceInfoByName [▶ 22]	Determine the device ID and the AmsNetId, knowing the device name

Fieldbus-specific and device-specific functions

CANopen

Name	Description
IOF_CAN_Layer2Command [▶ 62]	Carry out a layer 2 command

Beckhoff Lightbus

Name	Description
IOF_LB_BreakLocationTest [▶ 44]	Optical fiber ring break location test
IOF_LB_ParityCheck [▶ 45]	Read parity counter
IOF_LB_ParityCheckWithReset [▶ 47]	Read and reset parity counter

SERCOS

Name	Description
IOF_SER_GetPhase [▶ 90]	Read the current phase
IOF_SER_ResetErr [▶ 92]	Reset the error buffer
IOF_SER_SaveFlash [▶ 91]	Save parameter in flash
IOF_SER_SetPhase [▶ 93]	Set the current phase
IOF_SER_IDN_Read [▶ 94]	Read Sercos drive parameters
IOF_SER_IDN_Write [▶ 95]	Write Sercos drive parameters
IOF_SER_DRIVE_Backup [▶ 96]	Backup and restore the Sercos drive parameters to/from a file
IOF_SER_DRIVE_BackupEx [▶ 98]	Backup and restore the Sercos drive parameters to/from a file (extended functionality)
IOF_SER_DRIVE_Reset [▶ 100]	Drive reset of a Sercos drive by command on parameter S-0-0099 (IDN99)

Profibus DPV1 (Sinamics

Name	Description
F_CreateDpv1ReadReqPkg [▶ 71]	Generate DPV1 telegram for parameter reading
FB_Dpv1Read [▶ 74]	Send DPV1 telegram for parameter reading
F_SplitDpv1ReadResPkg [▶ 72]	Evaluate DPV1 response telegram for parameter reading
F_CreateDpv1WriteReqPkg [▶ 72]	Generate DPV1 telegram for parameter writing
FB_Dpv1Write [▶ 76]	Send DPV1 telegram for parameter writing
F_SplitDpv1WriteResPkg [▶ 73]	Evaluate DPV1 response telegram for parameter writing

Profinet DPV1 (Sinamics)

Name	Description
F_CreateDpv1ReadReqPkgPNET [▶ 79]	Generate DPV1 telegram for parameter reading
FB_Dpv1ReadPNET [▶ 82]	Send DPV1 telegram for parameter reading
F_SplitDpv1ReadResPkgPNET [▶ 80]	Evaluate DPV1 response telegram for parameter reading
F_CreateDpv1WriteReqPkgPNET [▶ 79]	Generate DPV1 telegram for parameter writing
FB_Dpv1WritePNET [▶ 84]	Send DPV1 telegram for parameter writing
F_SplitDpv1WriteResPkgPNET [▶ 81]	Evaluate DPV1 response telegram for parameter writing

NOV/DP-RAM

Name	Description
FB_NovRamReadWrite [▶ 63]	Write data to the NOVRAM or read data from the NOVRAM
FB_NovRamReadWriteEx [▶ 65]	Write data to the NOVRAM or read data from the NOVRAM. Checks whether a special method of accessing the memory is necessary and copies the data accordingly in the correct way (e.g. when accessing the CX_9000 NOV-RAM).
FB_GetDPRAMInfo [▶ 67]	Read the address pointer and the configured size from the NOV/DP-RAM
FB_GetDPRAMInfoEx [▶ 69]	Read the address pointer and the configured size from the NOV/DP-RAM (Extension)

AX200x Profibus

Function blocks for access to the AX200X via Profibus: overview [▶ 36].

ASI master terminal

Function blocks for access to an ASI master terminal: overview [▶ 26].

Beckhoff UPS (under Windows UPS service)

Name	Description
FB_GetUPSStatus [▶ 48]	Read the status of the UPS from the PLC.

Third-party vendor devices

INTERBUS Phoenix IBS SC/I-T functions

Phoenix IBS SC/I-T functions: overview [▶ 103].

Name	Description
SCIT_ActivateConfiguration [▶ 105]	Executes the **Activate_Configuration** command
SCIT_DeactivateConfiguration [▶ 106]	Executes the **Deactivate_Configuration** command
SCIT_StartDataTransfer [▶ 107]	Executes the **Start_Data_Transfer** command
SCIT_StopDataTransfer [▶ 108]	Executes the **Stop_Data_Transfer** command
SCIT_AlarmStop [▶ 109]	Executes the **Alarm_Stop** command
SCIT_ControlActiveConfiguration [▶ 110]	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.
SCIT_GetErrorInfo [▶ 111]	Returns the error type and error location of an Interbus device after a bus error
SCIT_ConfDevErrAll [▶ 112]	Acknowledge periphery errors of all devices

ads-tec

Name	Description
FB_ReadAdsTecSysData [▶ 113]	Reads the system data/diagnostic data

RAID_Controller

The following function blocks are available for RAID controller services.

Name	Description
FB_RAIDFindCntlr [▶ 86]	Returns the number of RAID controllers and the corresponding RAID controller IDs
FB_RAIDGetInfo [▶ 87]	Supplies RAID information containing the number of RAID controller sets and the maximum number of RAID drives per set.
FB_RAIDGetStatus [▶ 89]	Returns the RAID set index, the RAID type, the RAID status, the number of RAID drives and the status of the RAID drives.

3 Function blocks

3.1 General IO FBs

3.1.1 IOF_DeviceReset

The IOF_DeviceReset function block resets an IO device (e.g. a fieldbus card). The function corresponds to the online reset function from the TwinCAT->I/O->Devices->Device xyz context menu.

VAR_INPUT

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

NETID: The network address of the TwinCAT computer on which the function is to be executed can be entered here (type: T_AmsNetId). 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 during the hardware configuration.

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

TMOUT: States the length of the timeout that may not be exceeded during execution of the 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 a feedback is received.

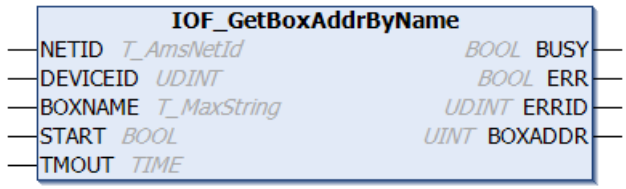
ERR: If an error should occur during the transfer of the command, then this output is set once the BUSY output was reset.

ERRID: Supplies the ADS error number [▶ 153] when the ERR output is set.

Requirements

Development environment	Target platform	IO hardware	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	PC or CX (x86, x64, ARM)	All IO devices	Tc2_IoFunctions (IO)

3.1.2 IOF_GetBoxAddrByName



The IOF_GetBoxAddrByName function block determines the fieldbus address of a box (box = slave, module, station), knowing the box name 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 fiber 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.

VAR_INPUT

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

NETID: The network address of the TwinCAT computer on which the function is to be executed can be entered here (type: T_AmsNetID). 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 during the hardware configuration.

BOXNAME: The box name as a string (type: T_MaxString).

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

TMOUT: States the length of the timeout that may not be exceeded during execution of the 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 a feedback is received.

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

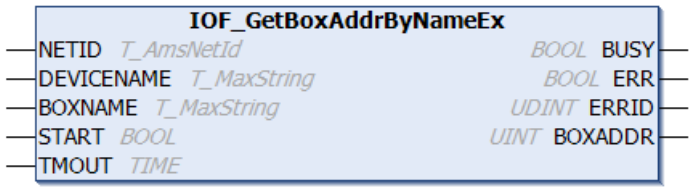
ERRID: Supplies the ADS error number [▶ 153] when the ERR output is set.

BOXADDR: The fieldbus address of the box.

Requirements

Development environment	Target platform	IO hardware	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	PC or CX (x86, x64, ARM)	All IO devices	Tc2_IoFunctions (IO)

3.1.3 IOF_GetBoxAddrByNameEx



The IOF_GetBoxAddrByNameEx function block determines the fieldbus address of a box (box = slave, module, station), knowing the box name and the device name. 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 fiber 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.

VAR_INPUT

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

NETID: The network address of the TwinCAT computer on which the function is to be executed can be entered here (type: T_AmsNetID). If it is to be run on the local computer, an empty string can be entered.

DEVICENAME: The device name of an IO device as a string (type: T_MaxString).

BOXNAME: The box name as a string (type: T_MaxString).

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

TMOUT: States the length of the timeout that may not be exceeded during execution of the 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 a feedback is received.

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

ERRID: Supplies the ADS error number [▶ 153] when the ERR output is set.

BOXADDR: The fieldbus address or logical address of the box.

Requirements

Development environment	Target platform	IO hardware	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	PC or CX (x86, x64, ARM)	All IO devices	Tc2_IoFunctions (IO)

3.1.4 IOF_GetBoxCount

The IOF_GetBoxCount function block reads the number of configured and active boxes (box = slave, module, station) of an IO device.

VAR_INPUT

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

NETID: The network address of the TwinCAT computer on which the function is to be executed can be entered here (type: T_AmsNetID). 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 during the hardware configuration.

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

TMOUT: States the length of the timeout that may not be exceeded during execution of the 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 a feedback is received.

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

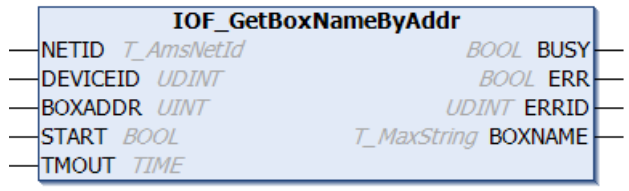
ERRID: Supplies the ADS error number [▶ 153] when the ERR output is set.

BOXCOUNT: The number of boxes.

Requirements

Development environment	Target platform	IO hardware	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	PC or CX (x86, x64, ARM)	All IO devices	Tc2_IoFunctions (IO)

3.1.5 IOF_GetBoxNameByAddr



The IOF_GetBoxNameByAddr function block determines the box name, knowing 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 fiber ring). When successful the function block returns the box identifier configured in TwinCAT as a string.

VAR_INPUT

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

NETID: The network address of the TwinCAT computer on which the function is to be executed can be entered here (type: T_AmsNetID). 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 during the hardware configuration.

BOXADDR: The fieldbus address of the box.

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

TMOUT: States the length of the timeout that may not be exceeded during execution of the 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 a feedback is received.

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

ERRID: Supplies the ADS error number [▶ 153] when the ERR output is set.

BOXNAME: The box name as a string (type: T_MaxString).

Requirements

Development environment	Target platform	IO hardware	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	PC or CX (x86, x64, ARM)	All IO devices	Tc2_IoFunctions (IO)

3.1.6 IOF_GetBoxNetId

Some boxes (slave modules) can be assigned an AmsNetId during the configuration in TwinCAT. The AmsNetId can then be used to execute firmware functions on the box. The IOF_GetBoxNetId function block determines the TwinCAT network address, knowing the device ID of the master and the fieldbus address or logical address in the fieldbus. The device IDs are specified by the TwinCAT system during the configuration and cannot be configured by the user.

VAR_INPUT

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

NETID: The network address of the TwinCAT computer on which the function is to be executed can be entered here (type: T_AmsNetID). 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 block is activated by a positive edge at this input.

TMOUT: States the length of the timeout that may not be exceeded during execution of the 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 a feedback is received.

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

ERRID: Supplies the ADS error number [▶ 153] when the ERR output is set.

BoxNetId: The TwinCAT network address of the box as a string (type: T_AmsNetID).

Requirements

Development environment	Target platform	IO hardware	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	PC or CX (x86, x64, ARM)	All IO devices	Tc2_IoFunctions (IO)

3.1.7 IOF_GetDeviceCount

The IOF_GetDeviceCount function block reads the number of configured and active IO devices.

VAR_INPUT

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

NETID: The network address of the TwinCAT computer on which the function is to be executed can be entered here (type: T_AmsNetID). If it is to be run on the local computer, an empty string can be entered.

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

TMOUT: States the length of the timeout that may not be exceeded during execution of the 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 a feedback is received.

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

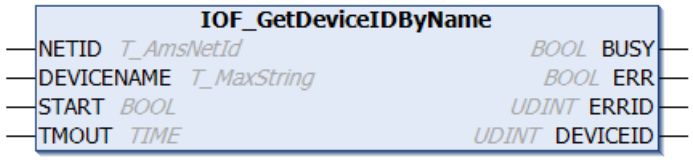
ERRID: Supplies the ADS error number [▶ 153] when the ERR output is set.

DEVICECOUNT: The number of I/O devices.

Requirements

Development environment	Target platform	IO hardware	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	PC or CX (x86, x64, ARM)	All IO devices	Tc2_IoFunctions (IO)

3.1.8 IOF_GetDeviceIDByName



The IOF_GetDeviceIDByName function block determines the device ID of an IO device, knowing the device name. When successful, the function block returns the device ID specified by the TwinCAT system during the configuration. The device IDs cannot be configured by the user.

VAR_INPUT

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

NETID: The network address of the TwinCAT computer on which the function is to be executed can be entered here (type: T_AmsNetID). If it is to be run on the local computer, an empty string can be entered.

DEVICENAME: The device name of an IO device (type: T_MaxString).

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

TMOUT: States the length of the timeout that may not be exceeded during execution of the 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 a feedback is received.

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

ERRID: Supplies the ADS error number [▶ 153] when the ERR output is set.

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

Requirements

Development environment	Target platform	IO hardware	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	PC or CX (x86, x64, ARM)	All IO devices	Tc2_IoFunctions (IO)

3.1.9 IOF_GetDeviceIDs

The IOF_GetDeviceIDs function block reads the device IDs of all configured and active IO devices 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 by the TwinCAT system during the configuration and cannot be configured by the user.

VAR_INPUT

```
VAR_INPUT
    NETID    : T_AmsNetId;
    LEN      : UDINT;
    DESTADDR : PVOID;
```

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

NETID: The network address of the TwinCAT computer on which the function is to be executed can be entered here (type: T_AmsNetID). 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 block is activated by a positive edge at this input.

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

VAR_OUTPUT

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

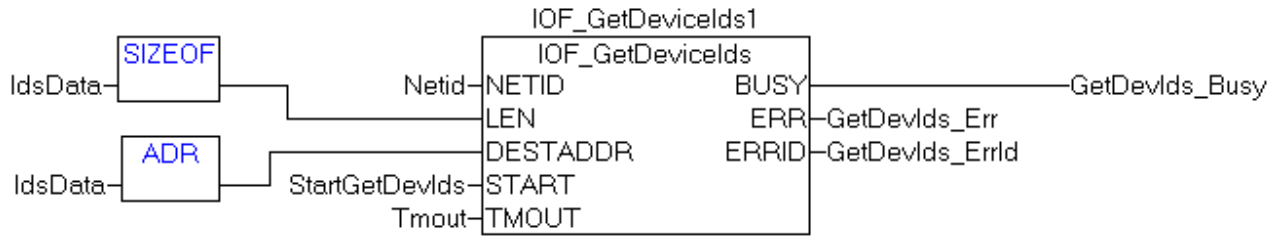
BUSY: When the function block is activated this output is set. It remains set until a feedback is received.

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

ERRID: Supplies the ADS error number [▶ 153] when the ERR output is set.

Example:

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



Requirements

Development environment	Target platform	IO hardware	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	PC or CX (x86, x64, ARM)	All IO devices	Tc2_IoFunctions (IO)

3.1.10 IOF_GetDeviceInfoByName

The IOF_GetDeviceInfoByName function block determines the device ID of an IO device and its TwinCAT network address, knowing the device name. The device IDs cannot be configured by the user.

VAR_INPUT

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

NETID: The AmsNetId of the TwinCAT computer on which the function is to be executed can be entered here (type: T_AmsNetID). If it is to be run on the local computer, an empty string can be entered.

DEVICENAME: The device name of an IO device (type: T_MaxString).

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

TMOUT: States the length of the timeout that may not be exceeded during execution of the 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 a feedback is received.

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

ERRID: Supplies the ADS error number [▶ 153] when the ERR output is set.

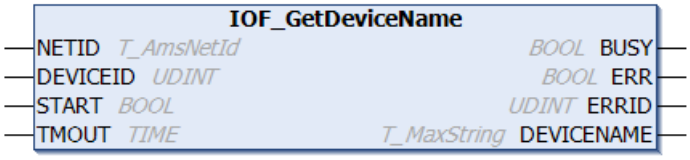
DEVICEID: The device ID of an I/O device.

DEVICENETID: The network address of an IO device (type: T_AmsNetID).

Requirements

Development environment	Target platform	IO hardware	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	PC or CX (x86, x64, ARM)	All IO devices	Tc2_IoFunctions (IO)

3.1.11 IOF_GetDeviceName



The IOF_GetDeviceName function block reads the device name of an IO device. The device identifier can be specified by the user during the configuration in the TwinCAT system. 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 := DEFAULT_ADS_TIMEOUT;
END_VAR
```

NETID: The network address of the TwinCAT computer on which the function is to be executed can be entered here (type: T_AmsNetID). 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 during the hardware configuration.

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

TMOUT: States the length of the timeout that may not be exceeded during execution of the 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 a feedback is received.

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

ERRID: Supplies the ADS error number [▶ 153] when the ERR output is set.

DEVICENAME: The device name of the IO device (type: T_MaxString).

Requirements

Development environment	Target platform	IO hardware	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	PC or CX (x86, x64, ARM)	All IO devices	Tc2_IoFunctions (IO)

3.1.12 IOF_GetDeviceNetId

Some IO devices can be assigned a TwinCAT network address during the configuration in the TwinCAT system (e.g. FC310x Profibus card or CP9030 card). The network address can then be used to execute firmware functions on the device. The IOF_GetDeviceNetId function block determines the network address, knowing the device ID. The device IDs are specified by the TwinCAT system during the configuration and cannot be configured by the user.

VAR_INPUT

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

NETID: The network address of the TwinCAT computer on which the function is to be executed can be entered here (type: T_AmsNetID). If it is to be run on the local computer, an empty string can be entered.

DEVICEID: The ID of the device whose TwinCAT network address is to be read

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

TMOUT: States the length of the timeout that may not be exceeded during execution of the 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 a feedback is received.

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

ERRID: Supplies the ADS error number [▶ 153] when the ERR output is set.

DeviceNetId: The AmsNetId of the device as a string (type: T_AmsNetID).

Requirements

Development environment	Target platform	IO hardware	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	PC or CX (x86, x64, ARM)	All IO devices	Tc2_IoFunctions (IO)

3.1.13 IOF_GetDeviceType

The IOF_GetDeviceType function block determines the type of device, knowing the device ID. The device IDs are specified by the TwinCAT system during the configuration and cannot be configured by the user.

VAR_INPUT

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

NETID: The network address of the TwinCAT computer on which the function is to be executed can be entered here (type: T_AmsNetID). 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 block is activated by a positive edge at this input.

TMOUT: States the length of the timeout that may not be exceeded during execution of the 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 a feedback is received.

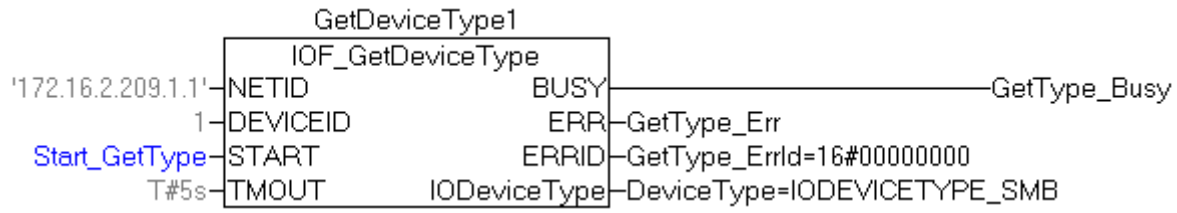
ERR: If an error should occur during the transfer of the command, then this output is set once the BUSY output was reset.

ERRID: Supplies the ADS error number [▶ 153] when the ERR output is set.

IODeviceType: The device type constant (type: IODEVICETYPES [▶ 124]).

Example:

```
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 platform	IO hardware	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	PC or CX (x86, x64, ARM)	All IO devices	Tc2_IoFunctions (IO)

3.2 ASI master terminal

3.2.1 Overview

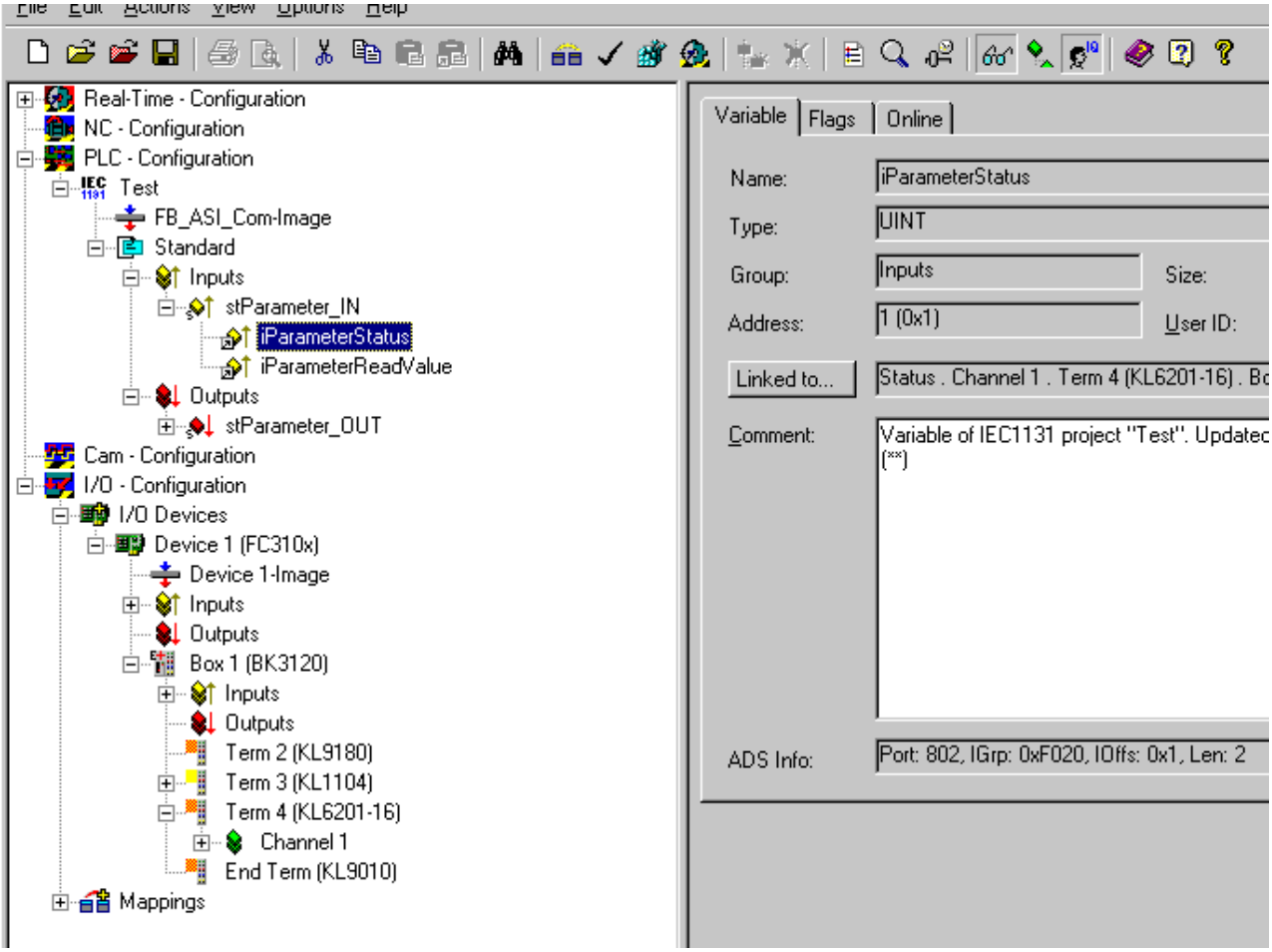
Function blocks for accessing the ASI master terminal.

Function blocks:

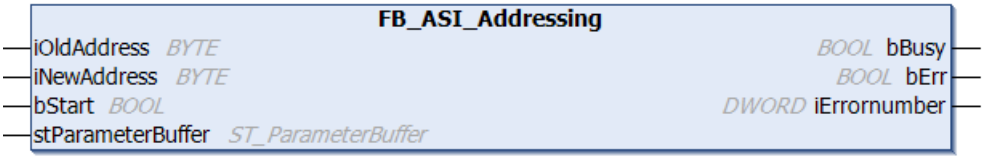
Name	Description
FB_ASI_Addressing [▶ 27]	Specify or modify addresses of the ASI slaves
FB_ASI_SlaveDiag [▶ 28]	Cyclic slave diagnostics (e.g. counter states)
FB_ASI_ReadParameter [▶ 29]	Universal FB for reading all the parameters of an ASI slave
FB_ASI_WriteParameter [▶ 31]	Universal FB for setting all the parameters of an ASI slave
FB_ReadInput_analog [▶ 34]	Read analog values
FB_WriteOutput_analog [▶ 35]	Write analog values
FB_ASI_Processdata_digital [▶ 32]	Read/write digital values
FB_ASI_ParameterControl [▶ 33]	Background communication. **This block must always be called cyclically!!!**

Linking into the System Manager

The library has one input structure: ST_Parameter_IN, and one output structure: ST_Parameter_OUT. These must be instanced 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.



3.2.2 FB_ASI_Addressing



VAR_IN_OUT

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

stParameterBuffer: Data buffer for the background communication (type: ST_ParameterBuffer [▶ 131]).

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; (* Error code 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 a command at the inputs.

Command-specific error code (decimal)	Description
1	Communication timeout
2	ASI slave address does not exist
3 - 10	Reserved
11	ASI slave is not activated (slave is not in LAS)
12	An error occurred during communication.
13	Data exchange bit (CN.4) not set

Requirements

Development environment	Target platform	IO hardware	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	PC or CX (x86)	ASI master terminal	Tc2_IoFunctions (IO)

3.2.3 FB_ASI_SlaveDiag

VAR_IN_OUT

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

stParameterBuffer: Data buffer for the background communication (type: ST_ParameterBuffer [▶ 131]).

VAR_INPUT

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

iSlaveaddress: slave address

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

bCounterReset: resetting of the current counter.

bReadLES: List of the identified ASI Slaves (LES).

bReadWrite: 0=READ, 1=WRITE.

bReadLAS: List of the 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;(* Error code 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 a command at the inputs.

Command-specific error code (decimal)	Description
1	Communication timeout
2	ASI slave address does not exist
3 - 10	Reserved
11	ASI slave is not activated (slave is not in LAS)
12	An error occurred during communication.
13	Data exchange bit (CN.4) not set

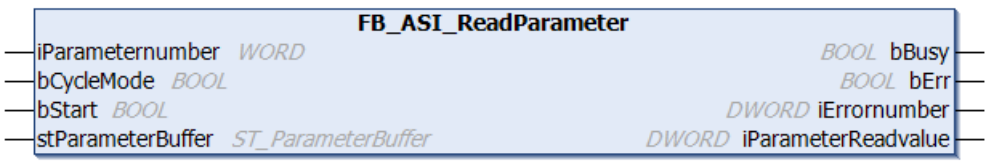
iCountervalue: Counter value.

iSlaveList: LES or LAS.

Requirements

Development environment	Target platform	IO hardware	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	PC or CX (x86)	ASI master terminal	Tc2_IoFunctions (IO)

3.2.4 FB_ASI_ReadParameter



VAR_IN_OUT

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

stParameterBuffer: Data buffer for the background communication (type: ST_ParameterBuffer [▶ 131]).

VAR_INPUT

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

iParameterNumber: parameter number.

bCycleMode: 0: Acyclic, 1:Cyclic (permanent Read/Write). If this bit is set, the bBusy output is only cancelled when the bStart input is pulled to FALSE. No current value is present at the output yet if the bStart input is pulled to FALSE too early.

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;(* Error code 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 a command at the inputs.

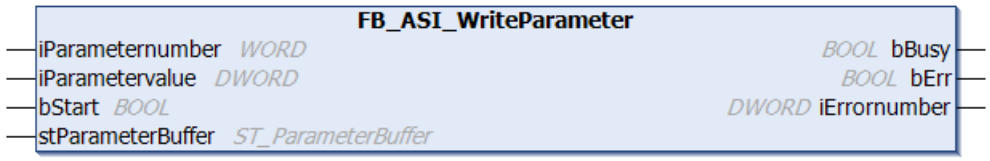
Command-specific error code (decimal)	Description
1	Communication timeout
2	ASI slave address does not exist
3 - 10	Reserved
11	ASI slave is not activated (slave is not in LAS)
12	An error occurred during communication.
13	Data exchange bit (CN.4) not set

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

Requirements

Development environment	Target platform	IO hardware	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	PC or CX (x86)	ASI master terminal	Tc2_IoFunctions (IO)

3.2.5 FB_ASI_WriteParameter



VAR_IN_OUT

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

stParameterBuffer: Data buffer for the background communication (type: ST_ParameterBuffer [▶ 131]).

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;(* Error code 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 a command at the inputs.

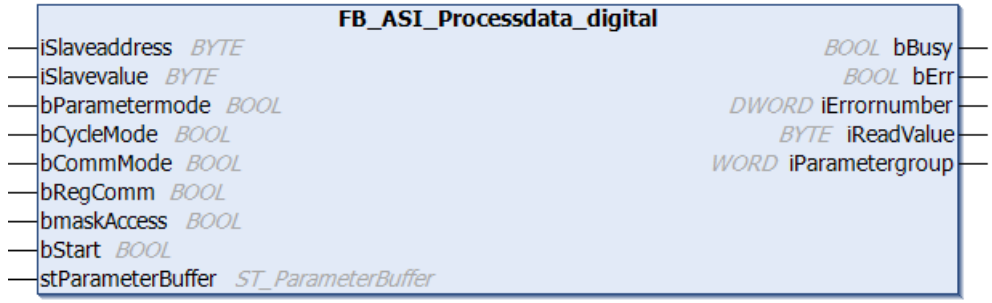
Command-specific error code (decimal)	Description
1	Communication timeout
2	ASI slave address does not exist
3 - 10	Reserved
11	ASI slave is not activated (slave is not in LAS)
12	An error occurred during communication.
13	Data exchange bit (CN.4) not set

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

Requirements

Development environment	Target platform	IO hardware	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	PC or CX (x86)	ASI master terminal	Tc2_IoFunctions (IO)

3.2.6 FB_ASI_Processdata_digital



VAR_IN_OUT

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

stParameterBuffer: Data buffer for the background communication (type: ST_ParameterBuffer [▶ 131]).

VAR_INPUT

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

iSlaveaddress: slave address.

iSlavevalue: Process value.

bParametermode: 0: Read, 1: Write.

bCycleMode: 0: Acyclic, 1:Cyclic (permanent Read/Write). If this bit is set, the bBusy output is only cancelled when the bStart input is pulled to FALSE. No current value is present at the output yet if the bStart input is pulled to FALSE too early.

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;(* Error code 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 a command at the inputs.

Command-specific error code (decimal)	Description
1	Communication timeout
2	ASI slave address does not exist
3 - 10	Reserved
11	ASI slave is not activated (slave is not in LAS)
12	An error occurred during communication.
13	Data exchange bit (CN.4) not set

iReadvalue: Process value.

iParametergroup: parameter group.

Requirements

Development environment	Target platform	IO hardware	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	PC or CX (x86)	ASI master terminal	Tc2_IoFunctions (IO)

3.2.7 FB_ASI_ParameterControl

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

Call the block

This block must always be called cyclically

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

stParameterBuffer: Data buffer for the background communication (type: ST_ParameterBuffer [▶ 131]).

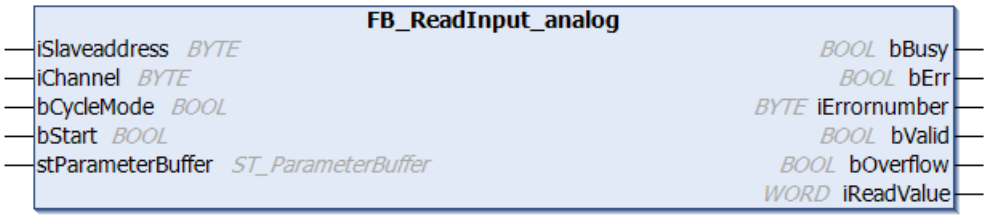
stParameter_IN: Input data from the ASI terminal (type: ST_Parameter_IN [▶ 129]).

stParameter_OUT: Input data from the ASI terminal (type: ST_Parameter_OUT [▶ 130]).

Requirements

Development environment	Target platform	IO hardware	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	PC or CX (x86)	ASI master terminal	Tc2_IoFunctions (IO)

3.2.8 FB_ReadInput_analog



VAR_IN_OUT

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

stParameterBuffer: Data buffer for the background communication (type: ST_ParameterBuffer [▶ 131]).

VAR_INPUT

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

iSlaveaddress: slave address.

iChannel: slave channel.

bCycleMode: 0: Acyclic, 1:Cyclic (permanent Read/Write). If this bit is set, the bBusy output is only cancelled when the bStart input is pulled to FALSE. No current value is present at the output yet if the bStart input is pulled to FALSE too early.

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;(* Error code of ASI-Master *)
    bValid       : BOOL;
    bOverflow    : BOOL;
    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 a command at the inputs.

Command-specific error code (decimal)	Description
1	Communication timeout
2	ASI slave address does not exist
3 - 10	Reserved
11	ASI slave is not activated (slave is not in LAS)
12	An error occurred during communication.
13	Data exchange bit (CN.4) not set

bValid: Validity of the values read.

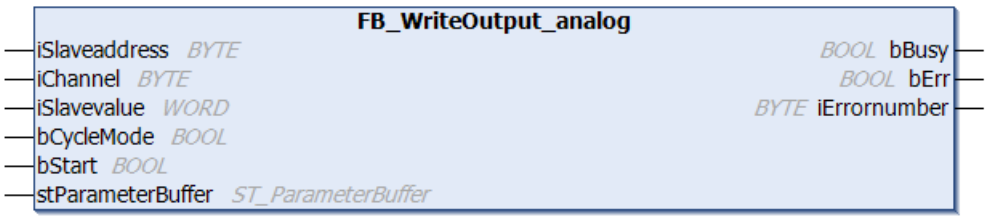
bOverflow: Slave has a value outside of its range of values.

iReadvalue: Process value.

Requirements

Development environment	Target platform	IO hardware	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	PC or CX (x86)	ASI master terminal	Tc2_IoFunctions (IO)

3.2.9 FB_WriteOutput_analog



VAR_IN_OUT

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

stParameterBuffer: Data buffer for the background communication (type: ST_ParameterBuffer [▶ 131]).

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 that is to be written.

bCycleMode: 0: Acyclic, 1:Cyclic (permanent Read/Write). If this bit is set, the bBusy output is only cancelled when the bStart input is pulled to FALSE. No current value is present at the output yet if the bStart input is pulled to FALSE too early.

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;(* Error code 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 a command at the inputs.

Command-specific error code (decimal)	Description
1	Communication timeout
2	ASI slave address does not exist
3 - 10	Reserved
11	ASI slave is not activated (slave is not in LAS)
12	An error occurred during communication.
13	Data exchange bit (CN.4) not set

bValid: Validity of the values read.

bOverflow: Slave has a value outside of its range of values.

Requirements

Development environment	Target platform	IO hardware	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	PC or CX (x86)	ASI master terminal	Tc2_IoFunctions (IO)

3.3 AX200x Profibus

3.3.1 Overview

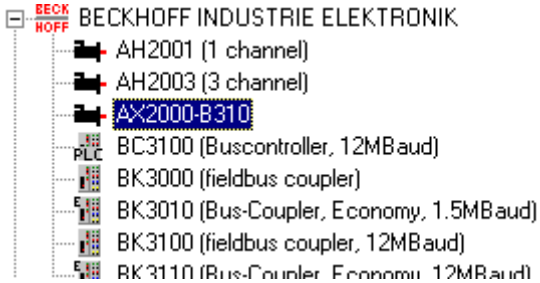
Function blocks for access to the AX20XX via Profibus. The requirement for operation on the Profibus is the use of an FC310x with a firmware version higher than 1.20.

Function blocks

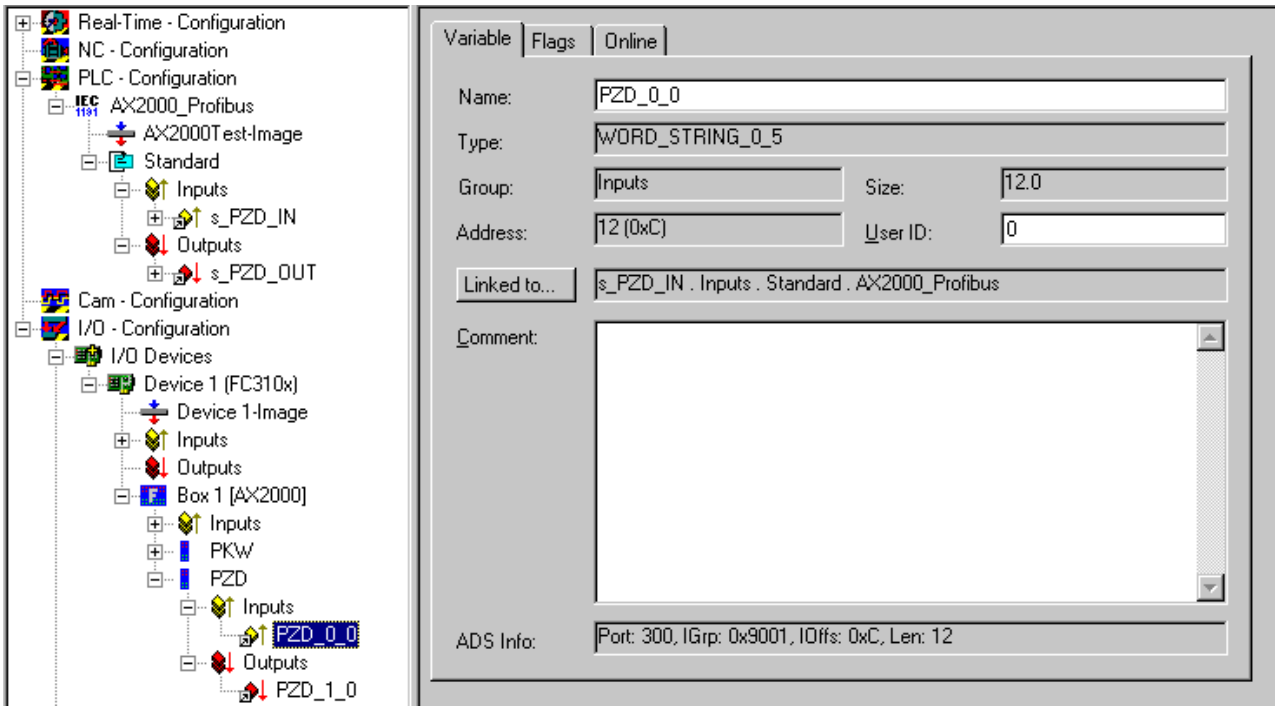
Name	Description
FB_AX2000_Parameter [▶ 40]	Write/read the data for the parameterization of the drive. **It must be remembered that the "STOP" input of the AX2000AXACT block must be held TRUE while a parameter that will change the operating mode is being written.**
FB_AX2000_AXACT [▶ 38]	Start axis actions (must always be called cyclically)
FB_AX2000_Jogmode [▶ 39]	Jogging operation
FB_AX2000_Reference [▶ 41]	Set the reference point or start a homing run
FB_AX200X_Profibus [▶ 42]	This block combines the three preceding blocks. It offers the complete interface to the AX2000 with access to all functions (except 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.



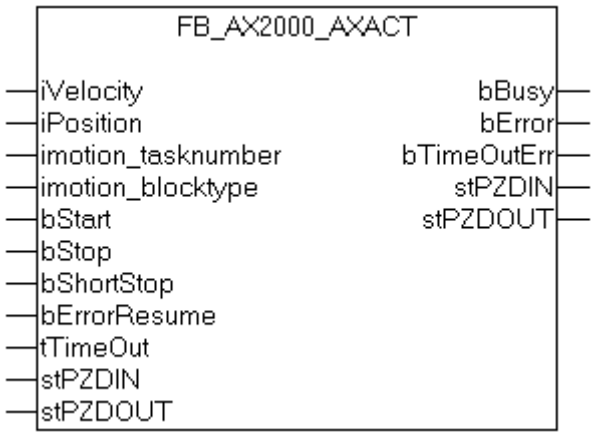
The I/O variables can now be linked directly to the corresponding I/O variables in the PLC application in the "PZD" (process data) module for the AX2000 box. The "PKW" module is not linked, because the PKW data is transmitted by ADS.



Notes on use of the function blocks

- Instances of the I/O structures stPZD_IN and stPZD_OUT must be created and addressed in order 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.

3.3.2 FB_AX2000_AXACT



VAR_IN_OUT

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

stPZDIN : Data words from the drive to the PLC (type: ST_PZD_IN [▶ 133]).

stPZDOUT: Data words from the PLC to the drive (type: ST_PZD_OUT [▶ 133]).

VAR_INPUT

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

bMode_DigitalSpeed: This is set if the drive is to be placed in the ‘digital speed’ operating mode during the initialization.

iDigitalSpeed: Speed in the ‘digital speed’ operating mode.

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 allowed for the execution of the command.

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.

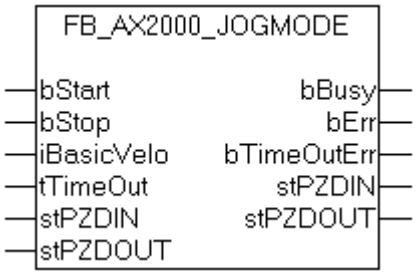
bError: This output shows the error status.

bTimeOutErr : TimeOut error.

Requirements

Development environment	Target platform	IO hardware	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	PC or CX (x86)	AX2000 Profibus box	Tc2_IoFunctions (IO)

3.3.3 FB_AX2000_JogMode



Jogging operation.

VAR_IN_OUT

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

stPZDIN : Data words from the drive to the PLC (type: ST_PZD_IN [▶ 133]).

stPZDOUT: Data words from the PLC to the drive (type: ST_PZD_OUT [▶ 133]).

VAR_INPUT

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

bStart: Start jogging mode.

bStop : Stop jogging mode.

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

tTimeOut: Maximum time allowed for the execution of the command.

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.

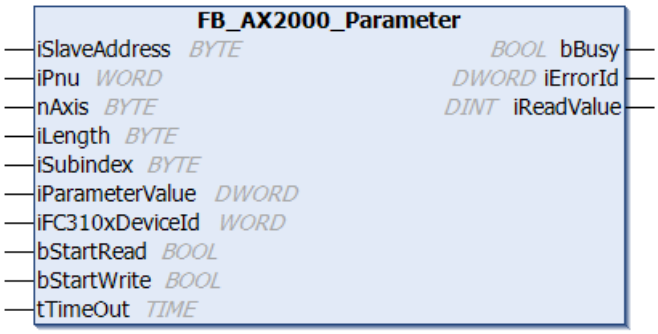
bErr: This output shows the error status.

bTimeOutErr : TimeOut error.

Requirements

Development environment	Target platform	IO hardware	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	PC or CX (x86)	AX2000 Profibus box	Tc2_IoFunctions (IO)

3.3.4 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 := DEFAULT_ADS_TIMEOUT;
END_VAR
```

iSlaveAddress: Station address.

iPnu: Selection of the parameter to be read or written. List of the available parameter numbers [▶ 148].

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 [▶ 38] block.

tTimeOut: Maximum time allowed for the execution of the command.

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.

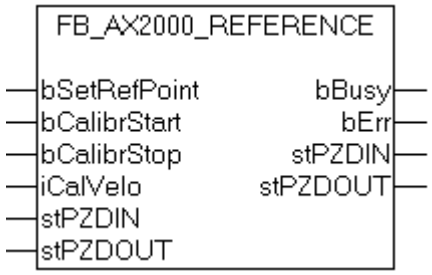
iErrorId: contains the command-specific error code of the most recently executed command. Is reset to 0 by the execution of a command at the inputs.

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

Requirements

Development environment	Target platform	IO hardware	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	PC or CX (x86)	AX2000 Profibus box	Tc2_IoFunctions (IO)

3.3.5 FB_AX2000_Reference



Homing.

VAR_IN_OUT

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

stPZDIN : Data words from the drive to the PLC (type: ST_PZD_IN [▶ 133]).

stPZDOUT: Data words from the PLC to the drive (type: ST_PZD_OUT [▶ 133]).

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 : Setting the reference point.

bCalibrStart: Start the homing run.

bCalibrStop: Stop the homing run.

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.

VAR_OUTPUT

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

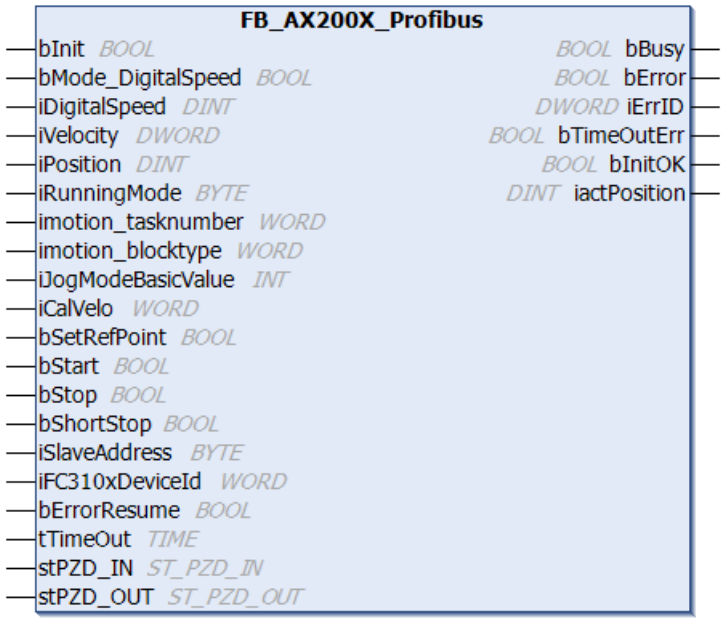
bBusy: This output remains TRUE until the block has executed a command. While Busy = TRUE, no new command will be accepted at the inputs. Please note that it is not the execution of the service but its acceptance whose time is monitored.

bErr: This output shows the error status.

Requirements

Development environment	Target platform	IO hardware	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	PC or CX (x86)	AX2000 Profibus box	Tc2_IoFunctions (IO)

3.3.6 FB_AX200X_Profibus



VAR_IN_OUT

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

stPZD_IN: Data words from the drive to the PLC (type: ST_PZD_IN [▶ 133]).

stPZD_OUT: Data words from the PLC to the drive (type: ST_PZD_OUT [▶ 133]).

VAR_INPUT

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

bInit: Initialization of the drive. If bInit is TRUE then operating mode 2, "positioning", is set in the drive.

bMode_DigitalSpeed: This is set if the drive is to be placed in the ‘digital speed’ operating mode during the initialization.

iDigitalSpeed: Speed in the ‘digital speed’ operating mode.

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

iPosition: Target position.

iRunningMode: 0: digital speed, 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: Maximum time allowed for the execution of the command.

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 shows the error status.

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

bTimeOutErr : TimeOut error.

bInitOK : Initialization state of the drive, bInit:= TRUE: The drive is initialized, and is in operating mode 2, "positioning".

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

Requirements

Development environment	Target platform	IO hardware	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	PC or CX (x86)	AX2000 Profibus box	Tc2_IoFunctions (IO)

3.4 Beckhoff Lightbus

3.4.1 IOF_LB_BreakLocationTest

The IOF_LB_BreakLocationTest function block carries out a break location test in a Beckhoff Lightbus optical fiber ring and can locate possible break locations. 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 cannot be located.

VAR_INPUT

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

NETID: The network address of the TwinCAT computer on which the function is to be executed can be entered here (type: T_AmsNetID). 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 during the hardware configuration.

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

TMOUT: States the length of the timeout that may not be exceeded during execution of the 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 a feedback is received.

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

ERRID: Supplies the ADS error number [▶ 153] when the ERR output is set.

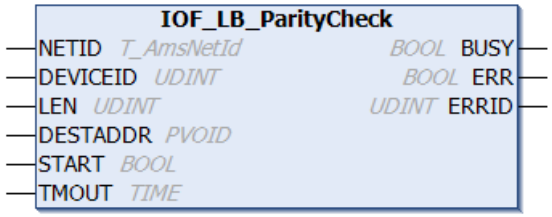
BREAK: This flag is set if a break location is detected in the optical fiber ring.

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

Requirements

Development environment	Target platform	IO hardware	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	None. This functionality is not supported by TwinCAT 3 at present!	Beckhoff Lightbus Master: C1220 ISA; FC200x PCI	Tc2_IoFunctions (IO)

3.4.2 IOF_LB_ParityCheck



The IOF_LB_ParityCheck function block reads the parity error counters of the Beckhoff Lightbus modules (e.g. BK2000). In contrast to the function block IOF_LB_ParityCheckWithReset [▶ 47], the counter states are not reset. 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  : PVOID;
    START     : BOOL;
    TMOUT     : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

NETID: The network address of the TwinCAT computer on which the function is to be executed can be entered here (type: T_AmsNetID). 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 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 block is activated by a positive edge at this input.

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

VAR_OUTPUT

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

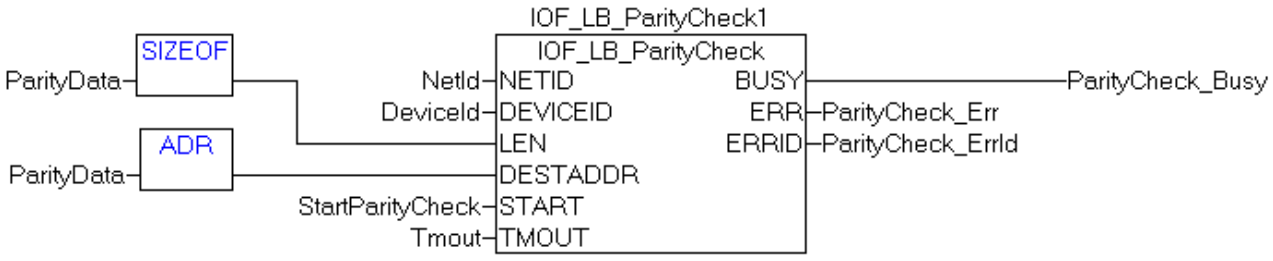
BUSY: When the function block is activated this output is set. It remains set until a feedback is received.

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

ERRID: Supplies the ADS error number [▶ 153] when the ERR output is set.

Example:

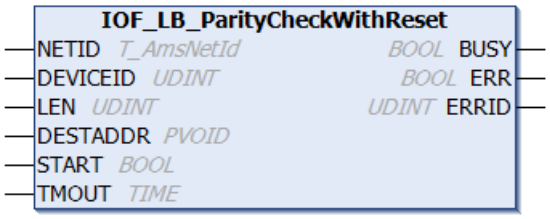
```
PROGRAM MAIN
VAR
    IOF_LB_ParityCheck1 : IOF_LB_ParityCheck;
    ParityData          : ARRAY[1..256] OF BYTE;
    StartParityCheck    : BOOL;
    ParityCheck_Busy    : BOOL;
    ParityCheck_Err     : BOOL;
    ParityCheck_ErrId   : UDINT;
END_VAR
```



Requirements

Development environment	Target platform	IO hardware	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	None. This functionality is not supported by TwinCAT 3 at present!	Beckhoff Lightbus Master: C1220 ISA; FC200x PCI	Tc2_IoFunctions (IO)

3.4.3 IOF_LB_ParityCheckWithReset



The IOF_LB_ParityCheckWithReset function block reads the parity error counters of the Beckhoff Lightbus modules (e.g. BK2000). The counters are then reset. 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  : PVOID;
    START     : BOOL;
    TMOUT     : TIME;
END_VAR
```

NETID: The network address of the TwinCAT computer on which the function is to be executed can be entered here (type: T_AmsNetID). 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 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 block is activated by a positive edge at this input.

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

VAR_OUTPUT

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

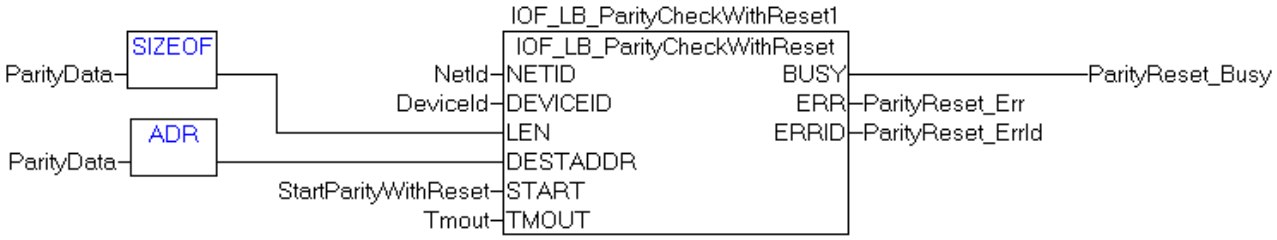
BUSY: When the function block is activated this output is set. It remains set until a feedback is received.

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

ERRID: Supplies the ADS error number [▶ 153] when the ERR output is set.

Example:

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



Requirements

Development environment	Target platform	IO hardware	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	None. This functionality is not supported by TwinCAT 3 at present!	Beckhoff Lightbus Master: C1220 ISA; FC200x PCI	Tc2_IoFunctions (IO)

3.5 Beckhoff UPS (configured with Windows UPS Service

3.5.1 FB_GetUPSStatus

Requirements:

- Beckhoff UPS software components have been installed:
  - Windows 7, Windows Embedded Standard 7 and higher: Configuration dialog under "*Start->Programs->Beckhoff->UPS Software Components"*.
  - NT4, Win2K, WinXP, WinXP embedded: Additional tabs under "*Control Panel->Power Options->Beckhoff UPS Configuration"* or "*Control Panel->Power Options->UPS*".
  - Beckhoff CE devices with 24V UPS support are delivered with a special Beckhoff Battery Driver for Windows CE. In these devices the driver is included in the standard CE image.
- The UPS has been activated and configured. You can find more information about UPS configuration in the corresponding advanced UPS software and device documentation.
  - Windows 7, Windows Embedded Standard 7 and higher: Configuration dialog under "*Start->Programs->Beckhoff->UPS Software Components"*.
  - NT4, Win2K, WinXP, WinXP embedded: Configuration dialog under "*Control Panel->Power Options->Beckhoff UPS Configuration"*.
  - Windows CE: By default the UPS function is disabled and must be enabled via a RegFile. Newer images have a configuration dialog under "*Start->Control Panel->BECKHOFF UPS Configuration"*.

The function block FB_GetUPSStatus reads the status of the UPS hardware from the PLC. The function block is level triggered, which means that the status information of the UPS 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: A string with the network address of the TwinCAT computer whose UPS status is to be read can be entered here (type: T_AmsNetID). If it is to be run on the local computer, an empty string can be entered.

nPort: The ADS port number (type: T_AmsPort). Set this value to zero. Other port numbers are reserved for future applications.

bEnable: The UPS status is read cyclically if this input is set.

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_UPSstatus 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 [▶ 153] or the command-specific error code (table) when the *bError* output is set.

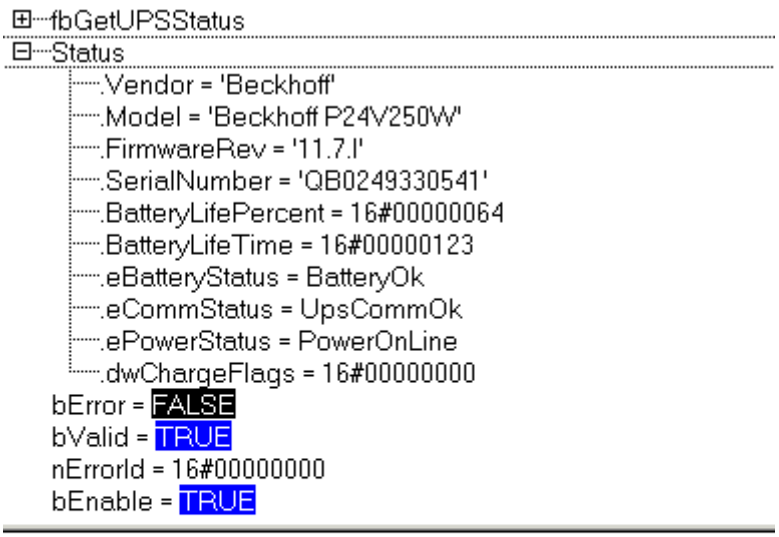
Error Codes	Error description
0x0000	No error
0x8001	UPS configuration error. It is possible that the UPS is not configured correctly or that no UPS is configured at all.
0x8002	Communication error. Communication with the UPS was interrupted.
0x8003	Error during the reading of the status data.

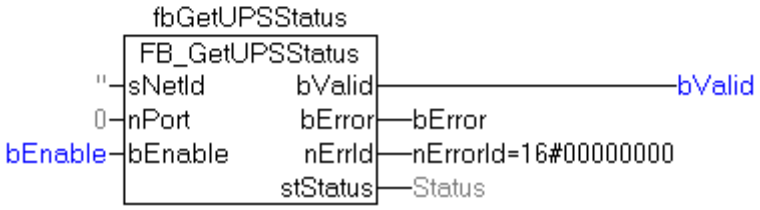
stStatus: Structure with the status information of the UPS (type: ST_UPSStatus [▶ 137]).

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

Example:

Online data with status information of a UPS:



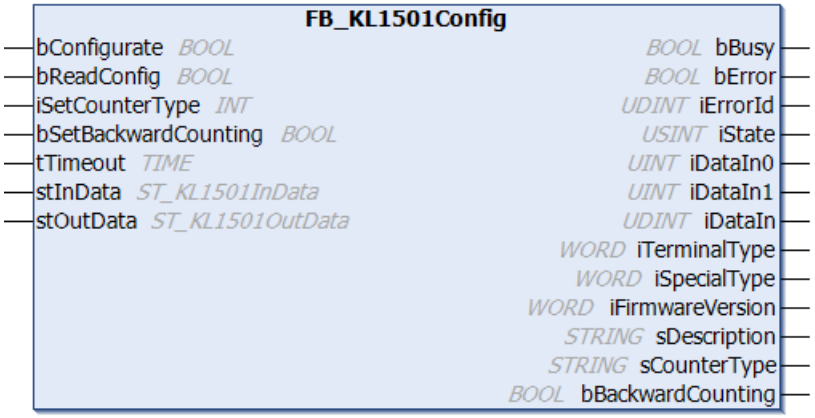


Requirements

Development environment	Target platform	UPS hardware	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	PC or CX (x86, x64, ARM)	• 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); • The APC devices that come supplied with Beckhoff Industrial PC support the Smart protocol and can be configured with the Windows UPS service.	Tc2_IoFunctions (IO)

3.6 Bus Terminal configuration

3.6.1 FB_KL1501Config



Function block for parameterizing a KL1501: Single-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

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

bConfigurate: A rising edge starts the configuration page. First, the general terminal data, i.e. "terminal designation", "special version" and "firmware version", are read. Then, the specified settings are written to the corresponding registers and finally read again for verification and information. The read information is displayed at the function block outputs. During this sequence the output *bBusy* is TRUE, and no further command, e.g. *bReadConfig*, is accepted.

bReadConfig: A rising edge only starts a read sequence. The general terminal data, i.e. "terminal designation", "special version" and "firmware version", are read, followed by the set configuration parameters. The read information is displayed at the function block outputs. During the read sequence the output *bBusy* is TRUE, and no further command, e.g. *bConfigurate*, is accepted.

iSetCounterType: Counter type input. The setting is based on the table below.

bSetBackwardCounting: TRUE at this input reverses the count direction.

tTimeout: The terminal configuration and reading of the configuration must be complete within the time entered here. Otherwise an error with a corresponding error number is issued at the outputs *bError* and *iErrorId*.

iSetCounterType	Counter type
0	32 bit up/down counter
1	2 x 16 bit up counter
2	32-bit gated counter, gate input Low disables the counter
3	32-bit gated counter, gate input High disables the counter

VAR_OUTPUT

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

bBusy: As long as the read or configuration sequence is in progress, this output is TRUE.

bError: This output is switched to TRUE, if an error occurred while a command (Configure or Read) is executed. The command-specific error code is contained in *iErrorId*.

iErrorId: contains the command-specific error code of the most recently executed command. It is reset to 0 by activating the function block again via the inputs *bConfigurate* or *bReadConfig*. See Error codes [▶ 153].

iState: Corresponds to the status variable of the process data, i.e. *stInData*.iState, see VAR_IN_OUT. During command execution (*bBusy* = TRUE) this output is set to 0. This output is suitable for status assessment during normal terminal operation. Spurious states during configuration and reading through the register communication are hidden.

iDataIn0: Corresponds to the data variable of the process data, i.e. *stInData.arrDataIn[0]*, see VAR_IN_OUT. During command execution (*bBusy* = TRUE) this output retains the value it had before the command was called. This output is suitable for direct process data processing during normal terminal operation. Spurious states during configuration and reading through the register communication are hidden.

iDataIn1: Corresponds to the data variable of the process data, i.e. *stInData.arrDataIn[1]*, see VAR_IN_OUT. During command execution (*bBusy* = TRUE) this output retains the value it had before the command was called. This output is suitable for direct process data processing during normal terminal operation. Spurious states during configuration and reading through the register communication are hidden.

iDataIn: This variable of type UDINT is used for improved evaluation, if a 32-bit counter is selected. It consists of the two variables *iDataIn0* and *iDataIn1* (both of type UINT) referred to above. *iDataIn0* is used for the low-order part, *iDataIn1* for the high-order part.

iTerminalType: Contents of register 8 (terminal designation). When used with the correct terminal, the content should be 0x05DD (1501dec).

iSpecialType: Contents of register 29 (special version).

iFirmwareVersion: Contents of register 9 (firmware version).

sDescription: Terminal designation, special version and firmware version as string (e.g. 'terminal KL1501-0000 / firmware 1C').

sCounterType: Set counter mode in plain text.

bBackwardCounting: TRUE: The count direction was reversed.

VAR_IN_OUT

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

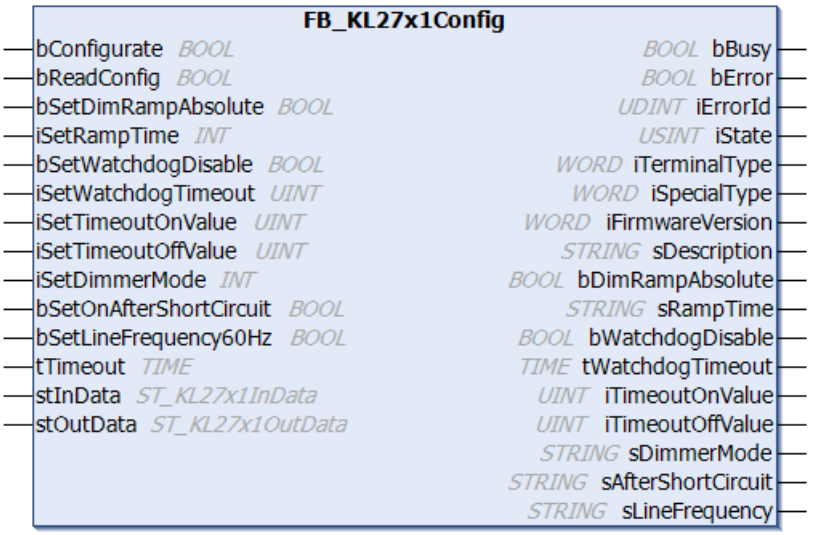
stInData : Reference to the structure of the input process image (type: ST_KL1501InData [▶ 141]).

stOutData : Reference to the structure of the output process image (type: ST_KL1501OutData [▶ 141]).

Requirements

Development environment	Target platform	IO Hardware	PLC libraries to include
TwinCAT v3.1.4018.26	PC/CX	KL1501	Tc2_IoFunctions from v3.3.5.0

3.6.2 FB_KL27x1Config



Function block for parameterizing a KL2751 / KL2761: Single-channel dimmer terminal.

VAR_INPUT

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

bConfigurate: A rising edge starts the configuration page. First, the general terminal data, i.e. "terminal designation", "special version" and "firmware version", are read. Then, the specified settings are written to the corresponding registers and finally read again for verification and information. The read information is displayed at the function block outputs. During this sequence the output *bBusy* is TRUE, and no further command, e.g. *bReadConfig*, is accepted.

bReadConfig: A rising edge only starts a read sequence. The general terminal data, i.e. "terminal designation", "special version" and "firmware version", are read, followed by the set configuration parameters. The read information is displayed at the function block outputs. During the read sequence the output *bBusy* is TRUE, and no further command, e.g. *bConfigurate*, is accepted.

bSetDimRampAbsolute: FALSE: The set ramp time *iSetRampTime* refers to the complete data area (0 - 32767). The smaller the discontinuity, the shorter the ramp time. TRUE: Each switching step, whatever the size, requires the same time, as entered under *iSetRampTime*.

iSetRampTime: Ramp time input. The setting is based on the table below.

bSetWatchdogDisable: The internal watchdog is disabled.

iSetWatchdogTimeout: Watchdog time as multiple of 10 ms.

iSetTimeoutOnValue: This input specifies the light value that is output when a fieldbus error occurs with current process data > 0.

iSetTimeoutOffValue: This input specifies the light value that is output when a fieldbus error occurs with current process data = 0.

iSetDimmerMode: This input is used to set the dimmer mode. The setting is based on the table below.

bSetOnAfterShortCircuit: FALSE: After a short circuit the light remains switched off. TRUE: The light is switched on again after a short circuit.

bSetLineFrequency60Hz: FALSE: mains frequency = 50 Hz. TRUE: mains frequency = 60 Hz.

tTimeout: The terminal configuration and reading of the configuration must be complete within the time entered here. Otherwise an error with a corresponding error number is issued at the outputs *bError* and *iErrorId*.

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 phase control
2	Leading edge phase control
3	Rectifier mode, positive (positive half-wave with leading edge phase control)
4	Rectifier mode, negative (negative half-wave with leading edge phase control)

VAR_OUTPUT

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

bBusy: As long as the read or configuration sequence is in progress, this output is TRUE.

bError: This output is switched to TRUE, if an error occurred while a command (Configure or Read) is executed. The command-specific error code is contained in *iErrorId*.

iErrorId: contains the command-specific error code of the most recently executed command. It is reset to 0 by activating the function block again via the inputs *bConfigurate* or *bReadConfig*. See Error codes [▶ 153].

iState: Corresponds to the status variable of the process data, i.e. *stInData*.iState, see VAR_IN_OUT. During command execution (*bBusy* = TRUE) this output is set to 0. This output is suitable for status assessment during normal terminal operation. Spurious states during configuration and reading through the register communication are hidden.

iDataIn: Corresponds to the data variable of the process data, i.e. *stInData.iDataIn*, see VAR_IN_OUT. During command execution (*bBusy* = TRUE) this output retains the value it had before the command was called. This output is suitable for direct process data processing during normal terminal operation. Spurious states during configuration and reading through the register communication are hidden.

iTerminalType: Contents of register 8 (terminal designation). If the correct terminal is used, the content should be 0x0ABF (2751dec) or 0x0AC9 (2761dec).

iSpecialType: Contents of register 29 (special version).

iFirmwareVersion: Contents of register 9 (firmware version).

sDescription: Terminal designation, special version and firmware version as string (e.g. 'terminal KL27x1-0000 / firmware 1C').

bDimRampAbsolute: TRUE: The dimming ramp is set absolute, i.e. each switching step requires the same ramp time, as specified under *iSetRampTime*.

sRampTime: Set ramp time in plain text.

bWatchdogDisable: TRUE: Watchdog is disabled.

tWatchdogTimeout: Set watchdog time.

iTimeoutOnValue: Set light value, which is output when a fieldbus error occurs with current process data > 0.

–iTimeoutOffValue: Set light value, which is output when a fieldbus error occurs with current process data = 0.

sDimmerMode: Set dimmer mode in plain text.

sAfterShortCircuit: Set behavior after short circuit in plain text.

sLineFrequency: Set mains frequency in plain text.

VAR_IN_OUT

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

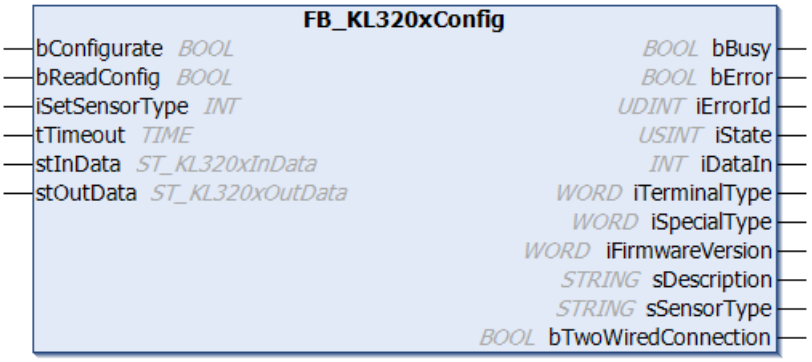
stInData : Reference to the structure of the input process image (type: ST_KL27x1InData [▶ 141]).

stOutData : Reference to the structure of the output process image (type: ST_KL27x1OutData [▶ 142]).

Requirements

Development environment	Target platform	IO Hardware	PLC libraries to include
TwinCAT v3.1.4018.26	PC/CX	KL2751, KL2761	Tc2_IoFunctions from v3.3.5.0

3.6.3 FB_KL320xConfig



Function block for parameterizing a KL3201, KL3202 or KL3204: 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

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

bConfigurate: A rising edge starts the configuration page. First, the general terminal data, i.e. "terminal designation", "special version" and "firmware version", are read. Then, the specified settings are written to the corresponding registers and finally read again for verification and information. The read information is displayed at the function block outputs. During this sequence the output *bBusy* is TRUE, and no further command, e.g. *bReadConfig*, is accepted.

bReadConfig: A rising edge only starts a read sequence. The general terminal data, i.e. "terminal designation", "special version" and "firmware version", are read, followed by the set configuration parameters. The read information is displayed at the function block outputs. During the read sequence the output *bBusy* is TRUE, and no further command, e.g. *bConfigurate*, is accepted.

iSetSensorType: This input is used to set the sensor. The setting is based on the table below.

tTimeout: The terminal configuration and reading of the configuration must be complete within the time entered here. Otherwise an error with a corresponding error number is issued at the outputs *bError* and *iErrorId*.

iSetSensorType	Element
0	PT100
1	NI100
2	PT1000
3	PT500
4	PT200
5	NI1000
6	NI120
7	Output 10.0 Ω - 5000.0 Ω
8	Output 10.0 Ω – 1200.0 Ω
9	PT1000 - two-wire connection - **not permitted with KL3204!**

VAR_OUTPUT

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

bBusy: As long as the read or configuration sequence is in progress, this output is TRUE.

bError: This output is switched to TRUE, if an error occurred while a command (Configure or Read) is executed. The command-specific error code is contained in *iErrorId*.

iErrorId: contains the command-specific error code of the most recently executed command. It is reset to 0 by activating the function block again via the inputs *bConfigurate* or *bReadConfig*. See Error codes [▶ 153].

iState: Corresponds to the status variable of the process data, i.e. *stInData*.iState, see VAR_IN_OUT. During command execution (*bBusy* = TRUE) this output is set to 0. This output is suitable for status assessment during normal terminal operation. Spurious states during configuration and reading through the register communication are hidden.

iDataIn: Corresponds to the data variable of the process data, i.e. *stInData.iDataIn*, see VAR_IN_OUT. During command execution (*bBusy* = TRUE) this output retains the value it had before the command was called. This output is suitable for direct process data processing during normal terminal operation. Spurious states during configuration and reading through the register communication are hidden.

iTerminalType: Contents of register 8 (terminal designation). The register content must match the terminal used: 0xC81 for KL3201, 0xC82 for KL3202 and 0xC84 for KL3204.

iSpecialType: Contents of register 29 (special version).

iFirmwareVersion: Contents of register 9 (firmware version).

sDescription: Terminal designation, special version and firmware version as string (e.g. 'terminal KL320x-0010 / firmware 1C').

sSensorType: Set sensor type in plain text.

bTwoWiredConnection: The sensor type is parameterized in the two-wire connection.

VAR_IN_OUT

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

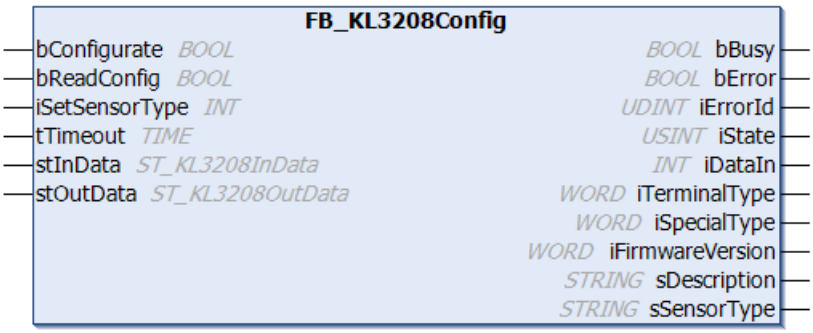
stInData : Reference to the structure of the input process image (type: ST_KL320xInData [▶ 142]).

stOutData : Reference to the structure of the output process image (type: ST_KL320xOutData [▶ 142]).

Requirements

Development environment	Target platform	IO Hardware	PLC libraries to include
TwinCAT v3.1.4018.26	PC/CX	KL3201, KL3202, KL3204	Tc2_IoFunctions from v3.3.5.0

3.6.4 FB_KL3208Config



Function block for parameterizing 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

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

bConfigurate: A rising edge starts the configuration page. First, the general terminal data, i.e. "terminal designation", "special version" and "firmware version", are read. Then, the specified settings are written to the corresponding registers and finally read again for verification and information. The read information is displayed at the function block outputs. During this sequence the output *bBusy* is TRUE, and no further command, e.g. *bReadConfig*, is accepted.

bReadConfig: A rising edge only starts a read sequence. The general terminal data, i.e. "terminal designation", "special version" and "firmware version", are read, followed by the set configuration parameters. The read information is displayed at the function block outputs. During the read sequence the output *bBusy* is TRUE, and no further command, e.g. *bConfigurate*, is accepted.

iSetSensorType: This input is used to set the sensor. The setting is based on the table below.

tTimeout: The terminal configuration and reading of the configuration must be complete within the time entered here. Otherwise an error with a corresponding error number is issued at the outputs *bError* and *iErrorId*.

iSetSensorType	Element
0	PT1000
1	NI1000
2	RSNI1000 (NI1000 based on Landis & Staefa characteristics: 1000 Ω at 0 °C and 1500 Ω at 100 °C)
3	NTC1K8
4	NTC1K8_TK
5	NTC2K2
6	NTC3K
7	NTC5K
8	NTC10K
9	NTC10KPRE
10	NTC10K_3204
11	NTC10KTYP2
12	NTC10KTYP3
13	NTC10KDALE
14	NTC10K3A221
15	NTC20K
16	Potentiometer, resolution 0.1 Ω
17	Potentiometer, resolution 1 Ω
18	NTC100K

VAR_OUTPUT

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

bBusy: As long as the read or configuration sequence is in progress, this output is TRUE.

bError: This output is switched to TRUE, if an error occurred while a command (Configure or Read) is executed. The command-specific error code is contained in *iErrorId*.

iErrorId: contains the command-specific error code of the most recently executed command. It is reset to 0 by activating the function block again via the inputs *bConfigurate* or *bReadConfig*. See Error codes [▶ 153].

iState: Corresponds to the status variable of the process data, i.e. *stInData*.iState, see VAR_IN_OUT. During command execution (*bBusy* = TRUE) this output is set to 0. This output is suitable for status assessment during normal terminal operation. Spurious states during configuration and reading through the register communication are hidden.

iDataIn: Corresponds to the data variable of the process data, i.e. *stInData.iDataIn*, see VAR_IN_OUT. During command execution (*bBusy* = TRUE) this output retains the value it had before the command was called. This output is suitable for direct process data processing during normal terminal operation. Spurious states during configuration and reading through the register communication are hidden.

iTerminalType: Contents of register 8 (terminal designation). When used with the correct terminal, the content should be 0x0C88 (3208dec).

iSpecialType: Contents of register 29 (special version).

iFirmwareVersion: Contents of register 9 (firmware version).

sDescription: Terminal designation, special version and firmware version as string (e.g. 'terminal KL3208-0010 / firmware 1C').

sSensorType: Set sensor type in plain text.

VAR_IN_OUT

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

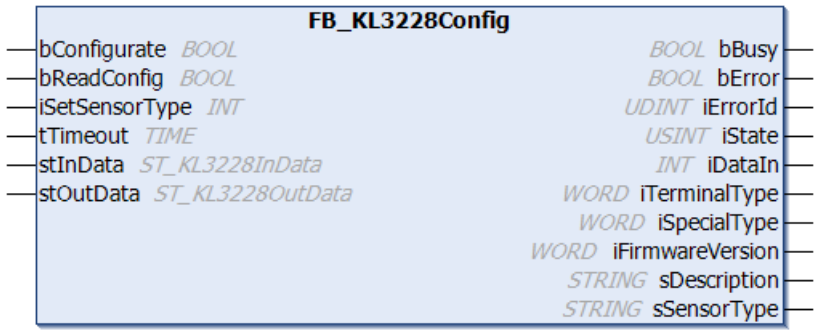
stInData : Reference to the structure of the input process image (type: ST_KL3208InData [▶ 143]).

stOutData : Reference to the structure of the output process image (type: ST_KL3208OutData [▶ 143]).

Requirements

Development environment	Target platform	IO Hardware	PLC libraries to include
TwinCAT v3.1.4018.26	PC/CX	KL3208	Tc2_IoFunctions from v3.3.5.0

3.6.5 FB_KL3228Config



Function block for parameterizing a KL3228: 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

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

bConfigurate: A rising edge starts the configuration page. First, the general terminal data, i.e. "terminal designation", "special version" and "firmware version", are read. Then, the specified settings are written to the corresponding registers and finally read again for verification and information. The read information is displayed at the function block outputs. During this sequence the output *bBusy* is TRUE, and no further command, e.g. *bReadConfig*, is accepted.

bReadConfig: A rising edge only starts a read sequence. The general terminal data, i.e. "terminal designation", "special version" and "firmware version", are read, followed by the set configuration parameters. The read information is displayed at the function block outputs. During the read sequence the output *bBusy* is TRUE, and no further command, e.g. *bConfigurate*, is accepted.

iSetSensorType: This input is used to set the sensor. The setting is based on the table below.

tTimeout: The terminal configuration and reading of the configuration must be complete within the time entered here. Otherwise an error with a corresponding error number is issued at the outputs *bError* and *iErrorId*.

iSetSensorType	Element
0	PT1000
1	NI1000
2	RSNI1000 (NI1000 based on Landis & Staefa characteristics: 1000 Ω at 0 °C and 1500 Ω at 100 °C)

VAR_OUTPUT

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

bBusy: As long as the read or configuration sequence is in progress, this output is TRUE.

bError: This output is switched to TRUE, if an error occurred while a command (Configure or Read) is executed. The command-specific error code is contained in *iErrorId*.

iErrorId: contains the command-specific error code of the most recently executed command. It is reset to 0 by activating the function block again via the inputs *bConfigurate* or *bReadConfig*. See Error codes [▶ 153].

iState: Corresponds to the status variable of the process data, i.e. *stInData*.iState, see VAR_IN_OUT. During command execution (*bBusy* = TRUE) this output is set to 0. This output is suitable for status assessment during normal terminal operation. Spurious states during configuration and reading through the register communication are hidden.

iDataIn: Corresponds to the data variable of the process data, i.e. *stInData.iDataIn*, see VAR_IN_OUT. During command execution (*bBusy* = TRUE) this output retains the value it had before the command was called. This output is suitable for direct process data processing during normal terminal operation. Spurious states during configuration and reading through the register communication are hidden.

iTerminalType: Contents of register 8 (terminal designation). When used with the correct terminal, the content should be 0x0C9C (3228dec).

iSpecialType: Contents of register 29 (special version).

iFirmwareVersion: Contents of register 9 (firmware version).

sDescription: Terminal designation, special version and firmware version as string (e.g. 'terminal KL3228-0000 / firmware 1C').

sSensorType: Set sensor type in plain text.

VAR_IN_OUT

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

stInData : Reference to the structure of the input process image (type: ST_KL3228InData [▶ 143]).

stOutData : Reference to the structure of the output process image (type: ST_KL3228OutData [▶ 144]).

Requirements

Development environment	Target platform	IO Hardware	PLC libraries to include
TwinCAT v3.1.4018.26	PC/CX	KL3228	Tc2_IoFunctions from v3.3.5.0

3.7 CANopen

3.7.1 IOF_CAN_Layer2Command

The IOF_CAN_Layer2Command function block sends a 10-byte long command to layer 2 of a CAN master.

VAR_INPUT

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

NETID: The network address of the TwinCAT computer on which the function is to be executed can be entered here (type: T_AmsNetID). 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 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 block is activated by a positive edge at this input.

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

VAR_OUTPUT

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

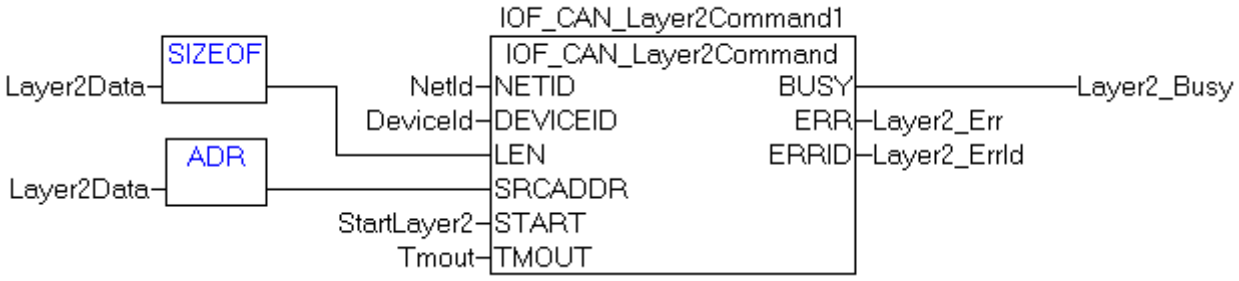
BUSY: When the function block is activated this output is set. It remains set until a feedback is received.

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

ERRID: Supplies the ADS error number [▶ 153] when the ERR output is set.

Example:

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



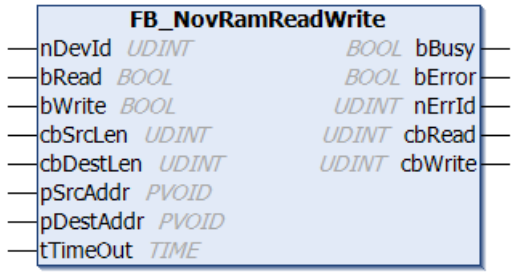
Requirements

Development environment	Target platform	IO hardware	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	None. This functionality is not supported by TwinCAT 3 at present!	HILSCHER CIF3xx COM master card	Tc2_IoFunctions (IO)

3.8 NOV/DP-RAM

3.8.1 FB_NovRamReadWrite

The following description refers to TwinCAT Version 2.8. Starting from TwinCAT version 2.9 [Build 927], a function block will no longer be required for the writing/reading of PLC data to be secured in the NOVRAM.



The *FB_NovRamReadWrite* function block accesses the NOV-RAM of the FCxxxx-000**2** fieldbus cards 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, *bRead* and *bWrite*, are set simultaneously, the data are first written into the NOV-RAM and then read back.

Comments:

In order to determine the NOV-RAM address pointer, the *FB_NovRamReadWrite* function block internally uses an instance of the ADSREAD function block. This address pointer is only determined when the *FB_NovRamReadWrite* function block is called for the first time or in the event of a change in *nDevId*. This

task requires several PLC cycles. The MEMCPY function is used to write data into the NOV-RAM or to read 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    : PVOID;
    pDestAddr   : PVOID;
    tTimeOut    : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

nDevId: The device ID 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 during the 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 NULL) 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 NULL).

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 with the data to be written into the NOV-RAM.

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: Supplies the ADS error number [▶ 153] when the *bError* output is set.

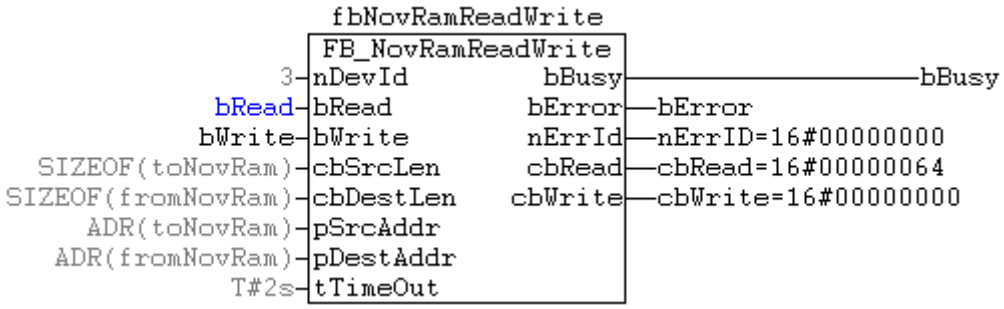
cbRead: Number of successfully read data bytes.

cbWrite: Number of successfully written data bytes.

Example:

```
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
```

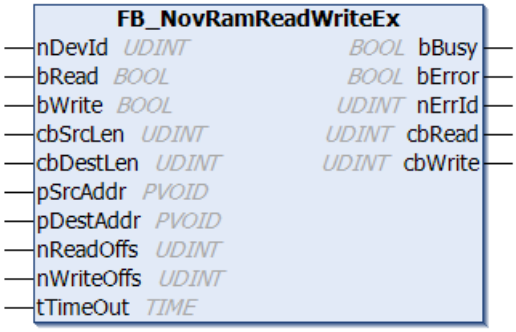


In the example a rising edge at the *bRead* input led to 100 bytes of data being read from the NOV-RAM and copied into the *fromNovRam* array.

Requirements

Development environment	Target platform	IO Hardware	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	PC or CX (x86)	FCxxxx cards with NOVRAM (FCxxxx-0002)	Tc2_IoFunctions (IO)

3.8.2 FB_NovRamReadWriteEx



The *FB_NovRamReadWriteEx* function block accesses the NOV-RAM (e.g. of the FCxxxx-0002 fieldbus cards, CX9000 NOVRAM, 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   : PVOID;
    pDestAddr  : PVOID;
    nReadOffs  : UDINT;
    nWriteOffs : UDINT;
    tTimeOut   : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

nDevId: The device ID 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 with the data to be written into the NOV-RAM.

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: Supplies the ADS error number [▶ 153] when the *bError* output is set.

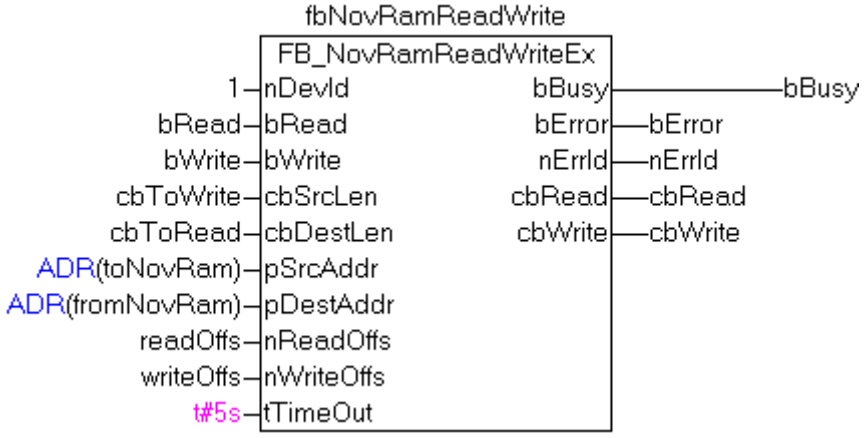
cbRead: Number of successfully read data bytes.

cbWrite: Number of successfully written data bytes.

Example:

```
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 platform	IO Hardware	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	PC or CX (x86)	FCxxxx cards with NOVRAM (FCxxxx-0002)	Tc2_IoFunctions (IO)

3.8.3 FB_GetDPRAMInfo

The *FB_GetDPRAMInfo* function block determines 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 within the TwinCAT system. The MEMCPY, MEMSET or MEMCMP functions of the PLC program can then be used for write/read access to any memory offset.

This function block returns a *Service Not Supported* error (16#701) if the NOV/DP-RAM requires a special access method (e.g. BYTE, aligned WORD). In this case the FB_NovRamReadWriteEx block can be used to read data from the NOV/DP-RAM or to write data into the NOV/DP RAM, e.g. the NOV-RAM of the *CX9000* permits only BYTE accesses, so the FB_NovRamReadWriteEx block should be used in this case.

VAR_INPUT

```
VAR_INPUT
    nDevId      : UDINT;
    bExecute    : BOOL;
    tTimeOut    : TIME := DEFAULT_ADS_TIMEOUT;
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 during the 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: Supplies the ADS error number [▶ 153] when the *bError* output is set.

stInfo: A structure containing the address and size of the NOV/DP-RAM (type: ST_NovRamAddrInfo [▶ 128]).

Example:

A pointer to an instance of the ST_NOVRAM structure variable is declared in MAIN. This pointer is initialized with the address of the NOV/DP-RAM at the start of the program. By referencing the pointer, the individual structure elements (and the NOV/DP-RAM) can be accessed for writing or reading.

```
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 platform	IO Hardware	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	PC or CX (x86)	FCxxxx cards with NOVRAM (FCxxxx-0002)	Tc2_IoFunctions (IO)

3.8.4 FB_GetDPRAMInfoEx

The FB_GetDPRAMInfo function block determines 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 within the TwinCAT system.

Along with the address (pCardAddress) and the size (iCardMemSize) of the NOV/DP-RAM the stInfo output also returns the type of access (eAccessType).

If the NOV/DP-RAM does not require special access (BYTE or WORD aligned), the MEMCPY, MEMSET or MEMCMP functions can be used in the PLC program to be able to have read/write access to a random memory offset. If the NOV/DP-RAM requires special access (BYTE or WORD aligned), then access is only possible with the appropriate data size. For this purpose use FB_NovRamReadWriteEx.

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

nDevId: 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 during the 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
    bBusy     :    BOOL;
    bError    :    BOOL;
    nErrid    :    UDINT;
    stInfo    :    ST_NovRamAddrInfoEx;
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 the execution.

nErrId: Supplies the ADS error number [▶ 153] when the bError output is set.

stInfo: A structure containing the address and size of the NOV/DP-RAM (type: ST_NovRamAddrInfoEx [▶ 129]).

Example

In MAIN a pointer to a BYTE is declared. This pointer is initialized with the address of the NOV/DP-RAM at the start of the program. Referencing the pointer makes read/write access to the NOV/DP-RAM possible.

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

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

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

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

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

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

Requirements

Development environment	Target platform	IO Hardware	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	PC or CX (x86)	FCxxxx cards with NOVRAM (FCxxxx-0002)	Tc2_IoFunctions (IO)

3.9 Profibus DPV1 (Sinamics)

3.9.1 F_CreateDpv1ReadReqPkg

The “F_CreateDpv1ReadReqPkg” function creates a DPV1 telegram for an FB_Dpv1Read [▶ 74] of one or more parameters of a drive or the controller unit of a Sinamics Profidrive (Profidrive Specification 3.1). Since Profidrives use the Motorola format and IPCs the Intel format, the function automatically swaps the byte order of parameters with data types with more than one byte in the DPV1 telegram.

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

FUNCTION F_CreateDpv1ReadReqPkg : USINT

VAR_INPUT

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

pDpv1ReqData: Pointer to an array of 240 bytes containing the DPV1 read telegram. This telegram is created by the function.

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

iDriveID: The ID is 1 for the controller unit, 2 for the drive object A and 3 for the drive object B of a double/ triple drive. The drive ID is set in the starter software. 1 – 16 is possible.

VAR_IN_OUT

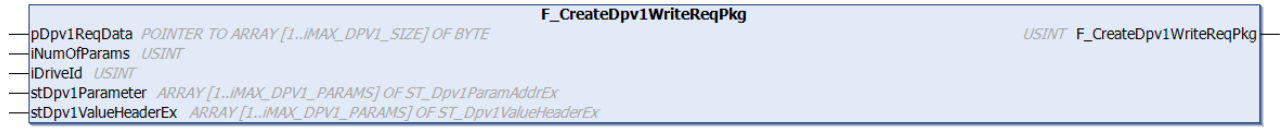
```
VAR_IN_OUT
    stDpv1Parameter : ARRAY [1..iMAX_DPV1_PARAMS] OF ST_Dpv1ParamAddrEx;(* list of parameters *)
END_VAR
```

stDpv1Parameter: Array of 39 parameters to be added to the DPV1 read telegram (type: ST_Dpv1ParamAddrEx [▶ 127]).

Requirements

Development environment	Target platform	IO hardware	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	PC or CX (x86)	Beckhoff FC310x PCI, CX1500-M310, EL6731	Tc2_IoFunctions (IO)

3.9.2 F_CreateDpv1WriteReqPkg



The “F_CreateDpv1WriteReqPkg” function creates a DPV1 telegram for an FB_Dpv1Write [▶ 76] of one or more parameters of a drive or the controller unit of a Sinamics Profidrive (Profidrive Specification 3.1). Since Profidrives use the Motorola format and IPCs the Intel format, the function automatically swaps the byte order of parameters with data types with more than one byte in the DPV1 telegram.

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

FUNCTION F_CreateDpv1WriteReqPkg : USINT

VAR_INPUT

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

pDpv1ReqData: Pointer to an array of 240 bytes containing the DPV1 write telegram. This telegram is created by the function.

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

iDriveID: The ID is 1 for the controller unit, 2 for the drive object A and 3 for the drive object B of a double/ triple drive. The 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
```

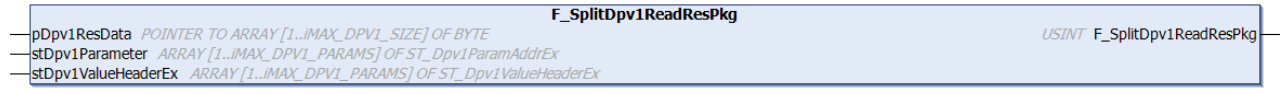
stDpv1Parameter: Array of 39 parameters to be added to the DPV1 write telegram (type: ST_Dpv1ParamAddrEx [▶ 127]).

stDpv1ValueHeaderEx: Array of 39 parameter values to be added to the DPV1 write telegram (type: ST_Dpv1ValueHeaderEx [▶ 128]).

Requirements

Development environment	Target platform	IO hardware	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	PC or CX (x86)	Beckhoff FC310x PCI, CX1500-M310, EL6731	Tc2_IoFunctions (IO)

3.9.3 F_SplitDpv1ReadResPkg



The “F_SplitDpv1ReadResPkg” function creates a DPV1 telegram for an FB_Dpv1Read [▶ 74] of one or more parameters of a drive or the controller unit of a Sinamics Profidrive (Profidrive Specification 3.1). Since Profidrives use the Motorola format and IPCs the Intel format, the function automatically swaps the byte order of parameters with data types with more than one byte in the DPV1 telegram.

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

FUNCTION F_SplitDpv1ReadResPkg : USINT

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 containing the DPV1 read response telegram. This telegram is evaluated by the 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
```

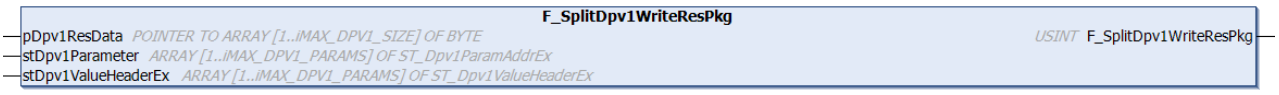
stDpv1Parameter: Array of 39 parameters that were added to the DPV1 read telegram (type: ST_Dpv1ParamAddrEx [▶ 127]).

stDpv1ValueHeaderEx: Array of 39 parameter values that were read from the drive (type: ST_Dpv1ValueHeaderEx [▶ 128]).

Requirements

Development environment	Target platform	IO hardware	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	PC or CX (x86)	Beckhoff FC310x PCI, CX1500-M310, EL6731	Tc2_IoFunctions (IO)

3.9.4 F_SplitDpv1WriteResPkg



The “F_SplitDpv1WriteResPkg” function creates a DPV1 telegram for an FB_Dpv1Write [▶ 76] of one or more parameters of a drive or the controller unit of a Sinamics Profidrive (Profidrive Specification 3.1). Since Profidrives use the Motorola format and IPCs the Intel format, the function automatically swaps the byte order of parameters with data types with more than one byte in the DPV1 telegram.

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

FUNCTION F_SplitDpv1WriteResPkg : USINT

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 containing the DPV1 write response telegram. This telegram is evaluated by the 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
```

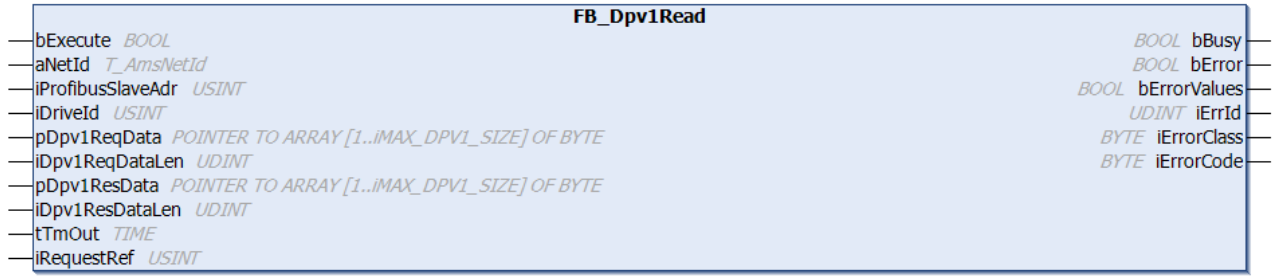
stDpv1Parameter: Array of 39 parameters that were added to the DPV1 write telegram (type: ST_Dpv1ParamAddrEx [▶ 127]).

stDpv1ValueHeaderEx: Array of 39 parameter values that were read from the drive (type: ST_Dpv1ValueHeaderEx [▶ 128]).

Requirements

Development environment	Target platform	IO hardware	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	PC or CX (x86)	Beckhoff FC310x PCI, CX1500-M310, EL6731	Tc2_IoFunctions (IO)

3.9.5 FB_Dpv1Read



The “FB_Dpv1Read” function block reads one or more parameters of a Sinamics Profidrive via DPV1 (Profidrive Specification 3.1). The DPV1 read telegram must be created with F_CreateDpv1ReadReqPkg [▶ 71] before a rising edge is present at bExecute. The DPV1 response telegram must be evaluated with F_SplitDpv1ReadResPkg [▶ 72] after a falling edge appears at bBusy.

The execution of this function block requires some time, depending on the number of parameters to be read. The function block sends the DPV1 telegram and polls for a response telegram.

VAR_INPUT

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

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

aNetId: The network address of the Profibus master device (see ADS tab of the Profibus master device in the I/O configuration in the TwinCAT system, type: T_AmsNetID).

iProfibusSlaveAdr: The Profibus slave DP address of the drive. This is an address for several axes, specified in the TwinCAT system in the I/O configuration.

iDriveID: The ID is 1 for the controller unit, 2 for the drive object A and 3 for the drive object B of a double/ triple drive. The drive ID is set in the starter software. 1 – 16 is possible.

pDpv1ReqData: Pointer to an array of 240 bytes containing the DPV1 read telegram. This telegram must be created by the F_CreateDpv1ReadReqPkg [▶ 71] function before the DPV1 read is activated via bExecute.

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

pDpv1ResData: Pointer to an array of 240 bytes containing the DPV1 read response telegram. This telegram must be evaluated by the F_SplitDpv1ReadResPkg [▶ 72] function after a falling edge appears at bBusy.

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

tTmOut: Maximum time allowed for the execution of the command.

VAR_OUTPUT

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

bBusy: This output goes TRUE as soon as the function block is activated via bExecute and remains TRUE as long as the block has not received a response.

bError: In case of errors this output goes TRUE and bBusy goes FALSE.

bErrorValues: TRUE if the DPV1 read was unsuccessful or only partly successful. The error causes are returned via the error ID (as well as class and code).

nErrId: Returns the ADS error number [▶ 153] or function-block-specific error numbers if bError = TRUE.

nErrClass: Profidrive error class.

nErrCode: Profidrive error code.

Function-block-specific error codes	Description
0x2	incorrect response reference
0x3	DPV1 read was erroneous or partly erroneous
0x4	incorrect response ID
other error IDs	see ADS error code

Error classes	Description	Error code
0x0 - 0x9	reserved	-
0xA	Application errors	0x0: read error 0x1: write error 0x2: module error 0x3 - 0x7: reserved 0x8: version conflict 0x9: not supported 0xA - 0xF: user-dependent
0xB	Access errors	0x0: invalid index (no data block DB47, parameter access not supported) 0x1: write length error 0x2: invalid slot 0x3: type conflict 0x4: invalid range 0x5: state conflict (access to DB47 temporarily not possible because of internal process states) 0x6: access denied 0x7: invalid range (write error in the DB47 header) 0x8: invalid parameter 0x9: invalid type 0xA - 0xF: user-dependent
0xC	Resource errors	0x0: read conflict 0x1: write conflict 0x2: resource busy 0x3: resource not attainable 0x4 - 0x7: reserved 0x8 - 0xF: user-dependent
0xD - 0xF	User-defined errors	-

VAR_IN_OUT

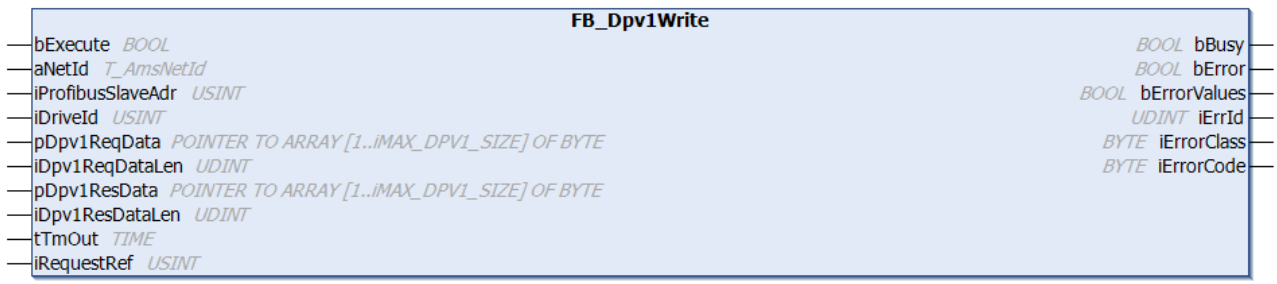
```
VAR_OUTPUT
    iRequestRef : USINT; (* 1..127; 0: reserved *)
END_VAR
```

iRefRequest: Reference that is automatically incremented with each telegram. The reference is required for the allocation of the responses to the write/read requests.

Requirements

Development environment	Target platform	IO hardware	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	PC or CX (x86)	Beckhoff FC310x PCI, CX1500-M310, EL6731	Tc2_IoFunctions (IO)

3.9.6 FB_Dpv1Write



The “FB_Dpv1Write” function block writes one or more parameters of a Sinamics Profidrive via DPV1 (Profidrive Specification 3.1). The DPV1 write telegram must be created with F_CreateDpv1WriteReqPkg [▶ 72] before a rising edge is present at bExecute. The DPV1 response telegram must be evaluated with F_SplitDpv1WriteResPkg [▶ 73] after a falling edge appears at bBusy.

The execution of this function block requires some time, depending on the number of parameters to be read. The function block sends the DPV1 telegram and polls for a response telegram.

VAR_INPUT

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

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

aNetId: The network address of the Profibus master device (see ADS tab of the Profibus master device in the I/O configuration in the TwinCAT system, type: T_AmsNetID).

iProfibusSlaveAdr: The Profibus slave DP address of the drive. This is an address for several axes, specified in the TwinCAT system in the I/O configuration.

iDriveID: The ID is 1 for the controller unit, 2 for the drive object A and 3 for the drive object B of a double/ triple drive. The drive ID is set in the starter software. 1 – 16 is possible.

pDpv1ReqData: Pointer to an array of 240 bytes containing the DPV1 write telegram. This telegram must be created by the F_CreateDpv1WriteReqPkg [▶ 72] function before the DPV1 write is activated via bExecute.

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

pDpv1ResData: Pointer to an array of 240 bytes containing the DPV1 write response telegram. This telegram must be evaluated by the F_SplitDpv1WriteResPkg [▶ 73] function after a falling edge appears at bBusy.

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

tTmOut: Maximum time allowed for the execution of the command.

VAR_OUTPUT

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

bBusy: This output goes TRUE as soon as the function block is activated via bExecute and remains TRUE as long as the block has not received a response.

bError: In case of errors this output goes TRUE and bBusy goes FALSE.

bErrorValues: TRUE if the DPV1 write was unsuccessful or only partly successful. The error causes are returned via the error ID (as well as class and code).

nErrId: Returns the ADS error number [▶ 153] or function-block-specific error numbers if bError = TRUE.

nErrClass: Profidrive error class.

nErrCode: Profidrive error code.

Function-block-specific error codes	Description
0x2	incorrect response reference
0x3	DPV1 write was erroneous or partly erroneous
0x4	incorrect response ID
other error IDs	see ADS error code

Error classes	Description	Error code
0x0 - 0x9	reserved	-
0xA	Application errors	0x0: read error 0x1: write error 0x2: module error 0x3 - 0x7: reserved 0x8: version conflict 0x9: not supported 0xA - 0xF: user-dependent
0xB	Access errors	0x0: invalid index (no data block DB47, parameter access not supported) 0x1: write length error 0x2: invalid slot 0x3: type conflict 0x4: invalid range 0x5: state conflict (access to DB47 temporarily not possible because of internal process states) 0x6: access denied 0x7: invalid range (write error in the DB47 header) 0x8: invalid parameter 0x9: invalid type 0xA - 0xF: user-dependent
0xC	Resource errors	0x0: read conflict 0x1: write conflict 0x2: resource busy 0x3: resource not attainable 0x4 - 0x7: reserved 0x8 - 0xF: user-dependent
0xD - 0xF	User-defined errors	-

VAR_IN_OUT

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

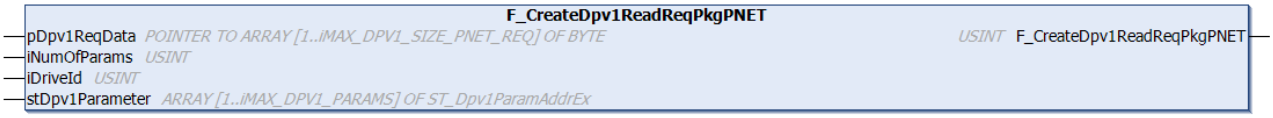
iRefRequest: Reference that is automatically incremented with each telegram. The reference is required for the allocation of the responses to the write/read requests.

Requirements

Development environment	Target platform	IO hardware	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	PC or CX (x86)	Beckhoff FC310x PCI, CX1500-M310, EL6731	Tc2_IoFunctions (IO)

3.10 Profinet DPV1 (Sinamics)

3.10.1 F_CreateDpv1ReadReqPkgPNET



The “F_CreateDpv1ReadReqPkg” function creates a DPV1 telegram for an FB_Dpv1ReadPNET [▶ 82] of one or more parameters of a drive or the controller unit of a Sinamics Profidrive (Profidrive Specification 3.1) that is connected via Profinet. Since Profidrives use the Motorola format and IPCs the Intel format, the function automatically swaps the byte order of parameters with data types with more than one byte in the DPV1 telegram.

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

FUNCTION F_CreateDpv1ReadReqPkgPNET : USINT

VAR_INPUT

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

pDpv1ReqData: Pointer to an array of 240 bytes containing the DPV1 read telegram. This telegram is created by the function.

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

iDriveID: The ID is 1 for the controller unit, 2 for the drive object A and 3 for the drive object B of a double/ triple drive. The drive ID is set in the starter software. 1 – 16 is possible.

VAR_IN_OUT

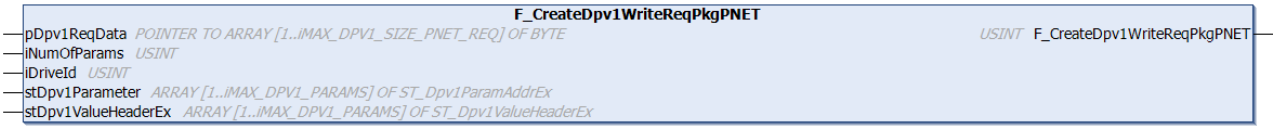
```
VAR_IN_OUT
    stDpv1Parameter : ARRAY [1..iMAX_DPV1_PARAMS] OF ST_Dpv1ParamAddrEx;(* list of parameters *)
END_VAR
```

stDpv1Parameter: Array of 39 parameters to be added to the DPV1 read telegram (type: ST_Dpv1ParamAddrEx [▶ 127]).

Requirements

Development environment	Target platform	IO hardware	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	PC or CX (x86)	Beckhoff EL6632	Tc2_IoFunctions (IO)

3.10.2 F_CreateDpv1WriteReqPkgPNET



The “F_CreateDpv1WriteReqPkgPNET” function creates a DPV1 telegram for an FB_Dpv1WritePNET [▶ 84] of one or more parameters of a drive or the controller unit of a Sinamics Profidrive (Profidrive Specification 3.1) that is connected via Profinet. Since Profidrives use the Motorola format and IPCs the Intel format, the function automatically swaps the byte order of parameters with data types with more than one byte in the DPV1 telegram.

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

FUNCTION F_CreateDpv1WriteReqPkgPNET : USINT

VAR_INPUT

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

pDpv1ReqData: Pointer to an array of 240 bytes containing the DPV1 write telegram. This telegram is created by the function.

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

iDriveID: The ID is 1 for the controller unit, 2 for the drive object A and 3 for the drive object B of a double/ triple drive. The 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
```

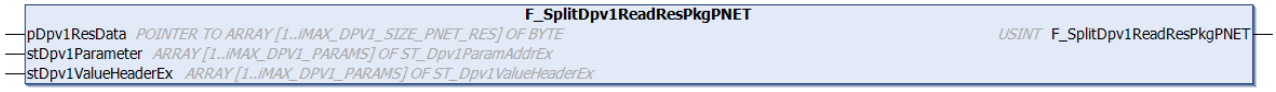
stDpv1Parameter: Array of 39 parameters to be added to the DPV1 write telegram (type: ST_Dpv1ParamAddrEx [▶ 127]).

stDpv1ValueHeaderEx: Array of 39 parameter values to be added to the DPV1 write telegram (type: ST_Dpv1ValueHeaderEx [▶ 128]).

Requirements

Development environment	Target platform	IO hardware	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	PC or CX (x86)	Beckhoff EL6632	Tc2_IoFunctions (IO)

3.10.3 F_SplitDpv1ReadResPkgPNET



The “F_SplitDpv1ReadResPkgPNET” function creates a DPV1 telegram for an FB_Dpv1ReadPNET [▶ 82] of one or more parameters of a drive or the controller unit of a Sinamics Profidrive (Profidrive Specification 3.1) that is connected via Profinet. Since Profidrives use the Motorola format and IPCs the Intel format, the function automatically swaps the byte order of parameters with data types with more than one byte in the DPV1 telegram.

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

FUNCTION F_SplitDpv1ReadResPkgPNET : USINT

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 containing the DPV1 read response telegram. This telegram is evaluated by the 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
```

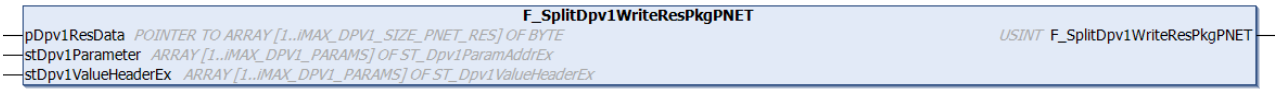
stDpv1Parameter: Array of 39 parameters that were added to the DPV1 read telegram (type: ST_Dpv1ParamAddrEx [▶ 127]).

stDpv1ValueHeaderEx: Array of 39 parameter values that were read from the drive (type: ST_Dpv1ValueHeaderEx [▶ 128]).

Requirements

Development environment	Target platform	IO hardware	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	PC or CX (x86)	Beckhoff EL6632	Tc2_IoFunctions (IO)

3.10.4 F_SplitDpv1WriteResPkgPNET



The “F_SplitDpv1WriteResPkgPNET” function creates a DPV1 telegram for an FB_Dpv1WritePNET [▶ 84] of one or more parameters of a drive or the controller unit of a Sinamics Profidrive (Profidrive Specification 3.1) that is connected via Profinet. Since Profidrives use the Motorola format and IPCs the Intel format, the function automatically swaps the byte order of parameters with data types with more than one byte in the DPV1 telegram.

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

FUNCTION F_SplitDpv1WriteResPkgPNET : USINT

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 containing the DPV1 write response telegram. This telegram is evaluated by the 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 that were added to the DPV1 write telegram (type: ST_Dpv1ParamAddrEx [▶ 127]).

stDpv1ValueHeaderEx: Array of 39 parameter values that were read from the drive (type: ST_Dpv1ValueHeaderEx [▶ 128]).

Requirements

Development environment	Target platform	IO hardware	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	PC or CX (x86)	Beckhoff EL6632	Tc2_IoFunctions (IO)

3.10.5 FB_Dpv1ReadPNET

The “FB_Dpv1ReadPNET” function block reads one or more parameters of a Sinamics Profidrive via DPV1 (Profidrive Specification 3.1) via Profinet. The DPV1 read telegram must be created with F_CreateDpv1ReadReqPkgPNET [▶ 79] before a rising edge is present at bExecute. The DPV1 response telegram must be evaluated with F_SplitDpv1ReadResPkgPNET [▶ 80] after a falling edge appears at bBusy.

The execution of this function block requires some time, depending on the number of parameters to be read. The function block sends the DPV1 telegram and polls for a response telegram.

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 block is activated by a positive edge at this input.

aNetId: The network address of the Profibus master device (see ADS tab of the Profibus master device in the I/O configuration in the TwinCAT system, type: T_AmsNetID).

iProfinetPort: The Profinet port number of the drive. This is an address for several axes, specified in the TwinCAT system in the I/O configuration.

iDriveID: The ID is 1 for the controller unit, 2 for the drive object A and 3 for the drive object B of a double/ triple drive. The drive ID is set in the starter software. 1 – 16 is possible.

pDpv1ReqData: Pointer to an array of 240 bytes containing the DPV1 read telegram. This telegram must be created by the F_CreateDpv1ReadReqPkg [▶ 71] function before the DPV1 read is activated via bExecute.

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

pDpv1ResData: Pointer to an array of 240 bytes containing the DPV1 read response telegram. This telegram must be evaluated by the F_SplitDpv1ReadResPkg [▶ 72] function after a falling edge appears at bBusy.

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

tTmOut: Maximum time allowed for the execution of the command.

VAR_OUTPUT

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

bBusy: This output goes TRUE as soon as the function block is activated via bExecute and remains TRUE as long as the block has not received a response.

bError: In case of errors this output goes TRUE and bBusy goes FALSE.

bErrorValues: TRUE if the DPV1 read was unsuccessful or only partly successful. The error causes are returned via the error ID (as well as class and code).

nErrId: Returns the ADS error number [▶ 153] or function-block-specific error numbers if bError = TRUE.

nErrClass: Profidrive error class.

nErrCode: Profidrive error code.

Function-block-specific error codes	Description
0x2	incorrect response reference
0x3	DPV1 read was erroneous or partly erroneous
0x4	incorrect response ID
other error IDs	see ADS error code

Error classes	Description	Error code
0x0 - 0x9	reserved	-
0xA	Application errors	0x0: read error 0x1: write error 0x2: module error 0x3 - 0x7: reserved 0x8: version conflict 0x9: not supported 0xA - 0xF: user-dependent
0xB	Access errors	0x0: invalid index (no data block DB47, parameter access not supported) 0x1: write length error 0x2: invalid slot 0x3: type conflict 0x4: invalid range 0x5: state conflict (access to DB47 temporarily not possible because of internal process states) 0x6: access denied 0x7: invalid range (write error in the DB47 header) 0x8: invalid parameter 0x9: invalid type 0xA - 0xF: user-dependent
0xC	Resource errors	0x0: read conflict 0x1: write conflict 0x2: resource busy 0x3: resource not attainable 0x4 - 0x7: reserved 0x8 - 0xF: user-dependent
0xD - 0xF	User-defined errors	-

VAR_IN_OUT

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

iRefRequest: Reference that is automatically incremented with each telegram. The reference is required for the allocation of the responses to the write/read requests.

Requirements

Development environment	Target platform	IO hardware	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	PC or CX (x86)	Beckhoff EL6632	Tc2_IoFunctions (IO)

3.10.6 FB_Dpv1WritePNET

The “FB_Dpv1WritePNET” function block writes one or more parameters of a Sinamics Profidrive via DPV1 (Profidrive Specification 3.1) via Profinet. The DPV1 write telegram must be created with F_CreateDpv1WriteReqPkgPNET [▶ 79] before a rising edge is present at bExecute. The DPV1 response telegram must be evaluated with F_SplitDpv1WriteResPkgPNET [▶ 81] after a falling edge appears at bBusy.

The execution of this function block requires some time, depending on the number of parameters to be read. The function block sends the DPV1 telegram and polls for a response telegram.

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 block is activated by a positive edge at this input.

aNetId: The network address of the Profibus master device (see ADS tab of the Profibus master device in the I/O configuration in the TwinCAT system, type: T_AmsNetID).

iProfinetPort: The Profinet port number of the drive. This is an address for several axes, specified in the TwinCAT system in the I/O configuration.

iDriveID: The ID is 1 for the controller unit, 2 for the drive object A and 3 for the drive object B of a double/ triple drive. The drive ID is set in the starter software. 1 – 16 is possible.

pDpv1ReqData: Pointer to an array of 240 bytes containing the DPV1 write telegram. This telegram must be created by the F_CreateDpv1WriteReqPkgPNET [▶ 79] function before the DPV1 write is activated via bExecute.

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

pDpv1ResData: Pointer to an array of 240 bytes containing the DPV1 write response telegram. This telegram must be evaluated by the F_SplitDpv1WriteResPkgPNET [▶ 81] function after a falling edge appears at bBusy.

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

tTmOut: Maximum time allowed for the execution of the command.

VAR_OUTPUT

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

bBusy: This output goes TRUE as soon as the function block is activated via bExecute and remains TRUE as long as the block has not received a response.

bError: In case of errors this output goes TRUE and bBusy goes FALSE.

bErrorValues: TRUE if the DPV1 write was unsuccessful or only partly successful. The error causes are returned via the error ID (as well as class and code).

nErrId: Returns the ADS error number [▶ 153] or function-block-specific error numbers if bError = TRUE.

nErrClass: Profidrive error class.

nErrCode: Profidrive error code.

Function-block-specific error codes	Description
0x2	incorrect response reference
0x3	DPV1 write was erroneous or partly erroneous
0x4	incorrect response ID
other error IDs	see ADS error code

Error classes	Description	Error code
0x0 - 0x9	reserved	-
0xA	Application errors	0x0: read error 0x1: write error 0x2: module error 0x3 - 0x7: reserved 0x8: version conflict 0x9: not supported 0xA - 0xF: user-dependent
0xB	Access errors	0x0: invalid index (no data block DB47, parameter access not supported) 0x1: write length error 0x2: invalid slot 0x3: type conflict 0x4: invalid range 0x5: state conflict (access to DB47 temporarily not possible because of internal process states) 0x6: access denied 0x7: invalid range (write error in the DB47 header) 0x8: invalid parameter 0x9: invalid type 0xA - 0xF: user-dependent
0xC	Resource errors	0x0: read conflict 0x1: write conflict 0x2: resource busy 0x3: resource not attainable 0x4 - 0x7: reserved 0x8 - 0xF: user-dependent
0xD - 0xF	User-defined errors	-

VAR_IN_OUT

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

iRefRequest: Reference that is automatically incremented with each telegram. The reference is required for the allocation of the responses to the write/read requests.

Requirements

Development environment	Target platform	IO hardware	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	PC or CX (x86)	Beckhoff EL6632	Tc2_IoFunctions (IO)

3.11 RAID Controller

3.11.1 FB_RAIDFindCntlr

This function returns the counter value of the RAID controller and the corresponding controller IDs.

NOTE

The function block should be called once only in a PLC program!

The system performance can be dramatically affected by the cyclic calling of this function block.

VAR_INPUT

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

sNETID: A string containing the AMS network ID of the target device to which the ADS command is sent (type: T_AmsNetID).

bWrtRd: The ADS command is triggered by the rising edge of this input.

tTimeOut: States the length of the timeout that may not be exceeded during execution of the command.

VAR_OUTPUT

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

stRAIDCntlrFound: Contains the number of RAID controllers found and their RAID controller IDs (type: ST_RAIDCntlrFound [▶ 134]).

nBytesRead: Number of successfully returned data bytes.

bBusy: This output remains TRUE until the block has executed a command, but no longer than the time present at the 'Timeout' input. No new command will be accepted at the inputs as long as bBusy = TRUE. Please note that it is not the execution of the service, but the time during which it be may executed that is monitored here.

bError: This output is set to TRUE if an error occurs during the execution of a command. The command-specific error code is located in 'nErrorId'. If the block has an timeout error, 'bError' is set to TRUE and 'nErrorId' is 1861 (hexadecimal 0x745). It is set to FALSE if a command is executed at the inputs.

nErrorID: Contains the command-specific error code of the commands executed last; it is reset to 0 by a command at the inputs.

Requirements

Development environment	Target platform	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	PC or CX (x86, x64, ARM)	Tc2_IoFunctions (IO)

3.11.2 FB_RAIDGetInfo

This function block returns a RAID info. containing the number of RAID controller sets and the maximum number of drives per set.

NOTE

The function block should be called once only in a PLC program!

The system performance can be dramatically affected by the cyclic calling of this function block.

VAR_INPUT

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

sNETID: A string containing the AMS network ID of the target device to which the ADS command is sent (type: T_AmsNetID).

bWrtRd: The ADS command is triggered by the rising edge of this input.

nRAIDCntlrID: The RAID controller ID. (Note: can be read with FB_RAIDCntlrFind)

tTimeOut: States the length of the timeout that may not be exceeded during execution of the command.

VAR_OUTPUT

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

stRAIDInfo: Returns a RAID info. containing the number of RAID controller sets and the maximum number of drives per set (type: ST_RAIDInfo [▶ 135]).

nBytesRead: Number of successfully returned data bytes.

bBusy: This output remains TRUE until the block has executed a command, but no longer than the time present at the ‘Timeout’ input. No new command will be accepted at the inputs as long as bBusy = TRUE. Please note that it is not the execution of the service, but the time during which it be may executed that is monitored here.

bError: This output is set to TRUE if an error occurs during the execution of a command. The command-specific error code is located in ‘nErrorId’. If the block has an timeout error, ‘bError’ is set to TRUE and ‘nErrorId’ is 1861 (hexadecimal 0x745). It is set to FALSE if a command is executed at the inputs.

nErrorID: Contains the command-specific error code of the commands executed last; it is reset to 0 by a command at the inputs.

Requirements

Development environment	Target platform	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	PC or CX (x86, x64, ARM)	Tc2_IoFunctions (IO)

3.11.3 FB_RAIDGetStatus

This function block returns the RAID set index, the RAID type, the RAID status, the number of RAID drives and the status of the RAID drives.

NOTE

Call once per second at the most!

The system performance can be dramatically affected by the cyclic calling of this function block.

VAR_INPUT

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

sNETID: A string containing the AMS network ID of the target device to which the ADS command is sent (type: T_AmsNetID).

bWrtRd: The command is triggered by the rising edge of this input.

stRAIDConfigReq: RAID configuration request parameters are defined in this structure (type: ST_RAIDConfigReq [▶ 134]). It contains the controller ID and the index of the RAID set.

tTimeOut: States the length of the timeout that may not be exceeded during execution of the command.

VAR_OUTPUT

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

stRAIDStatusRes: This structure contains the RAID status response, the RAID set index, the RAID type, the RAID status, the number of RAID drives and the status of the RAID drives (type: ST_RAIDStatusRes [▶ 135]).

nBytesRead: Number of successfully returned data bytes.

bBusy: This output remains TRUE until the block has executed a command, but no longer than the time present at the ‘Timeout’ input. No new command will be accepted at the inputs as long as bBusy = TRUE. Please note that it is not the execution of the service, but the time during which it be may executed that is monitored here.

bError: This output is set to TRUE if an error occurs during the execution of a command. The command-specific error code is located in ‘nErrorId’. If the block has an timeout error, ‘bError’ is set to TRUE and ‘nErrorId’ is 1861 (hexadecimal 0x745). It is set to FALSE if a command is executed at the inputs.

nErrorID: Contains the command-specific error code of the commands executed last; it is reset to 0 by a command at the inputs.

Requirements

Development environment	Target platform	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	PC or CX (x86, x64, ARM)	Tc2_IoFunctions (IO)

3.12 SERCOS

3.12.1 IOF_SER_GetPhase

The “IOF_SER_GetPhase” function block determines the current communication phase on the SERCOS ring. The communication phase can have a value from 0 to 4.

VAR_INPUT

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

NETID: The network address of the TwinCAT computer on which the ADS command is to be executed can be entered here (type: T_AmsNetId). 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 during the hardware configuration.

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

TMOUT: States the length of the timeout that may not be exceeded during execution of the 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 a feedback is received.

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

ERRID: Supplies the ADS error number [▶ 153] when the ERR output is set.

PHASE: The current communication phase in the SERCOS ring.

Requirements

Development environment	Target platform	IO hardware	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	PC (x86)	Sercans SCS-P ISA; Sercans SCS-P PCI; Beckhoff FC750x PCI	Tc2_IoFunctions (IO)

3.12.2 IOF_SER_SaveFlash

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.

NOTE

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 := DEFAULT_ADS_TIMEOUT;
END_VAR
```

NETID: The network address of the TwinCAT computer on which the ADS command is to be executed can be entered here (type: T_AmsNetId). 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 during the hardware configuration.

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

TMOUT: States the length of the timeout that may not be exceeded during execution of the 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 a feedback is received.

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

ERRID: Supplies the ADS error number [▶ 153] when the ERR output is set.

Requirements

Development environment	Target platform	IO hardware	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	PC (x86)	Sercans SCS-P ISA; Sercans SCS-P PCI; Beckhoff FC750x PCI	Tc2_IoFunctions (IO)

3.12.3 IOF_SER_ResetErr

The “IOF_SER_ResetErr” function block clears the following errors of a SERCOS master:

- The errors in the existing drives;
- The diagnostic status in the diagnostics channel of the existing drives;
- The system error;

VAR_INPUT

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

NETID: The network address of the TwinCAT computer on which the ADS command is to be executed can be entered here (type: T_AmsNetId). 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 during the hardware configuration.

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

TMOUT: States the length of the timeout that may not be exceeded during execution of the 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 a feedback is received.

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

ERRID: Supplies the ADS error number [▶ 153] when the ERR output is set.

Requirements

Development environment	Target platform	IO hardware	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	PC (x86)	Sercans SCS-P ISA; Sercans SCS-P PCI; Beckhoff FC750x PCI	Tc2_IoFunctions (IO)

3.12.4 IOF_SER_SetPhase

The “IOF_SER_SetPhase” function block carries out the phase boot-up in the SERCOS ring to a certain value.

VAR_INPUT

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

NETID: The network address of the TwinCAT computer on which the ADS command is to be executed can be entered here (type: T_AmsNetId). 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 during the hardware configuration.

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

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

TMOUT: States the length of the timeout that may not be exceeded during execution of the 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 a feedback is received.

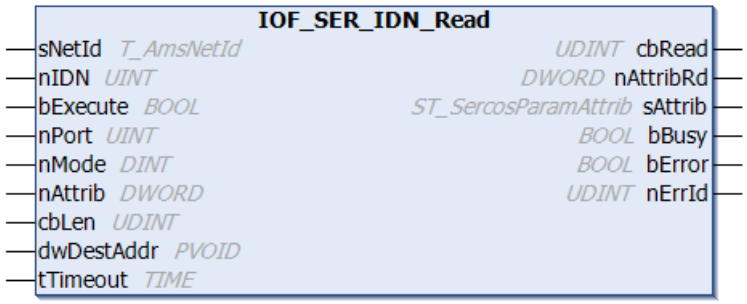
ERR: If an error should occur during the transfer of the command, then this output is set once the BUSY output was reset.

ERRID: Supplies the ADS error number [▶ 153] when the ERR output is set.

Requirements

Development environment	Target platform	IO hardware	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	PC (x86)	Sercans SCS-P ISA; Sercans SCS-P PCI; Beckhoff FC750x PCI	Tc2_IoFunctions (IO)

3.12.5 IOF_SER_IDN_Read



The “IOF_SER_IDN_Read” function block allows a value to be read from an S- or P-parameter of a Sercos drive. Data type and size are determined automatically on the basis of the attribute.

VAR_INPUT

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

sNetId: The network address of the TwinCAT computer on which the ADS command is to be executed can be entered here (type: T_AmsNetId). If it is to be run on the local computer, an empty string can be entered.

nIDN: contains the Sercos parameter number to be accessed for reading. nIDN must lie between 0 and 32767 for S-parameters and between 32768 and 65535 for P-parameters.

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

nPort: The port number nPort is assigned by the TwinCAT system during the hardware configuration.

nMode: The read mode determines which part of the parameter is to be read.
nMode = 0: Value
nMode = 2: Name
nMode = 3: Attribute (always read in order to determine data type and size, unless nAttrib is <> 0)
nMode = 4: Unit (not available for every parameter)
nMode = 5: Minimum (not available for every parameter)
nMode = 6: Maximum (not available for every parameter)

nAttrib: Attribute of the parameter, if known. If nAttrib = 0, IOF_SER_IDN_Write first reads the parameter attribute of the drive before the value is written to the parameter of the drive.

cbLen: Maximum length of the data buffer that is to accept the value.

dwDestAddr: Address of the data buffer that is to accept the value.

tTimeout: States the length of the timeout that may not be exceeded during execution of the 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: Attribute of the parameter; can be saved for the next access (nAttrib) to the parameter.
sAttrib: contains the attribute nAttribRd of the Sercos parameter, broken down into individual variables (type: ST_SercosParamAttrib [▶ 136]).
bBusy: When the function block is activated this output is set. It remains set until a feedback is received.

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

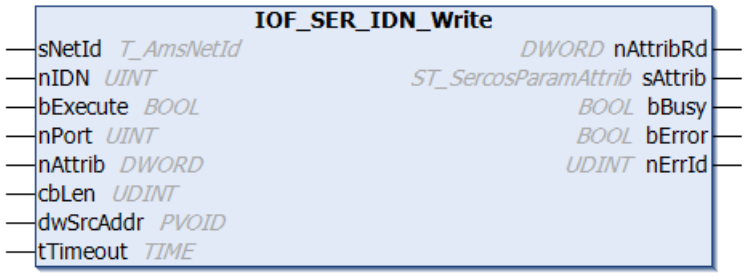
nErrId: Returns the ADS error number [▶ 153] or the specific function block error number if the ERR output is set.

specific function block error number	Description
0x1003	Incorrect parameter mode
0x1004	Incorrect parameter data size

Requirements

Development environment	Target platform	IO hardware	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	PC (x86)	Sercans SCS-P ISA; Sercans SCS-P PCI; Beckhoff FC750x PCI	Tc2_IoFunctions (IO)

3.12.6 IOF_SER_IDN_Write



The “IOF_SER_IDN_Write” function block allows a value to be written to an S- or P-parameter of a Sercos drive. Data type and size are determined automatically on the basis of the attribute.

VAR_INPUT

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

sNetId: The network address of the TwinCAT computer on which the ADS command is to be executed can be entered here (type: T_AmsNetId). If it is to be run on the local computer, an empty string can be entered.

nIDN: contains the Sercos parameter number to be accessed for writing. nIDN must lie between 0 and 32767 for S-parameters and between 32768 and 65535 for P-parameters.

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

nPort: The port number nPort is assigned by the TwinCAT system during the hardware configuration.

nAttrib: Attribute of the parameter, if known. If nAttrib = 0, IOF_SER_IDN_Write first reads the parameter attribute of the drive before the value is written to the parameter of the drive.

cbLen: Length of the data buffer containing the value.

dwSrcAddr: Address of the data buffer containing the value.

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

VAR_OUTPUT

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

nAttribRd: Attribute of the parameter; can be saved for the next access (nAttrib) to the parameter.

sAttrib: contains the attribute nAttribRd of the Sercos parameter, broken down into individual variables (type: ST_SercosParamAttrib [▶ 136]).

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

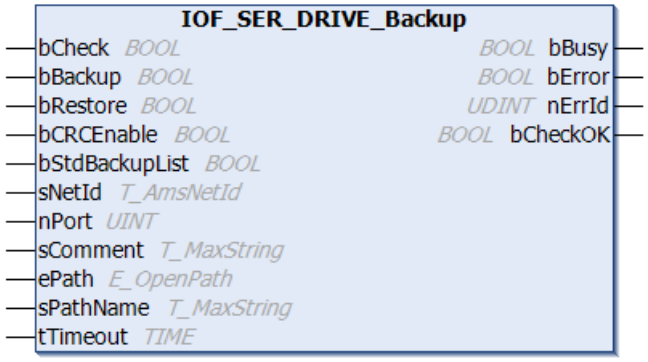
bError: If an error should occur during the transfer of the command, then this output is set once the bBusy output was reset.

nErrId: Supplies the ADS error number [▶ 153] when the ERR output is set.

Requirements

Development environment	Target platform	IO hardware	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	PC (x86)	Sercans SCS-P ISA; Sercans SCS-P PCI; Beckhoff FC750x PCI	Tc2_IoFunctions (IO)

3.12.7 IOF_SER_DRIVE_Backup



The “IOF_SER_DRIVE_Backup” function block allows the backup and restore of the drive data (S- and P-parameters) of the PLC in a binary file. The list of S- and P-parameter data to be backed up is taken by default from the Sercos parameter IDN192. Backup and restore require the SERCOS parameter mode (phase 2).

If bStdBackupList = TRUE (standard), the IDN192 parameter is taken as the list of the data to be backed up; otherwise IDN17, the list of all Sercos parameters, is taken. Restore requires a backup file that was created with the IDN192 parameter, since some parameters in the IDN17 list are write protected.

Backup and restore create a CRC16-CCITT and a 16-bit checksum and save them in the IDN142 parameter, if available.

The file format of the backup file is described in the backup file format [▶ 146].

VAR_INPUT

```
VAR_INPUT
    bCheck        : BOOL;
    bBackup       : BOOL;
    bRestore      : BOOL;
    bCRCEnable    : BOOL := TRUE;
    bStdBackupList: BOOL := TRUE;
    sNetId        : T_AmsNetId;
    nPort         : UINT;
    sComment      : T_MaxString;
    ePath         : E_OpenPath := PATH_BOOTPATH;
    sPathName     : T_MaxString := 'DRIVEPAR.BIN';
    tTimeout      : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

bCheck: The check by CRC and checksum is activated by a rising edge at this input. CRC and checksum are saved persistently and in the IDN142 parameter after a backup or a restore. bCheckOK is set to TRUE if the value from the IDN142 parameter matches the persistent data, otherwise bCheckOK is set to FALSE.

bBackup: the backup is activated by a rising edge at this input.

bRestore: the restore is activated by a rising edge at this input.

bCRCEnable: The CRC16-CCITT and the 16-bit checksum are activated via bCRCEnable = TRUE. The CRC and the checksum are saved in the IDN142 parameter if bCRCEnable = TRUE.

bStdBackupList: determines which parameter list is used for the backup. By default IDN192 (bStdBackupList = TRUE) is used for the backup. If bStdBackupList = FALSE, the list of all parameters, IDN017, is used. Restore requires a backup file created with the IDN192 list.

sNetId: The network address of the TwinCAT computer on which the ADS command is to be executed can be entered here (type: T_AmsNetId). If it is to be run on the local computer, an empty string can be entered.

nPort: The port number nPort is assigned by the TwinCAT system during the hardware configuration.

sComment: A comment that is written in the file header of the backup file (type: T_MaxString).

ePath: Determines the path of the backup file. If ePath = PATH_BOOTPATH, the TwinCAT boot path is taken. If ePath = PATH_GENERIC, the path specified in sPathName is taken (type: E_OpenPath).

sPathName: Contains the file name (if using the boot path) or the complete path and file name if using the generic path (type: T_MaxString).

tTimeout: States the length of the timeout that may not be exceeded during execution of the 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 a feedback is received.

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

nErrId: Returns the ADS error number [▶ 153] or the specific function block error number if the ERR output is set.

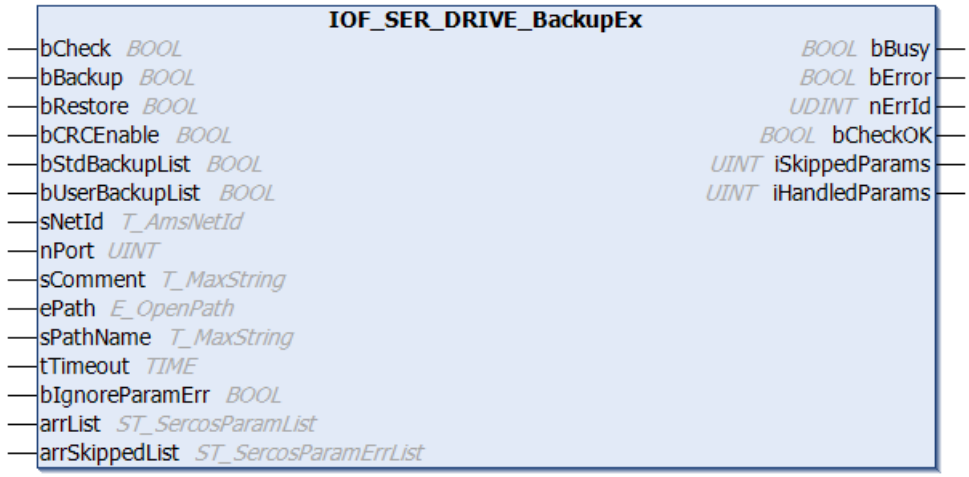
specific function block error number	Description
0x1003	Incorrect parameter mode
0x1004	Incorrect parameter data size
0x1005	Incorrect backup parameter type
0x1006	Backup parameter list was not IDN 192

bCheckOk: TRUE if the checksum test was successful.

Requirements

Development environment	Target platform	IO hardware	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	PC (x86)	Sercans SCS-P ISA; Sercans SCS-P PCI; Beckhoff FC750x PCI	Tc2_IoFunctions (IO)

3.12.8 IOF_SER_DRIVE_BackupEx



The “IOF_SER_DRIVE_BackupEx” function block permits the backup and restore of the drive data (S- and P-parameters) via the PLC to a binary file or back to the drive. The list of S- and P-parameter data to be backed up is taken by default from the Sercos parameter IDN192. Backup and restore require the SERCOS parameter mode (phase 2).

If bStdBackupList = TRUE (default), the IDN192 parameter is taken as the list of data to be backed up.
If bUserBackupList = TRUE, the arrList parameter list is taken as the list of data to be backed up.
Otherwise the list of all Sercos parameters, IDN17, is used.

Restore requires a backup file that was created with the IDN192 parameter or with a user parameter list. Some parameters in the IDN17 list are write protected.

Backup and restore can create a CRC16-CCITT and a 16-bit checksum and save them in the IDN142 parameter, if available. The bCRCEnable option is deactivated (FALSE) by default.

The file format of the backup file is described in the backup file format [▶ 146].

VAR_INPUT

```
VAR_INPUT
    bCheck          : BOOL;
    bBackup         : BOOL;
    bRestore        : BOOL;
    bCRCEnable      : BOOL;
    bStdBackupList  : BOOL     := TRUE;
    bUserBackupList : BOOL;
    sNetId          : T_AmsNetId;
    nPort           : UINT;
    sComment        : T_MaxString;
    ePath           : E_OpenPath := PATH_BOOTPATH;
    sPathName       : T_MaxString := 'DRIVEPAR.BIN';
    tTimeout        : TIME := DEFAULT_ADS_TIMEOUT;
    bIgnoreParamErr : BOOL;
END_VAR
```

bCheck: The check by CRC and checksum is activated by a rising edge at this input. CRC and checksum are saved persistently and in the IDN142 parameter after a backup or a restore. bCheckOK is set to TRUE if the value from the IDN142 parameter matches the persistent data, otherwise bCheckOK is set to FALSE.

bBackup: the backup is activated by a rising edge at this input.

bRestore: the restore is activated by a rising edge at this input.

bCRCEnable: The CRC16-CCITT and the 16-bit checksum are activated via bCRCEnable = TRUE. The CRC and the checksum are saved in the IDN142 parameter if bCRCEnable = TRUE.

bStdBackupList: determines which parameter list is used for the backup. By default IDN192 (bStdBackupList = TRUE) is used for the backup. If bStdBackupList = FALSE, the list of all parameters, IDN017, is used. Restore requires a backup file created with the IDN192 list.

bUserBackupList: Determines whether a user-defined parameter list, arrList, is used for the backup. By default IDN192 (bStdBackupList = TRUE) is used for the backup. arrList is used if bStdBackupList = FALSE and bUserBackupList = TRUE. Restore requires a backup file created with the IDN192 list or a user-defined parameter list.

sNetId: The network address of the TwinCAT computer on which the ADS command is to be executed can be entered here (type: T_AmsNetId). If it is to be run on the local computer, an empty string can be entered.

nIDN: contains the Sercos parameter number to be accessed for writing. nIDN must lie between 0 and 32767 for S-parameters and between 32768 and 65535 for P-parameters.

nPort: The port number nPort is assigned by the TwinCAT system during the hardware configuration.

sComment: A comment that is written in the file header of the backup file (type: T_MaxString).

ePath: Determines the path of the backup file (type: E_OpenPath). If ePath = PATH_BOOTPATH, the TwinCAT boot path is taken. If ePath = PATH_GENERIC, the path specified in sPathName is taken.

sPathName: Contains the file name (if using the boot path) or the complete path and file name if using the generic path (type: T_MaxString).

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

bIgnoreParamErr: Determines whether the backup/restore is to be continued or cancelled in case of parameter reading or writing errors. By default the process is cancelled in case of error (bIgnoreParamErr = FALSE). If ignoring of errors is activated (bIgnoreParamErr = TRUE), the parameter number and the error numbers are saved in the list of skipped parameters, arrSkippedList.

VAR_IN_OUT

```
VAR_IN_OUT
    arrList        : ST_SercosParamList;
    arrSkippedList : ST_SercosParamErrList;
END_VAR
```

arrList: In case of a standard backup via IDN192 (bStdBackupList = TRUE), this list contains the backup parameters from IDN192 after the backup. In case of a user-defined backup (bUserBackupList = TRUE and bStdBackupList = FALSE), this list must contain the list of parameters to be backed up before the backup. If the backup takes place by means of IDN 17 (bStdBackupList = FALSE and bStdBackupList = FALSE), all the parameters listed in IDN 17 will be backed up in arrList. (Typ: ST_SercosParamList [▶ 137]).

arrSkippedList: Contains a list of the skipped parameters (the parameter number and the error numbers, type: ST_SercosParamErrList [▶ 136]).

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 a feedback is received.

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

nErrId: Returns the ADS error number [▶ 153] or the specific function block error number if the ERR output is set.

specific function block error number	Description
0x1003	Incorrect parameter mode
0x1004	Incorrect parameter data size
0x1005	Incorrect backup parameter type
0x1006	Backup parameter list was not IDN 192 or user-defined

bCheckOk: TRUE if the checksum test was successful.

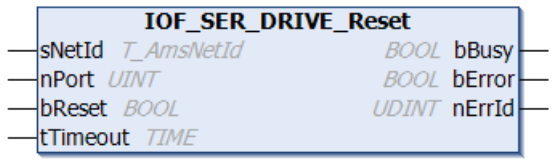
iSkippedParams: Contains the number of skipped parameters (see arrSkippedList) if ignoring of parameter errors was active (bIgnoreParamErr = TRUE).

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

Requirements

Development environment	Target platform	IO hardware	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	PC (x86)	Sercans SCS-P ISA; Sercans SCS-P PCI; Beckhoff FC750x PCI	Tc2_IoFunctions (IO)

3.12.9 IOF_SER_DRIVE_Reset



The “IOF_SER_DRIVE_Reset” function block resets a Sercos drive. Drive errors are cleared.

VAR_INPUT

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

sNetId: The network address of the TwinCAT computer on which the ADS command is to be executed can be entered here (type: T_AmsNetID). If it is to be run on the local computer, an empty string can be entered.

nPort: The port number nPort is assigned by the TwinCAT system during the hardware configuration.

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

tTimeout: States the length of the timeout that may not be exceeded during execution of the 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 a feedback is received.

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

nErrId: Supplies the ADS error number [▶ 153] when the ERR output is set.

Requirements

Development environment	Target platform	IO hardware	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	PC (x86)	Sercans SCS-P ISA; Sercans SCS-P PCI; Beckhoff FC750x PCI	Tc2_IoFunctions (IO)

3.13 TcTouchLock

TcTouchLock is a function of Beckhoff IPC Panel.

For more information, see the documentationTouch_Lock.

The hardware requirements can also be found there, in the chapter Requirements.

3.13.1 FB_TcTouchLock_AcquireFocus

The function block FB_TcTouchLock_AcquireFocus is used to avoid simultaneous inputs via several multi-touch Control Panels connected to an IPC, which would be disruptive. To this end, the focus is placed on one of the connected Control Panels, while inputs from all other connected Control Panels are blocked. The function block FB_TcTouchLock_AcquireFocus can be used to request and release the focus.

If the focus is requested from a multi-touch Control Panel when another multi-touch Control Panel currently has the focus, the focus must first be released by this Panel. Once the release has taken place, the focus is automatically set to the device that is waiting for it.

The multi-touch Control Panels that are accessed by the function block must first be configured using the command line application TcTouchLockService.exe. A unique identification number must be assigned to each device.

VAR_INPUT

```
VAR_INPUT
    bEnable     : BOOL;
    sSetID      : STRING(32);
    tLEDTime    : TIME := 200;
END_VAR
```

bEnable: TRUE = request focus, FALSE = release focus

sSetID: ID of the device

tLEDTime: The output LED flashes at the specified interval (100 ms – 1 s) while the focus is requested

VAR_OUTPUT

```
VAR_OUTPUT
    bAcquired  : BOOL := FALSE;(* Focus status information *)
    bLED       : BOOL := FALSE;(* LED control output *)
    bBusy      : BOOL;(* TRUE => function in progress *)
    bError:    : BOOL;(* Error flag *)
    nErrID     : UDINT;(* Error code *)
END_VAR
```

bAcquired: TRUE if the client has the focus; FALSE if the client loses the focus.

bLED: This output has the following meaning, depending on the mode:

Mode	Meaning
Constant TRUE	The Panel has the focus
Constant FALSE	The Panel does not have the focus
Toggeles	The Panel is waiting for the focus

bBusy: TRUE, as long as the function block is active.

bError: TRUE if an ADS error occurs during transmission of the command. The bBusy output is reset beforehand.

nErrId: Supplies the ADS error number [▶ 153] or the command-specific error code (table) when the *bError* output is set.

Error Codes	Error description
0x0000	No error
0x0006	Target port not found

Sample: Control of touch focus via special key

Manual setting of the focus can be controlled via a special key on the Panel, for example. Since it must be possible to request the focus via the touchscreen when the input is locked, an input option outside the lockable touchscreen must be provided. The special key is linked to the corresponding input variable in the PLC program through the TwinCAT System Manager. One FB_TcTouchLock_AquireFocus instance is created per Panel and configured with the Panel ID. After pressing the special key on a Panel, the function block R_TRIG detects the rising edge, and the PLC program tries to set the touch focus via the corresponding FB_TcTouchLock_AquireFocus instance. The function block can also control an output (e.g. an LED), which signals whether the touch focus has been set successfully or whether an attempt is still being made to obtain the focus. Pressing the special key again resets the touch focus, allowing the touch focus to be set to another Panel.

For two Panels the PLC program looks like this:

```
PROGRAM MAIN
VAR
    button1 AT%IX0.0 : BOOL;
    button2 AT%IX0.1 : BOOL;

    led1 AT%QX0.0 : BOOL;
    led2 AT%QX0.1 : BOOL;

    fbPanel1 : FB_TcTouchLock_AcquireFocus := ( sSetID := 'A' );
    fbPanel2 : FB_TcTouchLock_AcquireFocus := ( sSetID := 'B' );

    trigger1 : R_TRIG;
    trigger2 : R_TRIG;
END_VAR
```

```
(* Panel 1 *)
trigger1( CLK := button1 );
IF trigger1.Q THEN
fbPanel1.bEnable := NOT fbPanel1.bEnable;
END_IF
fbPanel1(bLED=>LED1);

(* Panel 2 *)
trigger2( CLK := button2 );
IF trigger2.Q THEN
fbPanel2.bEnable := NOT fbPanel2.bEnable;
END_IF
fbPanel2(bLED=>LED2 );
```

Requirements

Development environment	Target platform	PLC libraries to be integrated (category group)
TwinCAT v3.1. >= 4022.31	PC or CX (x86, x64)	Tc2_IoFunctions (IO)

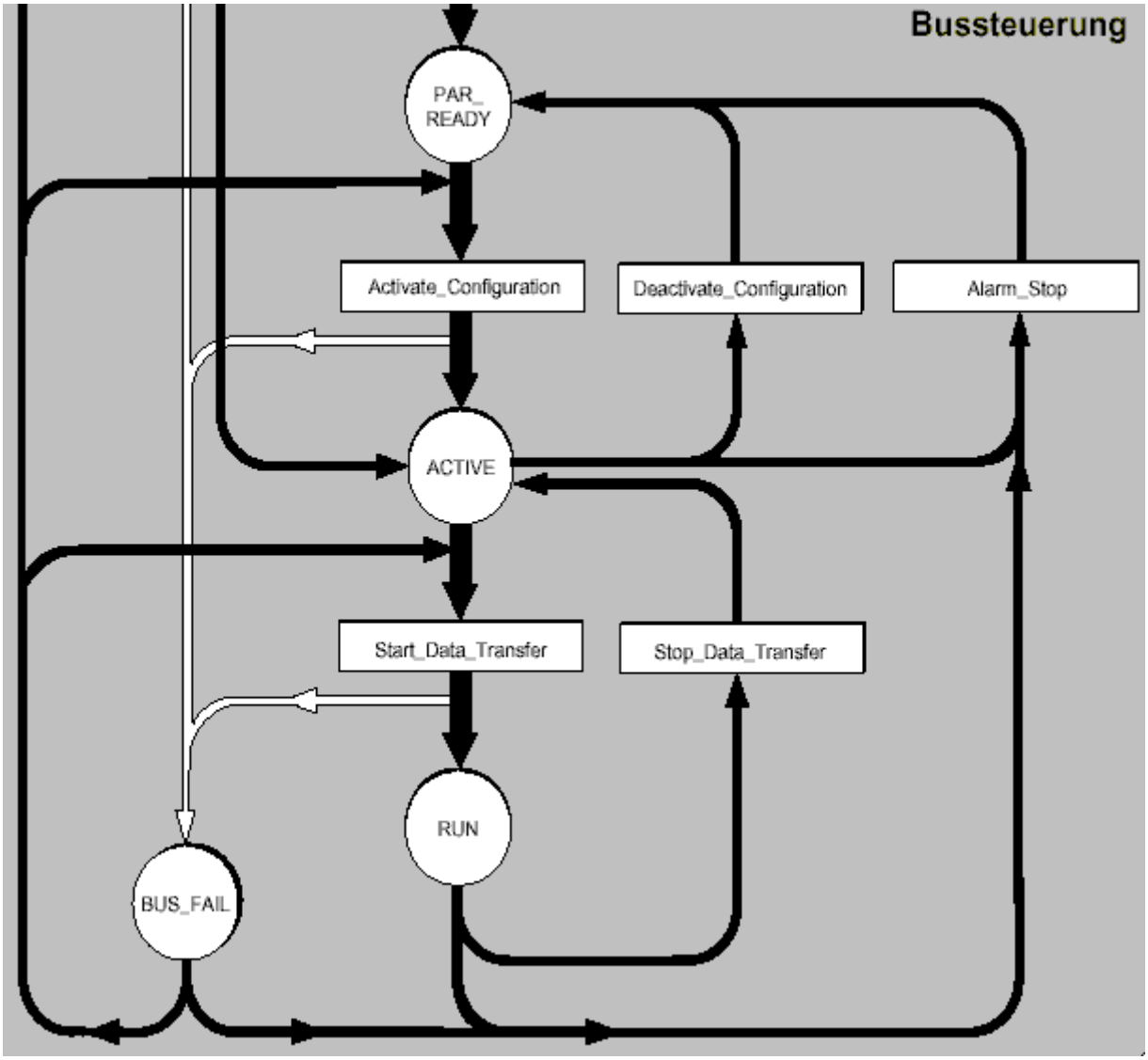
3.14 Third party devices

3.14.1 Phoenix IBS SC/I-T

3.14.1.1 Overview

The library offers a convenient possibility to execute the most important firmware services of the Phoenix IBS SC/I-T interbus 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 [▶ 105]: Executes the **Activate_Configuration** command.

SCIT_DeactivateConfiguration [▶ 106]: Executes the **Deactivate_Configuration** command.

SCIT_StartDataTransfer [▶ 107]: Executes the **Start_Data_Transfer** command.

SCIT_StopDataTransfer [▶ 108]: Executes the **Stop_Data_Transfer** command.

SCIT_AlarmStop [▶ 109]: Executes the **Alarm_Stop** command.

Configuration

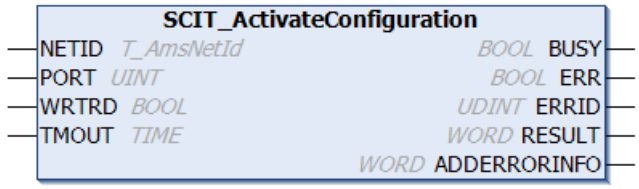
SCIT_ControlActiveConfiguration [▶ 110]: Is used to affect the active configuration of the Interbus devices. This command can be executed in the *PAR_READY* state as well as when in the *ACTIVE* or *RUN* states. Single, dependent and grouped devices can be activated and deactivated in this way.

Error diagnosis

SCIT_GetErrorInfo [▶ 111]: Returns the error type and error location of an Interbus device after a bus error.

SCIT_ConfDevErrAll [▶ 112]: Acknowledge periphery errors of all devices.

3.14.1.2 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. An **Activate_Configuration** sets the card in the ACTIVE [▶ 103] state.

VAR_INPUT

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

NETID: The network address of the computer in which the card is installed can be entered here (type: T_AmsNetID). If the card is located within the same system, it is also possible to give an empty string.

PORT: Contains the ADS port number of the card that was assigned by the TwinCAT system (type: T_AmsPort).

WRTRD: The function block is activated by a positive edge at this value.

TMOUT: Maximum time allowed for the execution of the command.

VAR_OUTPUT

```
VAR_INPUT
    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 error occurs while the command is being transferred, the ERR output is set once the BUSY signal is cancelled.

ERRID: Returns the ADS error number [▶ 153] in the case of error.

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 platform	IO Hardware	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	None. This functionality is not supported by TwinCAT 3 at present!	Phoenix: IBS SC/I-T; IBS SC/RI/RT-LK	Tc2_IoFunctions (IO)

3.14.1.3 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. An **Deactivate_Configuration** places the card in the PAR_READY [▶ 103] state and resets all the outputs.

VAR_INPUT

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

NETID: The network address of the computer in which the card is installed can be entered here (type: T_AmsNetID). If the card is located within the same system, it is also possible to give an empty string.

PORT: Contains the ADS port number of the card that was assigned by the TwinCAT system (type: T_AmsPort).

WRTRD: The function block is activated by a positive edge at this value.

TMOUT: Maximum time allowed for the execution of the command.

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 error occurs while the command is being transferred, the ERR output is set once the BUSY signal is cancelled.

ERRID: Returns the ADS error number [▶ 153] in the case of error.

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 platform	IO Hardware	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	None. This functionality is not supported by TwinCAT 3 at present!	Phoenix: IBS SC/I-T; IBS SC/RI/RT-LK	Tc2_IoFunctions (IO)

3.14.1.4 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. A **Start_Data_Transfer** places the card into the RUN [▶ 103] state.

VAR_INPUT

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

NETID: The network address of the computer in which the card is installed can be entered here (type: T_AmsNetID). If the card is located within the same system, it is also possible to give an empty string.

PORT: Contains the ADS port number of the card that was assigned by the TwinCAT system (type: T_AmsPort).

WRTRD: The function block is activated by a positive edge at this value.

TMOUT: Maximum time allowed for the execution of the command.

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 error occurs while the command is being transferred, the ERR output is set once the BUSY signal is cancelled.

ERRID: Returns the ADS error number [▶ 153] in the case of error.

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 platform	IO Hardware	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	None. This functionality is not supported by TwinCAT 3 at present!	Phoenix: IBS SC/I-T; IBS SC/RI/RT-LK	Tc2_IoFunctions (IO)

3.14.1.5 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. A **Stop_Data_Transfer** places the card into the ACTIVE [▶ 103] state, but the outputs are *not* reset.

VAR_INPUT

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

NETID: The network address of the computer in which the card is installed can be entered here (type: T_AmsNetID). If the card is located within the same system, it is also possible to give an empty string.

PORT: Contains the ADS port number of the card that was assigned by the TwinCAT system (type: T_AmsPort).

WRTRD: The function block is activated by a positive edge at this value.

TMOUT: Maximum time allowed for the execution of the command.

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 error occurs while the command is being transferred, the ERR output is set once the BUSY signal is cancelled.

ERRID: Returns the ADS error number [▶ 153] in the case of error.

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 platform	IO Hardware	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	None. This functionality is not supported by TwinCAT 3 at present!	Phoenix: IBS SC/I-T; IBS SC/RI/RT-LK	Tc2_IoFunctions (IO)

3.14.1.6 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. An **Alarm_Stop** places the card in the PAR_READY [▶ 103] state and resets all the outputs.

VAR_INPUT

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

NETID: The network address of the computer in which the card is installed can be entered here (type: T_AmsNetID). If the card is located within the same system, it is also possible to give an empty string.

PORT: Contains the ADS port number of the card that was assigned by the TwinCAT system (type: T_AmsPort).

WRTRD: The function block is activated by a positive edge at this value.

TMOUT: Maximum time allowed for the execution of the command.

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 error occurs while the command is being transferred, the ERR output is set once the BUSY signal is cancelled.

ERRID: Returns the ADS error number [▶ 153] in the case of error.

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 platform	IO Hardware	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	None. This functionality is not supported by TwinCAT 3 at present!	Phoenix: IBS SC/I-T; IBS SC/RI/RT-LK	Tc2_IoFunctions (IO)

3.14.1.7 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. 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 := DEFAULT_ADS_TIMEOUT;
    SWITCH_CODE  : WORD;
    DEVICE_NO    : WORD;
END_VAR
```

NETID: The network address of the computer in which the card is installed can be entered here (type: T_AmsNetID). If the card is located within the same system, it is also possible to give an empty string.

PORT: Contains the ADS port number of the card that was assigned by the TwinCAT system (type: T_AmsPort).

WRTRD: The function block is activated by a positive edge at this value.

TMOUT: Maximum time allowed for the execution of the command.

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: Specifies 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 error occurs while the command is being transferred, the ERR output is set once the BUSY signal is cancelled.

ERRID: Returns the ADS error number [▶ 153] in the case of error.

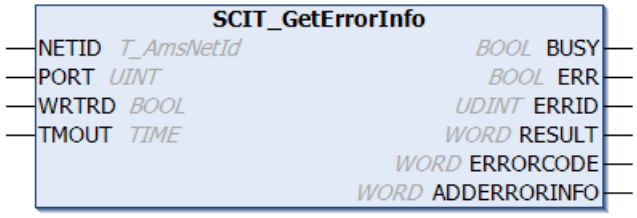
RESULT: Returns the result from the card (provided that fault-free ADS transport has occurred (ERR = FALSE)). RESULT = 0 identifies successful execution of the command. A value other than 0 contains the error number from the Phoenix card.

ADDERRINFO: If feedback from the card is negative, this contains additional error information (cf. the Phoenix card error descriptions).

Requirements

Development environment	Target platform	IO Hardware	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	None. This functionality is not supported by TwinCAT 3 at present!	Phoenix: IBS SC/I-T; IBS SC/RI/RT-LK	Tc2_IoFunctions (IO)

3.14.1.8 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.

VAR_INPUT

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

NETID: The network address of the computer in which the card is installed can be entered here (type: T_AmsNetID). If the card is located within the same system, it is also possible to give an empty string.

PORT: Contains the ADS port number of the card that was assigned by the TwinCAT system (type: T_AmsPort).

WRTRD: The function block is activated by a positive edge at this value.

TMOUT: Maximum time allowed for the execution of the command.

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 error occurs while the command is being transferred, the ERR output is set once the BUSY signal is cancelled.

ERRID: Returns the ADS error number [▶ 153] in the case of error.

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 platform	IO Hardware	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	None. This functionality is not supported by TwinCAT 3 at present!	Phoenix: IBS SC/I-T; IBS SC/RI/RT-LK	Tc2_IoFunctions (IO)

3.14.1.9 SCIT_ConfDevErrAll

The “SCIT_ConfDevErrAll” function block acknowledges periphery errors of all existing devices at the same time. The **Control_Device_Function** of the interbus card is called internally. The interbus card is addressed with the NETID and the PORT.

VAR_INPUT

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

NETID: The network address of the computer in which the card is installed can be entered here (type: T_AmsNetID). If the card is located within the same system, it is also possible to give an empty string.

PORT: Contains the ADS port number of the card that was assigned by the TwinCAT system (type: T_AmsPort).

WRTRD: The function block is activated by a positive edge at this value.

TMOUT: Maximum time that is not to be exceeded when executing the command.

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 an feedback is received.

ERR: If an error occurs while the command is being transferred, the ERR output is set once the BUSY signal is cancelled.

ERRID: Returns the ADS error number [▶ 153] in the case of error.

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.

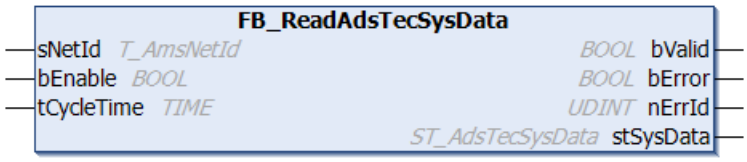
ADDERRORINFO: If feedback from the card is negative, this contains additional error information (cf. the Phoenix card error descriptions).

Requirements

Development environment	Target platform	IO Hardware	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	None. This functionality is not supported by TwinCAT 3 at present!	Phoenix: IBS SC/I-T; IBS SC/RI/RT-LK	Tc2_IoFunctions (IO)

3.14.2 ads-tec

3.14.2.1 FB_ReadAdsTecSysData



The *FB_ReadAdsTecSysData* function block reads the system data/diagnostic data of an ads-tec Industrial PC. 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: A string with the network address of the TwinCAT computer whose system data are to be read can be entered here (type: T_AmsNetID). If it is to be run on the local computer, an empty string can be entered.

bEnable: The block is reset by a rising edge (previous errors at the *bError* and *nErrId* outputs cleared). the system data are read cyclically if the input is set.

tCycleTime: The cyclic reading 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 occurred during the last reading cycle).

bError: This output is set if an error occurred when executing the function. The error is cleared with a rising edge at the *bEnable* input.

nErrId: Supplies the ADS error number [▶ 153] when the *bError* output is set.

stSysData: Structure with the system data/diagnostic data (type: ST_AdsTecSysData [▶ 126]).

Requirements

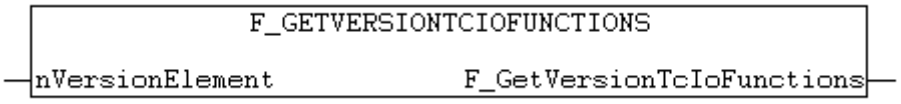
Development environment	Target platform	IO hardware	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	None. This functionality is not supported by TwinCAT 3 at present!	ads-tec PC	Tc2_IoFunctions (IO)

4 Functions

4.1 [Obsolete]

4.1.1 F_GetVersionTcIoFunctions

This function is obsolete and should not be used. Please use the global constant stLibVersion_Tc2_IoFunctions [▶ 145] in order to read version information from the PLC library.



This function reads version information from the PLC library.

FUNCTION F_GetVersionTcIoFunctions : UINT

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

nVersionElement : Version element to be read. Possible parameters:

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

Requirements

Development environment	Target platform	IO hardware	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	PC or CX (x86, x64, ARM)	All IO devices	Tc2_IoFunctions (IO)

4.1.2 F_GetVersionRAIDController

This function is obsolete and should not be used. Please use the global constant stLibVersion_Tc2_IoFunctions [▶ 145] in order to read version information from the PLC library.

This function reads version information from the PLC library.

FUNCTION F_GetVersionRAIDController : UINT

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

nVersionElement : Version element to be read. Possible parameters:

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

Requirements

Development environment	Target platform	IO hardware	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	PC or CX (x86, x64, ARM)	All IO devices	Tc2_IoFunctions (IO)

5 Data Types

5.1 E_PD_Dpv1Error

E_PD_Dpv1Error lists the DPV1-Error IDs:

```
TYPE E_PD_Dpv1Error :
(
    ePD_Err_ParamNumber               := 0,(* Unzulässige Parameternummer *)
    ePD_Err_ParamReadOnly             := 1,(* Parameterwert nicht änderbar *)
    ePD_Err_ValueOutOfRange           := 2,(* Untere oder obere Wertgrenze überschritten *)
    ePD_Err_InvalidSubIndex           := 3,(* Fehlerhafter Subindex *)
    ePD_Err_NoArray                   := 4,(* Kein Array *)
    ePD_Err_WrongDataType             := 5,(* Falscher Datentyp *)
    ePD_Err_OnlyResetPermitted        := 6,(* Kein Setzen errlaubt (nur Rücksetzen) *)
    ePD_Err_DescNotChangable          := 7,(* Beschreibungselement nicht änderbar *)
    ePD_Err_DescNotFound              := 9,(* Beschreibungselement nicht vorhanden *)
    ePD_Err_NoPermissionToChange      := 11,(* Keine Bedienhoheit *)
    ePD_Err_NoTextArray               := 15,(* Kein Textarray vorhanden *)
    ePD_Err_JobNotExecutable          := 17,(* Auftrag wegen Betriebszustand nicht ausführbar *)
    ePD_Err_ValueInvalid              := 20,(* Wert unzulässig *)
    ePD_Err_ResponseToLong            := 21,(* Antwort zu lang *)
    ePD_Err_ParamAddrInvalid          := 22,(* Parameteradresse unzulässig *)
    ePD_Err_FormatInvalid             := 23,(* Format unzulässig *)
    ePD_Err_NumOfValuesInvalid        := 24,(* Anzahl Werte nicht konsistent *)
    ePD_Err_DriveObjNotExisting       := 25,(* Antriebsobjekt existiert nicht *)
    ePD_Err_ParamDeactivated          := 101,(* Parameter momentan deaktiviert *)
    ePD_Err_ParamNoWrIfEnabled        := 107,(* Kein Schreibzugriff bei freigegebenem Regler *)
    ePD_Err_ParamInvalidUnit          := 108,(* Unbekannte Einheit *)
    ePD_Err_ParamNoWrIfNotInitFbk     := 109,(* Schreibzugriff nur in Inbetriebnahmezustand Geber *)
    ePD_Err_ParamWrIfInitMtr          := 110,(* Schreibzugriff nur in Inbetriebnahmezustand Motor *)
    ePD_Err_ParamWrIfInitDrv          := 111,
(* Schreibzugriff nur in Inbetriebnahmezustand Leistungsteil *)
    ePD_Err_ParamWrIfFastInit         := 112,(* Schreibzugriff nur in Schnellinbetriebnahme *)
    ePD_Err_ParamWrIfReady            := 113,(* Schreibzugriff nur in Bereit *)
    ePD_Err_ParamWrIfInitParamReset   := 114,
(* Schreibzugriff nur in Inbetriebnahmezustand Parameterreset *)
    ePD_Err_ParamWrIfInitSafety       := 115,
(* Schreibzugriff nur in Inbetriebnahmezustand Safety *)
    ePD_Err_ParamWrIfInitTechApp      := 116,
(* Schreibzugriff nur in Inbetriebnahmezustand Tech.Appl./Einheiten *)
    ePD_Err_ParamWrIfInit             := 117,(* Schreibzugriff nur in Inbetriebnahmezustand *)
    ePD_Err_ParamWrIfInitDwnLd        := 118,
(* Schreibzugriff nur in Inbetriebnahmezustand Download *)
    ePD_Err_ParamNoWrtIfDwnLd         := 119,(* Darf im Download nicht geschrieben werden *)
    ePD_Err_ParamWrIfInitDrvCfg       := 120,
(* Schreibzugriff nur in Inbetriebnahmezustand Antriebskonfiguration *)
    ePD_Err_ParamWrIfInitSetDrvType   := 121,
(* Schreibzugriff nur in Inbetriebnahmezustand Festlegung Antriebstyp *)
    ePD_Err_ParamWrIfInitDatasetCfg   := 122,
(* Schreibzugriff nur in Inbetriebnahmezustand Datensatz-Basiskonfiguration *)
    ePD_Err_ParamWrIfInitDevCfg       := 123,
(* Schreibzugriff nur in Inbetriebnahmezustand Gerätekonfiguration *)
    ePD_Err_ParamWrIfInitDevDwnLd     := 124,
(* Schreibzugriff nur in Inbetriebnahmezustand Gerätedownload *)
    ePD_Err_ParamWrIfInitDevPrmReset  := 125,
(* Schreibzugriff nur in Inbetriebnahmezustand Geräteparameterreset *)
    ePD_Err_ParamWrIfInitDevReady     := 126,
(* Schreibzugriff nur in Inbetriebnahmezustand Gerät bereit *)
    ePD_Err_ParamWrIfInitDevice       := 127,(* Schreibzugriff nur in Inbetriebnahmezustand Gerät *)
    ePD_Err_ParamNoWriteIfDwnLd       := 129,(* darf im Download nicht geschrieben werden *)
    ePD_Err_CtrlTakeOverBlocked       := 130,(* Übernahme der Steuerungshoheit über BICO gesperrt *)
    ePD_Err_ParamBicoSetInvalid       := 131,(* gewünschte BICO-Verschaltung unmöglich *)
    ePD_Err_ParamChangeBlocked        := 132,(* Parameteränderung gesperrt *)
    ePD_Err_ParamNoAccessDefined      := 133,(* Keine Zugriffsmethode definiert *)
    ePD_Err_BelowDefinedMinimum       := 200,(* Unterhalb aktuell gültiger Grenze *)
    ePD_Err_AboveDefinedMaximum       := 201,(* Oberhalb aktuell gültiger Grenze *)
    ePD_Err_WriteNotPermitted         := 204(* Schreiben nicht erlaubt *)
);
END_TYPE
```

Requirements

Development environment	Target platform	IO hardware	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	PC or CX (x86, x64, ARM)	Beckhoff FC310x PCI, CX1500-M310, EL6731, EL6632	Tc2_IoFunctions (IO)

5.2 E_BatteryStatus

Battery status.

```
TYPE E_BatteryStatus :
(
    BatteryUnknownStatus,
    BatteryOk,
    BatteryReplace
);
END_TYPE
```

Name	Value	Meaning
BatteryUnknownStatus	0	The battery status is unknown.
BatteryOk	1	The battery status is OK.
BatteryReplace	2	The battery should be replaced.

Requirements

Development environment	Target platform	UPS hardware	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	PC or CX (x86, x64, ARM)	• 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); • The APC devices that come supplied with Beckhoff Industrial PC support the Smart protocol and can be configured with the Windows UPS service.	Tc2_IoFunctions (IO)

5.3 E_PD_Datatype

E_PD_Datatype contains the possible data types of a Profidrive parameter.

```
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
```

Requirements

Development environment	Target platform	IO hardware	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	PC or CX (x86, x64, ARM)	Beckhoff FC310x PCI, CX1500-M310, EL6731, EL6632	Tc2_IoFunctions (IO)

5.4 E_RAIDDriveStatus

```
TYPE E_RAIDDriveStatus :
(
    eRAID_DRIVE_STATUS_OK         := 0,
    eRAID_DRIVE_STATUS_REBUILDING := 1,
    eRAID_DRIVE_STATUS_FAILED     := 2,
    eRAID_DRIVE_STATUS_DEGRADED   := 3
);
END_TYPE
```

Name	Value	Meaning
eRAID_DRIVE_STATUS_OK	0	Indicates that the physical drive is ready for operation.
eRAID_DRIVE_STATUS_DEGRADED	1	Indicates that the physical drive has sent a SMART message to the controller.
eRAID_DRIVE_STATUS_REBUILDING	2	Indicates that the physical drive is the target drive of a RAID set rebuild. If the rebuild has been successfully completed, the status changes to **eRAID_DRIVE_STATUS_OK**. The status is updated accordingly if the rebuild fails.
eRAID_DRIVE_STATUS_FAILED	3	Indicates that the physical drive has signaled unrecoverable errors to the controller or that the drive has started a vendor-specific action in order to remove the drive from the RAID set. There is no guarantee of the readiness for operation of the drive and loss of data has occurred or threatens.

Requirements

Development environment	Target platform	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	PC or CX (x86, x64, ARM)	Tc2_IoFunctions (IO)

5.5 E_RAIDDriveUsage

```
TYPE E_RAIDDriveUsage :
(
    eRAID_DRIVE_CONFIG_NOT_USED   := 0,
    eRAID_DRIVE_CONFIG_MEMBER     := 1,
    eRAID_DRIVE_CONFIG_SPARE      := 2
);
END_TYPE
```

Name	Value	Meaning
eRAID_DRIVE_CONFIG_NOT_USED	0	Indicates that the physical drive is not part of a RAID set.
eRAID_DRIVE_CONFIG_MEMBER	1	Indicates that the physical drive is part of a RAID set.
eRAID_DRIVE_CONFIG_SPARE	2	Indicates that the physical drive is part of this RAID set as a “hot swap spare”.

“hot swap spare” -> if a drive within the RAID group fails, it is replaced by the reserve drive (hot spare) while operation continues (hot swap). Redundancy is thus restored as quickly as possible.

Requirements

Development environment	Target platform	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	PC or CX (x86, x64, ARM)	Tc2_IoFunctions (IO)

5.6 E_RAIDStatus

```
TYPE E_RAIDStatus :
(
    eRAID_SET_STATUS_OK         := 0,
    eRAID_SET_STATUS_DEGRADED   := 1,
    eRAID_SET_STATUS_REBUILDING := 2,
    eRAID_SET_STATUS_FAILED     := 3
);
END_TYPE
```

Name	Value	Meaning
eRAID_SET_STATUS_OK	0	Indicates that the RAID set is ready for operation.
eRAID_SET_STATUS_DEGRADED	1	Indicates that the RAID set is no longer operating in an error-tolerant mode.
eRAID_SET_STATUS_REBUILDING	2	Indicates that the RAID set is rebuilding. This means that operation is restricted. Once the rebuild has been successfully completed, the status changes to **eRAID_SET_STATUS_OK**. The status is updated accordingly if the rebuild fails.
eRAID_SET_STATUS_FAILED	3	Indicates that the RAID set has failed. There is no guarantee of the readiness for operation of the RAID set and loss of data has occurred or threatens.

Requirements

Development environment	Target platform	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	PC or CX (x86, x64, ARM)	Tc2_IoFunctions (IO)

5.7 E_RAIDType

```
TYPE E_RAIDType :
(
    eRAID_TYPE_NONE  := 0,
    eRAID_TYPE_0     := 1,
    eRAID_TYPE_1     := 2,
    eRAID_TYPE_10    := 3,
    eRAID_TYPE_5     := 4,
    eRAID_TYPE_15    := 5,
    eRAID_TYPE_OTHER := 255
);
END_TYPE
```

Name	Value	Meaning
eRAID_TYPE_NONE	0	Indicates that the RAID set consists of a single drive. No set exists with the specified number.
eRAID_TYPE_0	1	Indicates that the RAID set is a striped set without error tolerance.
eRAID_TYPE_1	2	Indicates that the RAID set is a mirrored set.
eRAID_TYPE_10	3	Indicates that the RAID set is a striped and mirrored set.
eRAID_TYPE_5	4	Indicates that the RAID set is a parity set.
eRAID_TYPE_15	5	Indicates that the RAID set is advanced parity set.
eRAID_TYPE_OTHER	255	Indicates that the configuration of the RAID set does not correspond to the standard types.

Requirements

Development environment	Target platform	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	PC or CX (x86, x64, ARM)	Tc2_IoFunctions (IO)

5.8 E_SercosAttribLen

```
TYPE E_SercosAttribLen : (
    eLEN_2BYTE  := 1, (* 2 bytes, fixed length *)
    eLEN_4BYTE  := 2, (* 4 bytes, fixed length *)
    eLEN_8BYTE  := 3, (* 8 bytes, fixed length *)
    eLEN_V1BYTE := 4, (* 1 bytes, variable length *)
    eLEN_V2BYTE := 5, (* 2 bytes, variable length *)
    eLEN_V4BYTE := 6, (* 4 bytes, variable length *)
    eLEN_V8BYTE := 7 (* 8 bytes, variable length *)
);
END_TYPE
```

Requirements

Development environment	Target platform	IO hardware	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	PC (x86)	Sercans SCS-P ISA; Sercans SCS-P PCI; Beckhoff FC750x PCI	Tc2_IoFunctions (IO)

5.9 E_SercosAttribType

```
TYPE E_SercosAttribType :
(
    eType_BIN      := 0,
    eType_UNSIGNED := 1,
    eType_SIGNED   := 2,
    eType_HEX      := 3,
    eType_ASCII    := 4,
    eType_ID       := 5,
    eType_FLOAT    := 6
);
END_TYPE
```

Requirements

Development environment	Target platform	IO hardware	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	PC (x86)	Sercans SCS-P ISA; Sercans SCS-P PCI; Beckhoff FC750x PCI	Tc2_IoFunctions (IO)

5.10 E_UpsCommStatus

Status of communication with the UPS hardware.

```
TYPE E_UpsCommStatus :
(
    UpsCommUnknownStatus,
    UpsCommOk,
    UpsCommFailed
);
END_TYPE
```

Name	Value	Meaning
UpsCommUnknownStatus	0	The communication status is unknown.
UpsCommOk	1	Communication has been established with the UPS.
UpsCommFailed	2	Communication with the UPS was interrupted.

Requirements

Development environment	Target platform	UPS hardware	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	PC or CX (x86, x64, ARM)	• 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); • The APC devices that come supplied with Beckhoff Industrial PC support the Smart protocol and can be configured with the Windows UPS service.	Tc2_IoFunctions (IO)

5.11 E_UpsPowerStatus

Status of the power supply.

```
TYPE E_UpsPowerStatus :
(
    PowerUnknownStatus,
    PowerOnLine,
    PowerOnBattery
);
END_TYPE
```

Name	Value	Meaning
PowerUnknownStatus	0	The status of the power supply is unknown
PowerOnLine	1	Mains power supply.
Battery power supply.	2	Battery power supply.

Requirements

Development environment	Target platform	UPS hardware	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	PC or CX (x86, x64, ARM)	• 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); • The APC devices that come supplied with Beckhoff Industrial PC support the Smart protocol and can be configured with the Windows UPS service.	Tc2_IoFunctions (IO)

5.12 IODEVICETYPES

```
TYPE IODEVICETYPES:
(
IODEVICETYPE_UNKNOWN            := 0,(* Unknown device *)
IODEVICETYPE_C1220              := 1,(* Beckhoff Lightbus-Master *)
IODEVICETYPE_C1200              := 2,(* Beckhoff Lightbus-Master *)
IODEVICETYPE_SPC3               := 3,(* ProfiBus Slave (Siemens) *)
IODEVICETYPE_CIF30DPM           := 4,(* ISA ProfiBus-Master 2 kByte (Hilscher) *)
IODEVICETYPE_CIF40IBSM          := 5,(* ISA Interbus-S-Master 2 kByte (Hilscher) *)
IODEVICETYPE_BKHFPC             := 6,(* Beckhoff PC C2001*)
IODEVICETYPE_CP5412A2           := 7,(* ProfiBus-Master (Siemens)*)
IODEVICETYPE_SERCANSISA         := 8,(* Sercos Master (Indramat)*)
IODEVICETYPE_LPTPORT            := 9,(* Lpt Port*)
IODEVICETYPE_DPRAM              := 10,(* Generic DPRAM*)
IODEVICETYPE_COMPORT            := 11,(* COM Port*)
IODEVICETYPE_CIF30CAN           := 12,(* ISA CANopen-Master (Hilscher)*)
IODEVICETYPE_CIF30PB            := 13,(* ISA ProfiBus-Master 8 kByte (Hilscher)*)
IODEVICETYPE_BKHFCP2030         := 14,(* Beckhoff CP2030 (Pannel-Link)*)
IODEVICETYPE_IBSSCIT            := 15,(* Interbus-S-Master (Phoenix)*)
IODEVICETYPE_CIF30IBM           := 16,(* ISA Interbus-S-Master (Hilscher)*)
IODEVICETYPE_CIF30DNM           := 17,(* ISA DeviceNet-Master (Hilscher)*)
IODEVICETYPE_FCXXXX             := 18,(* Beckhoff-Filedbus card *)
IODEVICETYPE_CIF50PB            := 19,(* PCI ProfiBus-Master 8 kByte (Hilscher)*)
IODEVICETYPE_CIF50IBM           := 20,(* PCI Interbus-S-Master (Hilscher)*)
IODEVICETYPE_CIF50DNM           := 21,(* PCI DeviceNet-Master (Hilscher)*)
IODEVICETYPE_CIF50CAN           := 22,(* PCI CANopen-Master (Hilscher)*)
IODEVICETYPE_CIF60PB            := 23,(* PCMCIA ProfiBus-Master (Hilscher)*)
IODEVICETYPE_CIF60DNM           := 24,(* PCMCIA DeviceNet-Master (Hilscher)*)
IODEVICETYPE_CIF60CAN           := 25,(* PCMCIA CANopen-Master (Hilscher)*)
IODEVICETYPE_CIF104DP           := 26,(* PC104 ProfiBus-Master 2 kByte (Hilscher)*)
IODEVICETYPE_C104PB             := 27,(* PC104 ProfiBus-Master 8 kByte (Hilscher)*)
IODEVICETYPE_C104IBM            := 28,(* PC104 Interbus-S-Master 2 kByte (Hilscher)*)
IODEVICETYPE_C104CAN            := 29,(* PC104 CANopen-Master (Hilscher)*)
IODEVICETYPE_C104DNM            := 30,(* PC104 DeviceNet-Master (Hilscher)*)
IODEVICETYPE_BKHFCP9030         := 31,(* Beckhoff CP9030 (Pannel-Link with UPS)*)
IODEVICETYPE_SMB                := 32,(* Motherboard System Management Bus*)
IODEVICETYPE_PBMON              := 33,(* Beckhoff-PROFIBUS-Monitor*)
IODEVICETYPE_CP5613             := 34,(* PCI ProfiBus-Master (Siemens)*)
IODEVICETYPE_CIF60IBM           := 35,(* PCMCIA Interbus-S-Master (Hilscher)*)
IODEVICETYPE_FC200X             := 36,(* Beckhoff-Lightbus-I/II-PCI-Karte*)
IODEVICETYPE_FC3100_OLD         := 37,(* obsolete: dont use*)
IODEVICETYPE_FC3100             := 38,(* Beckhoff-Profibus-PCI*)
```

```
IODEVICETYPE_FC5100             := 39,(* Beckhoff-CanOpen-PCI*)
IODEVICETYPE_FC5200             := 41,(* Beckhoff-DeviceNet-PCI*)
IODEVICETYPE_BKHFNCBP           := 43,(* Beckhoff NC back plane*)
IODEVICETYPE_SERCANSPCI         := 44,(* Sercos Master (SICAN/IAM PCI)*)
IODEVICETYPE_ETHERNET           := 45,(* Virtual Ethernet Device*)
IODEVICETYPE_SERCONPCI          := 46,(* Sercon 410B or 816 Chip Master or Slave (PCI)*)
IODEVICETYPE_IBSSCRIRTLK        := 47,(* Interbus-S-Master with Slave-Module LWL Basis (Phoenix)*)
IODEVICETYPE_FC7500             := 48,(* Beckhoff-SERCOS-PCI*)
IODEVICETYPE_CIF30IBS           := 49,(* ISA Interbus-S-Slave (Hilscher)*)
IODEVICETYPE_CIF50IBS           := 50,(* PCI Interbus-S-Slave (Hilscher)*)
IODEVICETYPE_C104IBS            := 51,(* PC104 Interbus-S-Slave (Hilscher)*)
IODEVICETYPE_BKHFCP9040         := 52,(* Beckhoff CP9040 (CP-PC) *)
IODEVICETYPE_BKHFAH2000         := 53,(* Beckhoff AH2000 (Hydr. Backplane) *)
IODEVICETYPE_BKHFCP9035         := 54,(* Beckhoff CP9035 (PCI, Pannel-Link with UPS) *)
IODEVICETYPE_AH2000MC           := 55,(* Beckhoff-AH2000 with Profibus-MC *)
IODEVICETYPE_FC3100MON          := 56,(* Beckhoff-Profibus-Monitor-PCI *)
IODEVICETYPE_USB                := 57,(* Virtual USB Device *)
IODEVICETYPE_FC5100MON          := 58,(* Beckhoff-CANopen-Monitor-PCI *)
IODEVICETYPE_FC5200MON          := 59,(* Beckhoff-DeviceNet-Monitor-PCI *)
IODEVICETYPE_FC3100SLV          := 60,(* Beckhoff-Profibus-PCI Slave *)
IODEVICETYPE_FC5100SLV          := 61,(* Beckhoff-CanOpen-PCI Slave *)
IODEVICETYPE_FC5200SLV          := 62,(* Beckhoff-DeviceNet-PCI Slave *)
IODEVICETYPE_IBSSCITPCI         := 63,(* PCI Interbus-S-Master (Phoenix) *)
IODEVICETYPE_IBSSCRIRTLKPCI     := 64,(* PCI Interbus-S-Master with Slave-
Modulel LWL Basis (Phoenix) *)
IODEVICETYPE_CX1100_BK          := 65,(* Beckhoff-CX1100 terminal bus power supply *)
IODEVICETYPE_ENETRTMP           := 66,(* Ethernet real time miniport *)
IODEVICETYPE_CX1500_M200        := 67,(* PC104 Lightbus-Master *)
IODEVICETYPE_CX1500_B200        := 68,(* PC104 Lightbus-Slave *)
IODEVICETYPE_CX1500_M310        := 69,(* PC104 ProfiBus-Master *)
IODEVICETYPE_CX1500_B310        := 70,(* PC104 ProfiBus-Slave *)
IODEVICETYPE_CX1500_M510        := 71,(* PC104 CANopen-Master *)
IODEVICETYPE_CX1500_B510        := 72,(* PC104 CANopen-Slave *)
IODEVICETYPE_CX1500_M520        := 73,(* PC104 DeviceNet-Master *)
IODEVICETYPE_CX1500_B520        := 74,(* PC104 DeviceNet-Slave *)
IODEVICETYPE_CX1500_M750        := 75,(* PC104 Sercos-Master *)
IODEVICETYPE_CX1500_B750        := 76,(* PC104 Sercos-Slave *)
IODEVICETYPE_BX_BK              := 77,(* BX terminal bus interface *)
IODEVICETYPE_BX_M510            := 78,(* BX SSB-Master *)
IODEVICETYPE_BX_B310            := 79,(* BX ProfiBus-Slave *)
IODEVICETYPE_IBSSCRIRTPCI       := 80,(* PCI Interbus-S-
Master with slave module copper basis (Phoenix) *)
IODEVICETYPE_BX_B510            := 81,(* BX CANopen Slave *)
IODEVICETYPE_BX_B520            := 82,(* BX DeviceNet Slave *)
IODEVICETYPE_BC3150             := 83,(* BCxx50 ProfiBus Slave *)
IODEVICETYPE_BC5150             := 84,(* BCxx50 CANopen Slave *)
IODEVICETYPE_BC5250             := 85,(* BCxx50 DeviceNet Slave *)
IODEVICETYPE_EL6731             := 86,(* Beckhoff Profibus-EtherCAT Terminal *)
IODEVICETYPE_EL6751             := 87,(* Beckhoff CanOpen-EtherCAT Terminal *)
IODEVICETYPE_EL6752             := 88,(* Beckhoff DeviceNet-EtherCAT Terminal *)
IODEVICETYPE_COMPB              := 89,(* COM ProfiBus Master 8 kByte (Hilscher) *)
IODEVICETYPE_COMIBM             := 90,(* COM Interbus-S Master (Hilscher) *)
IODEVICETYPE_COMDNM             := 91,(* COM DeviceNet Master (Hilscher) *)
IODEVICETYPE_COMCAN             := 92,(* COM CANopen Master (Hilscher) *)
IODEVICETYPE_COMIBS             := 93,(* COM CANopen Slave (Hilscher) *)
IODEVICETYPE_ETHERCAT           := 94,(* EtherCAT in direct mode *)
IODEVICETYPE_PROFINETIOCONTROLLER:= 95,(* PROFINET Master *)
IODEVICETYPE_PROFINETIODEVICE   := 96,(* PROFINET Slave *)
IODEVICETYPE_EL6731SLV          := 97,(* Beckhoff Profibus Slave EtherCAT Terminal *)
IODEVICETYPE_EL6751SLV          := 98,(* Beckhoff CanOpen Slave EtherCAT Terminal *)
IODEVICETYPE_EL6752SLV          := 99,(* Beckhoff DeviceNet Slave EtherCAT Terminal *)
IODEVICETYPE_C104PPB            := 100,(* PC104+ ProfiBus Master 8 kByte (Hilscher) *)
IODEVICETYPE_C104PCAN           := 101,(* PC104+ CANopen Master (Hilscher) *)
IODEVICETYPE_C104PDNM           := 102,(* PC104+ DeviceNet Master (Hilscher) *)
IODEVICETYPE_BC8150             := 103,(* BCxx50 Serial Slave *)
IODEVICETYPE_BX9000             := 104,(* BX9000 Ethernet Slave *)
IODEVICETYPE_CX9000_BK          := 105,(* Beckhoff-CX9000 K-Bus Power Supply *)
IODEVICETYPE_EL6601             := 106,(* Beckhoff-RT-Ethernet-EtherCAT-Terminal *)
IODEVICETYPE_BC9050             := 107,(* BC9050 Ethernet Slave *)
IODEVICETYPE_BC9120             := 108,(* BC9120 Ethernet Slave *)
IODEVICETYPE_ENETADAPTER        := 109,(* Ethernet Miniport Adapter *)
IODEVICETYPE_BC9020             := 110,(* BC9020 Ethernet Slave *)
IODEVICETYPE_ETHERCATPROT       := 111,(* EtherCAT Protocol in direct mode *)
IODEVICETYPE_ETHERNETNVPROT     := 112,(* *)
IODEVICETYPE_ETHERNETPNMPROT    := 113,(* Profinet Controller *)
IODEVICETYPE_EL6720             := 114,(* Beckhoff-Lightbus-EtherCAT-Terminal *)
IODEVICETYPE_ETHERNETPNSPROT    := 115,(* Profinet Device*)
IODEVICETYPE_BKHFCP6608         := 116,(* Beckhoff CP6608(IXP PC) *)
IODEVICETYPE_PTP_IEEE1588       := 117,(* *)
IODEVICETYPE_EL6631SLV          := 118,(* EL6631-0010 Profinet Slave terminal *)
```

```
IODEVICETYPE_EL6631             := 119,(* EL6631 Profinet Master terminal *)
IODEVICETYPE_CX5000_BK          := 120,(* Beckhoff-CX5100 K-Bus power supply *)
IODEVICETYPE_PCIDEVICE          := 121,(* Generic PCI DPRAM (TCOM) *)
IODEVICETYPE_ETHERNETUPDPROT    := 122,(* UDP Protocol *)
IODEVICETYPE_ETHERNETAUTOPROT   := 123,(* Automation Protocol *)
IODEVICETYPE_CCAT               := 124,(* CCAT *)
IODEVICETYPE_CPLINK3            := 125,(* Virtuelles USB Device (remote via CPLINK3) *)
IODEVICETYPE_EL6632             := 126,(* EL6632 *)
IODEVICETYPE_CCAT_PBM           := 127,(* CCAT Profibus Master *)
IODEVICETYPE_CCAT_PBS           := 128,(* CCAT Profibus Slave *)
IODEVICETYPE_CCAT_CNM           := 129,(* CCAT CANopen Master *)
IODEVICETYPE_ETHERCATSLAVE      := 130,(* EtherCAT Slave *)
IODEVICETYPE_BACNET             := 131,(* BACnet device *)
IODEVICETYPE_CCAT_CNS           := 132,(* CCAT CANopen Slave *)
IODEVICETYPE_ETHIP_SCANNER      := 133,(* ETHERNET IP Master *)
IODEVICETYPE_ETHIP_ADAPTER      := 134,(* ETHERNET IP Slave (OLD) *)
IODEVICETYPE_CX8000_BK          := 135,(* Beckhoff-CX8100 Klemmenbus Netzteil -
 LEGACY use IODEVICETYPE_CX_BK *)
IODEVICETYPE_ETHERNETUDPPROT    := 136,(* Upd Protocol *)
IODEVICETYPE_BC9191             := 137,(* BC9191 Etherent Slave *)
IODEVICETYPE_ENETPROTOCOL       := 138,(* Real-Time Ethernet Protocol (BK90xx, AX2000-B900) *)
IODEVICETYPE_ETHIP_ADAPTEREX    := 139,(* ETHERNET IP Slave (NEW) *)
IODEVICETYPE_PNCONTR_CCAT_RT    := 140,(* Profinet Controller CCAT RT *)
IODEVICETYPE_PNCONTR_CCAT_IRT   := 141,(* Profinet Controller CCAT RT + IRT *)
IODEVICETYPE_PNDEV_CCAT_RT      := 142,(* Profinet Device CCAT RT *)
IODEVICETYPE_PNDEV_CCAT_IRT     := 143,(* Profinet Device CCAT RT + IRT *)
IODEVICETYPE_ETHERCATSIMULATION := 144,(* EtherCAT-Simulation *)
IODEVICETYPE_EL6652SLV          := 145,(* EL6652-0010 *)
IODEVICETYPE_PTP_VIA_CCAT       := 146,(* PTP CLock via CCAT *)
IODEVICETYPE_BACNETR9           := 147,(* BACnet Rev9 device *)
IODEVICETYPE_ETHERCATXFC        := 148,(* EtherCAT in xfc mode *)
IODEVICETYPE_CX2500_0030        := 149,(* CX2500-0030 RS232 Serial Communication Port *)
IODEVICETYPE_CX2500_0031        := 150,(* CX2500-0031 RS422/RS485 Serial Communication Port *)
IODEVICETYPE_EL6652MST          := 151,(* EL6652 *)

(* Reserved for new devices*)


IODEVICETYPE_MAX
);
END_TYPE
```

Requirements

Development environment	Target platform	IO hardware	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	PC or CX (x86, x64, ARM)	All IO devices	Tc2_IoFunctions (IO)

5.13 ST_AdsTecSysData

```
TYPE ST_AdsTecSysData
STRUCT
    bShiftKey    : BOOL;(* TRUE == Shift key pressed*)
    bRMouseKey   : BOOL;(* TRUE == Right mouse key pressed *)
    bHotKey      : BOOL;(* TRUE == Hotkey pressed *)
    bTaskChaKey  : BOOL;(* TRUE == Task change key pressed *)
    bABCKey      : BOOL;(* TRUE == ABC soft keyboard key pressed*)
    bRsrv1       : BOOL;
    bRsrv2       : BOOL;
    bRsrv3       : BOOL;
    bMainFanErr  : BOOL;(* TRUE == Main fan error*)
    bCpuFanErr   : BOOL;(* TRUE == CPU fan error*)
    bTempErr     : BOOL;(* TRUE == Internal temperature error ( temp > 50°C)*)
    bBatteryErr  : BOOL;(* TRUE == Battery error *)
    bRsrv4       : BOOL;
    bRsrv5       : BOOL;
    bRsrv6       : BOOL;
    bRsrv7       : BOOL;
    nMainNtcTemp : SINT;(* Main NTC temperature (-127°C .. + 127°C) *)
    nExtNtcTemp  : SINT;(* External NTC temperature (-127°C .. + 127°C) *)
    nRsrv8       : ARRAY[1..12] OF BYTE;
END_STRUCT
END_TYPE
```

bShiftKey: “Shift” key pressed (key on the far right in the front).

bRMouseKey: “Right mouse” button pressed.

bHotKey: “Hotkey” pressed.

bTaskChaKey: “Task change” key pressed.

bABCKey: “ABC soft keyboard” key pressed.

bMainFanErr: Main fan error.

bCpuFanErr: CPU fan error.

bTempErr: Temperature error (internal temperature > 50 °C).

bBatteryErr: Battery error (currently reserved).

nMainNtcTemp: Temperature value 1 (soldered NTC -127 °C to +127 °C).

nExtNtcTemp: Temperature level 2 (connectable NTC, does not exist in every device).

bRsrv1 - bRsrv7: Reserved.

nRsrv8: Reserved.

Requirements

Development environment	Target platform	IO hardware	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	None. This functionality is not supported by TwinCAT 3 at present!	ads-tec PC	Tc2_IoFunctions (IO)

5.14 ST_Dpv1ParamAddrEx

ST_Dpv1ParamAddrEx contains the data of a Profidrive parameter.

```
TYPE ST_Dpv1ParamAddrEx :
STRUCT
    iAttribute       : USINT;
    iNumOfElements   : USINT;
    iParameterNumber : UINT;
    iSubIndex        : UINT;
    iDataAddr        : PVOID;
    iDataSize        : UDINT;
    eFormat          : E_PD_Datatype;
    iNumOfValues     : UINT;
    iErrorValue      : UDINT;
    stError          : ST_PD_Dpv1Error;
END_STRUCT
END_TYPE
```

iAttribute: 0x10: value; 0x20: description; 0x30: text; 0x80..F0: manufacturer-specific; other values are reserved.

iNumOfElements: 1 to 234: number of elements; 0: special functions; other values are reserved.

iParameterNumber: 1 to 65535: parameter number; 0: reserved.

iSubIndex: 0 to 65535: sub-index.

iDataAddr: address of the buffer/address of the PLC variables.

iDataSize: size of the buffer/size of the PLC variables.

eFormat: 0x01..0x36: data type; 0x40: ZERO; 0x41: BYTE; 0x42: WORD; 0x43: DWORD; 0x44: error; other values are reserved. (Type: E_PD_Datatype [▶ 118]).

iNumOfValues: 0 to 234: number of values; other values are reserved.

iErrorValue: DPV1 error value.

stError: DPV1 error flag, DPV1 error enumeration type. (Type: ST_PD_Dpv1Error [▶ 131]).

Requirements

Development environment	Target platform	IO hardware	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	PC or CX (x86, x64, ARM)	Beckhoff FC310x PCI, CX1500-M310, EL6731, EL6632	Tc2_IoFunctions (IO)

5.15 ST_Dpv1ValueHeaderEx

ST_Dpv1ValueHeaderEx contains the data of a parameter in the DPV1 telegram and its string representation.

```
TYPE ST_Dpv1ValueHeaderEx :
STRUCT
    eFormat      : E_PD_Datatype;
    iNumOfValues : USINT;
    iOffset      : USINT;
    iDataLen     : UINT;
    strData      : STRING;
END_STRUCT
END_TYPE
```

eFormat: 0x01..0x36: data type; 0x40: ZERO; 0x41: BYTE; 0x42: WORD; 0x43: DWORD; 0x44: error; other values are reserved. (Type: E_PD_Datatype [▶ 118]).

iNumOfValues: 0 to 234: number of values; other values are reserved.

iOffset: Offset in the DPV1 response telegram.

iDataLen: Data length.

strData: Data as STRING.

Requirements

Development environment	Target platform	IO hardware	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	PC or CX (x86, x64, ARM)	Beckhoff FC310x PCI, CX1500-M310, EL6731, EL6632	Tc2_IoFunctions (IO)

5.16 ST_NovRamAddrInfo

```
TYPE ST_NovRamAddrInfo:
STRUCT
    pCardAddress : POINTER TO DWORD;
    iCardMemSize : UDINT;
END_STRUCT
END_TYPE
```

pCardAddress : The address pointer of the NOV/DP-RAM.

iCardMemSize : The configured NOV/DP-RAM size in bytes.

Requirements

Development environment	Target platform	IO hardware	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	PC or CX (x86, x64, ARM)	All IO devices	Tc2_IoFunctions (IO)

5.17 ST_NovRamAddrInfoEx

```
TYPE ST_NovRamAddrInfoEx:
STRUCT
   pCardAddress  :  POINTER TO DWORD;
   iCardMemSize  :  UDINT;
   eAccessType   :  E_IOACCESSTYPE
END_STRUCT
END_TYPE
```

pCardAddress: The address pointer of the NOV/DP-RAM.

iCardMemSize : The configured NOV/DP-RAM size in bytes.

eAccessType: The access type to the NOV/DP-RAM.

eIOAccess_Default: Normal access via MEMCPY function possible

eIOAccess_Byte: Only BYTE access via FOR loop possible

eIOAccess_WordSwap: Only WORT access + high/low byte swapping via FOR loop possible

Requirements

Development environment	Target platform	IO hardware	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	PC or CX (x86, x64, ARM)	All IO devices	Tc2_IoFunctions (IO)

5.18 ST_Parameter_IN

Input data of the ASI terminal.

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

Byte Offset	Bit Offset	Description
0	0-5	Status bits (as with previous terminals)
0	6	0: No diagnosis, 1: Diagnosis (as with previous terminals)
0	7	always 0: no register communication
1	0-3	0-3 reserved for extensions
1	4	Toggle bit for acknowledging order (in case of cyclic, bit 6 is copied from byte 0)
1	5	Receipt (0: NoError, 1: Error)
1	6	0: Cyclic, 1: Acyclic
1	7	0: parameter access, 1: ADS
2		Input data (cyclic), parameter value (acyclic) or error number bits 0-7
3		Input data (cyclic), parameter value (acyclic) or error number bits 8-15
4		Input data (cyclic), parameter value (acyclic) or error number bits 16-23
5		Input data (cyclic), parameter value (acyclic) or error number bits 24-31

Requirements

Development environment	Target platform	IO hardware	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	PC or CX (x86)	ASI master terminal	Tc2_IoFunctions (IO)

5.19 ST_Parameter_OUT

Output data to the ASI terminal.

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

Byte Offset	Bit Offset	Description
0	0-5	Parameter number bits 0-5 (or parameter offset)
0	6	In case of acyclic: 0: Read, 1: Write, in case of cyclic (always read/write) the bit is copied into the input data in order to have a direct allocation (the cyclic data could then also be modified)
0	7	0: parameter access, 1: Register communication
1	0-5	Parameter number bits 6-11 (or parameter page)
1	6	0: Cyclic, 1: Acyclic
1	7	0: parameter access, 1: ADS
2		Output data (cyclic) or parameter value (acyclic) bits 0-7
3		Output data (cyclic) or parameter value (acyclic) bits 8-15
4		Output data (cyclic) or parameter value (acyclic) bits 16-23
5		Output data (cyclic) or parameter value (acyclic) bits 24-32

Requirements

Development environment	Target platform	IO hardware	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	PC or CX (x86)	ASI master terminal	Tc2_IoFunctions (IO)

5.20 ST_ParameterBuffer

Data buffer for the I/O data of the ASI terminal

```
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
```

Requirements

Development environment	Target platform	IO hardware	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	PC or CX (x86)	ASI master terminal	Tc2_IoFunctions (IO)

5.21 ST_PD_Dpv1Error

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

bError: Error flag (TRUE => error, FALSE => no error).

eErrorID: Error code. (Type: E_PD_Dpv1Error [▶ 117]).

Requirements

Development environment	Target platform	IO hardware	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	PC or CX (x86, x64, ARM)	Beckhoff FC310x PCI, CX1500-M310, EL6731, EL6632	Tc2_IoFunctions (IO)

5.22 ST_PNET_CCDSTS

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

Requirements

Development environment	Target platform	IO hardware	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	PC or CX (x86)	Beckhoff EL6632	Tc2_IoFunctions (IO)

5.23 ST_PNIOConfigRecord

```
TYPE ST_PNIOConfigRecord :
STRUCT
    iRW          : UINT;
    iNumOfAR     : UINT;
    iAPI         : UDINT;
    iSlot        : UINT;
    iSubSlot     : UINT;
    stPNIORecord : ST_PNIORecord;
END_STRUCT
END_TYPE
```

iRW: 0: Read, 1: Write.

iNumOfAR: number of arguments.

iAPI: API number.

iSlot: slot number.

iSubSlot: sub-slot number.

stPNIORecord: (Type: ST_PNIORecord [▶ 133]).

Requirements

Development environment	Target platform	IO hardware	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	PC or CX (x86)	Beckhoff EL6632	Tc2_IoFunctions (IO)

5.24 ST_PNIORecord

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

Requirements

Development environment	Target platform	IO hardware	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	PC or CX (x86)	Beckhoff EL6632	Tc2_IoFunctions (IO)

5.25 ST_PNIOState

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

Requirements

Development environment	Target platform	IO hardware	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	PC or CX (x86)	Beckhoff EL6632	Tc2_IoFunctions (IO)

5.26 ST_PZD_IN

Data words from the drive to the PLC.

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

Requirements

Development environment	Target platform	IO hardware	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	PC or CX (x86)	AX2000 Profibus box	Tc2_IoFunctions (IO)

5.27 ST_PZD_OUT

Data words from the PLC to the drive.

```
TYPE ST_PZD_OUT :
STRUCT
    wCtrlW :WORD;
    PZD2   :WORD;
    PZD3   :WORD;
    PZD4   :WORD;
```

```
    PZD5   :WORD;
    PZD6   :WORD;
END_STRUCT
END_TYPE
```

Requirements

Development environment	Target platform	IO hardware	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	PC or CX (x86)	AX2000 Profibus box	Tc2_IoFunctions (IO)

5.28 ST_RAIDCntlrFound

```
TYPE ST_RAIDCntlrFound :
STRUCT
    nRAIDCntlrCount : UDINT;
    nRAIDCntlrIds   : ARRAY [1..g_nMAX_NUMBER_OF_RAID_CNTLRS] OF UDINT;
END_STRUCT
END_TYPE
```

nRAIDCntlrCount: Number of RAID controllers

nRAIDCntlrIds: ID of each RAID controller (default value is 4294967295 and therefore invalid).

g_nMAX_NUMBER_OF_RAID_CNTLRS is the maximum number of RAID controllers and is defined as a global constant = 10.

Requirements

Development environment	Target platform	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	PC or CX (x86, x64, ARM)	Tc2_IoFunctions (IO)

5.29 ST_RAIDConfigReq

```
TYPE ST_RAIDConfigReq :
STRUCT
    nRAIDCntlrID  : UDINT;
    nRAIDSetIndex : UDINT;
END_STRUCT
END_TYPE
```

nRAIDCntlrID: ID of the RAID controller

nRAIDSetIndex: Contains the number of the RAID set for which information is being requested. Please note that the set begins with index 0 in the case of Beckhoff CBx051 boards.

Requirements

Development environment	Target platform	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	PC or CX (x86, x64, ARM)	Tc2_IoFunctions (IO)

5.30 ST_RAIDDriveStatus

```
TYPE ST_RAIDDriveStatus :
STRUCT
    eRAIDDriveStatus  : E_RAIDDriveStatus;
    eRAIDDriveUsage   : E_RAIDDriveUsage;
    nSATAPort         : UINT;
    sRAIDDriveSerial  : STRING [39];
END_STRUCT
END_TYPE
```

eRAIDDriveStatus: Contains the status of the physical drive (type: E_RAIDDriveStatus [▶ 119]).

eRAIDDriveUsage: States whether the physical drive is part of the RAID set (type: E_RAIDDriveUsage [▶ 120]).

nSATAPort: Contains the SAS address of the physical drive. This box should contain a 0 if the drive has no SAS address, as is the case, for example, with a directly attached SATA drive.

sRAIDDriveSerial: Serial number of the RAID drive (40 letters).

Requirements

Development environment	Target platform	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	PC or CX (x86, x64, ARM)	Tc2_IoFunctions (IO)

5.31 ST_RAIDInfo

```
TYPE ST_RAIDInfo :
STRUCT
    nNumRAIDSets     : UDINT;
    nMaxDrivesPerSet : UDINT;
    bReserved        : ARRAY [1..92] OF BYTE;
END_STRUCT
END_TYPE
```

nNumRAIDSets: Number of currently defined RAID sets. 0 is returned if no sets have been defined yet.

nMaxDriverPerSet: Maximum number of physical drives in a logical RAID set. This can be an absolute maximum or the maximum currently defined for all sets.

bReserved: Reserved for internal purposes.

Requirements

Development environment	Target platform	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	PC or CX (x86, x64, ARM)	Tc2_IoFunctions (IO)

5.32 ST_RAIDStatusRes

```
TYPE ST_RAIDStatusRes :
STRUCT
    nRAIDSetIndex     : UDINT;
    eRAIDType         : E_RAIDType;
    eRAIDStatus       : E_RAIDStatus;
    nRAIDDriveCount   : UINT;
    nReserved         : UINT;
    stRAIDDriveStatus : ARRAY [1..g_nMAX_NUMBER_OF_RAID_DRIVES] OF ST_RAIDDriveStatus;
END_STRUCT
END_TYPE
```

nRAIDSetIndex: RAID set ID, as for the input.

eRAIDType: Contains the basic RAID type of the RAID set (type: E_RAIDType [▶ 121]).
eRAID_TYPE_NONE, indicates that the RAID set consists of a single drive, i.e. no set exists with the specified number.

eRAIDStatus: Contains the status of the RAID set (type: E_RAIDStatus [▶ 121]).

nRAIDDriveCount: Contains the number of drives in the RAID set

nReserved: Reserved.

stRAIDDriveStatus: Contains the status of the physical drive and the information regarding whether the physical drive is part of the RAID set (type: ST_RAIDDriveStatus [▶ 134]).

g_nMAX_NUMBER_OF_RAID_DRIVES is the number of RAID drives whose status can be read and is defined as a global constant = 10.

Requirements

Development environment	Target platform	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	PC or CX (x86, x64, ARM)	Tc2_IoFunctions (IO)

5.33 ST_SercosParamAttrib

ST_SercosParamAttrib contains the nAttrib attribute of the Sercos parameter, broken down into individual variables.

```
TYPE ST_SercosParamAttrib :
STRUCT
    nFactor       : UINT;
    eLength       : E_SercosAttribLen;
    bCommand      : BOOL;
    eType         : E_SercosAttribType;
    bReserved1    : BOOL;
    nComma        : USINT;
    bWriteProtCP2 : BOOL;
    bWriteProtCP3 : BOOL;
    bWriteProtCP4 : BOOL;
    bReserved2    : BOOL;
END_STRUCT
END_TYP
```

nFactor: bits 0..15.

eLength: bits 16..18. (Type: E_SercosAttribLen [▶ 122]).

bCommand: bit 19.

eType: bits 20..22. (Type: E_SercosAttribType [▶ 122]).

bReserved1: bit 23.

nComma: bits 24..27.

bWriteProtCP2: bit 28.

bWriteProtCP3: bit 29.

bWriteProtCP4: bit 30.

bReserved2: bit 31.

Requirements

Development environment	Target platform	IO hardware	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	PC (x86)	Sercans SCS-P ISA; Sercans SCS-P PCI; Beckhoff FC750x PCI	Tc2_IoFunctions (IO)

5.34 ST_SercosParamErrList

```
TYPE ST_SercosParamErrList :
STRUCT
    iActCount    : UINT;
```

```
    iMaxCount    : UINT;
    iList        : ARRAY [0..2047] OF UINT;
    iError       : ARRAY [0..2047] OF UDINT;
END_STRUCT
END_TYPE
```

iActCount: The number of parameters skipped (in this case 3 = 3 parameter errors).

iMaxCount: The number of parameters skipped (in this case 3 = 3 parameter errors).

iList: A field of up to 2048 parameter numbers where access errors occurred.

iError: A field of up to 2048 access error numbers.

Requirements

Development environment	Target platform	IO hardware	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	PC (x86)	Sercans SCS-P ISA; Sercans SCS-P PCI; Beckhoff FC750x PCI	Tc2_IoFunctions (IO)

5.35 ST_SercosParamList

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

iActCount: The current number of parameters in a list * 2. Sercos saves the number of bytes here; a parameter number consists of two bytes, e.g. 6 means 3 parameters.

iMaxCount: The maximum number of parameters in a list * 2. Sercos saves the number of bytes here; a parameter number consists of two bytes, e.g. 6 means 3 parameters.

iList: A field of up to 2048 parameter numbers.

Requirements

Development environment	Target platform	IO hardware	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	PC (x86)	Sercans SCS-P ISA; Sercans SCS-P PCI; Beckhoff FC750x PCI	Tc2_IoFunctions (IO)

5.36 ST_UPSStatus

```
TYPE ST_UPSStatus
STRUCT
    Vendor             : STRING; (* UPS vendor name  *)
    Model              : STRING; (* UPS model name *)
    FirmwareRev        : STRING; (* UPS firmware revision *)
    SerialNumber       : STRING; (* UPS serial number *)
    BatteryLifePercent : DWORD; (* The percent of battery capacity remaining in the UPS (0..100%) *)
    BatteryLifeTime    : DWORD; (* Remaining UPS run time, in minutes *)
    eBatteryStatus     : E_BatteryStatus; (* UPS battery state *)
    eCommStatus        : E_UpsCommStatus; (* Status of the communication path to the UPS *)
    ePowerStatus       : E_UpsPowerStatus; (* Status of utility-supplied power into the UPS *)
    nPowerFailCnt      : DWORD; (* Power Fail counter. Increments every time the UPS service detects
 power fail *)
    dwChargeFlags      : DWORD; (* Battery charge status flags. This member can be one or more of th
e following values.
    Bits0..7 := General battery status flags (if all bits are set to 0 => unknown status)
        Bit0 := High (bit set => high battery charge)
```

```
        Bit1 := Low (bit set => low battery charge)
        Bit2 := Critical (bit set  => battery is near empty)
        Bit3 := Charging (bit set  => battery is charging)
        Bits4..6 := reserved (all bits are 0)
        Bit7 := No Battery (bit set => battery not found or not connected, bit not set => battery is
 existing or unknown state)
   Bits8..15 := Special status information (if all bits are set to 0 => state ok or unknown state)
        Bit8 := UPS Fan Error (bit set => fan hardware reports an error, bit not set => fan is ok)
        Bit9 := Over Temperature (bit set => over temperature detected, bit not set => temperature i
s ok)
        Bit10 := Service Interval Notify (bit set => service interval time expired, bit not set =>se
rvice interval time not expired )
        Bit11 := Under Temperature (bit set => under temperature detected , bit not set => temperatu
re is ok )
        Bit12 := Fuse Not Ok (bit set => fuse broken or missed, bit not set => fuse ok)
        Bit13 := Alarm1 (reserved for later use, bit is 0)
        Bit14 := Alarm2 (reserved for later use, bit is 0)
        Bit15 := Alarm3 (reserved for later use, bit is 0)
    Bits16..31 := (reserved for later use, all bits are 0)
 *)
END_STRUCT
END_TYPE
```

Not all UPS models can supply every item of status information.

X: The status information is available with this model.

*) Available only if the model “Smart Signaling to any APC UPS & TwinCAT” has been configured.

Status information	Beckhoff BAPI v1	Beckhoff P24Vxxxx UPS	Beckhoff 24V UPS on the CP903x card	CX2100-09x4	APC Back-UPS Pro 280	APC Smart-UPS 420	Description
Vendor	X	X	X	X	X	X	Vendor name.
Model	X	X	X	X	X	X	Model string. Empty string if no UPS has been configured.
FirmwareRev	X	X	X	X	X	X	UPS firmware version information. Empty string if the UPS does not support this parameter.
SerialNumber	X	X	None	X	X	X	Serial number of the UPS. Empty string if the UPS does not support this parameter.
BatteryLifePercent	X	X	None	X	X	X	Remaining battery life in percent. The value is always zero if the UPS cannot supply this parameter.
BatteryLifeTime	X	X	None	X	X	X	Remaining battery life in minutes. The value is always zero if the UPS cannot supply this parameter.
eBatteryStatus	**BatteryOk**	**BatteryUnknownStatus** if no battery exists; from UPS software version >= 2.0.0.6 and UPS firmware >= 25.1.I **BatteryOk**	**BatteryUnknownStatus** if no battery exists. **BatteryOk**	**BatteryUnknownStatus** when no battery is present (applicable only for the model with "Smart Battery" and not with capacitors) **BatteryOk**	X	X	Battery status (type: E_BatteryStatus).
eCommStatus	X	X	X	X	X	X	Status of the communication with the UPS (type: E_UpsCommStatus).
ePowerStatus	X	X	X	X	X	X	Status of external power supply (type: E_UpsPowerStatus).
nPowerFailCnt	X	X	X	X	*X	*X	Power-fail counter. The counter is incremented if a voltage failure is detected by the UPS service.

Status information	Beckhoff BAPI v1	Beckhoff P24Vxxxx UPS	Beckhoff 24V UPS on the CP903x card	CX2100-09x4	APC Back-UPS Pro 280	APC Smart-UPS 420	Description
dwChargeFlags	**No Battery** (bit 7 set) from UPS firmware >=33.12-0 if no battery isconnected. **Service Interval Notify** (bit 10 set). The configured battery change interval service has expired	**No Battery** (bit 7 set) from UPS software version >= 2.0.0.6 and firmware >= 25.1.I. The existence of the battery is checked every minute. **UPS Fan Error** (bit 8 set) from UPS software version >= 2.0.0.7 and firmware >= 40.1.I. The UPS fan status is checked each minute. **Requires a newer (second) hardware revision!** **Service Interval Notify** (bit 10 set). The configured battery change interval service has expired. Implemented in the UPS software version >= 3.0.0.8;	**High** (bit 0 set) if battery is fully charged. **Charging** (bit 3 set) **No Battery** (bit 7 set) if no battery was found.	**No Battery** (bit 7 set). No communication to the battery (applicable only for the model with "Smart Battery" and not with capacitors). **Over Temperature** (bit 9 set) was detected and the charging of the battery was interrupted. **Requires a newer (second) hardware revision.** Implemented in the UPS software version >= 3.0.0.18. **Service Interval Notify** (bit 10 set). The configured battery service interval has expired. **Under Temperature** (bit 11 set) was detected and the charging of the battery was interrupted. **Requires a newer (second) hardware revision.** Implemented in the UPS software version >= 3.0.0.18. **Fuse Not Ok** (bit 12 set). The "Smart Battery" fuse is defective or not available. **Requires a newer (second) hardware revision.** Implemented in the UPS software version >= 3.0.0.18.	None	None	Battery charge status flags and special status information.

Requirements

Development environment	Target platform	UPS hardware	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	PC or CX (x86, x64, ARM)	• 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); • The APC devices that come supplied with Beckhoff Industrial PC support the Smart protocol and can be configured with the Windows UPS service.	Tc2_IoFunctions (IO)

5.37 ST_KL1501InData

Structure for linking in the input process image.

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

Requirements

Development environment	Target platform	IO Hardware	PLC libraries to include
TwinCAT v3.1.4018.26	PC/CX	KL1501	Tc2_IoFunctions from v3.3.5.0

5.38 ST_KL1501OutData

Structure for linking in the output process image.

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

Requirements

Development environment	Target platform	IO Hardware	PLC libraries to include
TwinCAT v3.1.4018.26	PC/CX	KL1501	Tc2_IoFunctions from v3.3.5.0

5.39 ST_KL27x1InData

Structure for linking in the input process image.

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

Requirements

Development environment	Target platform	IO Hardware	PLC libraries to include
TwinCAT v3.1.4018.26	PC/CX	KL2751, KL2761	Tc2_IoFunctions from v3.3.5.0

5.40 ST_KL27x1OutData

Structure for linking in the output process image.

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

Requirements

Development environment	Target platform	IO Hardware	PLC libraries to include
TwinCAT v3.1.4018.26	PC/CX	KL2751, KL2761	Tc2_IoFunctions from v3.3.5.0

5.41 ST_KL320xInData

Structure for linking in the input process image.

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

Requirements

Development environment	Target platform	IO Hardware	PLC libraries to include
TwinCAT v3.1.4018.26	PC/CX	KL3201, KL3202, KL3204	Tc2_IoFunctions from v3.3.5.0

5.42 ST_KL320xOutData

Structure for linking in the output process image.

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

Requirements

Development environment	Target platform	IO Hardware	PLC libraries to include
TwinCAT v3.1.4018.26	PC/CX	KL3201, KL3202, KL3204	Tc2_IoFunctions from v3.3.5.0

5.43 ST_KL3208InData

Structure for linking in the input process image.

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

Requirements

Development environment	Target platform	IO Hardware	PLC libraries to include
TwinCAT v3.1.4018.26	PC/CX	KL3208	Tc2_IoFunctions from v3.3.5.0

5.44 ST_KL3208OutData

Structure for linking in the output process image.

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

Requirements

Development environment	Target platform	IO Hardware	PLC libraries to include
TwinCAT v3.1.4018.26	PC/CX	KL3208	Tc2_IoFunctions from v3.3.5.0

5.45 ST_KL3228InData

Structure for linking in the input process image.

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

Requirements

Development environment	Target platform	IO Hardware	PLC libraries to include
TwinCAT v3.1.4018.26	PC/CX	KL3228	Tc2_IoFunctions from v3.3.5.0

5.46 ST_KL3228OutData

Structure for linking in the output process image.

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

Requirements

Development environment	Target platform	IO Hardware	PLC libraries to include
TwinCAT v3.1.4018.26	PC/CX	KL3228	Tc2_IoFunctions from v3.3.5.0

6 Global constants

6.1 Library version

All libraries have a certain version. The version is indicated in the PLC library repository, for example. A global constant contains the information about the library version:

Global_Version

```
VAR_GLOBAL CONSTANT
    stLibVersion_Tc2_IoFunctions : ST_LibVersion;
END_VAR
```

stLibVersion_Tc2_IoFunctions: Version number of the Tc2_IoFunctions library (type: ST_LibVersion).

To check whether the version you have is the version you need, use the function F_CmpLibVersion (defined in Tc2_System library).

All other options for comparing library versions, which you may know from TwinCAT 2, are outdated!

Requirements

Development environment	Target platform	IO hardware	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	PC or CX (x86, x64, ARM)	All IO devices	Tc2_IoFunctions (IO)

7 Appendix

7.1 SERCOS file format of the backup file

1. File header of the type ST_SercosFileHeader

2. n * data
a) Parameter header of the type ST_ParamHeader
b) Parameter data as bytes

Example of 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),
kann für jeden Parameter verschieden groß sein, je nach Typ oder
Listenlänge
```

Example of 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
cbCommentLen ( 4 Bytes), i.e. = 00 00 00 00 (= 0)
sComment ( 256 Bytes), i.e. = 00 00 00 00 00 00 00 ... 00
(256 * 00)
1st parameter ST_SercosParamHeader + Data (10 Bytes)
-------------------------------------------------------------------------------
nIDN ( 2 Bytes), i.e. = nnnncbSize( 2 Bytes), i.e. = 02 00 (= 2)nAttrib ( 4 Bytes), i.e. = xx xx xx
xx
arrData     (2 Bytes), i.e. = 12 342nd parameter ST_SercosParamHeader + Data (16 Bytes)

-------------------------------------------------------------------------------
nIDN ( 2 Bytes), i.e. = nnnncbSize( 2 Bytes), i.e. = 08 00 (= 8)nAttrib( 4 Bytes), i.e. = xx xx xx x
x
arrData ( 8 Bytes), i.e. = 12 34 56 78 9a bc de f03rd parameter ST_SercosParamHeader + Data (12 Byte
s)
-------------------------------------------------------------------------------
nIDN ( 2 Bytes), i.e. = nn nncbSize ( 2 Bytes), i.e. = 04 00 (= 4)nAttrib ( 4 Bytes), i.e. = xx xx x
x xx
arrData ( 4 Bytes), i.e. = 12 34 56 78
```

TYPE ST_SercosFileHeader (268 bytes)

The file header of the Sercos backup file is based on the ST_SercosFileHeader structure.

```
TYPE ST_SercosFileHeader :
STRUCT
    nVersion      : UDINT;(* 4 Bytes *)
```

```
    nListType     : UDINT;(* 4 Bytes *)
    cbCommentLen  : UDINT;(* 4 Bytes *)
    sComment      : T_MaxString;(* 256 Bytes *)
END_STRUCT
END_TYPE
```

nVersion: contains the file version, momentarily 1.

nListType: contains the IDN parameter list that was used for the backup. The default value is 192 (list of all backup parameters): the value is 0 in case of user-defined backup list. Alternatively, the list of all Sercos parameters (IDN 17) can be used. The restore, however, requires the list from parameter 192 or the user-defined list (0).

cbCommentLen: Length of the comment of the backup file.

sComment: Comment of the backup file. The string is written with all 256 characters.

TYPE ST_SercosParamHeader (8 bytes)

In the backup file, the file header is followed by a parameter header of the type ST_SercosParamHeader for each parameter.

```
TYPE ST_SercosParamHeader :
STRUCT
    nIDN     : UINT;(* 2 Bytes *)
    cbSize   : UINT;(* 2 Bytes *)
    nAttrib  : DWORD;(* 4 Bytes *)
END_STRUCT
END_TYPE
```

nIDN: Sercos parameter number.

cbSize: Length of the data in bytes that follow this header. Can be different for each parameter, depending on the parameter type or list length.

nAttrib: Attribute of the Sercos parameter (see ST_SercosParamData [▶ 136]); required for the determination of length and data type.

Parameter data (cbSize bytes)

The data directly follow each Sercos parameter header in the backup file. The number of data bytes is saved in the parameter header in cbSize.

Requirements

Development environment	Target platform	IO hardware	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	PC (x86)	Sercans SCS-P ISA; Sercans SCS-P PCI; Beckhoff FC750x PCI	Tc2_IoFunctions (IO)

7.2 AX200x Profibus Parameter Number

PNU	Data type	Access	Short description	SER-VOSTAR™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 identifier	-
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
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

PNU	Data type	Access	Short description	SER-VOSTAR™ASCII command
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	Velocity	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
1307	UINT32	r/w	Jerk limitation (deceleration)	O_DEC2
1308	UINT32	r/w	Number of the subsequent transport instruction	O_FN

PNU	Data type	Access	Short description	SER-VOSTAR™ASCII command
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	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	DC link voltage	VBUS
1411	INTEGER32	ro	Ballast power	PBAL
1412	INTEGER32	ro	I^2t - loading	I2T
1413	INTEGER32	ro	Period of operation	TRUN
Digital I/O-configuration				
1450	UINT32	r/w	Function of digital input 1	IN1MODE
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

PNU	Data type	Access	Short description	SER-VOSTAR™ASCII command
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
Motor parameters				
1550	UINT32	r/w	Brake configuration	MBRAKE
1551	UINT32	r/w	Motor number from the motor database	MNUMBER

7.3 Errorcodes

ErrId (hex)	ErrId (dez)	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 iSetCounterType.
0x8011	32785	FB_KL1501Config: Invalid counter type readouted. See output sCounterType.
0x8018	32792	FB_KL27x1Config: Invalid ramp time selected. See input iSetRampTime.
0x8019	32793	FB_KL27x1Config: Invalid dimmer mode selected. See input iSetDimmerMode.
0x801A	32794	FB_KL27x1Config: Invalid ramp time readouted. See output sRampTime.
0x801B	32795	FB_KL27x1Config: Invalid dimmer mode readouted. See output sDimmerMode.
0x801C	32796	FB_KL27x1Config: Invalid after-shortcut-behaviour readouted. See output sAfterShortCircuit.
0x801D	32797	FB_KL27x1Config: Invalid line frequency readouted. See output sLineFrequency.
0x8020	32800	FB_KL3208Config: Invalid sensor type selected. See input iSetSensorType.
0x8021	32801	FB_KL3208Config: Invalid sensor type readouted. See output sSensorType.
0x8028	32808	FB_KL320xConfig: Two-wired connection is not applicable for a KL3204.
0x8029	32809	FB_KL320xConfig: Invalid sensor type selected. See input iSetSensorType.
0x802A	32810	FB_KL320xConfig: Invalid sensor type readouted. See output sSensorType.
0x8030	32816	FB_KL3228Config: Invalid sensor type selected. See input iSetSensorType.
0x8031	32817	FB_KL3228Config: Invalid sensor type readouted. See output sSensorType.

7.4 ADS Return Codes

Grouping of error codes: 0x000 [▶ 153]..., 0x500 [▶ 154]..., 0x700 [▶ 155]..., 0x1000 [▶ 157]...

Global error codes

Hex	Dec	HRESULT	Name	Description
0x0	0	0x9811 0000	ERR_NOERROR	No error.
0x1	1	0x9811 0001	ERR_INTERNAL	Internal error.
0x2	2	0x9811 0002	ERR_NORTIME	No real-time.
0x3	3	0x9811 0003	ERR_ALLOCLOCKEDMEM	Allocation locked – memory error.
0x4	4	0x9811 0004	ERR_INSERTMAILBOX	Mailbox full – the ADS message could not be sent. Reducing the number of ADS messages per cycle will help.
0x5	5	0x9811 0005	ERR_WRONGRECEIVEHMSG	Wrong HMSG.
0x6	6	0x9811 0006	ERR_TARGETPORTNOTFOUND	Target port not found – ADS server is not started or is not reachable.
0x7	7	0x9811 0007	ERR_TARGETMACHINENOTFOUND	Target computer not found – AMS route was not found.
0x8	8	0x9811 0008	ERR_UNKNOWNCMDID	Unknown command ID.
0x9	9	0x9811 0009	ERR_BADTASKID	Invalid task ID.
0xA	10	0x9811 000A	ERR_NOIO	No IO.
0xB	11	0x9811 000B	ERR_UNKNOWNAMSCMD	Unknown AMS command.
0xC	12	0x9811 000C	ERR_WIN32ERROR	Win32 error.
0xD	13	0x9811 000D	ERR_PORTNOTCONNECTED	Port not connected.
0xE	14	0x9811 000E	ERR_INVALIDAMSLENGTH	Invalid AMS length.
0xF	15	0x9811 000F	ERR_INVALIDAMSNETID	Invalid AMS Net ID.
0x10	16	0x9811 0010	ERR_LOWINSTLEVEL	Installation level is too low –TwinCAT 2 license error.
0x11	17	0x9811 0011	ERR_NODEBUGINTAVAILABLE	No debugging available.
0x12	18	0x9811 0012	ERR_PORTDISABLED	Port disabled – TwinCAT system service not started.
0x13	19	0x9811 0013	ERR_PORTALREADYCONNECTED	Port already connected.
0x14	20	0x9811 0014	ERR_AMSSYNC_W32ERROR	AMS Sync Win32 error.
0x15	21	0x9811 0015	ERR_AMSSYNC_TIMEOUT	AMS Sync Timeout.
0x16	22	0x9811 0016	ERR_AMSSYNC_AMSERROR	AMS Sync error.
0x17	23	0x9811 0017	ERR_AMSSYNC_NOINDEXINMAP	No index map for AMS Sync available.
0x18	24	0x9811 0018	ERR_INVALIDAMSPORT	Invalid AMS port.
0x19	25	0x9811 0019	ERR_NOMEMORY	No memory.
0x1A	26	0x9811 001A	ERR_TCPSEND	TCP send error.
0x1B	27	0x9811 001B	ERR_HOSTUNREACHABLE	Host unreachable.
0x1C	28	0x9811 001C	ERR_INVALIDAMSFRAGMENT	Invalid AMS fragment.
0x1D	29	0x9811 001D	ERR_TLSSEND	TLS send error – secure ADS connection failed.
0x1E	30	0x9811 001E	ERR_ACCESSDENIED	Access denied – secure ADS access denied.

Router error codes

Hex	Dec	HRESULT	Name	Description
0x500	1280	0x9811 0500	ROUTERERR_NOLOCKEDMEMORY	Locked memory cannot be allocated.
0x501	1281	0x9811 0501	ROUTERERR_RESIZEMEMORY	The router memory size could not be changed.
0x502	1282	0x9811 0502	ROUTERERR_MAILBOXFULL	The mailbox has reached the maximum number of possible messages.
0x503	1283	0x9811 0503	ROUTERERR_DEBUGBOXFULL	The Debug mailbox has reached the maximum number of possible messages.
0x504	1284	0x9811 0504	ROUTERERR_UNKNOWNPORTTYPE	The port type is unknown.
0x505	1285	0x9811 0505	ROUTERERR_NOTINITIALIZED	The router is not initialized.
0x506	1286	0x9811 0506	ROUTERERR_PORTALREADYINUSE	The port number is already assigned.
0x507	1287	0x9811 0507	ROUTERERR_NOTREGISTERED	The port is not registered.
0x508	1288	0x9811 0508	ROUTERERR_NOMOREQUEUES	The maximum number of ports has been reached.
0x509	1289	0x9811 0509	ROUTERERR_INVALIDPORT	The port is invalid.
0x50A	1290	0x9811 050A	ROUTERERR_NOTACTIVATED	The router is not active.
0x50B	1291	0x9811 050B	ROUTERERR_FRAGMENTBOXFULL	The mailbox has reached the maximum number for fragmented messages.
0x50C	1292	0x9811 050C	ROUTERERR_FRAGMENTTIMEOUT	A fragment timeout has occurred.
0x50D	1293	0x9811 050D	ROUTERERR_TOBEREMOVED	The port is removed.

General ADS error codes

Hex	Dec	HRESULT	Name	Description
0x700	1792	0x9811 0700	ADSERR_DEVICE_ERROR	General device error.
0x701	1793	0x9811 0701	ADSERR_DEVICE_SRVNOTSUPP	Service is not supported by the server.
0x702	1794	0x9811 0702	ADSERR_DEVICE_INVALIDGRP	Invalid index group.
0x703	1795	0x9811 0703	ADSERR_DEVICE_INVALIDOFFSET	Invalid index offset.
0x704	1796	0x9811 0704	ADSERR_DEVICE_INVALIDACCESS	Reading or writing not permitted.
0x705	1797	0x9811 0705	ADSERR_DEVICE_INVALIDSIZE	Parameter size not correct.
0x706	1798	0x9811 0706	ADSERR_DEVICE_INVALIDDATA	Invalid data values.
0x707	1799	0x9811 0707	ADSERR_DEVICE_NOTREADY	Device is not ready to operate.
0x708	1800	0x9811 0708	ADSERR_DEVICE_BUSY	Device is busy.
0x709	1801	0x9811 0709	ADSERR_DEVICE_INVALIDCONTEXT	Invalid operating system context. This can result from use of ADS function blocks in different tasks. It may be possible to resolve this through multitasking synchronization in the PLC.
0x70A	1802	0x9811 070A	ADSERR_DEVICE_NOMEMORY	Insufficient memory.
0x70B	1803	0x9811 070B	ADSERR_DEVICE_INVALIDPARM	Invalid parameter values.
0x70C	1804	0x9811 070C	ADSERR_DEVICE_NOTFOUND	Not found (files, ...).
0x70D	1805	0x9811 070D	ADSERR_DEVICE_SYNTAX	Syntax error in file or command.
0x70E	1806	0x9811 070E	ADSERR_DEVICE_INCOMPATIBLE	Objects do not match.
0x70F	1807	0x9811 070F	ADSERR_DEVICE_EXISTS	Object already exists.
0x710	1808	0x9811 0710	ADSERR_DEVICE_SYMBOLNOTFOUND	Symbol not found.
0x711	1809	0x9811 0711	ADSERR_DEVICE_SYMBOLVERSIONIN-VALID	Invalid symbol version. This can occur due to an on-line change. Create a new handle.
0x712	1810	0x9811 0712	ADSERR_DEVICE_INVALIDSTATE	Device (server) is in invalid state.
0x713	1811	0x9811 0713	ADSERR_DEVICE_TRANSMODENOTSUPP	AdsTransMode not supported.
0x714	1812	0x9811 0714	ADSERR_DEVICE_NOTIFYHNDINVALID	Notification handle is invalid.
0x715	1813	0x9811 0715	ADSERR_DEVICE_CLIENTUNKNOWN	Notification client not registered.
0x716	1814	0x9811 0716	ADSERR_DEVICE_NOMOREHDLS	No further notification handle available.
0x717	1815	0x9811 0717	ADSERR_DEVICE_INVALIDWATCHSIZE	Notification size too large.
0x718	1816	0x9811 0718	ADSERR_DEVICE_NOTINIT	Device not initialized.
0x719	1817	0x9811 0719	ADSERR_DEVICE_TIMEOUT	Device has a timeout.
0x71A	1818	0x9811 071A	ADSERR_DEVICE_NOINTERFACE	Interface query failed.
0x71B	1819	0x9811 071B	ADSERR_DEVICE_INVALIDINTERFACE	Wrong interface requested.
0x71C	1820	0x9811 071C	ADSERR_DEVICE_INVALIDCLSID	Class ID is invalid.
0x71D	1821	0x9811 071D	ADSERR_DEVICE_INVALIDOBJID	Object ID is invalid.
0x71E	1822	0x9811 071E	ADSERR_DEVICE_PENDING	Request pending.
0x71F	1823	0x9811 071F	ADSERR_DEVICE_ABORTED	Request is aborted.
0x720	1824	0x9811 0720	ADSERR_DEVICE_WARNING	Signal warning.
0x721	1825	0x9811 0721	ADSERR_DEVICE_INVALIDARRAYIDX	Invalid array index.
0x722	1826	0x9811 0722	ADSERR_DEVICE_SYMBOLNOTACTIVE	Symbol not active.
0x723	1827	0x9811 0723	ADSERR_DEVICE_ACCESSDENIED	Access denied.
0x724	1828	0x9811 0724	ADSERR_DEVICE_LICENSENOTFOUND	Missing license.
0x725	1829	0x9811 0725	ADSERR_DEVICE_LICENSEEXPIRED	License expired.
0x726	1830	0x9811 0726	ADSERR_DEVICE_LICENSEEXCEEDED	License exceeded.
0x727	1831	0x9811 0727	ADSERR_DEVICE_LICENSEINVALID	Invalid license.
0x728	1832	0x9811 0728	ADSERR_DEVICE_LICENSESYSTEMID	License problem: System ID is invalid.
0x729	1833	0x9811 0729	ADSERR_DEVICE_LICENSENOTIMELIMIT	License not limited in time.
0x72A	1834	0x9811 072A	ADSERR_DEVICE_LICENSEFUTUREISSUE	License problem: Time in the future.
0x72B	1835	0x9811 072B	ADSERR_DEVICE_LICENSETIMETOLONG	License period too long.
0x72C	1836	0x9811 072C	ADSERR_DEVICE_EXCEPTION	Exception at system startup.
0x72D	1837	0x9811 072D	ADSERR_DEVICE_LICENSEDUPLICATED	License file read twice.
0x72E	1838	0x9811 072E	ADSERR_DEVICE_SIGNATUREINVALID	Invalid signature.
0x72F	1839	0x9811 072F	ADSERR_DEVICE_CERTIFICATEINVALID	Invalid certificate.
0x730	1840	0x9811 0730	ADSERR_DEVICE_LICENSEOEMNOT-FOUND	Public key not known from OEM.
0x731	1841	0x9811 0731	ADSERR_DEVICE_LICENSERESTRICTED	License not valid for this system ID.
0x732	1842	0x9811 0732	ADSERR_DEVICE_LICENSEDEMODENIED	Demo license prohibited.
0x733	1843	0x9811 0733	ADSERR_DEVICE_INVALIDFNCID	Invalid function ID.
0x734	1844	0x9811 0734	ADSERR_DEVICE_OUTOFRANGE	Outside the valid range.
0x735	1845	0x9811 0735	ADSERR_DEVICE_INVALIDALIGNMENT	Invalid alignment.

Hex	Dec	HRESULT	Name	Description
0x736	1846	0x9811 0736	ADSERR_DEVICE_LICENSEPLATFORM	Invalid platform level.
0x737	1847	0x9811 0737	ADSERR_DEVICE_FORWARD_PL	Context – forward to passive level.
0x738	1848	0x9811 0738	ADSERR_DEVICE_FORWARD_DL	Context – forward to dispatch level.
0x739	1849	0x9811 0739	ADSERR_DEVICE_FORWARD_RT	Context – forward to real-time.
0x740	1856	0x9811 0740	ADSERR_CLIENT_ERROR	Client error.
0x741	1857	0x9811 0741	ADSERR_CLIENT_INVALIDPARM	Service contains an invalid parameter.
0x742	1858	0x9811 0742	ADSERR_CLIENT_LISTEMPTY	Polling list is empty.
0x743	1859	0x9811 0743	ADSERR_CLIENT_VARUSED	Var connection already in use.
0x744	1860	0x9811 0744	ADSERR_CLIENT_DUPLINVOKEID	The called ID is already in use.
0x745	1861	0x9811 0745	ADSERR_CLIENT_SYNCTIMEOUT	Timeout has occurred – the remote terminal is not responding in the specified ADS timeout. The route setting of the remote terminal may be configured incorrectly.
0x746	1862	0x9811 0746	ADSERR_CLIENT_W32ERROR	Error in Win32 subsystem.
0x747	1863	0x9811 0747	ADSERR_CLIENT_TIMEOUTINVALID	Invalid client timeout value.
0x748	1864	0x9811 0748	ADSERR_CLIENT_PORTNOTOPEN	Port not open.
0x749	1865	0x9811 0749	ADSERR_CLIENT_NOAMSADDR	No AMS address.
0x750	1872	0x9811 0750	ADSERR_CLIENT_SYNCINTERNAL	Internal error in Ads sync.
0x751	1873	0x9811 0751	ADSERR_CLIENT_ADDHASH	Hash table overflow.
0x752	1874	0x9811 0752	ADSERR_CLIENT_REMOVEHASH	Key not found in the table.
0x753	1875	0x9811 0753	ADSERR_CLIENT_NOMORESYM	No symbols in the cache.
0x754	1876	0x9811 0754	ADSERR_CLIENT_SYNCRESINVALID	Invalid response received.
0x755	1877	0x9811 0755	ADSERR_CLIENT_SYNCPORTLOCKED	Sync Port is locked.

RTime error codes

Hex	Dec	HRESULT	Name	Description
0x1000	4096	0x9811 1000	RTERR_INTERNAL	Internal error in the real-time system.
0x1001	4097	0x9811 1001	RTERR_BADTIMERPERIODS	Timer value is not valid.
0x1002	4098	0x9811 1002	RTERR_INVALIDTASKPTR	Task pointer has the invalid value 0 (zero).
0x1003	4099	0x9811 1003	RTERR_INVALIDSTACKPTR	Stack pointer has the invalid value 0 (zero).
0x1004	4100	0x9811 1004	RTERR_PRIOEXISTS	The request task priority is already assigned.
0x1005	4101	0x9811 1005	RTERR_NOMORETCB	No free TCB (Task Control Block) available. The maximum number of TCBs is 64.
0x1006	4102	0x9811 1006	RTERR_NOMORESEMAS	No free semaphores available. The maximum number of semaphores is 64.
0x1007	4103	0x9811 1007	RTERR_NOMOREQUEUES	No free space available in the queue. The maximum number of positions in the queue is 64.
0x100D	4109	0x9811 100D	RTERR_EXTIRQALREADYDEF	An external synchronization interrupt is already applied.
0x100E	4110	0x9811 100E	RTERR_EXTIRQNOTDEF	No external sync interrupt applied.
0x100F	4111	0x9811 100F	RTERR_EXTIRQINSTALLFAILED	Application of the external synchronization interrupt has failed.
0x1010	4112	0x9811 1010	RTERR_IRQLNOTLESSOREQUAL	Call of a service function in the wrong context
0x1017	4119	0x9811 1017	RTERR_VMXNOTSUPPORTED	Intel VT-x extension is not supported.
0x1018	4120	0x9811 1018	RTERR_VMXDISABLED	Intel VT-x extension is not enabled in the BIOS.
0x1019	4121	0x9811 1019	RTERR_VMXCONTROLSMISSING	Missing function in Intel VT-x extension.
0x101A	4122	0x9811 101A	RTERR_VMXENABLEFAILS	Activation of Intel VT-x fails.

TCP Winsock error codes

Hex	Dec	Name	Description
0x274C	10060	WSAETIMEDOUT	A connection timeout has occurred - error while establishing the connection, because the remote terminal did not respond properly after a certain period of time, or the established connection could not be maintained because the connected host did not respond.
0x274D	10061	WSAECONNREFUSED	Connection refused - no connection could be established because the target computer has explicitly rejected it. This error usually results from an attempt to connect to a service that is inactive on the external host, that is, a service for which no server application is running.
0x2751	10065	WSAEHOSTUNREACH	No route to host - a socket operation referred to an unavailable host.
More Winsock error codes: Win32 error codes			

More Information:
www.beckhoff.com/te1000

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