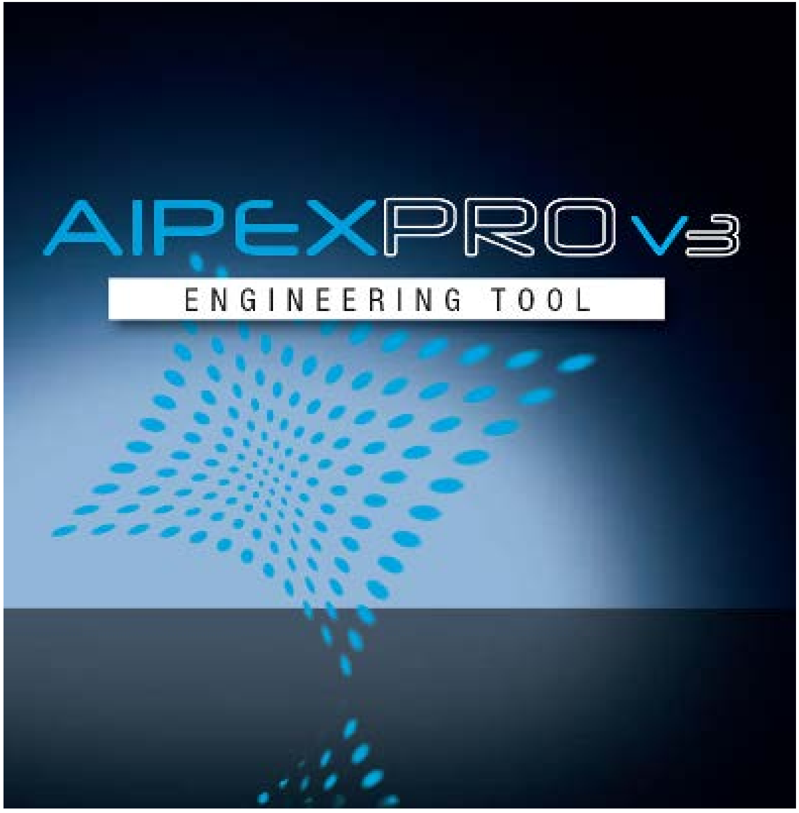


AMKASYN
Software description
AmkLibraries
IEC 61131-3 Library for PLC programming with CODESYS V3

Version: 2023/28
Part no.: 205210
Translation of the "Original Dokumentation"

AMKmotion
MEMBER OF THE ARBURG FAMILY

Imprint

Name: PDK_205210_V3_AmkLibraries_en

Version:

Version: 2023/28	
Change	**Letter symbol**
• AMKmotion Design • FiFileConnect - note 'SMBv1' network protocol added	LeS/KoJ

Previous version: 2019/45

Product version:

Product	Firmware Version (Part no.)
AIPEX PRO	≥ V3.04 (206872)

Reservation: We reserve the right to modify the content of the documentation as well as the delivery options for the product.

Publisher: AMKmotion GmbH + Co KG
Gaußstraße 37-39
73230 Kirchheim unter Teck
Germany

Phone +49 7021 50 05-0
Fax +49 7021 50 05-176

E-mail: info@amk-motion.com

Registration court: AG Stuttgart, HRA 230681, Kirchheim unter Teck,
Tax Id no.: DE 145 912 804

Complementary: AMKmotion Verwaltungsgesellschaft mbH, HRB 774646

Service: Phone +49 7021 50 05-190, Fax -193

For fast and reliable troubleshooting, you can help us by informing our Customer Service about the following:

- Type plate data for each unit
- Software version
- Device configuration and application
- Type of fault/problem and suspected cause
- Diagnostic messages (error messages)

E-mail service@amk-motion.com

Internet address: www.amk-motion.com

Content

1 Organization of the basic system libraries

The following basic system libraries are components of the AIPEX PRO (CODESYS V3) installation:

AmkBase	Base function specific to AMK	PDK_204986_V3_AmkBase
AmkSupport	Support functions specific to AMK	PDK_205002_V3_AmkSupport
AmkSystem	System functions specific to AMK	PDK_205004_V3_AmkSystem
AmkTabc	Table calculation blocks specific to AMK	PDK_205003_V3_AmkTabc
AmkCamEditor	Type definition specific to 3S	PDK_205008_V3_AmkCamEditor
AmkCom	Communication interface specific to AMK	PDK_205010_V3_AmkCom
AmkPmc	AmkPmc - Printing mark control specific to AMK	PDK_205009_V3_AmkPmc
AmkBaseElems	Base visualization function specific to AMK	PDK_109902_V3_AmkBaseElems
AmkDevAccBase	Base device access function specific to AMK	PDK_109904_V3_AmkDevAccBase
AmkDevAccess	Device access function specific to AMK	PDK_109903_V3_AmkDevAccess
AmkEasyDev	Simplified device interface	PDK_205150_V3_AmkEasyDev
AmkFile	File function specific to AMK	PDK_205144_V3_AmkFile
AmkSockets	Ethernet socket functions specific to AMK	PDK_205183_V3_AmkSockets
AmkTcp	Communication interface specific to AMK	PDK_205151_V3_AmkTcp
AmkUdp	UDP communication interface specific to AMK	PDK_205152_V3_AmkUdp
AmkSm3Drive	Sm3Drive blocks specific to AMK	PDK_205458_V3_AmkSm3Drive

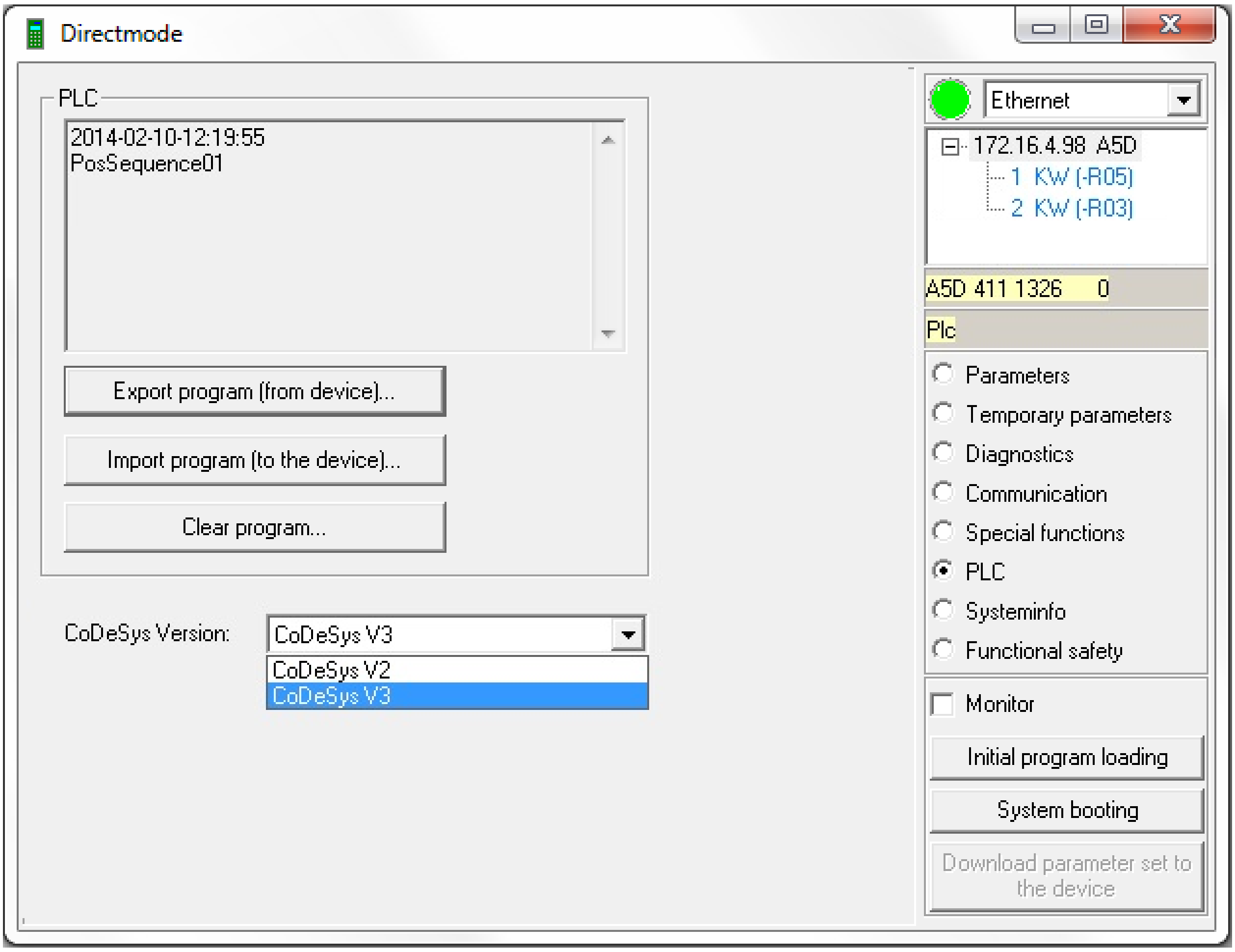
The blocks in the AFL application library (AMK Function Library, AMK part no. O913) are an extension of the basic system libraries. Available to purchase as optional extras, they carry out many tasks on the application programmer's behalf and speed up the application implementation process.

2 AMK control technology

All AMK controllers are supported by the AMK AIPEX PRO tool. From AIPEX PRO Version 3.00, the PLC function of AMK controllers can be programmed with CODESYS V3 by 3S Smart Software Solutions GmbH in IEC 61131.

AIPEX PRO "direct mode" is used to specify whether a controller that supports CODESYS V3 is to operate with CODESYS V2 or CODESYS V3 (see Figure 1).

Abbildung 1: CODESYS version selection



Alongside full "CODESYS V3" programming system scope, the user also has access to various AMK-specific libraries customized for drive functionality (see Table 1). The functional scope of these libraries essentially corresponds to that of the libraries embedded in CODESYS V2.3 which support automatic bus configuration. This means that it is relatively easy to convert existing CODESYS V2 projects into CODESYS V3 projects.

The current version of AIPEX PRO supports both CAN-based bus configuration (ACC = AMK CAN communication) and EtherCAT bus configuration in the context of automatic bus configuration.

Based on the respective controller, AIPEX PRO can be used to create sample projects (templates) for specific target systems which provide the starting point for a new CODESYS V3 project and the basis of automatic bus configuration.

AIPEX PRO V3 thus supports:

- The configuration of a device topology with all components that can be accessed via ACC or EtherCAT (controllers, drives, I/O modules, etc.).
- The programming of controllers with CODESYS V3 using AMK libraries.
- Functional access (via AMK function blocks) to all components that can be accessed via the buses.
- Communication (synchronous/asynchronous) between AMK PLC modules (via AMK function blocks).
- Automatic generation of the information required for bus communication on this basis.

Table 1:
Library overview of the AMK basic modules

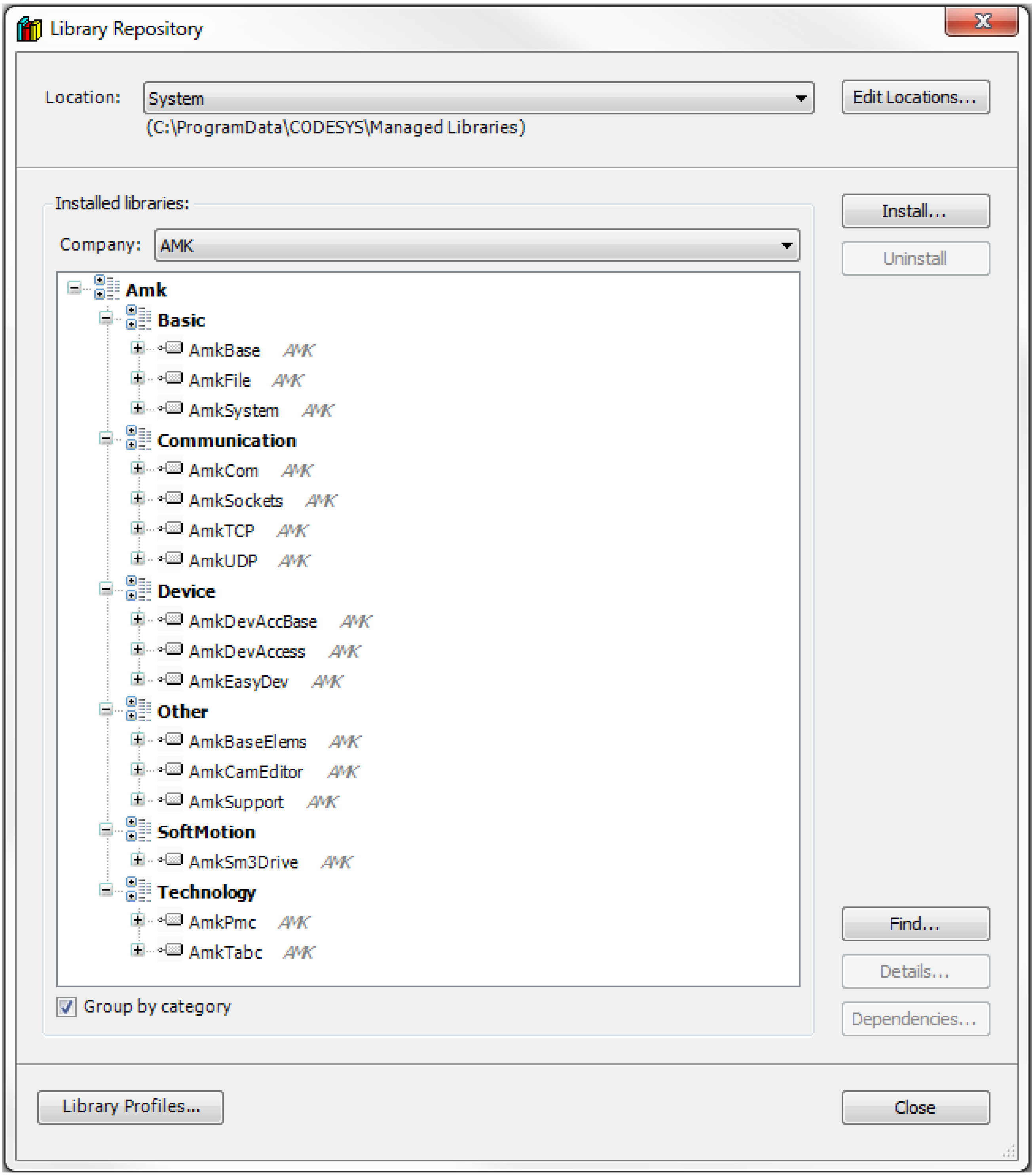
Topic	<Name>.library	Impl.1)	Lib.2)	Place 3)	Note
Basic	AmkBase	E	C	G	Base function
	AmkFile	E	C	G	File functions
	AmkSystem	I	C	B	System functionality
Communication	AmkCom	E	C	G	Communication functionality
	AmkSockets	E	C	G	Ethernet socket functions
	AmkTcp	I	C	B	TCP communication interface
	AmkUdp	I	C	B	UDP communication interface
Device	AmkDevAccBase	I	S	B	Base device access functionality
	AmkDevAccess	I	S	B	Device access functionality
	AmkEasyDev	I	S	B	Simplified AMK device interface
Other	AmkBaseElems	I	C	B	Basic visualization elements
	AmkCamEditor	I	S	B	CamEditor specific type definitions
	AmkSupport	I	C	B	Support of special hardware/technologies
SoftMotion	AmkSm3Drive	I	S	B	AMK Softmotion drive interface
Technology	AmkPmc	I	C	B	Register mark controller functionality
	AmkTabc	I	C	B	Table calculation blocks

1) Implementation: E = external/I = internal
    - External: implemented as a system component, programmed in 'C'.
    - Internal: implemented as an IEC program.
2) Library implementation: C = as compiled library/S = as source library
   A library implementation in the form of a compiled library or a source library.
    - Compiled libraries are more code efficient but cannot be analyzed in the source text. For this reason, it is not possible to 'step' into the libraries for test purposes.
    - Source libraries can be analyzed in test mode like the program (breakpoints, step-by-step operation, etc.).
3) Placeholder implementation: G = based on device description, B = based on library profile
   Placeholder implementation based on a device description or a library profile.
    - With 'placeholder description based on a device description', version resolution of the library takes place in the controller device description (controller version).
    - With 'placeholder description based on a library profile', version resolution of the library is based on the compiler version in CODESYS.

In this context, a distinction is made between:
- external and internal implementation.
    - External: implemented as a system component, programmed in "C".
    - Internal: implemented as an IEC program.
- A library implementation in the form of a "compiled library" or a source text library.
    - Compiled libraries are more code efficient but cannot be analyzed in the source text. For this reason, it is not possible to "step" into the libraries for test purposes.
    - Source text libraries can be analyzed in test mode like the program (breakpoints, step-by-step operation, etc.).
- Placeholder implementation based on a device description or a library profile.
    - With "placeholder description based on a device description", version resolution of the library takes place in the controller device description (controller version).
    - With "placeholder description based on a library profile", version resolution of the library is based on the compiler version in CODESYS.

In CODESYS V3, the AMK libraries are selected in the library manager. Figure 2 shows the reduced view listing libraries by AMK only.

Figure 2: Library selection



In AIPEX PRO, CODESYS V3 is installed as a largely standalone package.

However, in order to be able to use the automatic bus configuration, PLC projects (templates) must be created in AIPEX PRO Siehe 'Creating a PLC project' auf Seite 17.

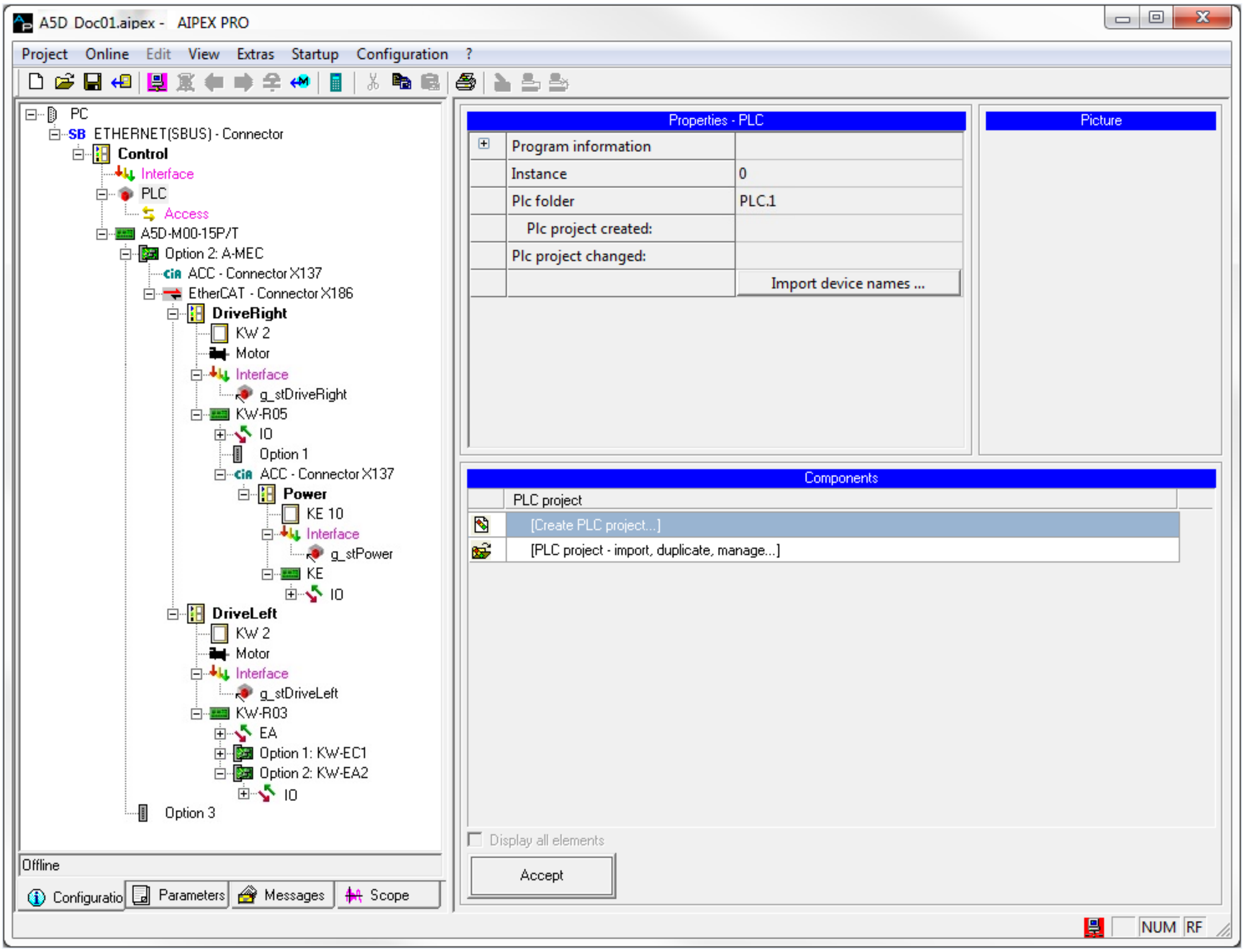
3 Working with CODESYS V3 in AIPEX PRO

3.1 Creating a PLC project

To create a new CODESYS project, simply:

1. Select the "PLC" in the device tree in AIPEX PRO.
2. Create a new project by selecting "Create PLC project" in the component window (see Figure 3; double-click) or select "Import / Open PLC project" to open an existing CODESYS V3 project (a project generated in AIPEX PRO).

Figure 3: Create / open PLC project



At the start of the process to create a new CODESYS V3 project, for example, the user is prompted to enter a project name (see Figure 4). Next, a device handle can be imported into the device tree of the new CODESYS V3 project with "Import device names" (see Figure 5), based on the device names of the AIPEX PRO project (see Figure 6).

Figure 4: Enter project name

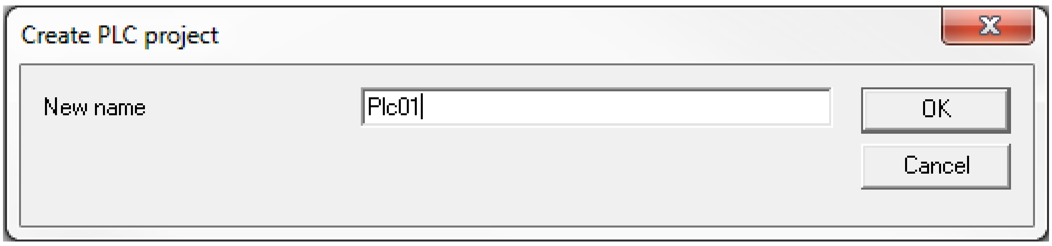
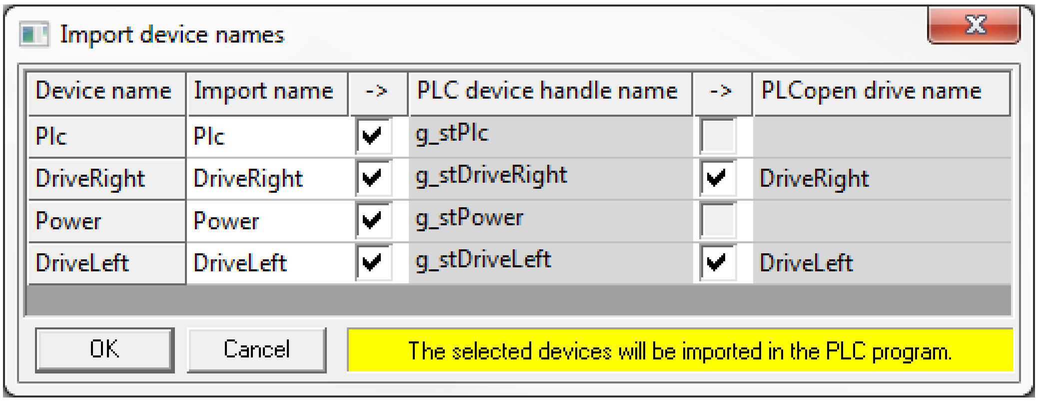


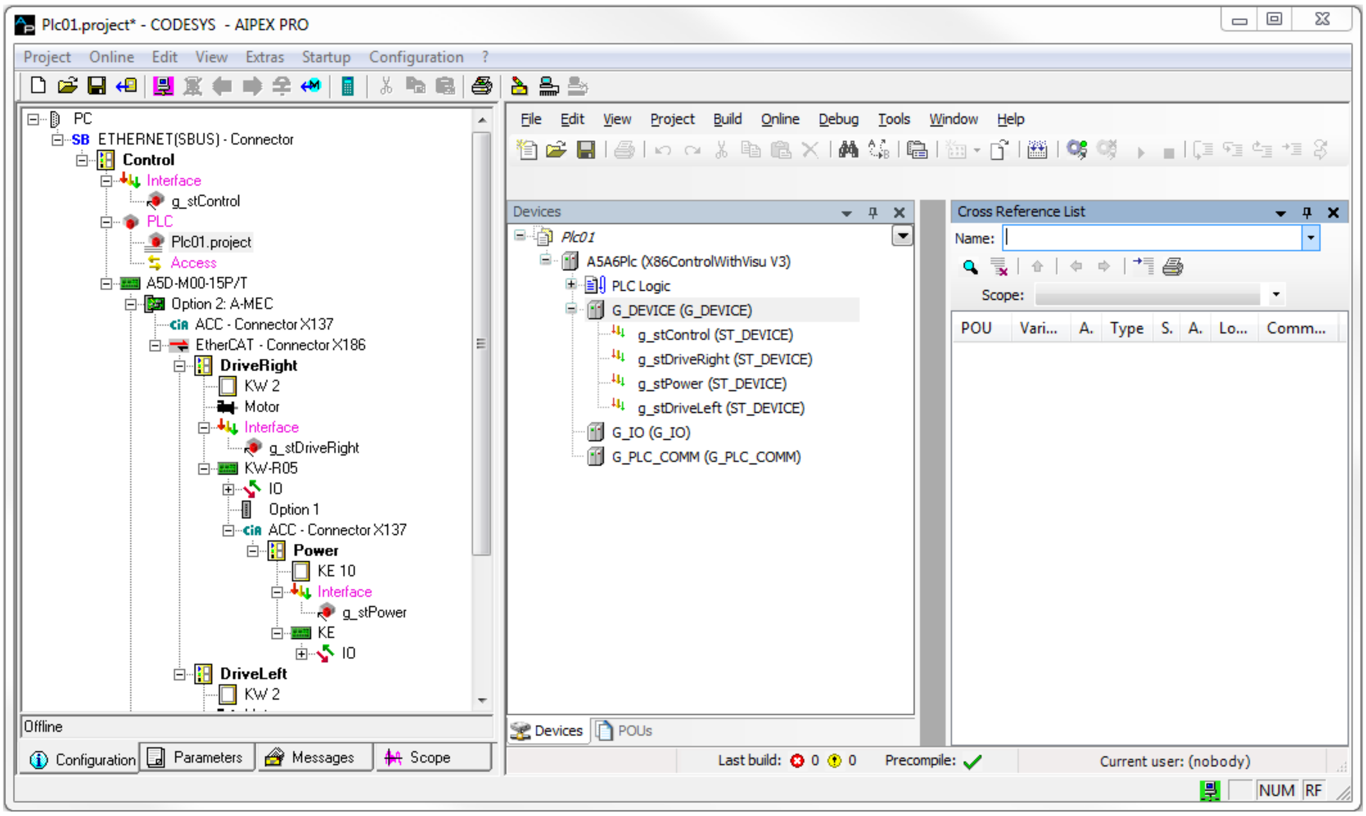
Figure 5: Import device names



Import device names

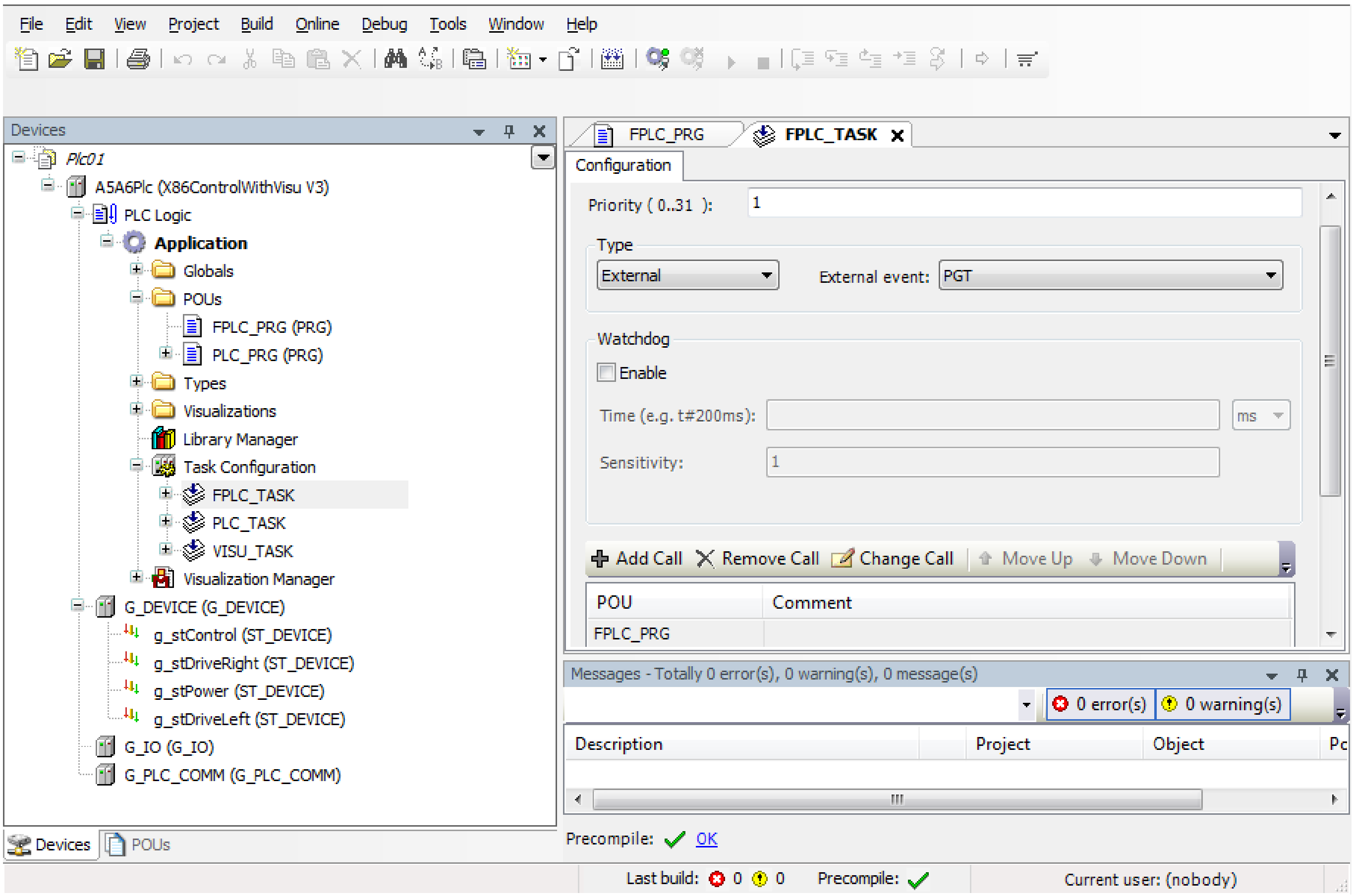
Device name	Import name	->	PLC device handle name	->	PLCopen drive name
Plc	Plc	☑	g_stPlc	☐	
DriveRight	DriveRight	☑	g_stDriveRight	☑	DriveRight
Power	Power	☑	g_stPower	☐	
DriveLeft	DriveLeft	☑	g_stDriveLeft	☑	DriveLeft

Figure 6: Device tree of the CODESYS V3 project



When the project is created, the target system of the "X86ControlWithVisu V3" controller is set (see Section 1.11) and a default task configuration is created (see Figure 7) according to the project template of the selected device (e.g. AxD-MC0-15T; see Figure 6).

Figure 7: Default task configuration



3.2 Controller configuration

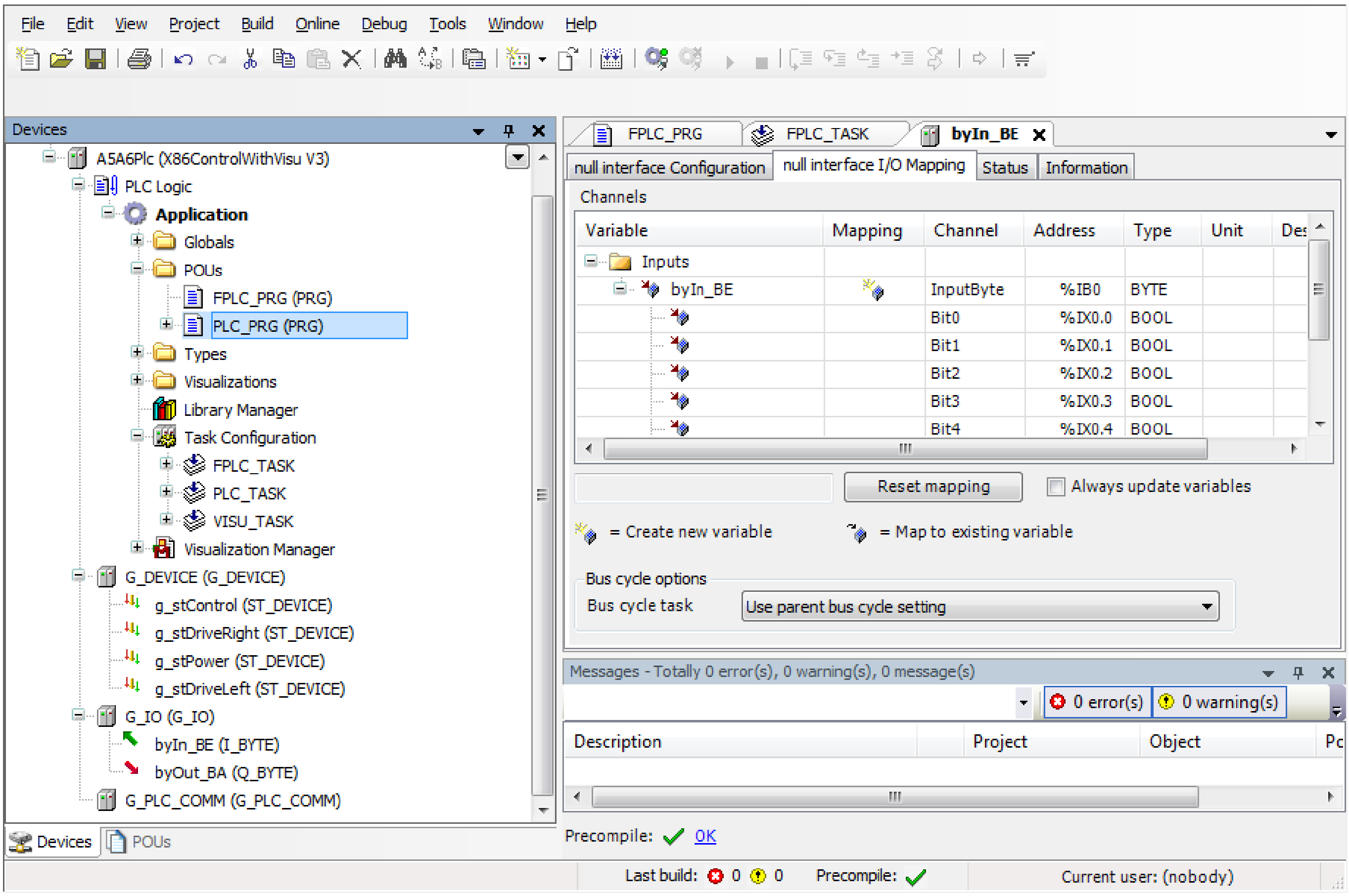
In the controller configuration, a fundamental distinction is made between the following configuration options (see Figure 8):

- G_DEVICE: global devices (to access device information from drives, power supply module, or controllers, for example),
- G_IO: global IO modules (to access binary inputs/outputs, for example),
- G_PLC_COMM: global PLC-PLC communication variable (for communication between PLCs).

Figure 8 illustrates a simply example for access to 4 devices, for example, with one binary input byte and one binary output byte. It is then possible to work in the PLC program with the handles created here (variable names: 'g_stPlc', 'g_stDriveRight', 'g_stPower', 'g_stDriveLeft', 'byIn_BE', 'byOut_BA').

Handles are assigned to the physical devices entirely independently of the programming during the bus configuration process which takes place automatically when the project is compiled (see Figure 15).

Figure 8: Controller configuration with devices and IO modules added



3.3 Library management

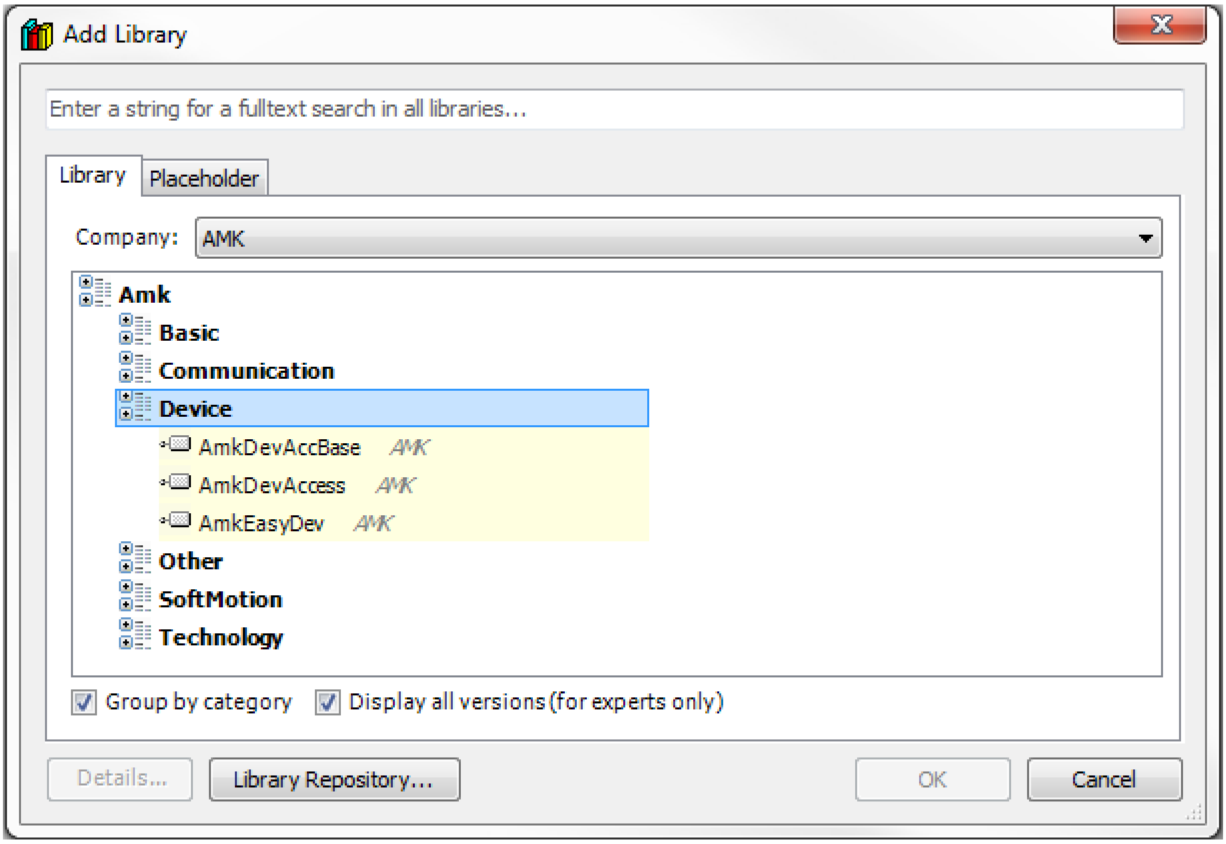
The next stage of the process is to select the required bus access blocks, for example. For automatic configuration, the following libraries are of primary importance:

- AmkDevAccess
- AmkEasyDev

bereitgestellt.

In the library selection, they are listed under "Company: AMK; Amk->Device" (see Figure 9).

Figure 9: Library selection



- "AmkDevAccess" provides a variety of blocks for basic device information (drive information, for example).
- "AmkEasyDev", on the other hand, provides more complex mechanisms for accessing devices (drives, power supply modules, controllers) which in turn are based on basic blocks from "AmkDevAccess".

The AmkDevAccess library is a component of the template project. If required, the AmkEasyDev library can be added specifically via the library manager.

Figure 10 shows how to add AmkEasyDev to the project with a placeholder. Figure 11 shows the library embedded in the project. The library version is resolved to 3.5.3.0 with the "AmkEasyDev" placeholder.

Figure 10: Add placeholder library

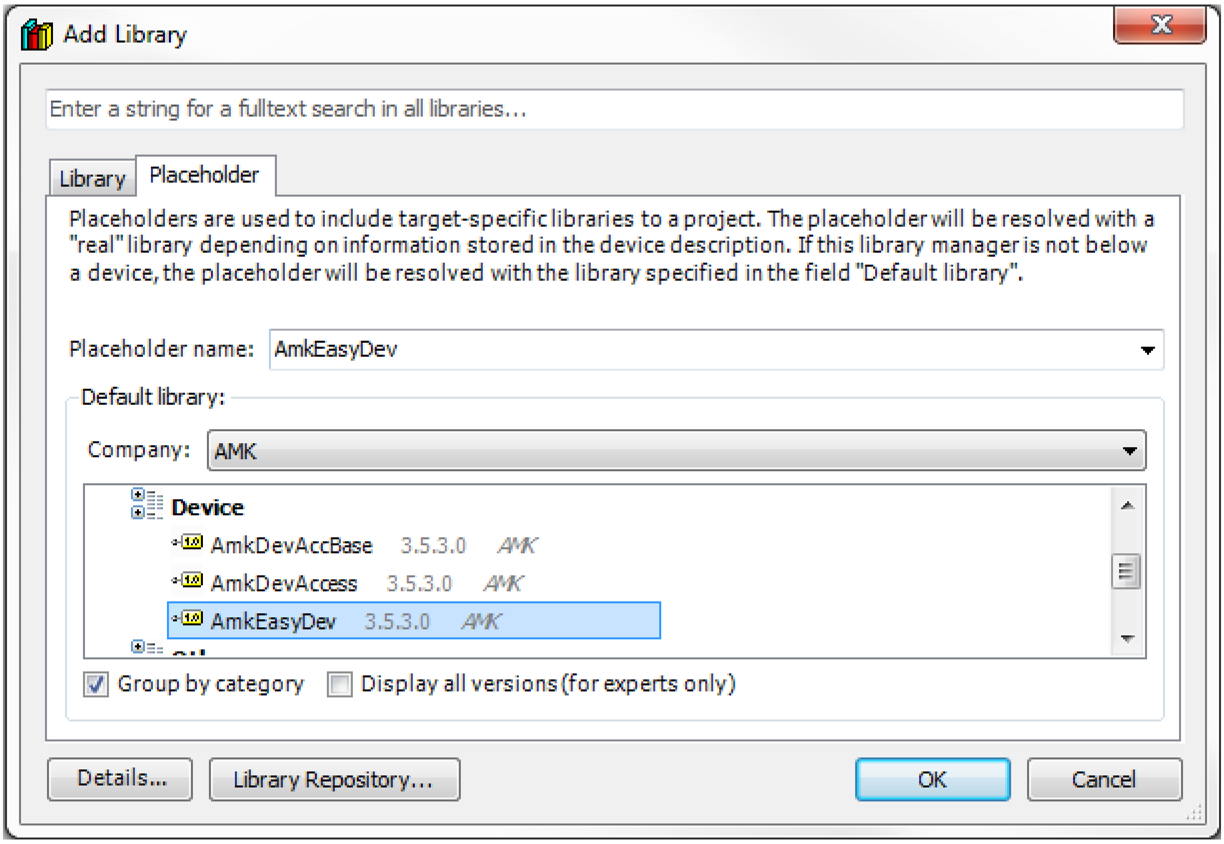


Figure 11: Add AmkEasyDev with the library manager

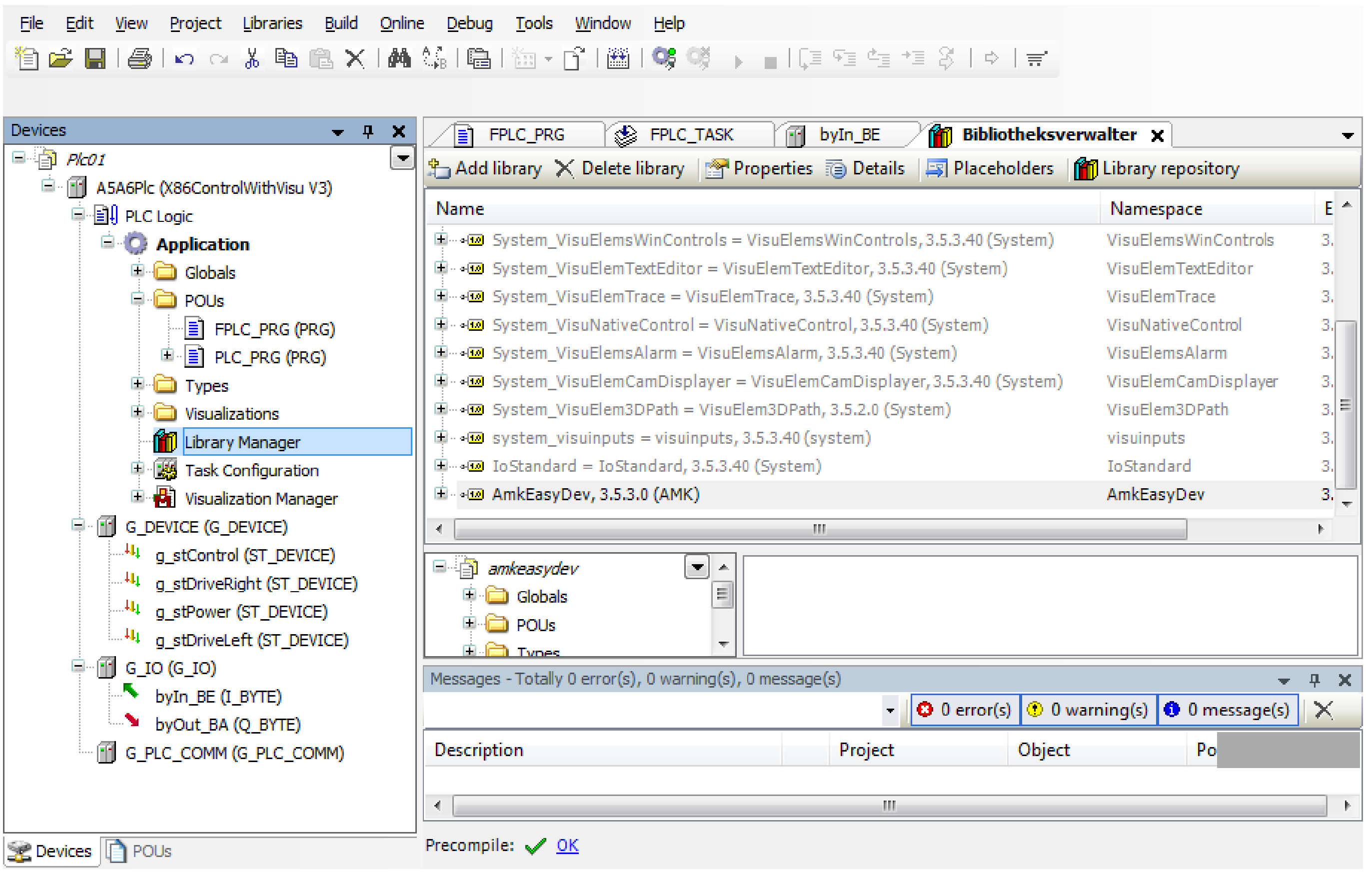
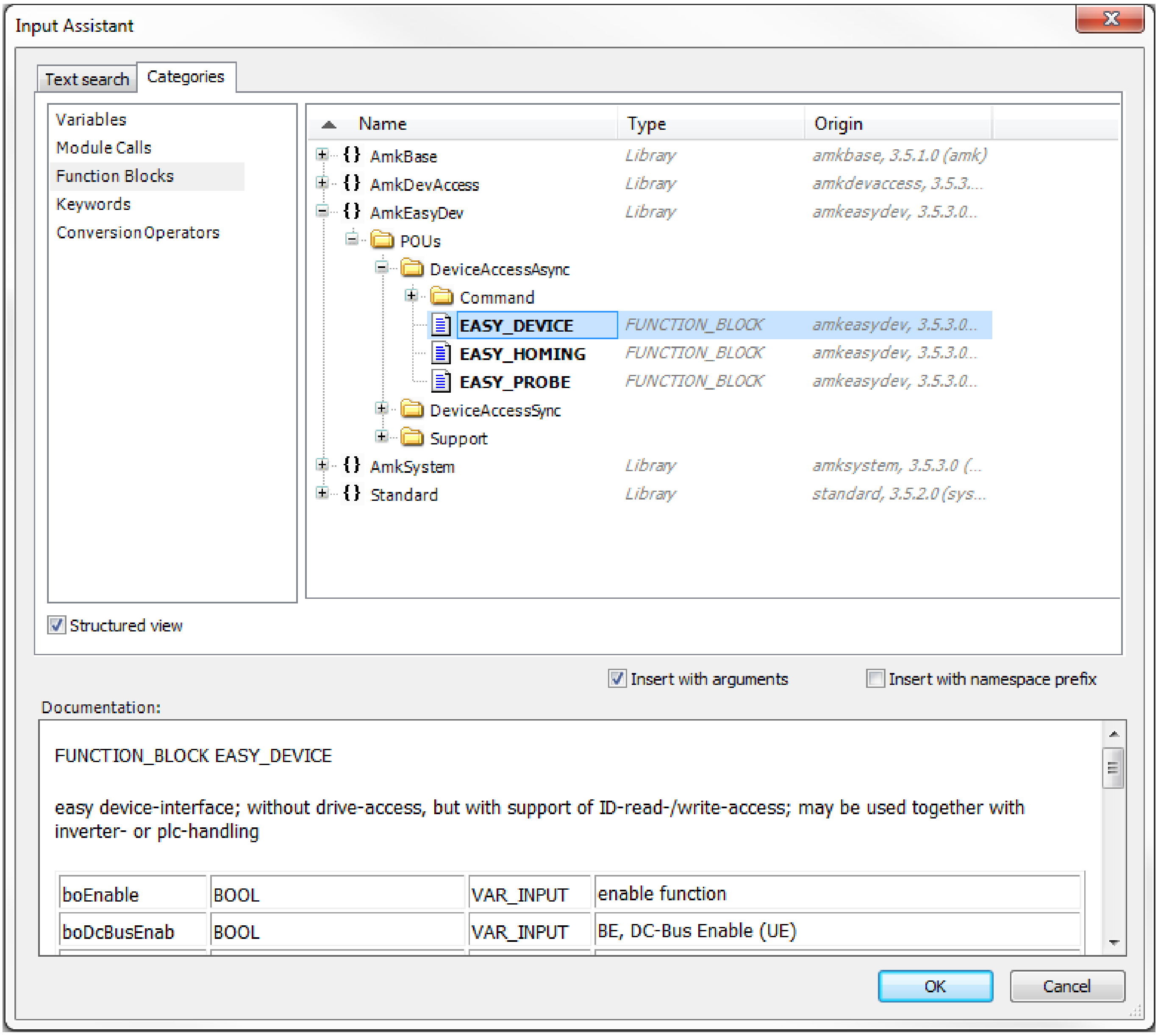


Figure 12 below shows the selection, e.g. of the 'EASY_DEVICE' block from the "AmkEasyDev" library.

Figure 12: Input assistance



3.4 Programming

The definition of 2 instances of the 'EASY_DEVICE' type is shown in Figure 14. These type instances are integrated into the "PLC_PRG" program, for example (using the input assistance shown in Figure 12 or the variables declaration shown in Figure 13); each one is linked to a handle defined in the controller configuration ('g_stDriveLeft', 'g_stPlc'). The IN_OUT variable 'stDevice' for these blocks is used for this purpose (see Figure 14).

Figure 13: Declare variable

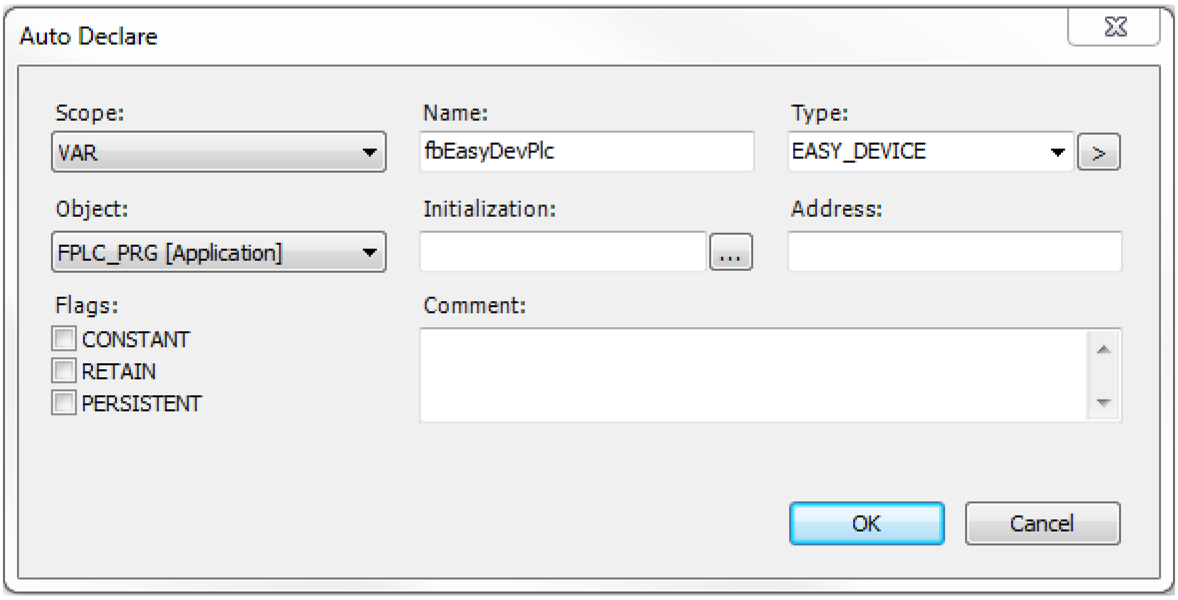
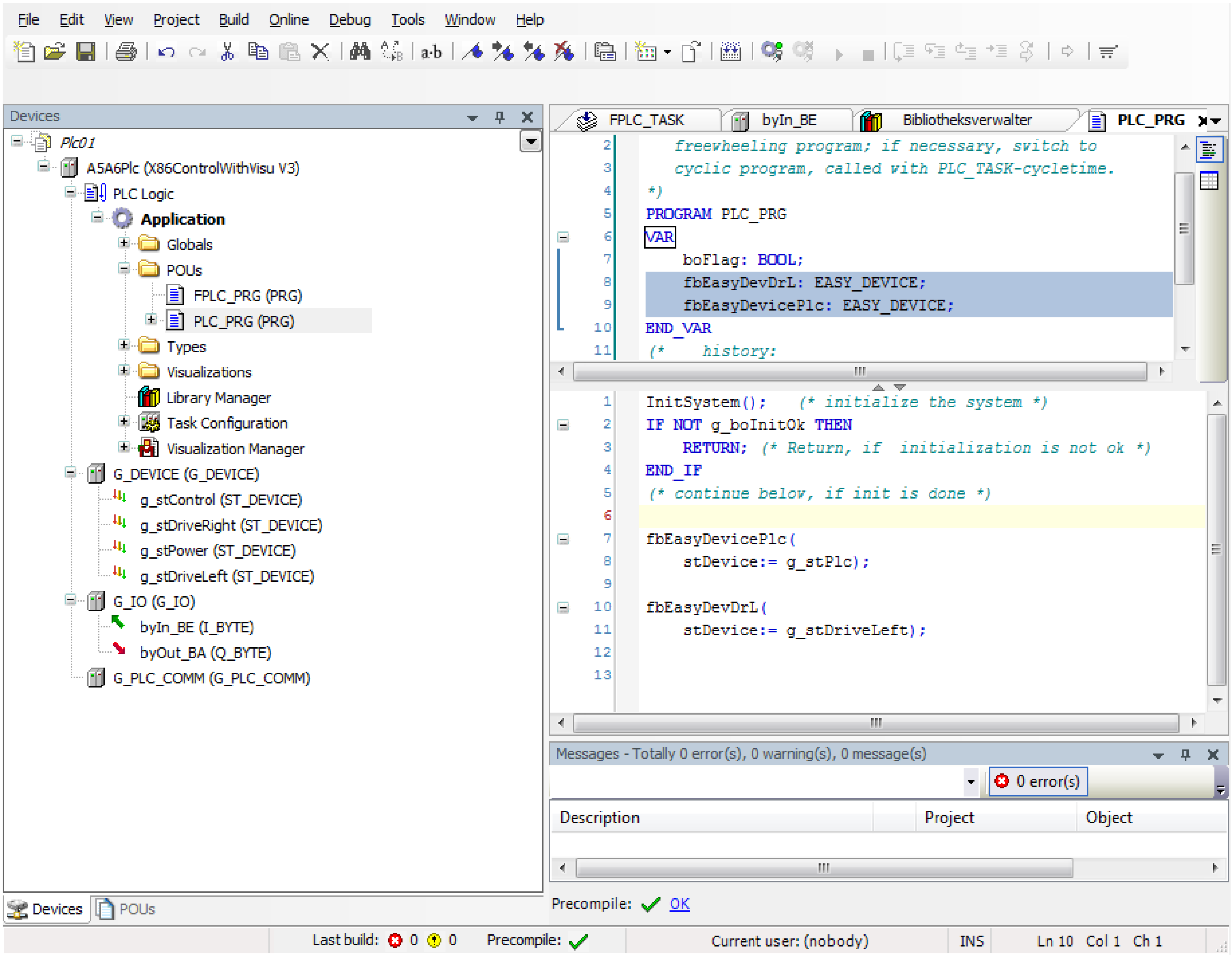


Figure 14: Default program "PLC_PRG"



3.5 Bus configuration

Handles are assigned to the physical devices during the bus configuration process, which is started by selecting "Create configuration" from the menu (see Figure 15).

The device and IO variables from the controller configuration that are used in the PLC program are analyzed, the device and IO information linked to these variables is derived, and "Plc01.project" is displayed in the assignment window (see Figure 16). Next, these variables can be assigned to the corresponding locations in the device tree (interface, or IO subelement) with the mouse (drag & drop). When the "Done" button is pressed, the necessary information for the corresponding bus configuration and the CODESYS V3 project is generated (see Figure 17).

Figure 15: Bus configuration

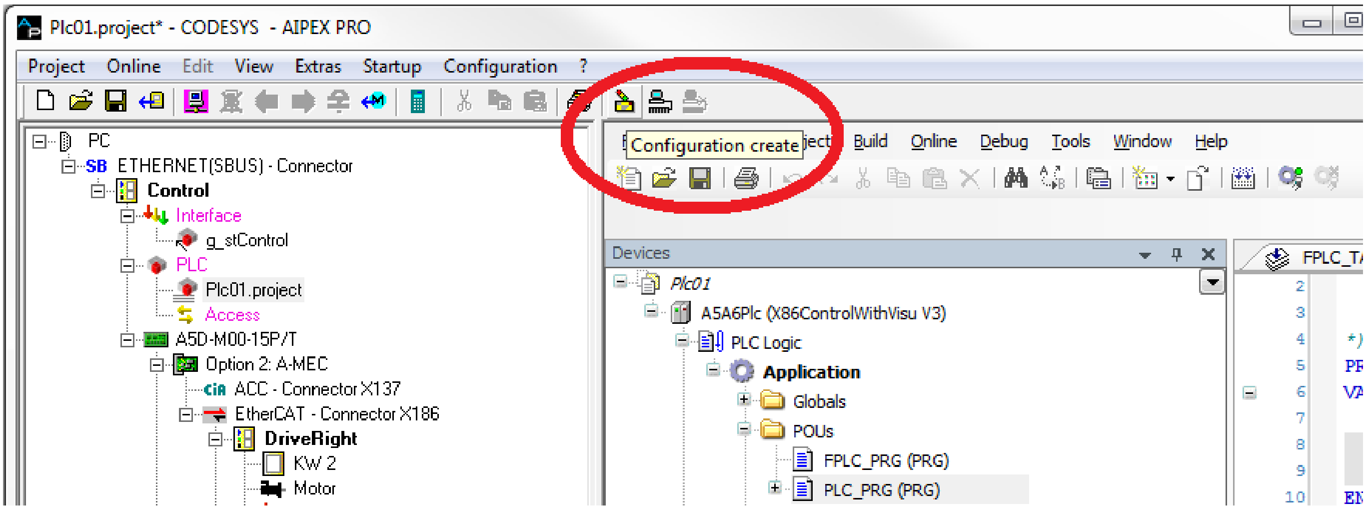


Figure 16: Device assignment window

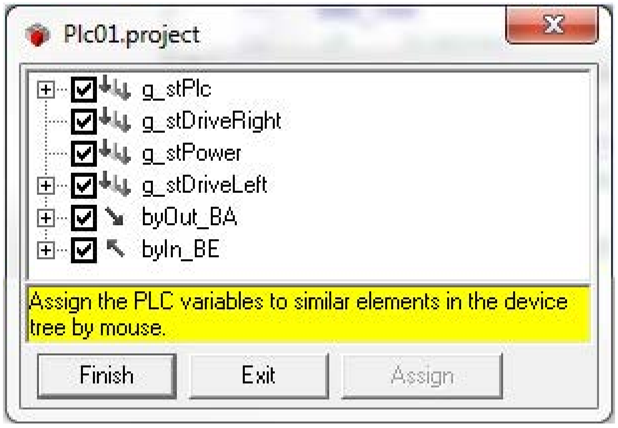
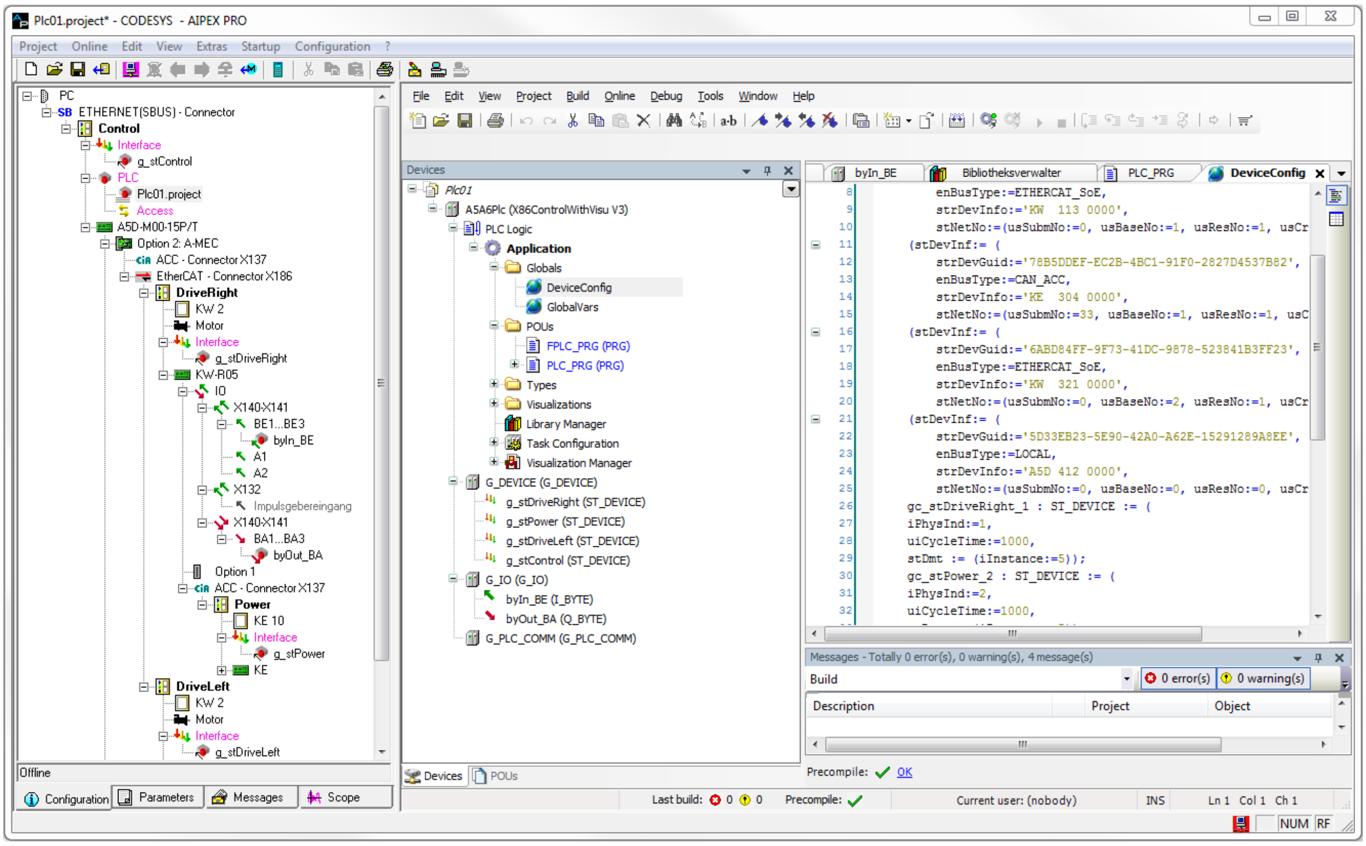


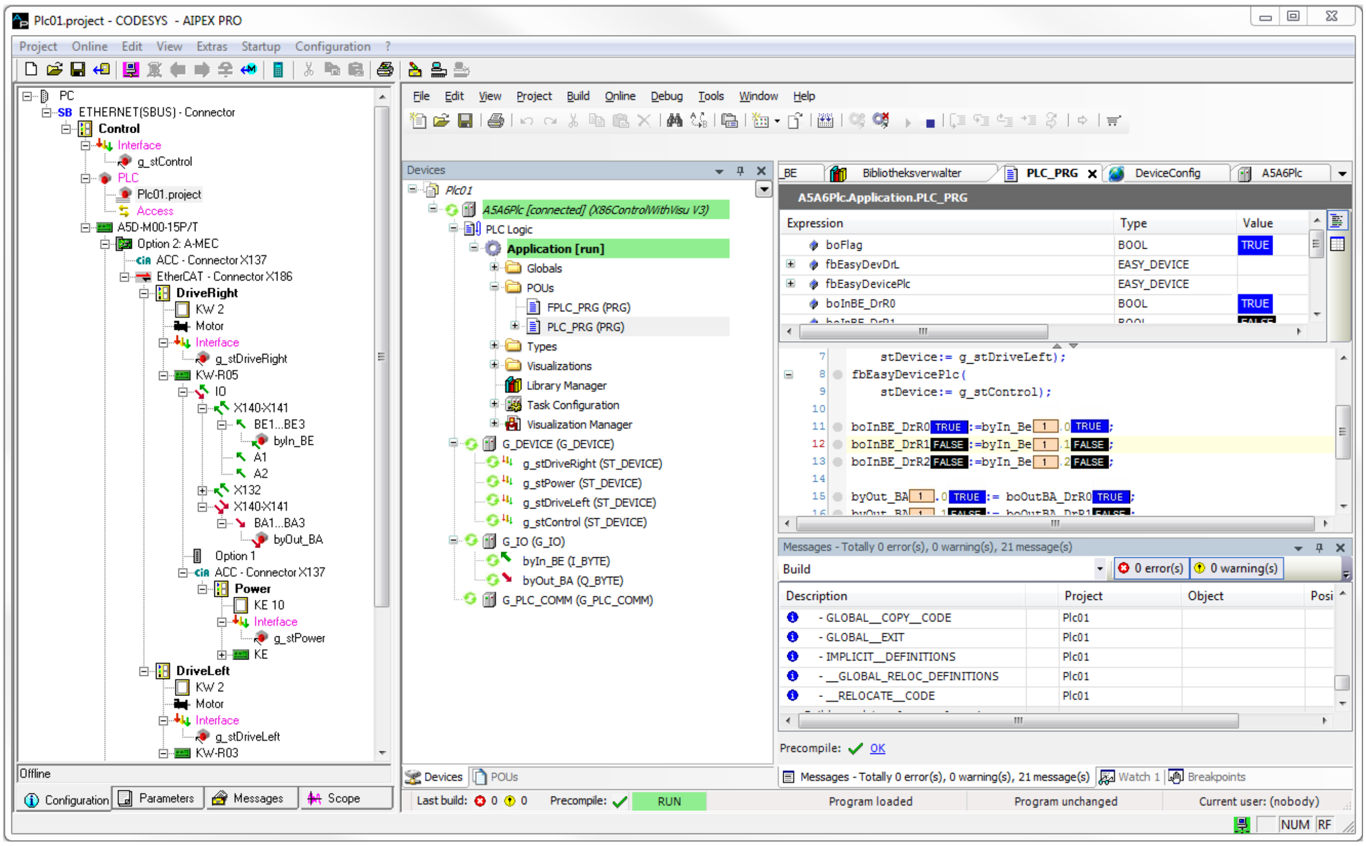
Figure 17: Device assignment and generated configuration



If AIPEX PRO is connected to the controller "online" and the "active path" in the communication settings for CODESYS also matches the controller (see Figure 18), the bus configuration and the CODESYS project are transferred to the controller automatically and the system is rebooted.

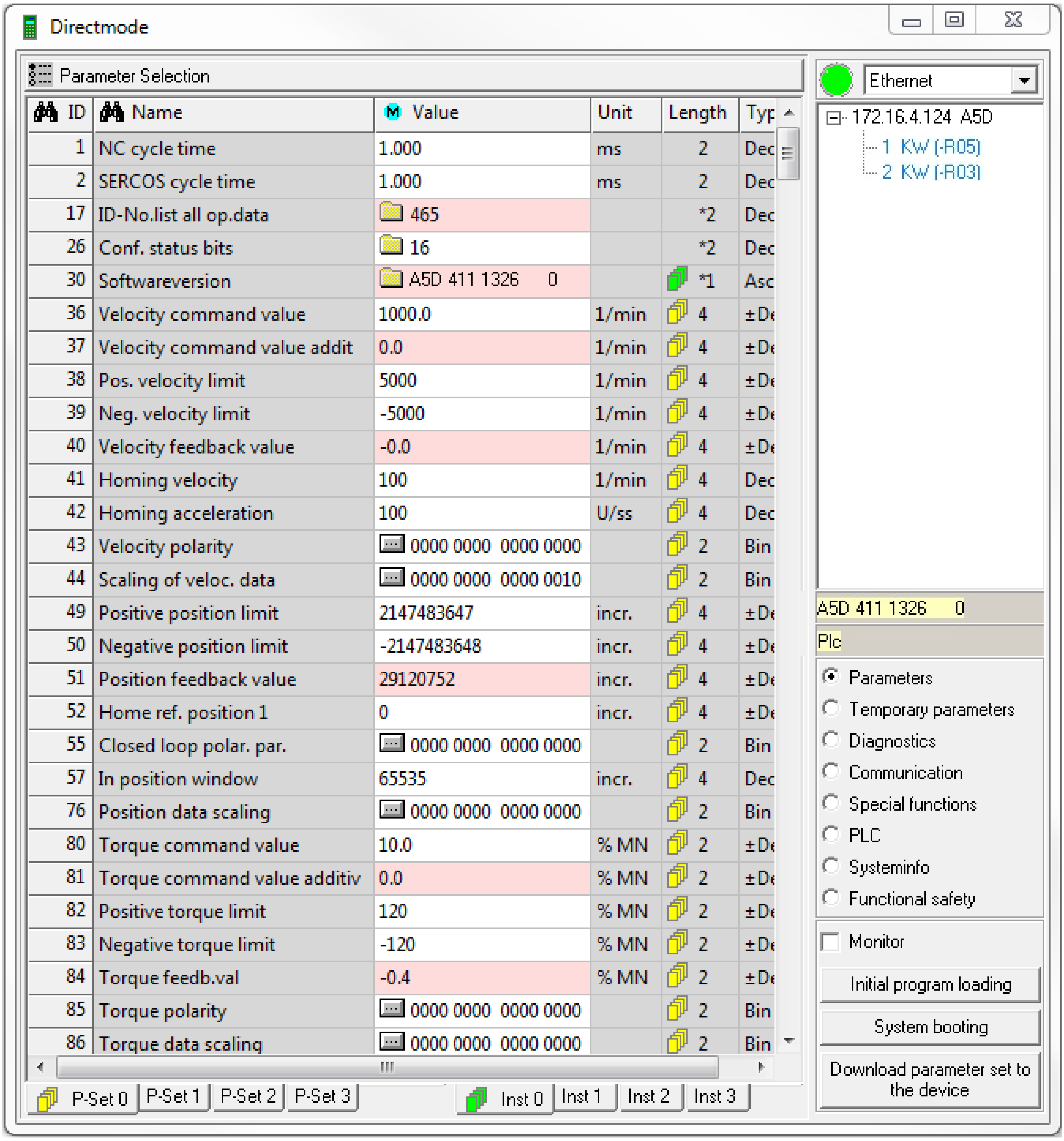
Following "login" on the controller, the IO can be accessed directly, for example (see Figure 19).

Figure 19: PLC project with bus configuration completed



Alternatively, "system booting" can be triggered by switching the system off and back on again or by selecting the corresponding command from the menu when AIPEX PRO is in "Directmode" (see Figure 20).

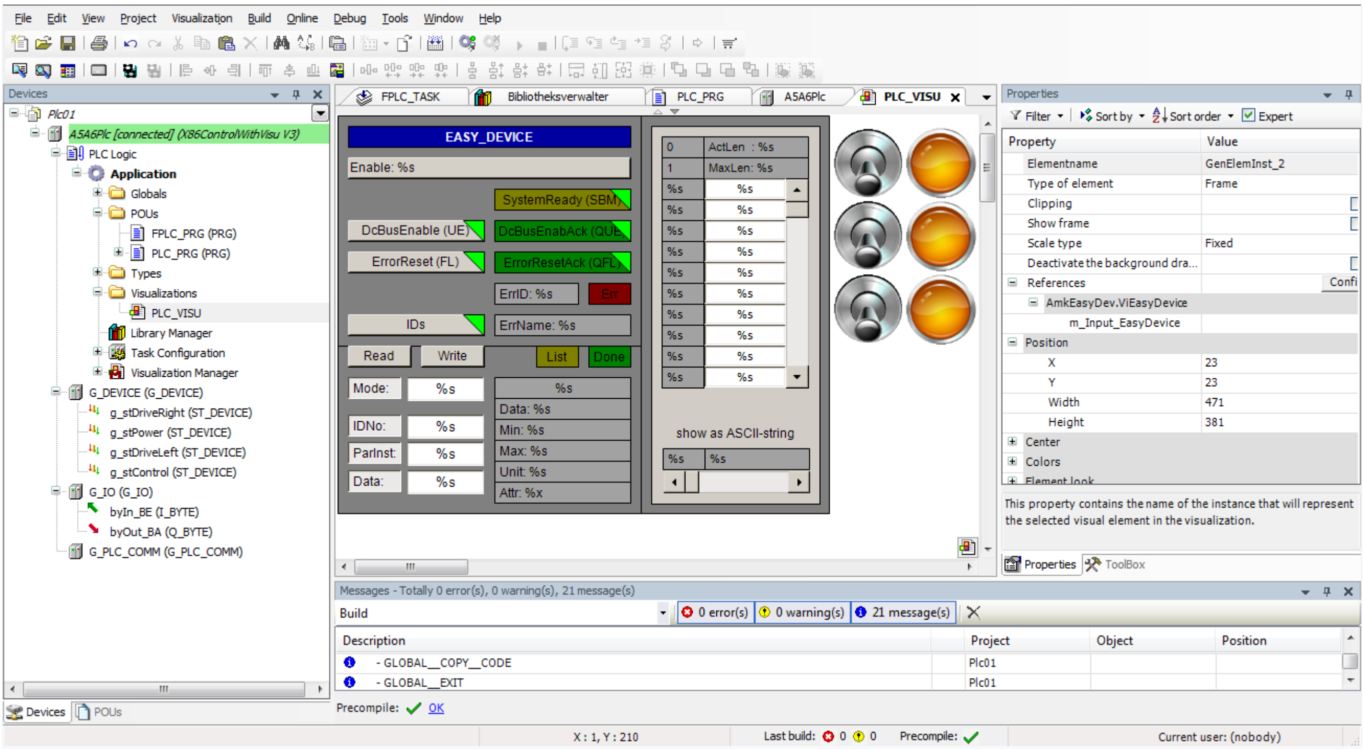
Figure 20: Direct mode



3.6 Visualization

Figure 21 still shows the corresponding block-specific visualization (e.g. 'ViEasyDevice') as available in the AmkEasyDev library block.

Figure 21: Block-specific visualizations



In the context of the frame concept of CODESYS V3, this can be linked with the corresponding instance of the 'EASY_DEVICE' block (e.g. PLC_PRG.fbPlc) (see Figure 21).

It can be used to create very simple projects (for test purposes, for example).

With some more complex visualizations, visualization elements that are not required can be deselected (see the green triangle in Figure 21). When 'boEnable' is set for the block, these elements are hidden in the visualization. They are not enabled in the corresponding block. The local variable 'wDisable' is integrated for this purpose (see the documentation for AmkEasyDev, for example).

3.7 Additional variable access

Additional variable access supports formal acces (read and write) to the device information that can be mapped via the bus. The associated procedure is as follows:

- First, the device interface of the required device is selected.
- Next, the information (variables) that can be mapped via the corresponding bus for this device is displayed (see Figure 23) by "double-clicking" "Additional variable access" (see Figure 22). The variables listed are filtered by ready/write selection. ("Reading" and "writing" are defined from the point of view of PLC programming.)
- Select a variable and press the "Add" button (see Figure 23) to display the PLC block assigned to the formal mapping (see Figure 24).
- Press the "OK" button to import this block into the PLC project that is currently open (see Figure 25).

The access blocks generated in this way provide read access ('GET_FDEV_...') and write access ('SET_FDEV_...') to device information. Moreover, the "Add readback function" button allows 'GET_FDEV_...' blocks to be generated to read back values written with 'SET_FDEV_...' blocks.

The term "formal" means a copy-only function in the context of these generated blocks (in contrast to the blocks in the AmkDevAccess library). The device information copied is thus both device-specific and bus-system-specific.

Figure 22: Additional variable access

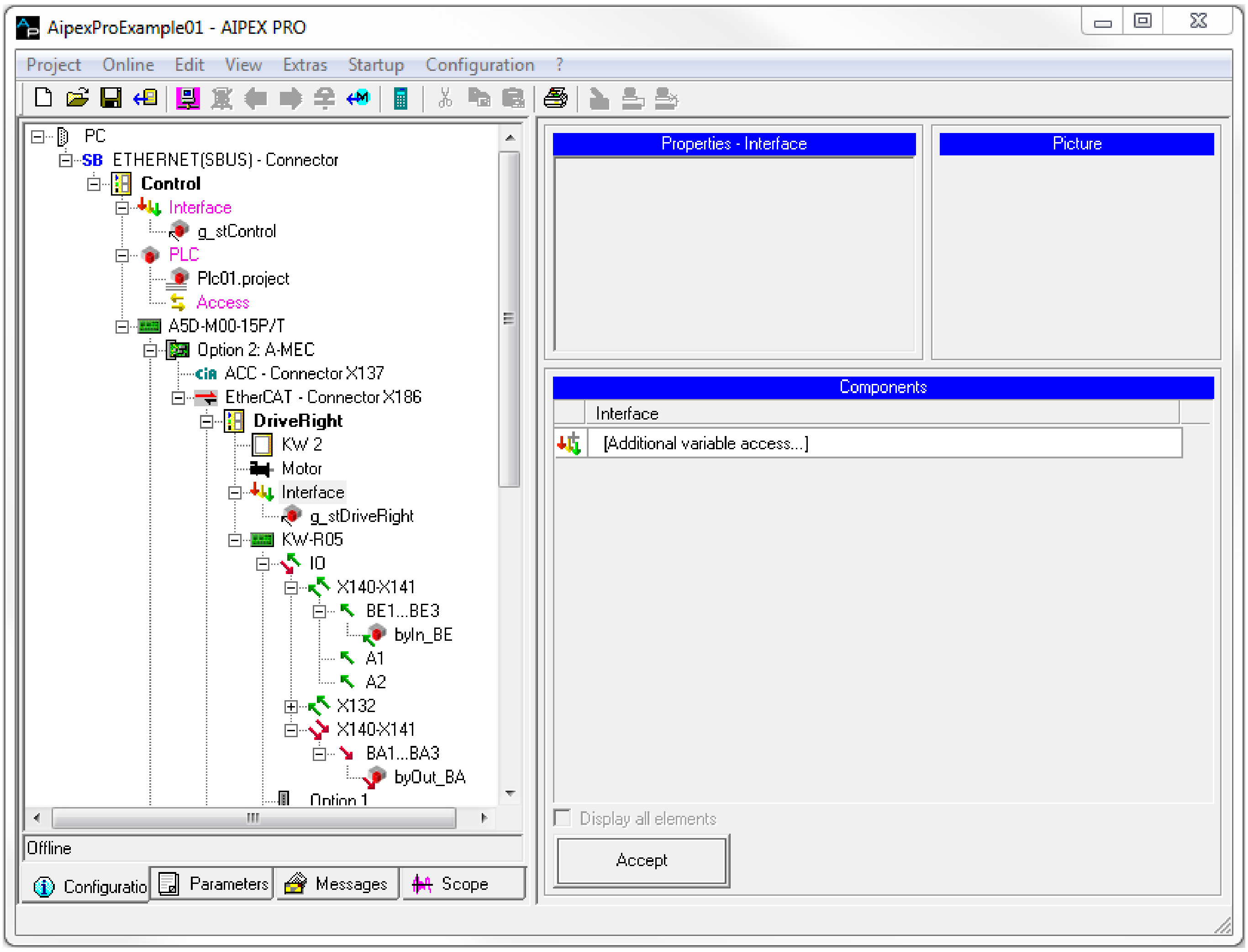


Figure 23: Variable selection (1)

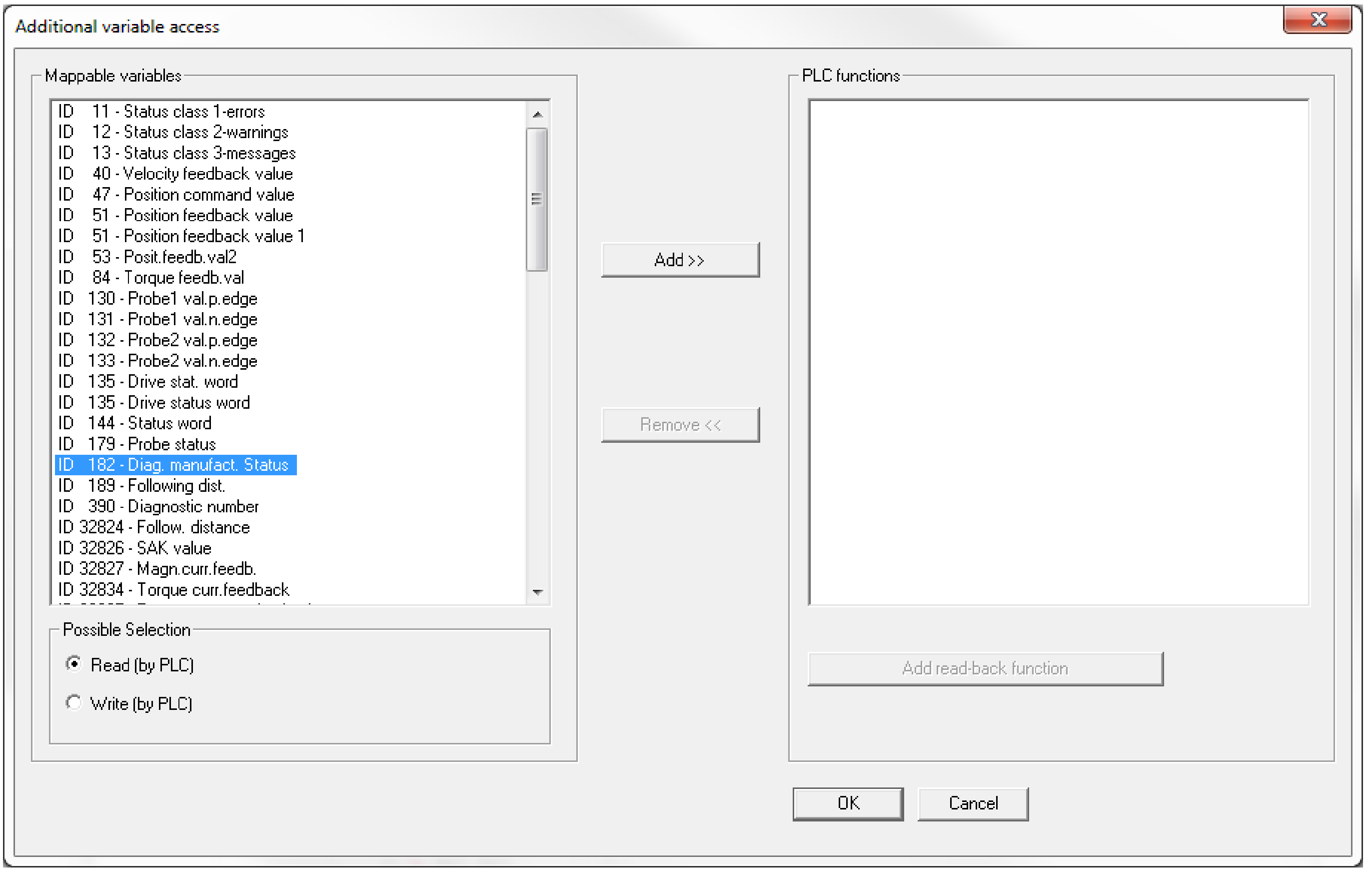


Figure 24: Variable selection (2)

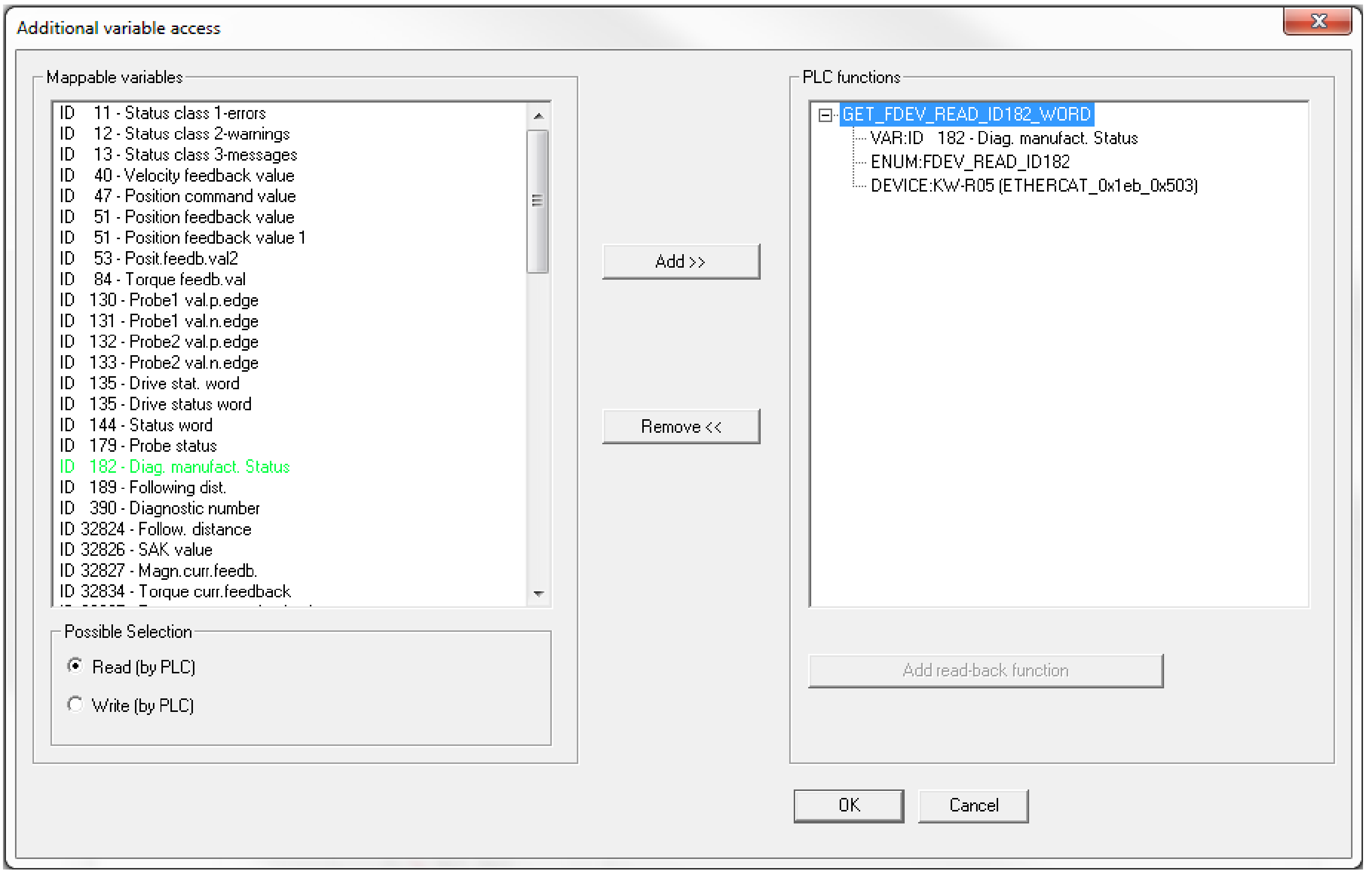
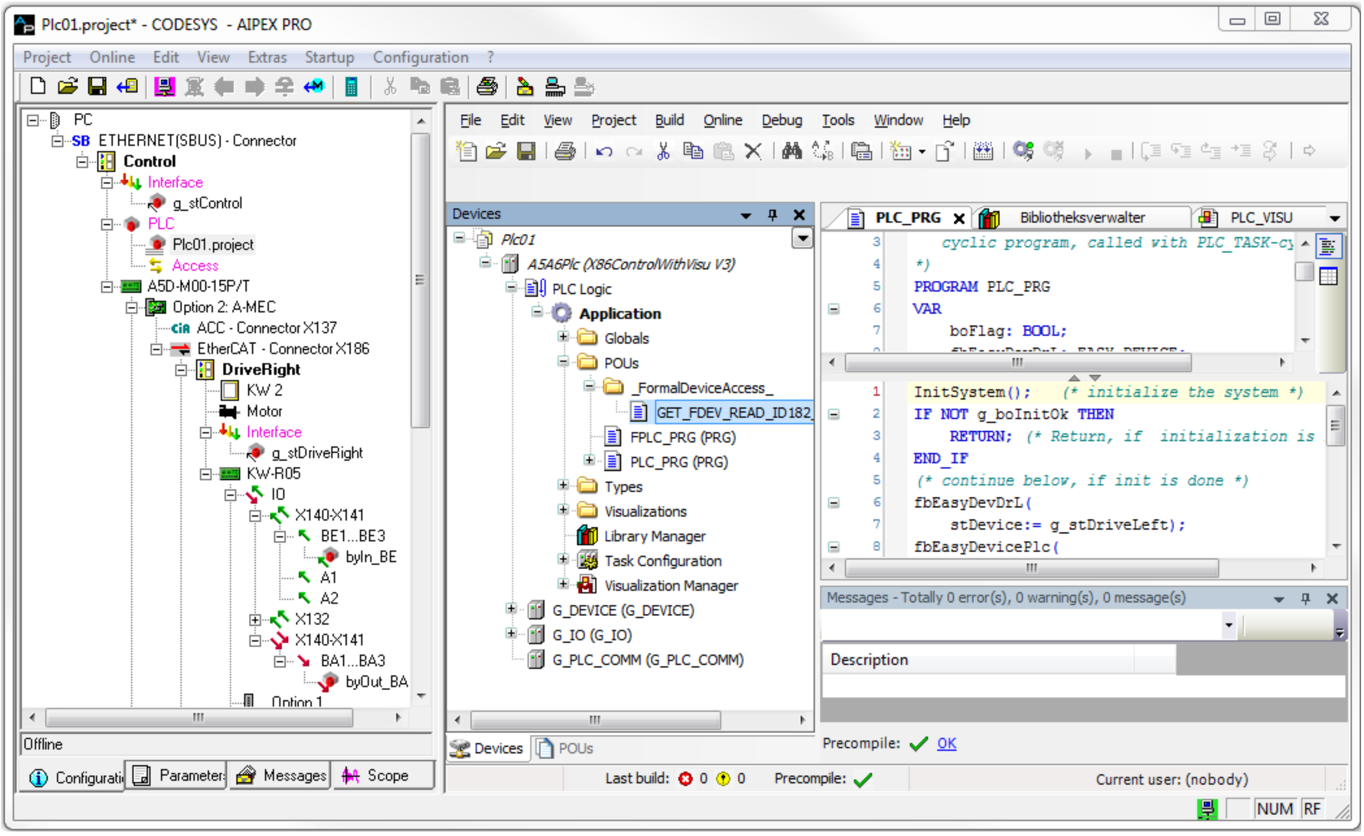
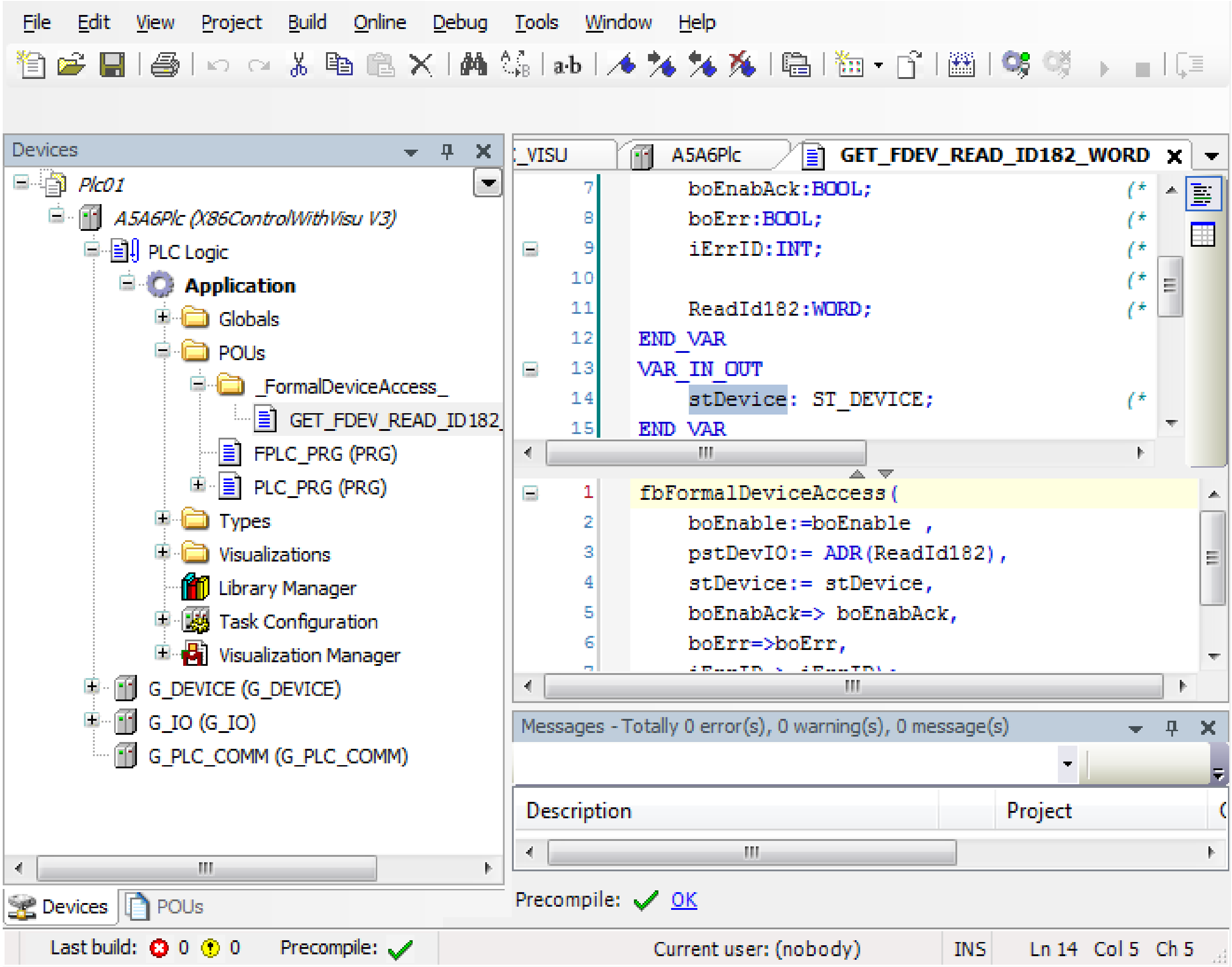


Figure 25: Block import



As shown in Figure 25, all formal variable access blocks generated are imported in the "_FormalDeviceAccess_" folder of the active PLC project. These blocks can then be used in the project like the blocks from the AmkDevAccess library. Like the AmkDevAccess blocks, the automatic bus configuration is based on the assignment of the 'stDevice' variables (see Figure 26) to a corresponding device (see Figure 15).

Figure 26: Block interface

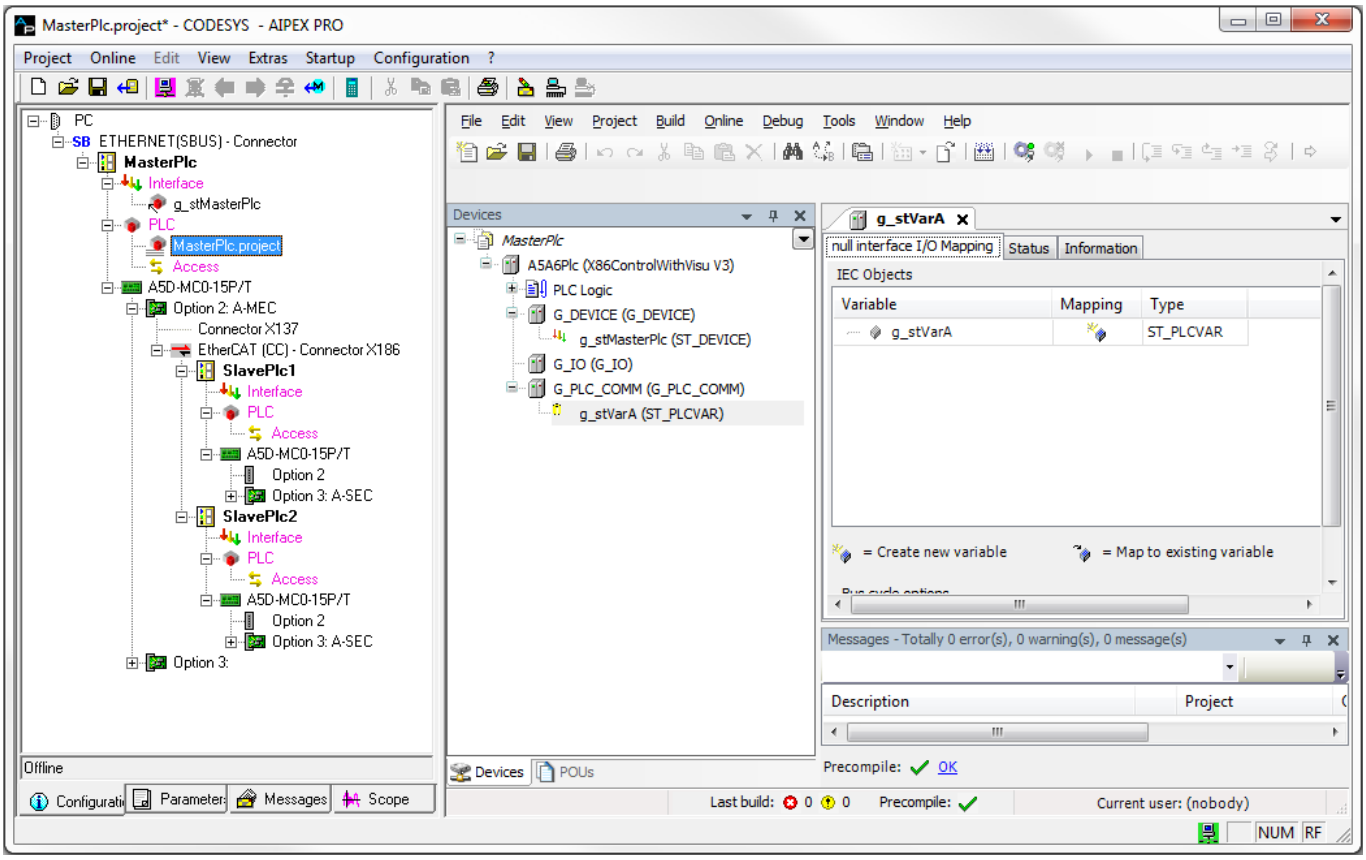


3.8 PLC-PLC communication

Data exchange (read and write) is now also possible between AMK controllers. The associated procedure is as follows:

- First, a handle is added to the PlcCommVars folder for PLC-PLC communication (see Figure 27: 'g_stVarA').

Figure 27: Master controller configuration



- Next, a block ('SET_PLCVAR_SYNC_INT', for example) is selected from the PlcVarAccess folder in the AmkDevAccess library for the synchronous sending of an INT variable (see Figure 28) and called in the synchronous FPLC_PRG (see Figure 29).

Figure 28: PlcVarAccess folder

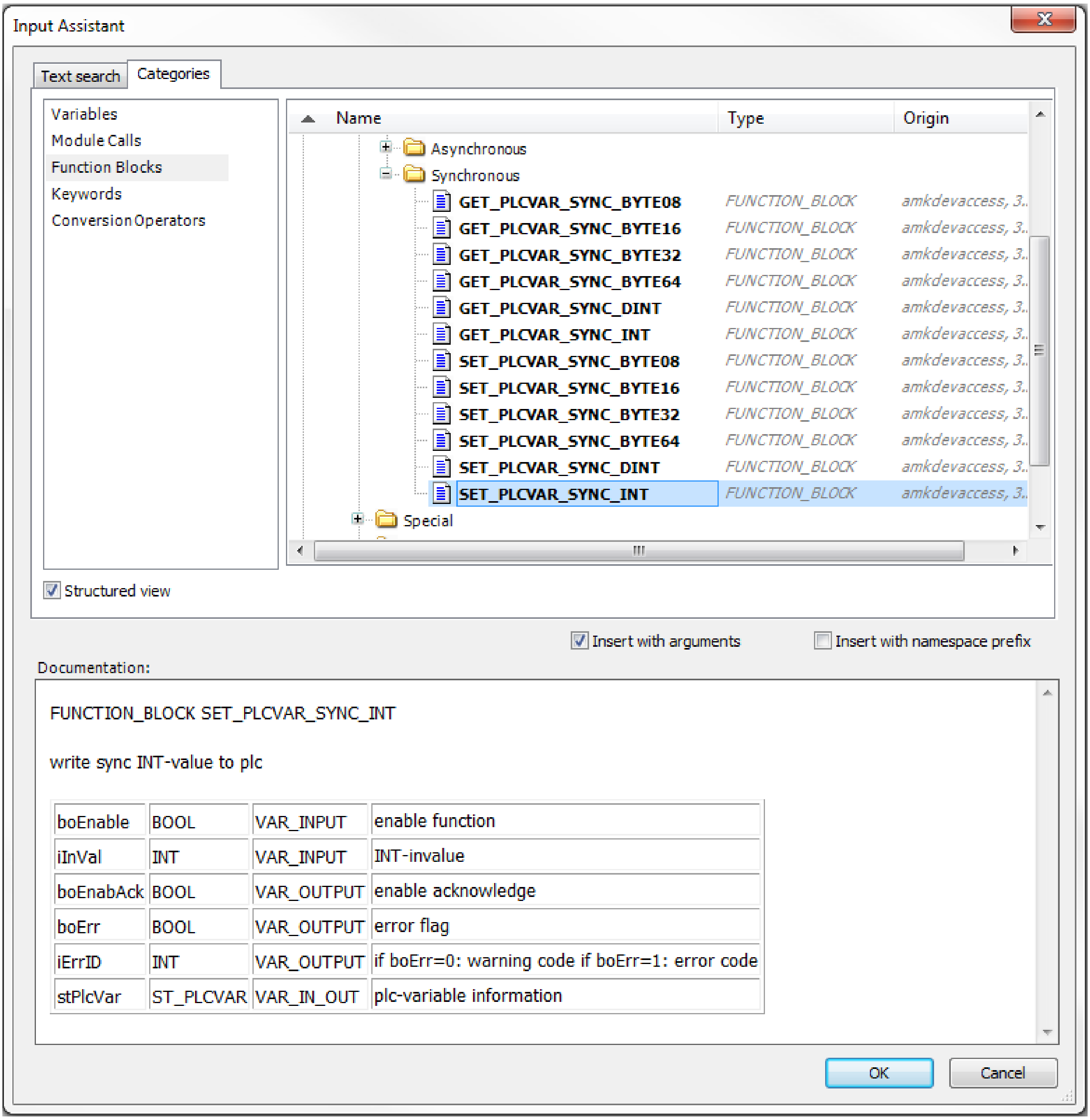
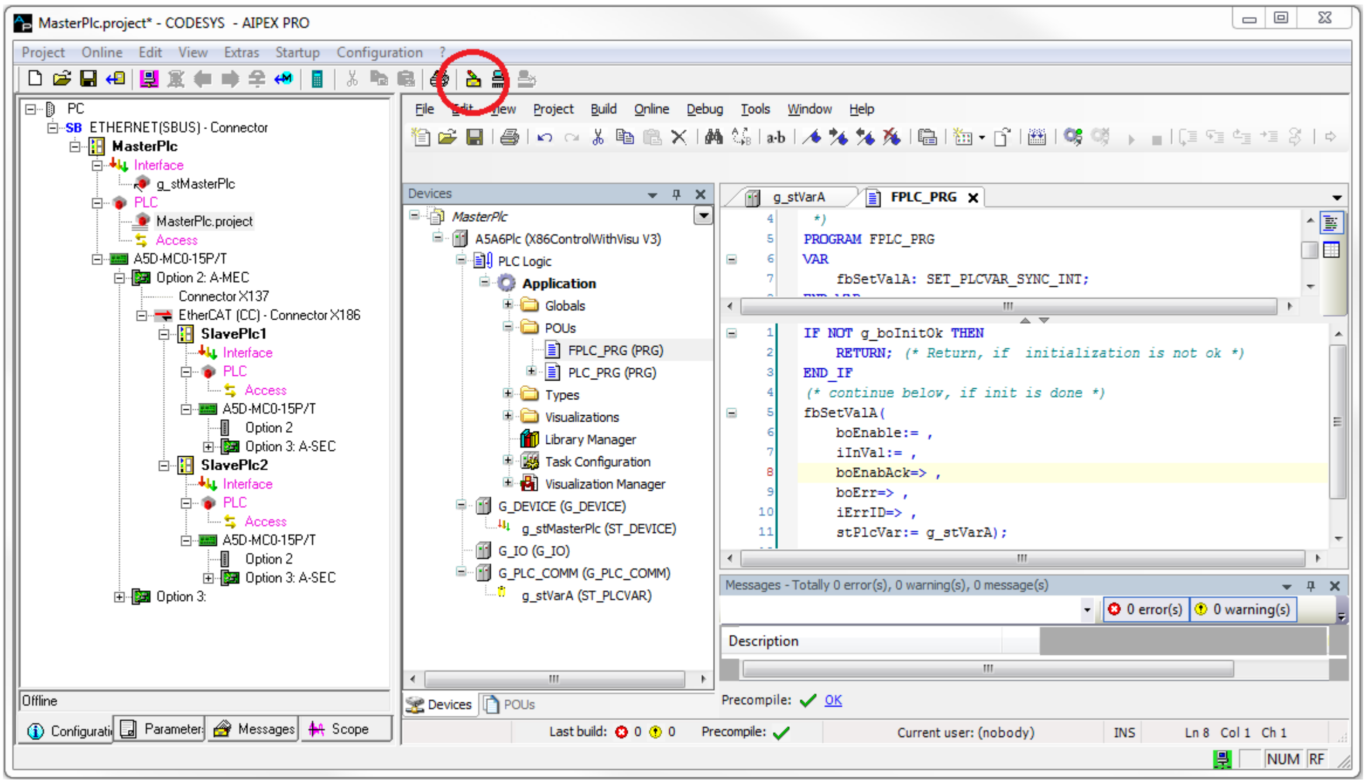


Figure 29: Master send block instance



- When "Create configuration" is selected (see the icon highlighted in Figure 29), the variable is displayed in the assignment window (see Figure 30) and can be dragged to the access branch associated with the required partner PLC (see SlavePlc1: Figure 31) in the device tree. This establishes the connection between the MasterPlc and SlavePlc1 with the 'g_stVarA' handle.

Figure 30: Assignment window

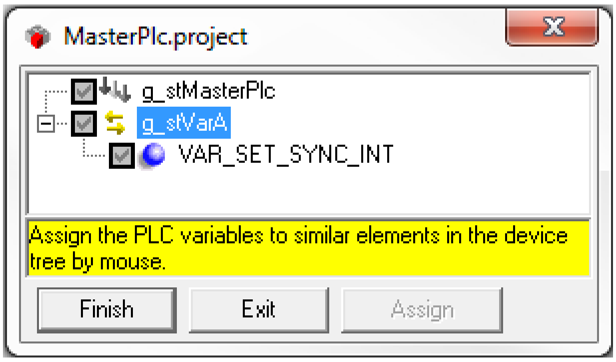
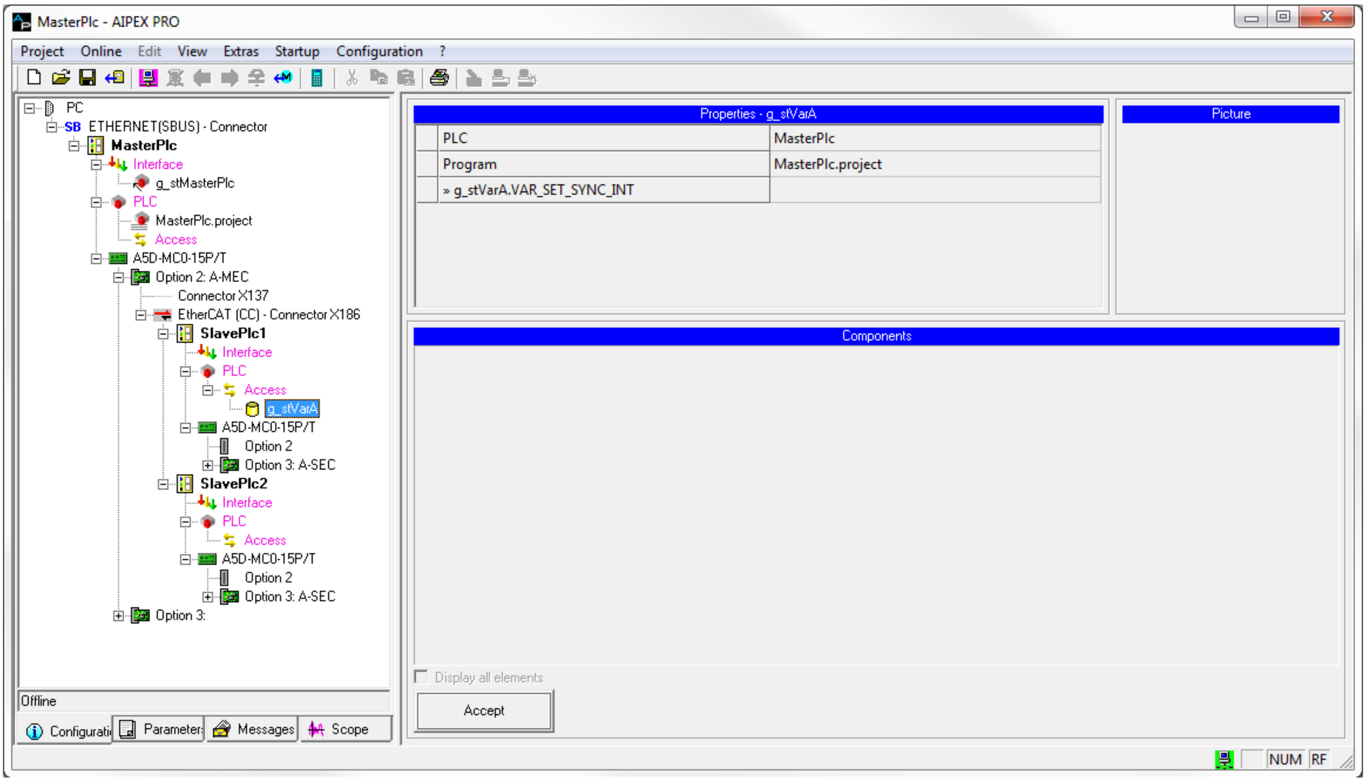


Figure 31: SlavePlc1 access branch



- The 'g_stVarA' handle is now listed automatically in the PLC configuration. Next, a corresponding block ('GET_PLCVAR_SYNC_INT', for example) is selected from the PlcVarAccess folder in the AmkDevAccess library for the synchronous receiving of an INT variable (see Figure 28) and called in the synchronous FPLC_PRG. The connection to the send block is established by adding the 'g_stVarA' handle (see Figure 32 or Figure 33).

Figure 32: SlavePlc1 controller configuration

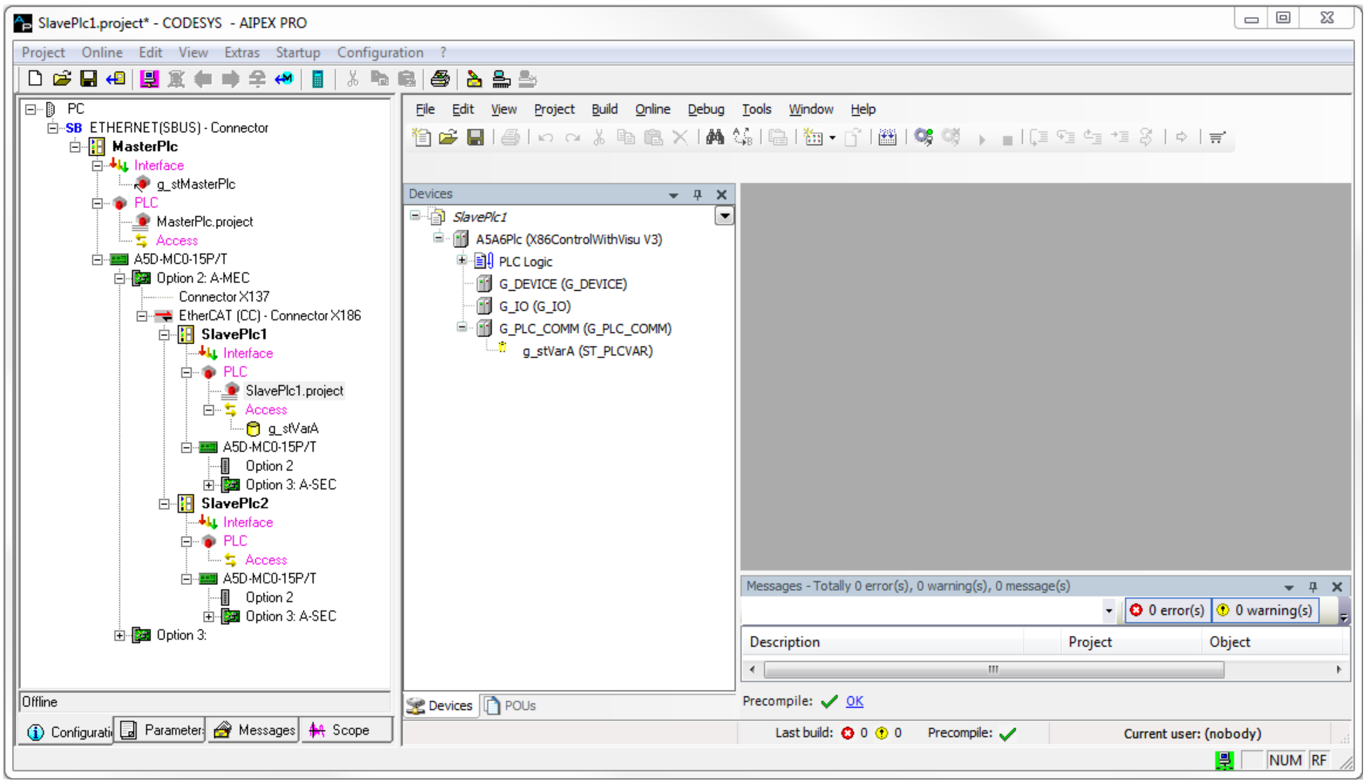
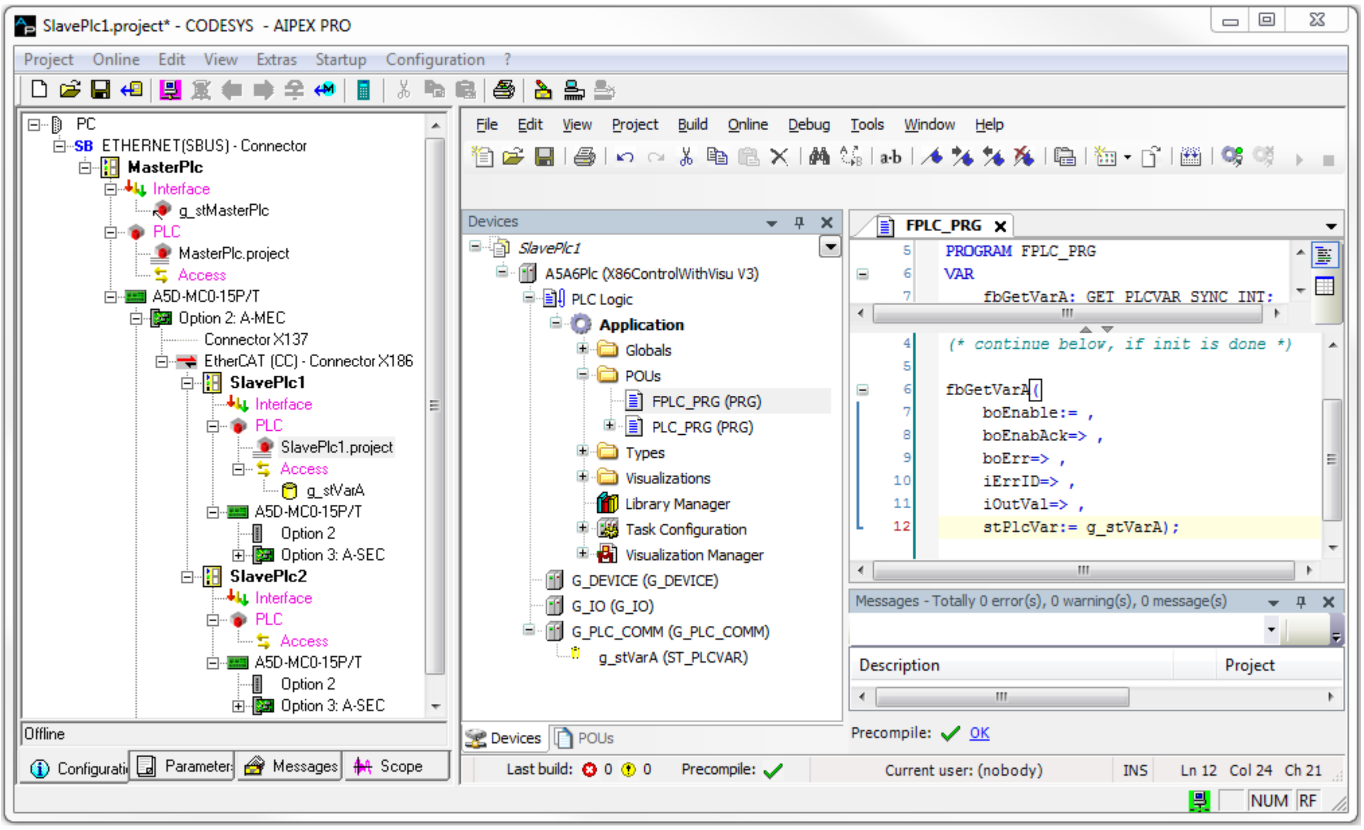
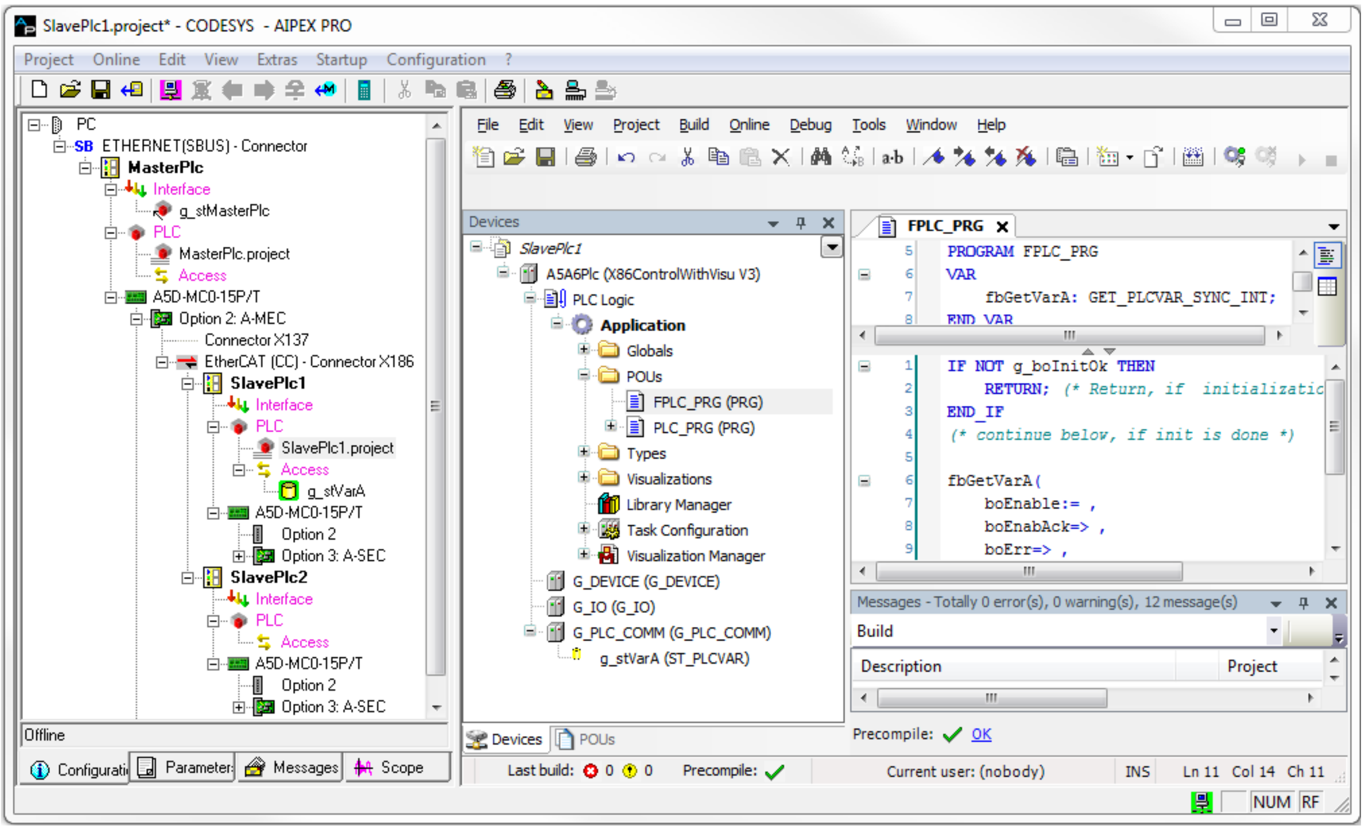


Figure 33: SlavePlc1 receive block instance



- The information required for the bus connection is generated with "Create configuration" on "SlavePlc1". The creation of the connection is displayed by the icon color in the device tree changing to green (see Figure 34).

Figure 34: Configured PLC-PLC connection branch



3.9 Support of specific IO

AIPEX PRO provides support for specific IO modules. As well as simplified access to specific XFC terminals, access to comparable functions of Axx-MxE- and iSA controller variants (controllers with local IO) is supported.

As the local TimeStamp IO for Axx-MxE- and iSA-controllers is compatible with the function of the XFC blocks (EL 1252 and EL 2252), these blocks can also be used to access these terminals.

Blocks from the AmkDevAccess library can be added for Axx-MxE and iSA (controllers with local IO) to access the local TimeStamp IO instances of the 'CAM_CONT_TS', 'GET_TS_INPUTS' and 'SET_TS_OUTPUTS' blocks (see Figure 35, Figure 36). These blocks are based on the process IO blocks from the AmkDevAccess library that are available by default (see Figure 37). Therefore, import via "Additional variable access" (see Figure 22) is not necessary.

Figure 35: TimeStamp IO

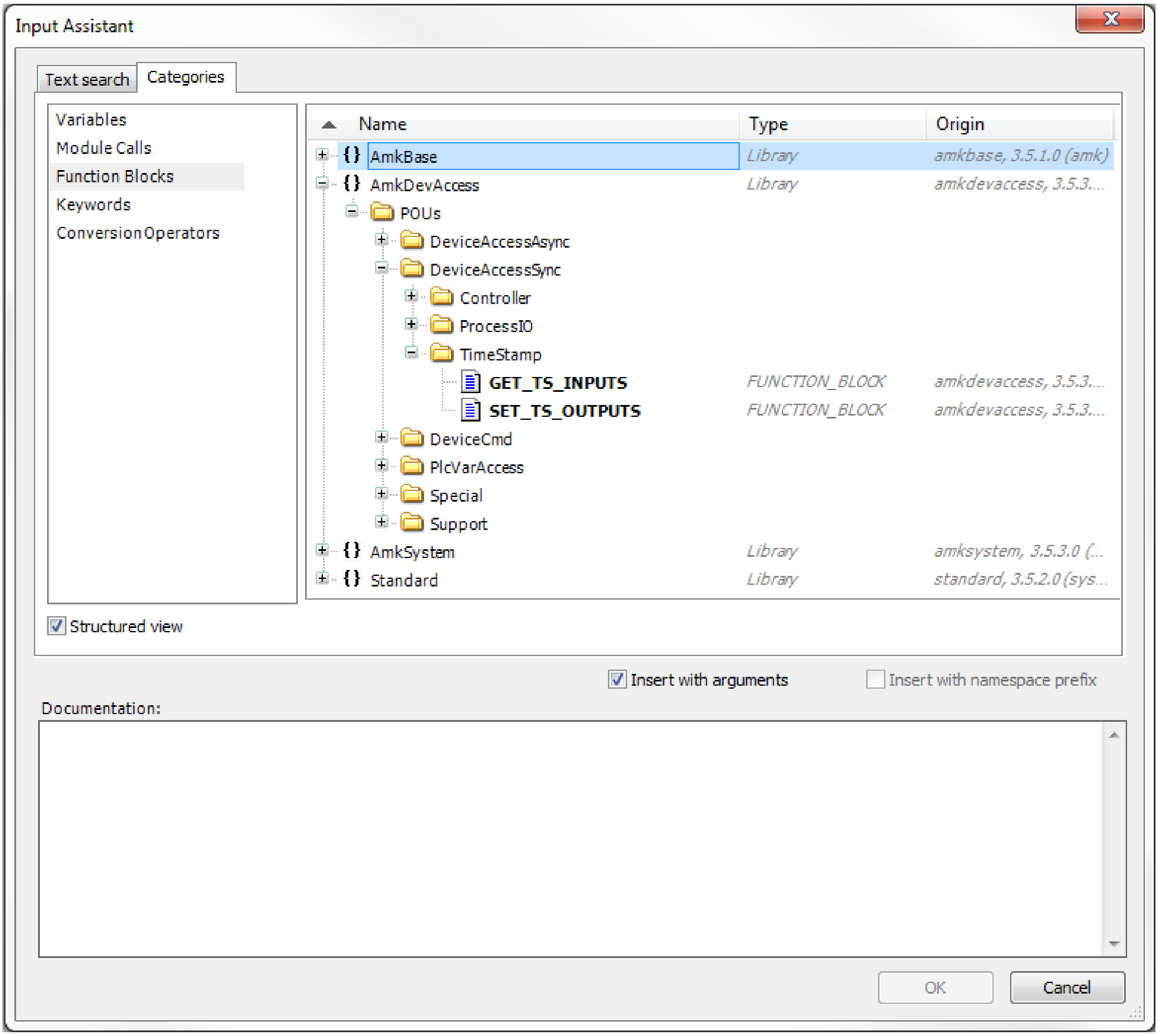


Figure 36: TimeStamp instance

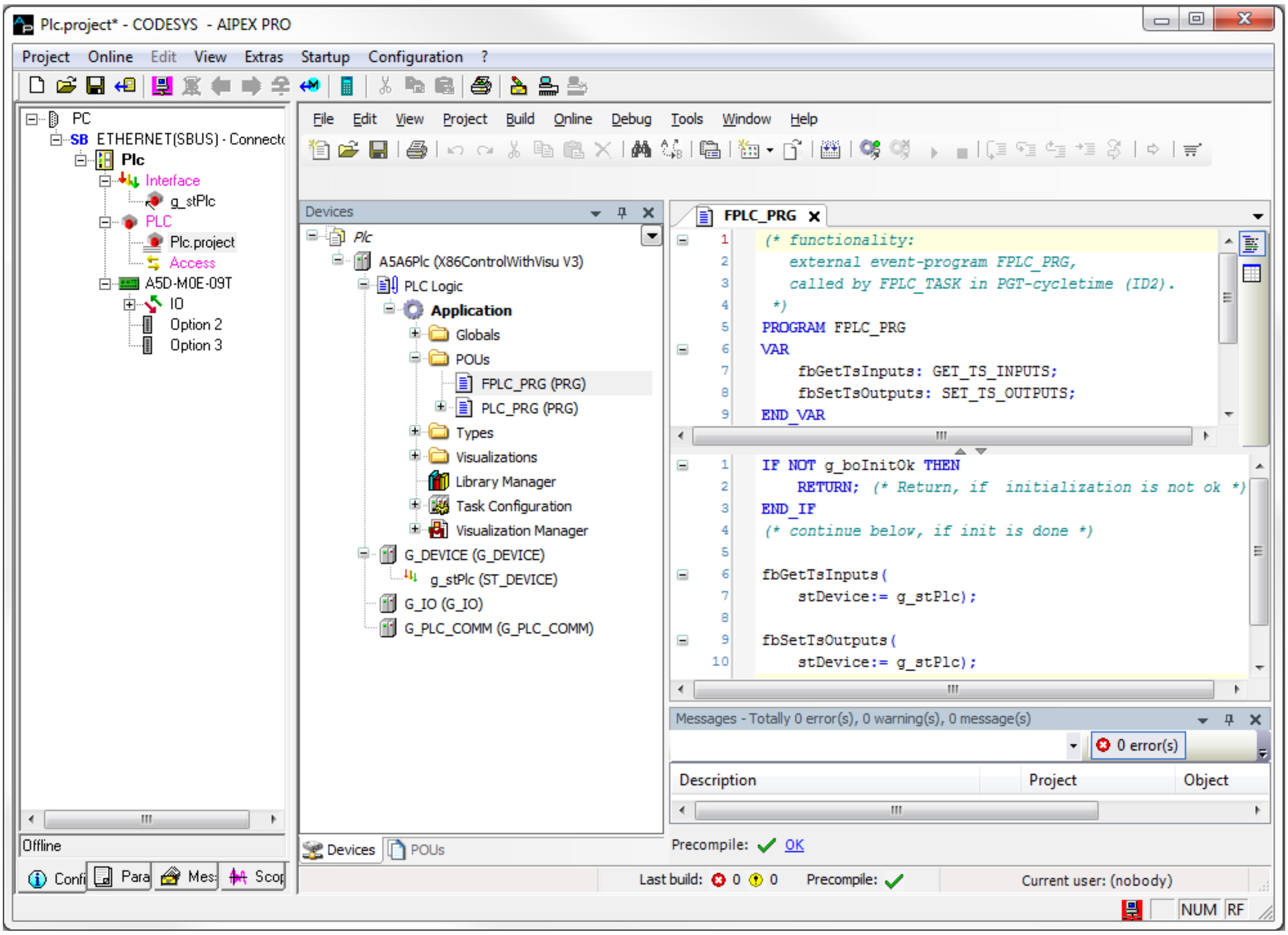
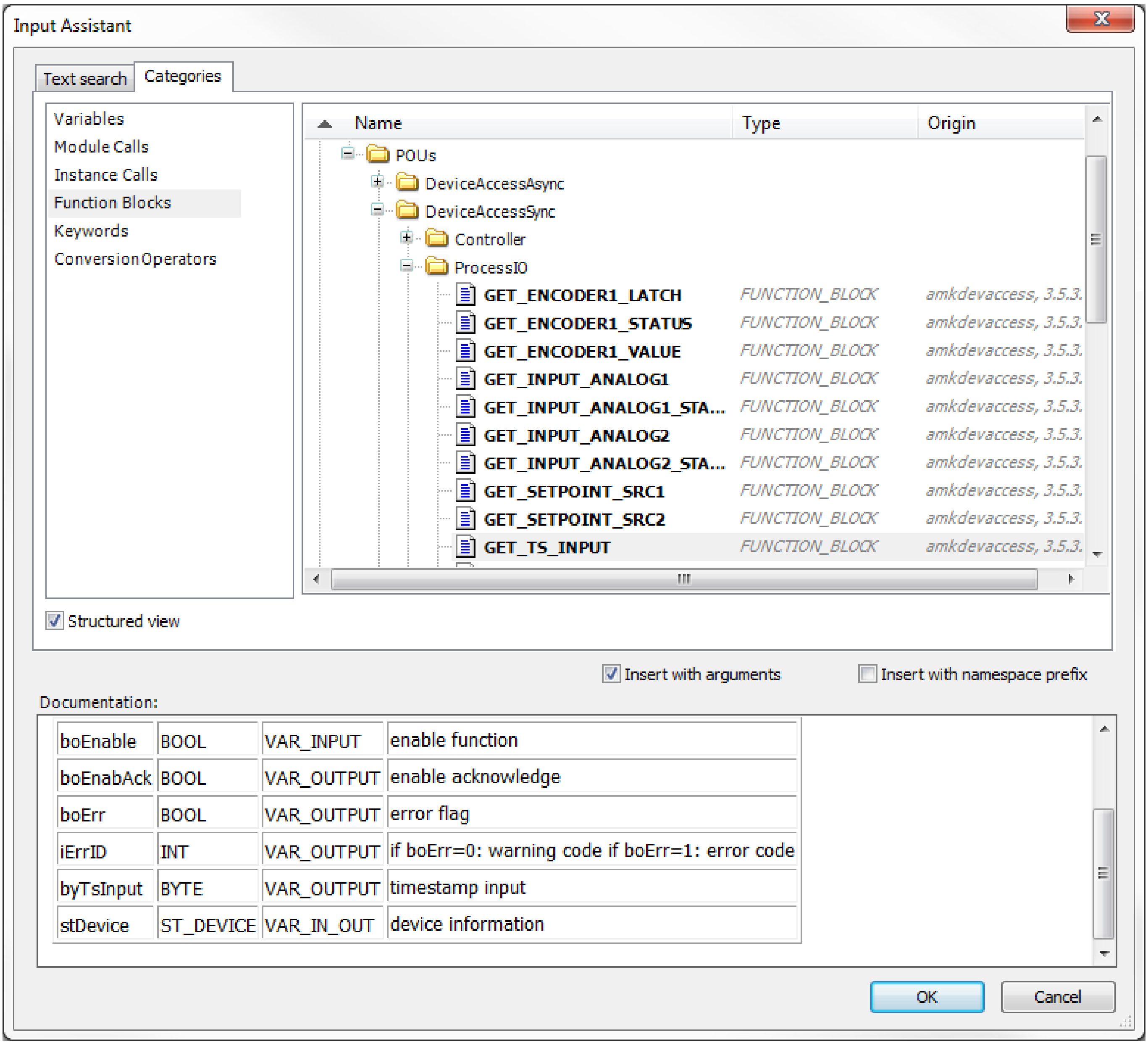


Figure 37: ProcessIO access blocks



The blocks in the "Device" group (see Figure 9) support device-driven automatic bus configuration. This facilitates functional access (based on function blocks) to internal controller information and components which can be accessed via buses (controllers, drives, IO modules, etc).

The base technology (system function) for this is provided by the AmkDevAccBase library; it is not relevant for the user (application programmer). The necessary function is made available to the programmer implicitly in the AmkDevAccess and AmkEasyDev libraries.

3.10 IO support

IO (input/output information, see Figure 38) is selected from the "Attach device" menu as shown in Figure 39.

Figure 38: IO selection

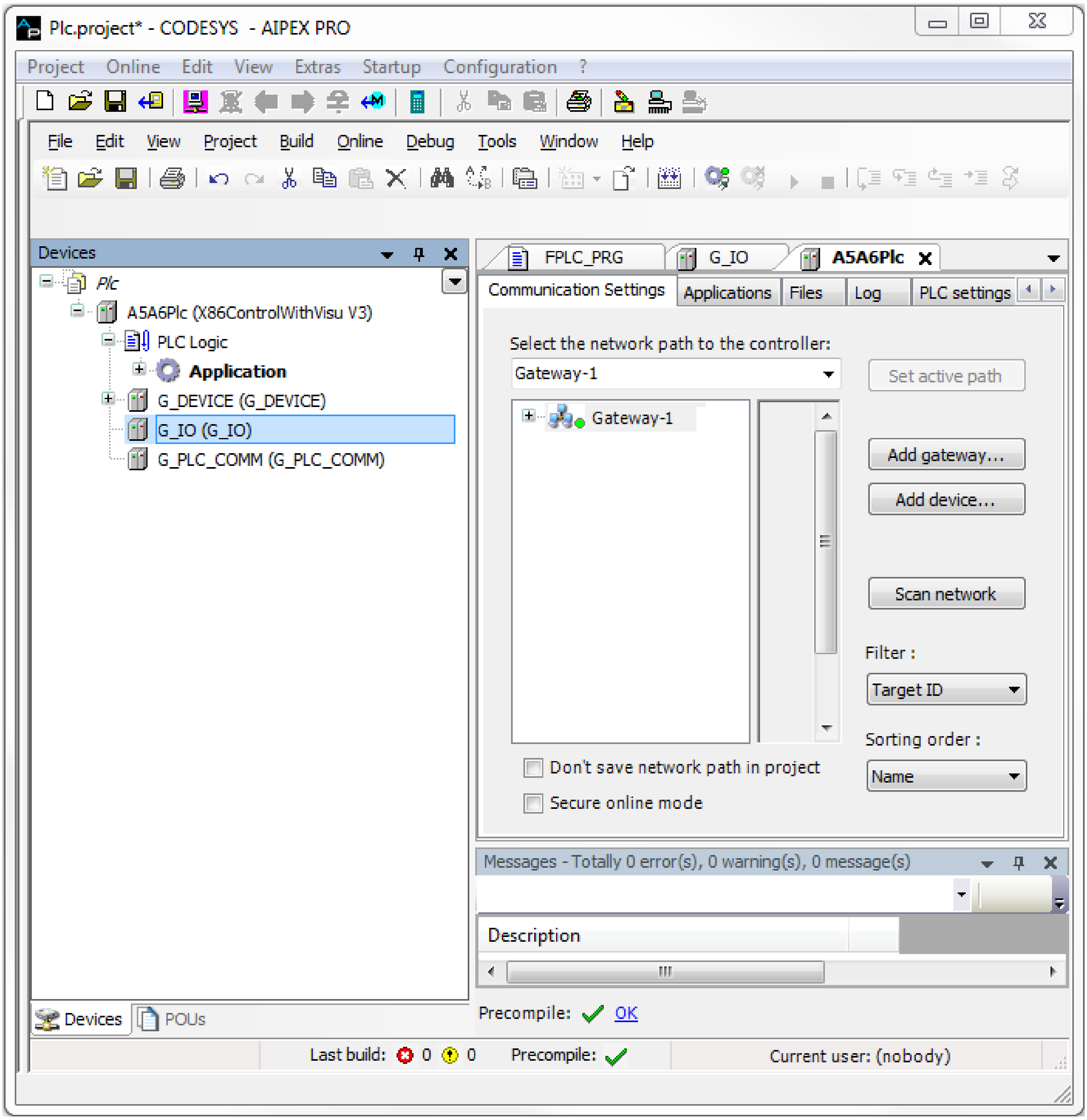
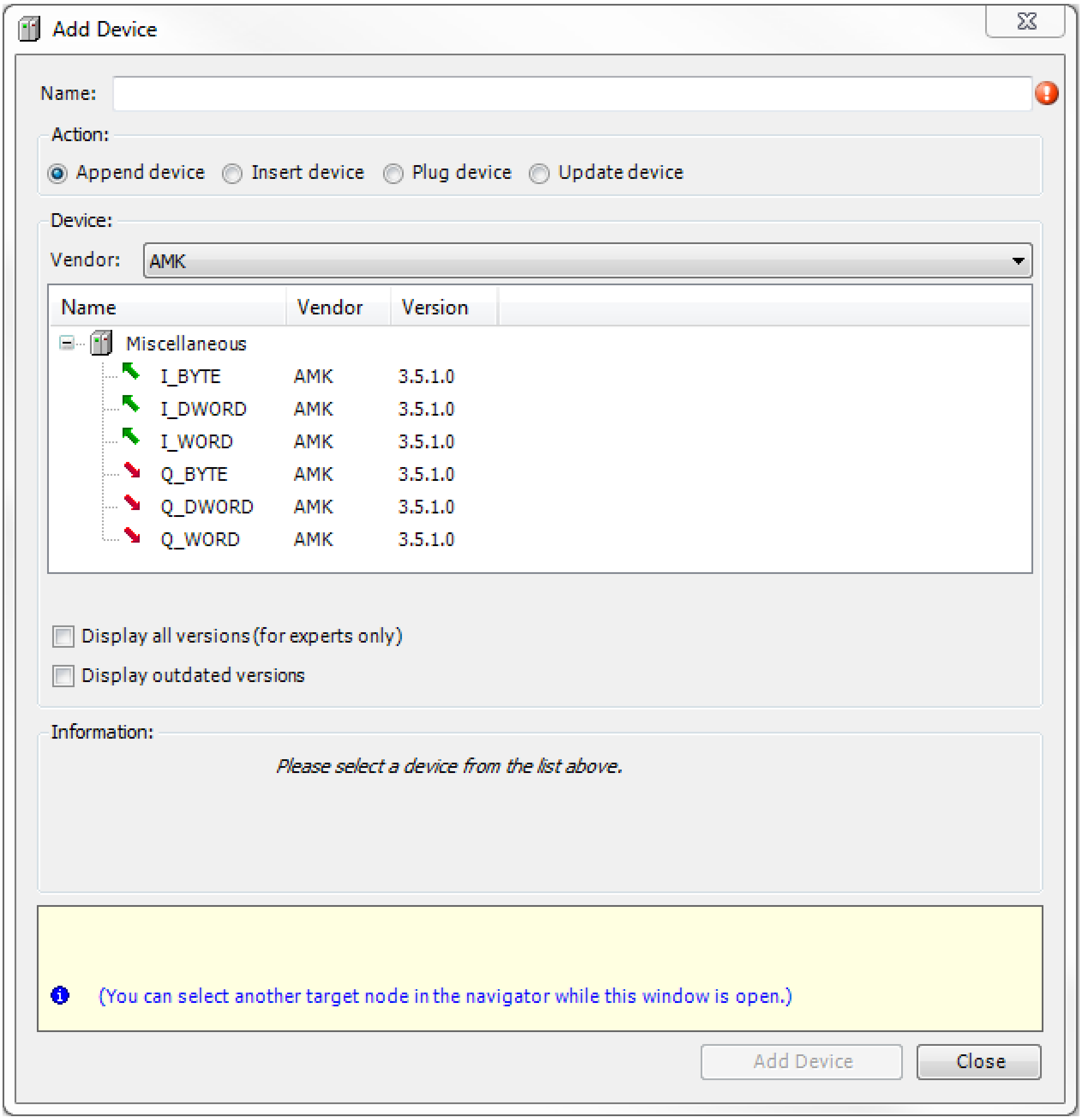


Figure 39: IO modules



A distinction is made between

- Input modules (I) of the following type:
  - BYTE
  - WORD
  - DWORD
- Output modules (Q) of the following type:
  - BYTE
  - WORD
  - DWORD

3.11 Device selection

Device descriptions specific to AMK have been created for AxD, AxS and iSA type controllers. In AIPEX PRO, the correct device description is set automatically based on the selected device when a CODESYS project is created (see Section 1.1).

A5/A6-D and A5/A6-S controllers

As shown in Figure 40, the following three basic variants have been created specifically for AMK for devices based on Intel processors (X86) (each further categorized by devices with and without visualization):

X86Control V3 X86ControlWithVisu V3	Ax devices without add-on option (PCO or PNC); without visualization (e.g. AxS devices). With visualization (e.g. AxD devices).
X86PLCopenControl V3 X86PLCopenControlWithVisu V3	Ax devices with PCO add-on option (PLCopen); without visualization (e.g. AxS devices). With visualization (e.g. AxD devices).
X86PLCopenCncControl V3 X86PLCopenCncControlWithVisu V3	Ax devices with PNC add-on option (PLCopen CNC); without visualization (e.g. AxS devices). With visualization (e.g. AxD devices).

A4/iSA controller

As shown in Figure 40, the following three basic variants have been created specifically for AMK for devices (ARM processor):

ArmControl V3 ArmControlWithVisu V3	iSA device without add-on option (PCO or PNC); without visualization. With visualization.
ArmPLCopenControl V3 ArmPLCopenControlWithVisu V3	iSA device with PCO add-on option (PLCopen); without visualization. With visualization.
ArmPLCopenCncControl V3 ArmPLCopenCncControlWithVisu V3	iSA device with PNC add-on option (PLCopen CNC); without visualization. With visualization.

As shown in Figure 41, the devices (controllers) are identified based on their target system ID and target system type. This results in the following interdependencies as listed in the table below:

X86 - A5/A6 Device	Target system ID	Target system type
X86Control V3 X86ControlWithVisu V3	16#10830006 16#10830005	16#1000 Controller 16#1000 Controller
X86PLCopenControl V3 X86PLCopenControlWithVisu V3	16#10830004 16#10830002	16#1006 Softmotion controller 16#1006 Softmotion controller
X86PLCopenCncControl V3 X86PLCopenCncControlWithVisu V3	16#10830003 16#10830001	16#1006 Softmotion controller 16#1006 Softmotion controller
Arm - A4/iSA device	**Target system ID**	**Target system type**
ArmControl V3 ArmControlWithVisu V3	16#10830016 16#10830015	16#1000 Controller 16#1000 Controller
ArmPLCopenControl V3 ArmPLCopenControlWithVisu V3	16#10830014 16#10830012	16#1006 Softmotion controller 16#1006 Softmotion controller
ArmPLCopenCncControl V3 ArmPLCopenCncControlWithVisu V3	16#10830013 16#10830011	16#1006 Softmotion controller 16#1006 Softmotion controller

In CODESYS V3, you can only log in on the appropriate controller with the correct device selection in the project!
So, if the PNC option is enabled on an AxD, for example, the "X86PLCopenCncControlWithVisu V3" device must always be set.

Figure 40: Device selection

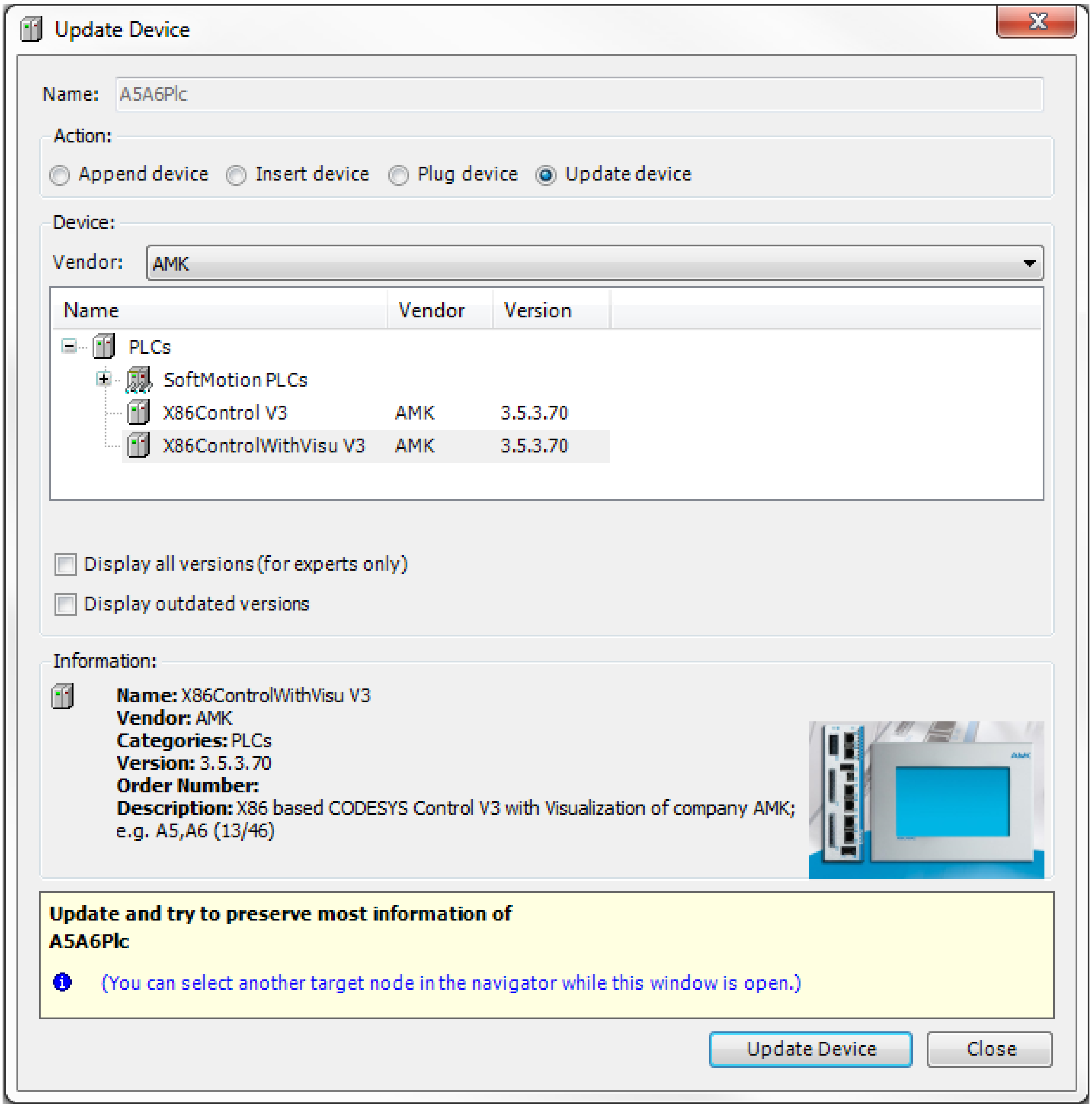
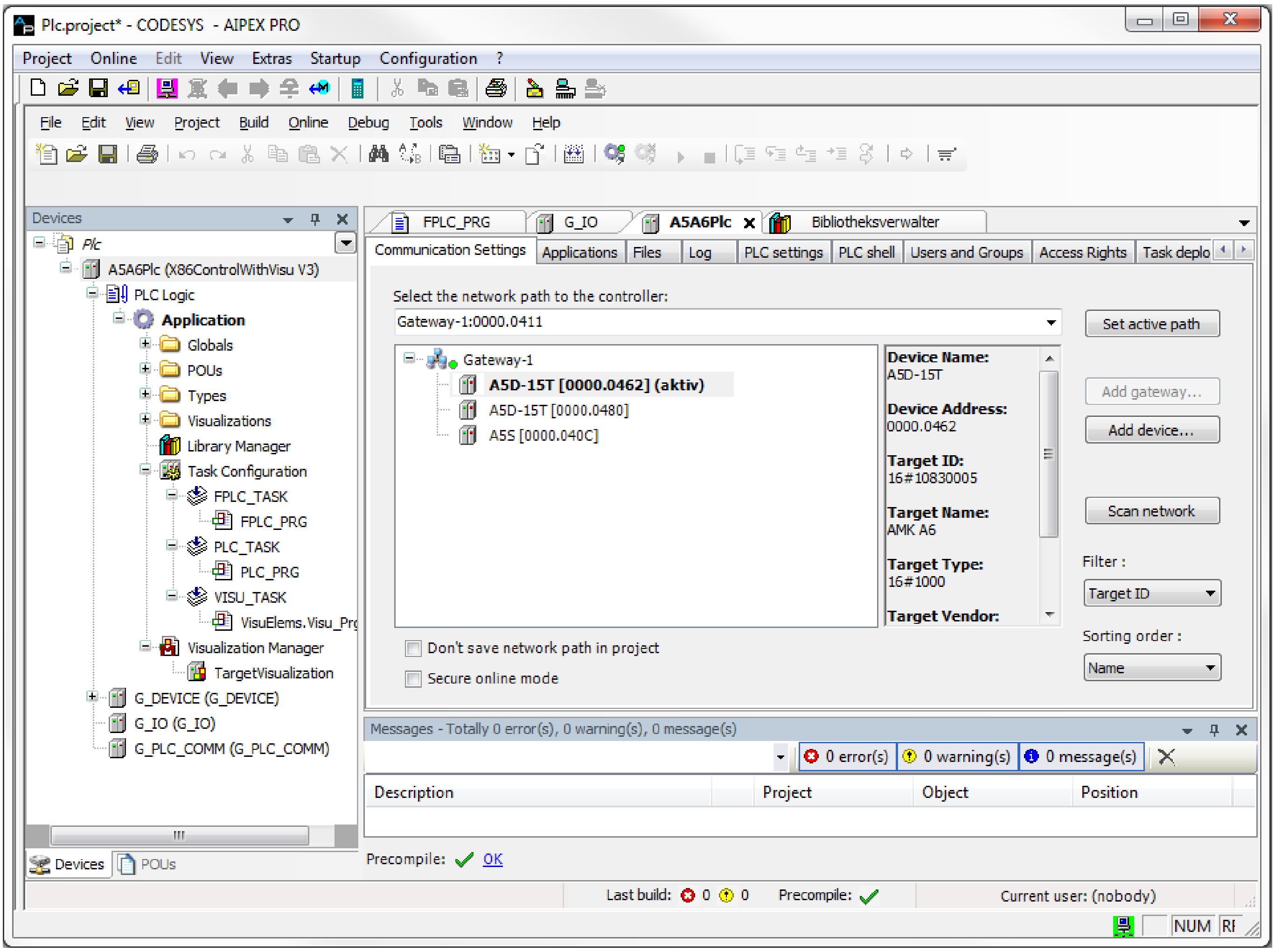


Figure 41: Device identification



In AIPEX PRO, the PCO (PLCopen) or PNC (PLCopen CNC) option is set in the computer card properties (see Figure 42).

The VIS (visualization) option is set automatically based on the selected computer card (e.g. AxD) or, for AxS devices, can be set via the "Web visualization" property in AIPEX PRO (see Figure 43).

Figure 42: Option selection (PLCopen/PLCopen CNC)

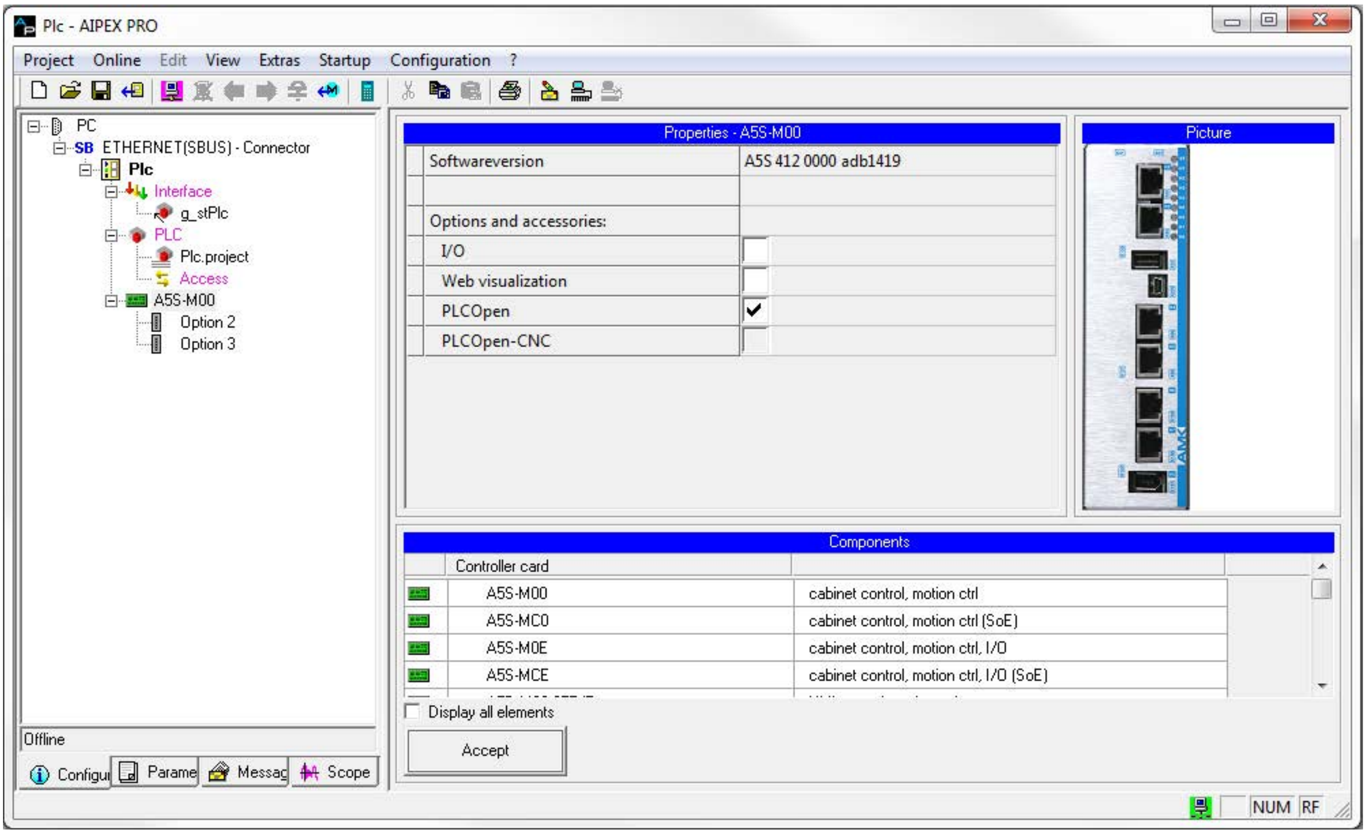
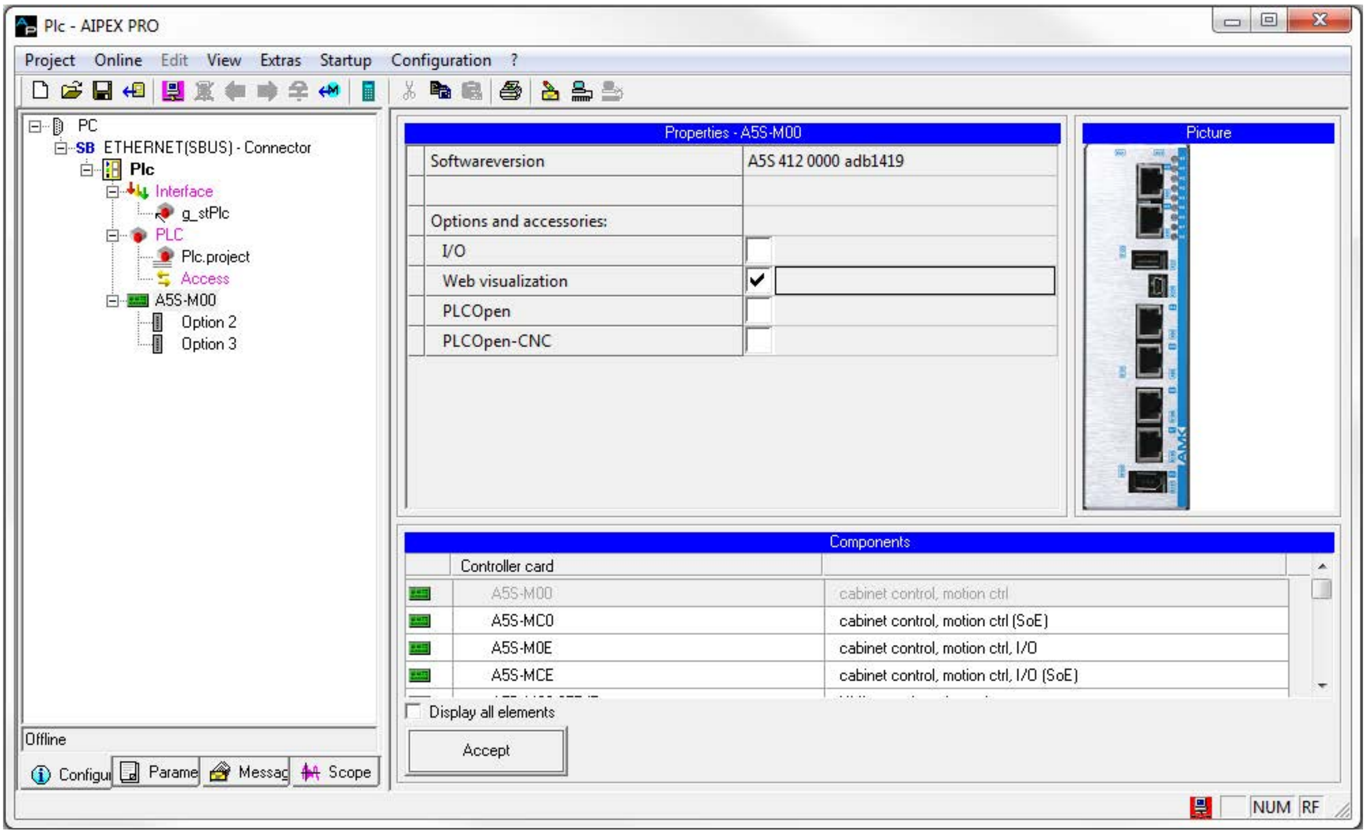


Figure 43: Option selection (web visualization)



3.12 Data exchange between A4/iSA and A5/A6 controllers with CODESYS V3

Apply for CODESYS V3 and affects the data transfer of structures with different elements (for example BOOL, WORD …) between and A5/A6 controller via TCP/IP, UDP, serial interfaces, file transfer or CODESYS network functions.

In order to adapt the different memory orientation of variables in different controllers, the alignment of a data structure can be explicitly defined in CODESYS V3 with { attribute 'pack_mode' := '<Value> '} (see: CODESYS help' attribute pack_mode ').

The following applies to AMK controllers:

	CODESYS V2	CODESYS V3
iSA, A4	'pack_mod' := '4' 1)	'pack_mod' := '8'
A5, A6	'pack_mod' := '1'	'pack_mod' := '4'

1) No LREAL variables can be exchanged with iSA and A4 'CODESYS V2 controllers' because in these controllers LREAL variables are implicitly used as REAL variables.

Example: Attribute 'pack_mode'

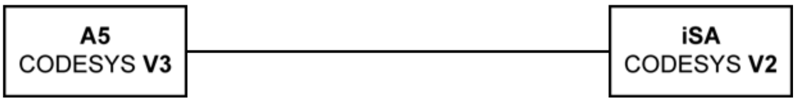
```
{attribute 'pack_mode' := '1'}
TYPE ST_A :
STRUCT
    byVarA: BYTE;
    wVarA:  WORD;
    byVarB: BYTE;
    dwVarA: DWORD;
    byVarC: BYTE;
END_STRUCT
END_TYPE
```

Example 1:

Mixed programming systems, CODESYS V3 and V2

A structure is adapted with {attribute 'pack_mode': = '4'} into a memory layout compatible with iSA (CODESYS V2).

{ attribute 'pack_mode' := '4' }
STRUCT
...

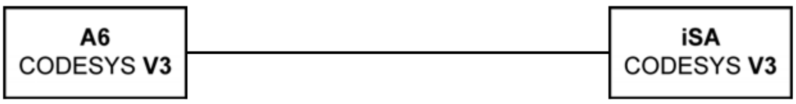


Example 2:

Identical programming systems, CODESYS V3

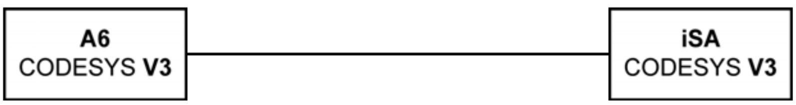
The attribute 'pack_mode' can be used on the controller A6 with the {attribute 'pack_mode': = '8' or alternatively on the iSA with the {attribute 'pack_mode': = '4'}.

{ attribute 'pack_mode' := '8' }
STRUCT
...



Alternative:

{ attribute 'pack_mode' := '4' }
STRUCT
...



3.13 OPC UA (Unified Architecture)

Requirements:

Hardware: A4 / iSA controller

Firmware: A4 / iSA ≥ V4.22

Software: AIPEX PRO ≥ V3.04 with Profil 'COSESYS V3.5 SP10 Patch 4

OPC UA (Open Platform Communications - Unified Architecture) is an industrial communication protocol between automation units (e. g. controllers, drives, operator panels, etc.) and is becoming increasingly important in connection with 'Industry 4.0'.

CODESYS V3 includes an 'OPC UA Server' from version 3.5.10.4 (integrated in AIPEX PRO version ≥ 3.04). The PLC data objects are integrated into the OPC UA communication with the CODESYS 'symbol configuration'.

Display CODESYS V3 'Log': runtime version 3.5.10.4

	03.04.2017 10:01:39	3.5.10.0 Jan 18 2017	CM
	03.04.2017 10:01:39	Copyright (c) 3S - Smart Software Solutions GmbH	CM
	03.04.2017 10:01:39	CODESYS Control V3	CM

Display CODESYS V3 'Log': OPC UA Server

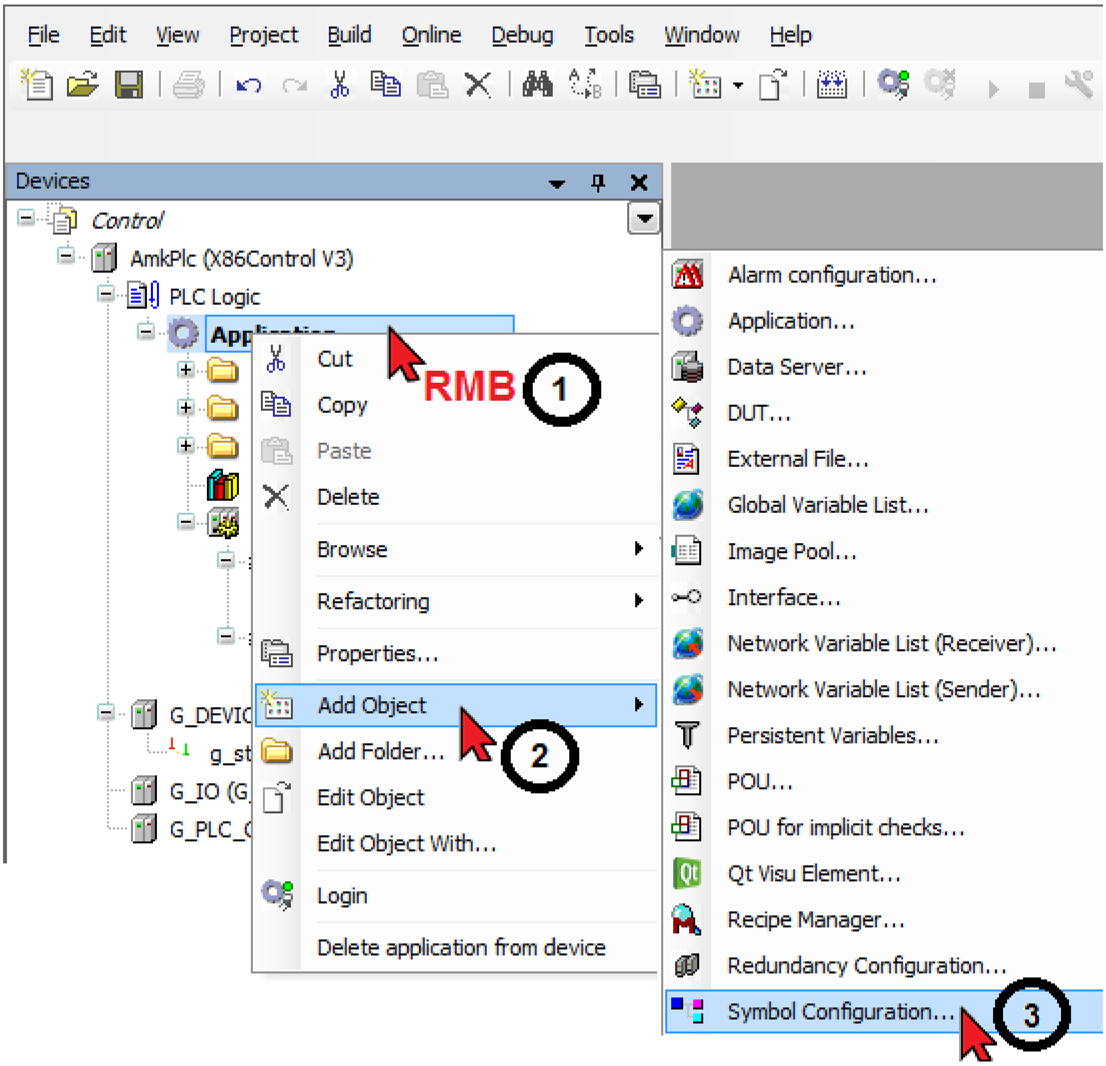
	03.04.2017 10:01:51	**	CmpOPCUAServer
	03.04.2017 10:01:51	URL:opc.tcp://172.16.4.2:4840	CmpOPCUAServer
	03.04.2017 10:01:51	URL:opc.tcp://127.0.0.1:4840	CmpOPCUAServer
	03.04.2017 10:01:51	URL:opc.tcp://A4D-07T:4840	CmpOPCUAServer
	03.04.2017 10:01:51	OPC UA Server	CmpOPCUAServer
	03.04.2017 10:01:51	**	CmpOPCUAServer
	03.04.2017 10:01:51	********************** Starting OPC UA Server! *************************	CmpOPCUAServer

Sample project

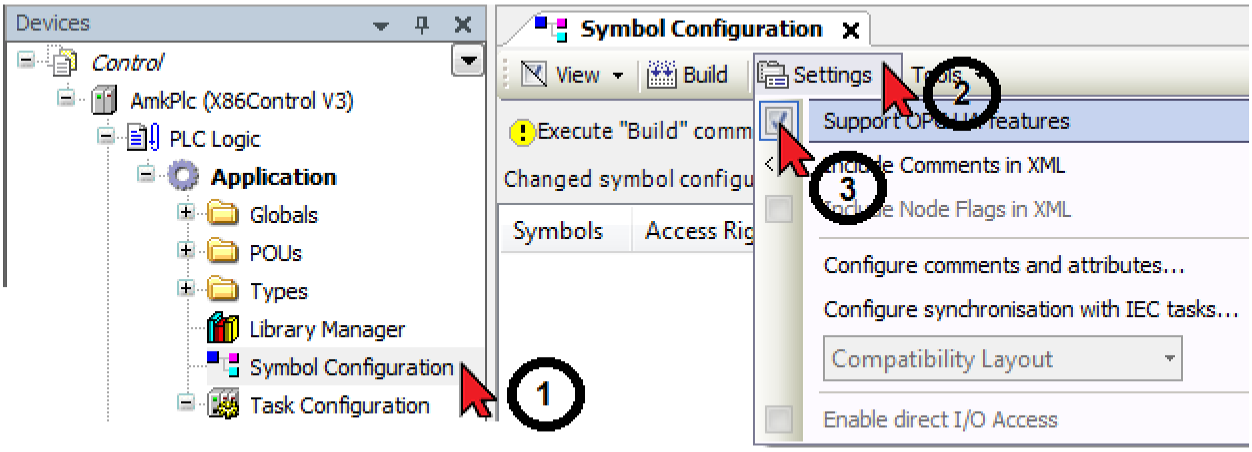
Access to PLC data objects is shown exemplary with the example project 'Plc_Easy01'

	03.04.2017 10:32:13	*** ApplInfo -- Info: Test OpcUa (17/14)., Date: Mo 2017-04-03 09:30:53 CEST ***	AmkIoDrv
	03.04.2017 10:32:13	*** ApplInfo -- Project: Plc_Easy01, Profile: CODESYS V3.5 SP10 Patch 1 AIPEX PRO, Version: 3.5.10.0, Autho...	AmkIoDrv
	03.04.2017 10:32:13	*** ProjInfo -- Info: Test OpcUa (17/14). ***	AmkIoDrv
	03.04.2017 10:32:13	*** ProjInfo -- Project: Plc_Easy01, Title: OpcUa_Easy, Version: 3.5.10.0, Author: EdH ***	AmkIoDrv
	03.04.2017 10:32:13	*** EVT_DownloadDone received: Application <Application> loaded ***	AmkIoDrv

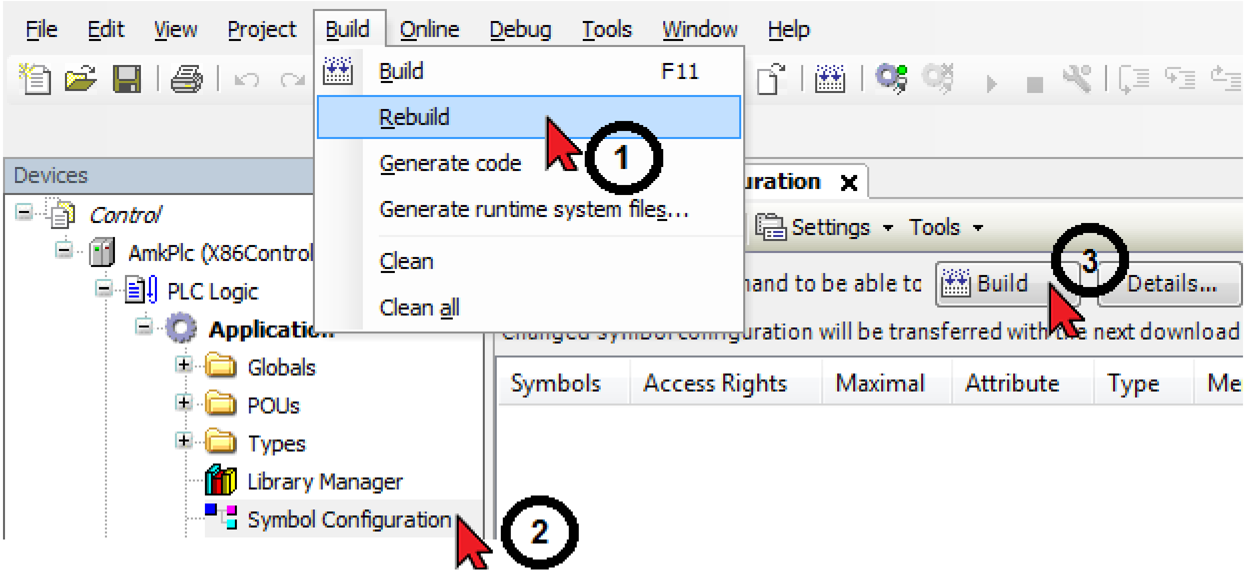
1. Insert the 'Symbol configuration' into the PLC project



2. Activate OPC UA

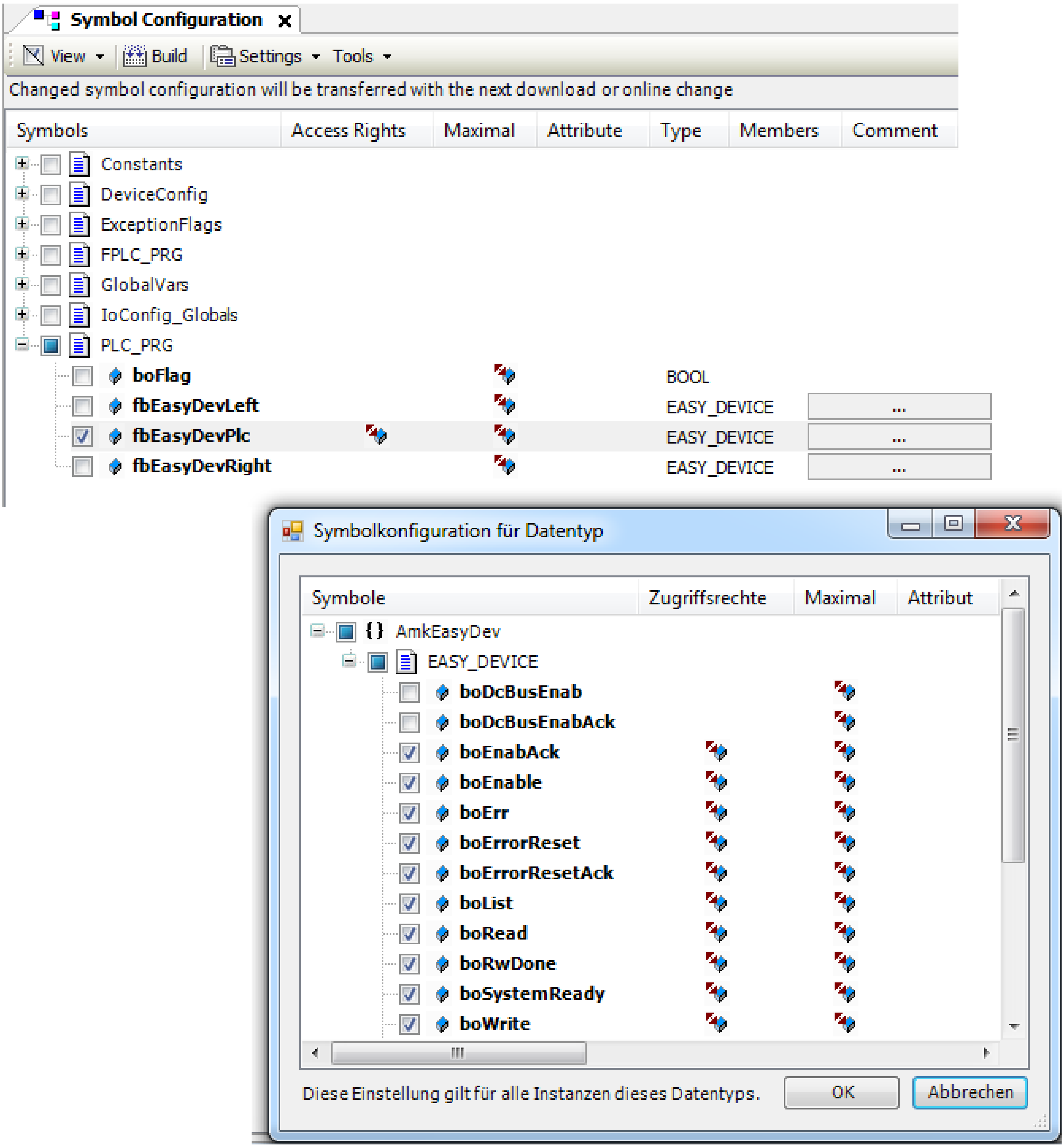


3. Create the variable list



4. Configure 'Symbol Configuration'

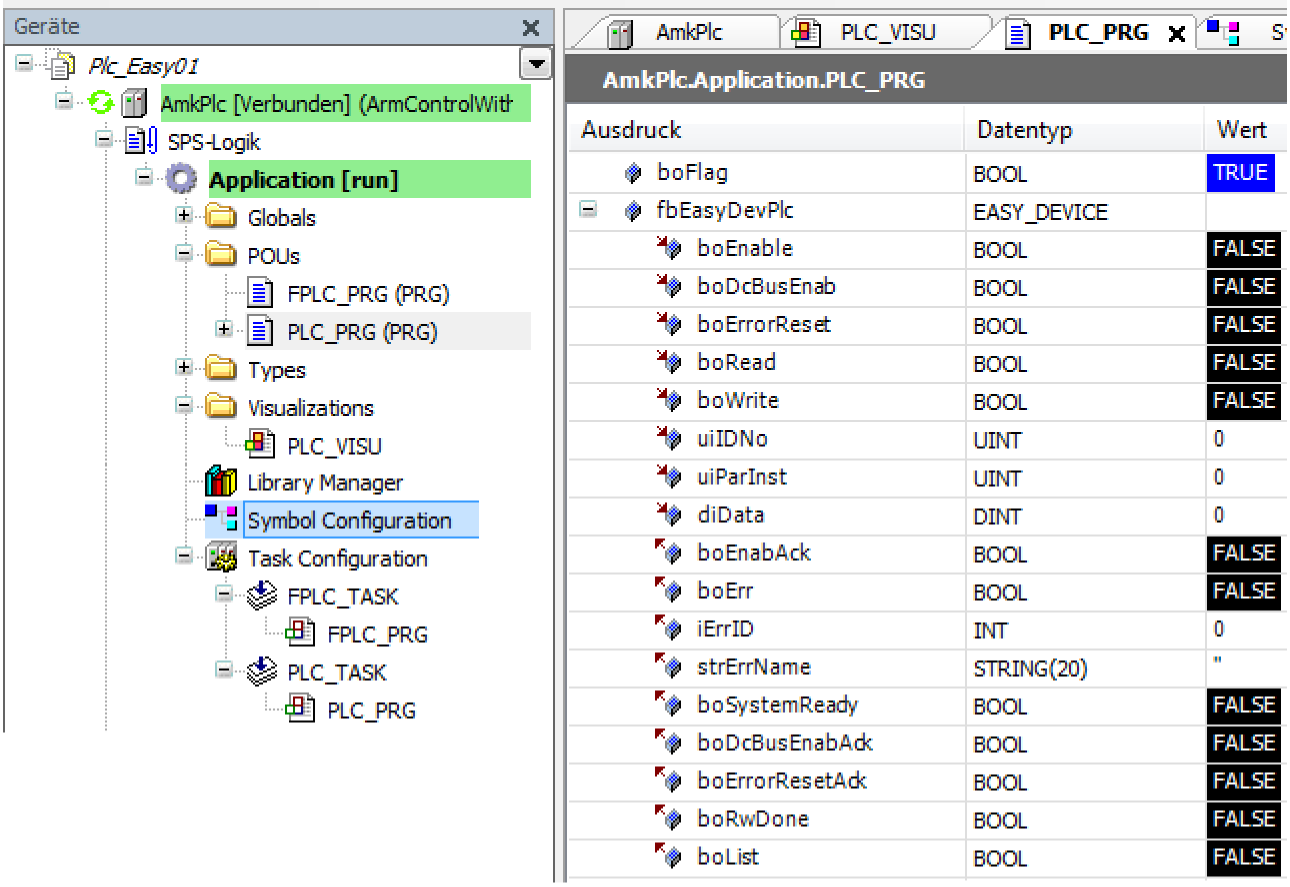
Activate the variables that are to be available for the OPC UA data exchange.



5. Project 'Create' and 'Log in'

The symbol configuration is transferred to the controller together with the application.

The figure shows the current data of the PLC.

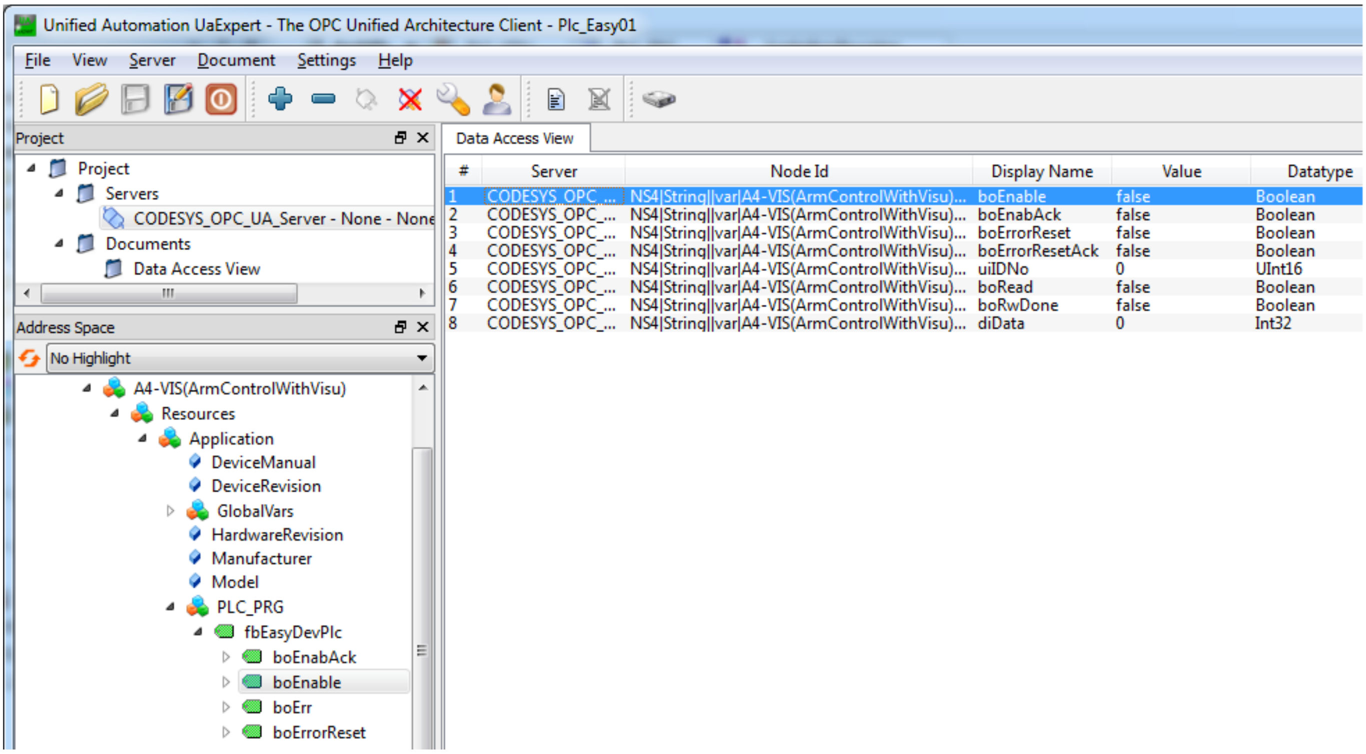


6. OPC UA Client

The figure shows the PLC data selected in the 'Symbol configuration'.

The free OPC UA client comes from the company Unified Automation Program: UaExpert.

Data can be exchanged in both directions. For example, in the client, control bits can be set in the drive.



4 AmkBase - Base function specific to AMK

AmkBase is an external AMK basic library which supports basic control functionality. It is divided into:

BasicFunctions	Basic functions
BasicSupport	Basic support functions
FastFunctions	Fast functions
System	System functions

4.1 BasicFunctions

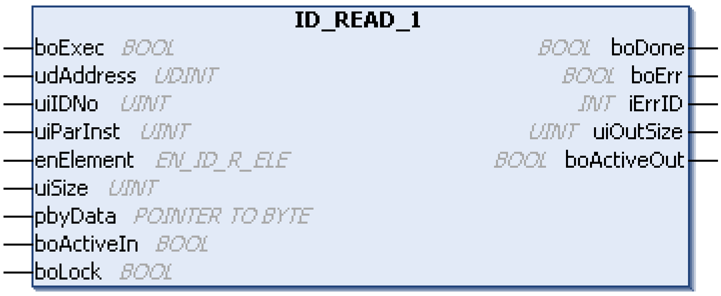
The following blocks are called from other libraries; they are not usually used directly by the application programmer.

ID_READ_1	Read AMK parameters (ID)
ID_WRITE_1	Write AMK parameters (ID)
TAB_CALC	Table calculation block

4.1.1 ID_READ_1 (FB)

The 'ID_READ_1' function block is used to read in AMK parameters across the system.
'ID_READ_1' is a base block that is called by other function blocks in the AMK libraries. (See AmkSystem system documentation.)

User interface



Input variables

Name	Type	Description
boExec	BOOL	Function execution: With a positive edge, the execution of the block starts. As long as 'boExec' = TRUE, the block is processed by the PLC. In the state 'boExec' = FALSE execution of the block is ended.
udAddress	UDINT	Routing address Based on the 'ST_NET_NO' structure
uiIDNo	UINT	ID number to be read out Special case: SDO index, if 'uiParInst' = 16#01xx
uiParInst	UINT	Parameter set number or instance number / instance of ID to be read Special case: SDO subindex, 'uiParInst' = 16#01xx (xx = subindex no.)

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>enElement</td><td>ENUM</td><td>EN_ID_R_ELE
Element of the parameter set / instance of the ID to be read.
<table>
<tr><td>Default</td><td>ID_R_ELE_DATA</td></tr>
</table>
<table>
<tr><td>Range</td><td>Meaning</td></tr>
<tr><td>ID_R_ELE_ NAME</td><td>Name</td></tr>
<tr><td>ID_R_ELE_ATTR</td><td>Attribute</td></tr>
<tr><td>ID_R_ELE_UNIT</td><td>Unit</td></tr>
<tr><td>ID_R_ELE_MIN</td><td>Minimum input value</td></tr>
<tr><td>ID_R_ELE_MAX</td><td>Maximum input value</td></tr>
<tr><td>ID_R_ELE_DATA</td><td>Value</td></tr>
</table></td></tr>
<tr><td>uiSize</td><td>UINT</td><td>Maximum data length available to accommodate the information to be read.
uiSize ≤ SIZEOF(variable) referenced by 'pbyData'!</td></tr>
<tr><td>pbyData</td><td>POINTER</td><td>POINTER TO READ DATA
Pointer referencing the structure / variable which is receiving the information read.</td></tr>
<tr><td>boActiveIn</td><td>BOOL</td><td>Active input to lock multiple parameter access attempts which cannot be interrupted</td></tr>
<tr><td>boLock</td><td>BOOL</td><td>Controller of the interlock mechanism
(in conjunction with 'boActiveIn' and 'boActiveOut')</td></tr>
</table>

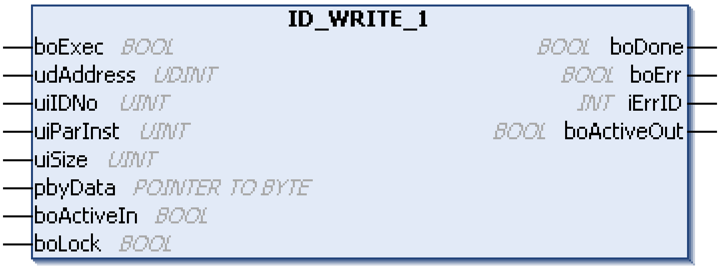
Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boDone</td><td>BOOL</td><td>Response that the function block has been completely executed.</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table></td></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Range: Siehe 'Error bit information' auf Seite 532.</td></tr>
<tr><td>uiOutSize</td><td>UINT</td><td>Current data length entered (read) in the structure referenced by the 'pbyData' pointer.</td></tr>
<tr><td>boActiveOut</td><td>BOOL</td><td>Active output to lock multiple parameter access attempts which cannot be interrupted</td></tr>
</table>

4.1.2 ID_WRITE_1 (FB)

The 'ID_WRITE_1' function block is used to write AMK parameters across the system.
'ID_WRITE_1' is a base block that is called by other function blocks in the AMK libraries. (see AmkSystem system documentation.)

User interface



Input variables

Name	Type	Description
boExec	BOOL	Function execution: With a positive edge, the execution of the block starts. As long as 'boExec' = TRUE, the block is processed by the PLC. In the state 'boExec' = FALSE execution of the block is ended.
udAddress	UDINT	Routing address Based on the 'ST_NET_NO' structure
uiIDNo	UINT	ID number to be written Special case: SDO index, if 'uiParInst' = 16#01xx
uiParInst	UINT	Parameter set number or instance number / instance Special case: SDO subindex, 'uiParInst' = 16#01xx (xx = subindex no.)
uiSize	UINT	Maximum data length of the information to be written. uiSize ≤ SIZEOF(variable) referenced by 'pbyData'!
pbyData	POINTER	POINTER TO WRITE DATA Pointer referencing the structure / variable which contains the information to be written.
boActiveIn	BOOL	Active input to lock multiple parameter access attempts which cannot be interrupted
boLock	BOOL	Controller of the interlock mechanism (in conjunction with 'boActiveIn' and 'boActiveOut')

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boDone</td><td>BOOL</td><td>Response that the function block has been completely executed.</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table></td></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Range: Siehe 'Error bit information' auf Seite 532.</td></tr>
<tr><td>boActiveOut</td><td>BOOL</td><td>Active output to lock multiple parameter access attempts which cannot be interrupted</td></tr>
</table>

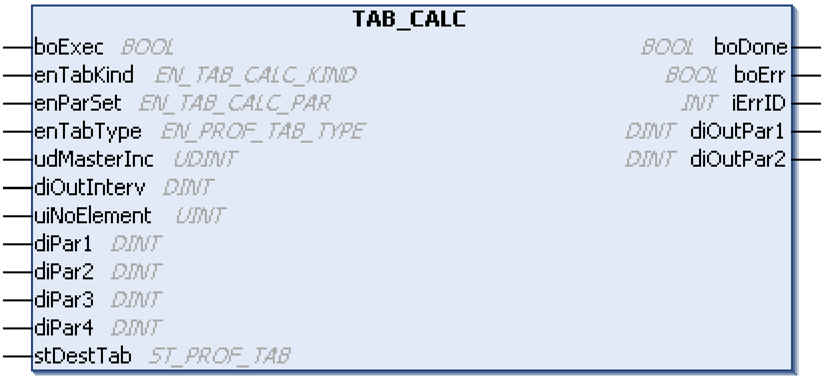
4.1.3 TAB_CALC (FB)

The 'TAB_CALC' function block is used to calculate table-based movement profiles.

'TAB_CALC' is a base block that is called by other function blocks in the AMK libraries. (see AmkTabc documentation)

The calculation of the table interpolation points commences as soon as 'boExec' sees a transition from FALSE -> TRUE. For reasons associated with processing time, the calculation process is distributed across a number of PLC cycles.
The calculation ends with 'boDone' or, in the event of an error, with 'boErr'. After this, 'boExec' should be set to FALSE.

User interface



Input variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boExec</td><td>BOOL</td><td>Function execution: With a positive edge, the execution of the block starts.
As long as 'boExec' = TRUE, the block is processed by the PLC.
In the state 'boExec' = FALSE execution of the block is ended.</td></tr>
<tr><td>enTabKind</td><td>ENUM</td><td>EN_TAB_CALC_KIND
Table type, for the specification of the fundamental table trend
<table>
<tr><td>Default</td><td>TAB_CALC_OP</td></tr>
</table>
<table>
<tr><th>Range</th><th>Meaning</th></tr>
<tr><td>TAB_CALC_IN</td><td>Phasing in table</td></tr>
<tr><td>TAB_CALC_OP</td><td>Operating table</td></tr>
<tr><td>TAB_CALC_OUT</td><td>Phasing out table</td></tr>
<tr><td>TAB_CALC_POSJLI</td><td>Positioning profile with jerk limitation</td></tr>
<tr><td>TAB_CALC_POS</td><td>Positioning profile without jerk limitation</td></tr>
</table></td></tr>
<tr><td>enParSet</td><td>ENUM</td><td>EN_TAB_CALC_PAR
Parameter set variant, for the selection of the description parameters
<table>
<tr><td>Default</td><td>TAB_CALC_PAR0</td></tr>
</table>
<table>
<tr><th>Range</th><th>Meaning</th></tr>
<tr><td>TAB_CALC_PAR0</td><td>First parameter set</td></tr>
<tr><td>TAB_CALC_PAR1</td><td>Second parameter set</td></tr>
<tr><td>TAB_CALC_PAR2</td><td>Third parameter set</td></tr>
</table></td></tr>
</table>

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>enTabType</td><td>ENUM</td><td>EN_PROF_TAB_TYPE
Table type, to differentiate between X and XY tables
<table><tr><td>Default</td><td>PROF_YTAB</td></tr></table>
<table>
<tr><th>Range</th><th>Meaning</th></tr>
<tr><td>PROF_YTAB</td><td>Equidistant X positions, Y positions defined by table value</td></tr>
<tr><td>PROF_XYTAB</td><td>X and Y position defined by table values</td></tr>
<tr><td>PROF_YTAB_NL</td><td>Equidistant X positions, Y positions defined by table value, not limited</td></tr>
<tr><td>PROF_XYTAB_NL</td><td>X and Y position defined by table values, not limited</td></tr>
</table></td></tr>
<tr><td>udMasterInc</td><td>UDINT</td><td>Increments of the master drive which produce a table cycle
Max. table X value
<table>
<tr><td>Range</td><td>0 ... 5000000</td></tr>
<tr><td>Unit</td><td>incr</td></tr>
<tr><td>Default</td><td>20000</td></tr>
</table></td></tr>
<tr><td>diOutInterv</td><td>DINT</td><td>Output interface defining the output increments per table cycle
Max. table Y value
<table>
<tr><td>Unit</td><td>incr</td></tr>
<tr><td>Default</td><td>20000</td></tr>
</table></td></tr>
<tr><td>uiNoElement</td><td>UINT</td><td>Element number of the last table element calculated,
number of table interpolation points
<table><tr><td>Default</td><td>100</td></tr></table></td></tr>
<tr><td>diPar1</td><td>DINT</td><td rowspan="4">Parameter n, based on the selected table type and parameter set variant
<table><tr><td>Default</td><td>360</td></tr></table></td></tr>
<tr><td>diPar2</td><td>DINT</td></tr>
<tr><td>diPar3</td><td>DINT</td></tr>
<tr><td>diPar4</td><td>DINT</td></tr>
</table>

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boDone</td><td>BOOL</td><td>Response that the function block has been completely executed.</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table></td></tr>
</table>

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Warning
<table>
<tr><th>Value</th><th>Meaning</th></tr>
<tr><td>1</td><td>Difference from adjacent point too great (>32767)</td></tr>
</table>
Error
<table>
<tr><th>Value</th><th>Meaning</th></tr>
<tr><td>1</td><td>Incorrect number of elements
the maximum number is dependent on the table type 'enTabType'</td></tr>
<tr><td>2</td><td>Incorrect parameter set variant
dependent on the table type 'enTabKind'</td></tr>
<tr><td>3</td><td>'udMasterInc' value too high</td></tr>
<tr><td>4</td><td>'diOutInterv' value too high / too low</td></tr>
<tr><td>5</td><td>'diPar1' illegal value
dependent on 'enTabType' and 'enTabKind'</td></tr>
<tr><td>6</td><td>'diPar2' illegal value
dependent on 'enTabType' and 'enTabKind'</td></tr>
<tr><td>7</td><td>'diPar3' illegal value
dependent on 'enTabType' and 'enTabKind'</td></tr>
<tr><td>8</td><td>'diPar4' illegal value
dependent on 'enTabType' and 'enTabKind'</td></tr>
<tr><td>9</td><td>Illegal synchronous point</td></tr>
<tr><td>10</td><td>Illegal phasing in point</td></tr>
<tr><td>11</td><td>Illegal phasing out point</td></tr>
<tr><td>12</td><td>Illegal sine starting point</td></tr>
<tr><td>13</td><td>Velocity too low</td></tr>
<tr><td>14</td><td>Acceleration too low</td></tr>
</table>
</td></tr>
<tr><td>diOutPar1</td><td>DINT</td><td rowspan="2">Output parameter based on the selected table type and parameter set variant</td></tr>
<tr><td>diOutPar2</td><td>DINT</td></tr>
</table>

Input and output variables

Name	Type	Description
stDestTab	STRUCT	ST_PROF_TAB Profile table structure

4.2 BasicSupport

The user has direct access to BasicSupport functions and function blocks.

ADD_LIMIT	Addition with limitation
ANALOG_TO_I	AMK AD converter
DI_TO_COUNT	AMK 32-bit counter output value converter
FboGetLocalTimeInfo	Get local time information
FboSetNetControl	Control of network behavior
FboTestFlagAndSet	Read-Modify-Write protected flag organization
FdiGetSysTime	Get time differences
FdwRandom	Random number generator

FuiGetNetStatus	Get network status
FwGetTargetInfo	Get information about the target system
PID_TO_KPKIKD	AMK PID converter
TIME_TO_COUNT	TimeStamp converter

4.2.1 ADD_LIMIT (FB)

The 'ADD_LIMIT' function block adds two DINT type variables. The result is also DINT.
The result can be limited to a minimum or a maximum value.

Thus:

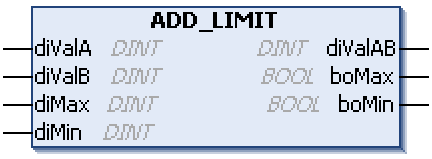
diValAB :=:= diValA + diValB where: diMin ≤ diValAB ≤ diMax

And:

boMin = TRUE if (diValA + diValB) < diMin
boMax = TRUE if (diValA + diValB) > diMax

Application note:
Combined with the 'PID_CTRL' function block, 'ADD_LIMIT' supports additive feed-forward control with limitation of the command variable, for example

User interface



Input variables

Name	Type	Description
diValA	DINT	Input value A, added to input value B
diValB	DINT	Input value B, added to input value A
diMax	DINT	Maximum permissible value
diMin	DINT	Minimum permissible value

Output variables

Name	Type	Description
diValAB	DINT	Output value AB Results from the addition of input value A and input value B taking the limit into account
boMax	BOOL	boMax = TRUE: The output value AB has been limited to the maximum permissible output value
boMin	BOOL	boMin = TRUE: The output value AB has been limited to the minimum permissible output value

4.2.2 ANALOG_TO_I (FB)

The 'ANALOG_TO_I' function block converts the 12-bit A/D converter number notation that is specific to AMK into a 16-bit two's complement.
Downstream, the converted values can be processed with standard blocks or used as input values for the PID controller.

A/D converter number format specific to AMK

Voltage value	Number notation specific to AMK	Two's complement
-10 V	16#0000	16#F800
0 V	16#0800	16#0000
+10 V	16#0FFF	16#07FF

User interface

Input variables

Name	Type	Description
iAnalog	INT	Analog value in AMK format (see table) Range: 0 ... 4095 Default: 2048

Output variables

Name	Type	Description
iValOut	INT	Analog value as two's component Range: -2048 ... 2047

4.2.3 DI_TO_COUNT (FB)

The 'DI_TO_COUNT' function block converts the 32-bit AMK pulse encoder information into various values that are relevant to the process.

User interface

Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.
diInVal	DINT	Input value Pulse encoder information 32-bit AMK data format: $\mathrm{diInVal}_{LW}$ = low word: latched counter reading for zero pulse $\mathrm{diInVal}_{HW}$ = high word: current 16-bit counter reading

Output variables

Name	Type	Description
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled
boErr	BOOL	The function block is in an error state FALSE: No error (permitted commanding or warning) TRUE: Error
iErrID	INT	Error identity number: Diagnostic number is output iErrID = 0: No error iErrID ≠ 0, boErr = TRUE: Error iErrID ≠ 0, boErr = FALSE: Warning
boRefPulse	BOOL	Homing pulse: The output adopts the value 'boRefPulse' = TRUE for one cycle Display of zero pulse detected through the input logic of the square-wave pulse encoder
diCount	DINT	32-bit Counter value, generated for each cycle from the changes in the value of the current 16-bit counter reading The value is reset on a positive edge at 'boEnable' ('diCount' = 0)
diOffset	DINT	Offset of the counter value to the homing pulse Unit: Incr $diOffset = diInVal_{LW} - diInVal_{HW}$

Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

4.2.4 FboGetLocalTimeInfo (F)

The 'FboGetLocalTimeInfo' function identifies the difference between the local time and Coordinated Universal Time (UTC).

User interface

Input variables

Name	Type	Description
pstLocalTimeInfo	POINTER	POINTER TO ST_LOCAL_TIME_INFO Pointer to the structure of the local time information

Output variables

Name	Type	Description
FboGetLocalTimeInfo	BOOL	Return value from get time call (not relevant at the current time)

4.2.5 FboSetNetControl (F)

The 'FboSetNetControl' function controls network behavior.

The 'uiControl' input variable is used to select control information. The function varies from one bus system to another; reference should be made to the corresponding network description.

ID33114 'Process number' displays the current status.

Variants for EtherCAT

- Bus start and stop
  Bit 0: Start / Stop (bit 1 - bit 15 = 0)
- User-defined control of the bus states
  Bit 8 - Bit 15: Request of the user-defined bus state (bit 0 - bit 7 = 0)
  Bit 0 = 1 (Bus active)

User interface

Input variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>uiChannel</td><td>UINT</td><td>Selection of communication instance
Network channel number / instance according to ID34140 'AS BUS protocol'
<table><tr><td>Range</td><td>0 ... 7</td></tr></table>
<table>
<tr><th>Instance</th><th>Use</th></tr>
<tr><td>0</td><td>-</td></tr>
<tr><td>1</td><td>ACC bus master</td></tr>
<tr><td rowspan="5">2</td><td>Profibus slave (option A-SPB)</td></tr>
<tr><td>EtherCAT slave (option A-SEC)</td></tr>
<tr><td>CAN / ACC bus slave (option A-SCN)</td></tr>
<tr><td>EtherNet/IP (option A-SIP)</td></tr>
<tr><td>Profinet IO Device (option A-SPN)</td></tr>
<tr><td>3</td><td>I/O option</td></tr>
<tr><td>4</td><td>1st Ethernet X20</td></tr>
<tr><td>5</td><td>EtherCAT master (option A-MEC)</td></tr>
<tr><td>6</td><td>Reserved</td></tr>
<tr><td>7</td><td>2nd Ethernet X60</td></tr>
</table></td></tr>
<tr><td>uiAxisNo</td><td>UINT</td><td>Axis ID number
'uiAxisNo' = 0: Selection of communication modules on the same physical system (PLC internal)
State changes to the slaves are not allowed.</td></tr>
<tr><td>uiControl</td><td>UINT</td><td>Bus-specific network control
<table><tr><td>Value
(e.g. for CAN)</td><td>'uiControl' bit 0 = 1: activation of initialization
(bit 1 ... 15 not used)</td></tr></table>
<table>
<tr><th>EtherCAT</th><th colspan="2">bit 8 - bit 15
user-defined</th><th>bit 1 - bit 7</th><th>bit 0
bus control</th></tr>
<tr><td>Variant 1:
Bus start and stop</td><td colspan="2">not used</td><td>not used</td><td>bit 0 = 1 Start
bit 0 = 0 Stop</td></tr>
<tr><td rowspan="6">Variant 2:
User-defined</td><td>Basic state</td><td>0x0</td><td rowspan="6">not used</td><td rowspan="6">bit 0 = 1 active
bit 0 = 0 not active</td></tr>
<tr><td>BUS_INIT</td><td>0x1</td></tr>
<tr><td>BUS_PREOP</td><td>0x2</td></tr>
<tr><td>BUS_ BOOTSTRAP</td><td>0x3</td></tr>
<tr><td>BUS_SAFEOP</td><td>0x4</td></tr>
<tr><td>BUS_OP</td><td>0x8</td></tr>
</table></td></tr>
</table>

Name	Type	Description
uiMask	UINT	Bit selection mask Selection of individual items of bit information from 'uiControl' for manipulation.

Output variables

Name	Type	Description
FboSetNetControl	BOOL	Acknowledges the network control order

Example EtherCAT

Input variable uiControl:

Command	Meaning
0x0000	Switch to the basic state
0x0001	Switch to highest possible state
0x0801	Switch to Operational Mode

Allowed global state changes with EtherCAT

Actual state			Allowed state			Remarks
Basic state	=	0	BUS_INIT	=	0x0101	Basic state after Power On.
			BUS_PREOP	=	0x0201	During operation, only a bus stop can switch to the basic state.
			BUS_SAFEOP	=	0x0401	
			BUS_OP	=	0x0801	E. g. in the case of a network extension with a new node
BUS_INIT	=	1	BUS_INIT	=	0x0101	-
			BUS_PREOP	=	0x0201	
			BUS_SAFEOP	=	0x0401	
			BUS_OP	=	0x0801	
BUS_PREOP	=	2	BUS_INIT	=	0x0101	-
			BUS_PREOP	=	0x0201	
			BUS_SAFEOP	=	0x0401	
			BUS_OP	=	0x0801	
BUS_SAFEOP	=	4	BUS_INIT	=	0x0101	Switch to PREOP not allowed
			BUS_SAFEOP	=	0x0401	
			BUS_OP	=	0x0801	
BUS_OP	=	8	BUS_INIT	=	0x0101	Switch to PREOP and SAFEOP not allowed
			BUS_OP	=	0x0801	

4.2.6 FboTestFlagAndSet (F)

The 'FboTestFlagAndSet' uses a BOOL type flag variable (semaphore) to execute a Read-Modify-Write operation.

This type of flag variable can be used, for example, to organize data exchange between programs associated with different (preemptive) tasks which interrupt one another.

User interface

Input variables

Name	Type	Description
pboFlag	POINTER	POINTER TO BOOL Flag pointer transferred, for example, with 'ADR' (boFlag)

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>FboTestFlagAndSet</td><td>BOOL</td><td>Result of flag query<table><tr><td>TRUE</td><td>If 'pboFlag^' = FALSE
'pboFlag^' := TRUE is set implicitly without interruption by another task being possible between the comparison and the setting of the value</td></tr><tr><td>FALSE</td><td>If 'pboFlag^' = TRUE</td></tr></table></td></tr>
</table>

4.2.7 FdiGetSysTime (F)

The 'FdiGetSysTime' queries time differences based on an internal system time base.

The time base is updated independently of the PS cycle time but dependent upon the target system. The maximum time difference that can be measured is also determined by the target system.

User interface

Input variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>iSelect</td><td>INT</td><td>Time selection used to select the time determination mode<table><tr><td>Value</td><td>Meaning</td></tr><tr><td>0</td><td>Time difference; time between two calls</td></tr></table></td></tr>
</table>

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>FdiGetSysTime</td><td>DINT</td><td>System time<table><tr><td>Unit</td><td>µs</td></tr></table></td></tr>
</table>

4.2.8 FdwRandom (F)

The 'FdwRandom' function is a random generator.

User interface

Input variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>wHandle</td><td>WORD</td><td>Handle for selection of algorithm
<table>
<tr><td>Value</td><td>Meaning</td></tr>
<tr><td>0</td><td>Virtually-binary white noise at bit 0</td></tr>
<tr><td>1</td><td>Optimized random algorithm
The periodicity of the algorithm is 2^{32}-1</td></tr>
</table>
</td></tr>
</table>

Output variables

Name	Type	Description
FdwRandom	DWORD	Random number

4.2.9 FuiGetNetStatus (F)

The 'FuiGetNetStatus' function queries the network status.
The meaning of the status information is determined by the corresponding bus system; reference should be made to the associated descriptions.

User interface

Input variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>uiChannel</td><td>UINT</td><td>Selection of communication instance
Network channel number / instance according to ID34140 'AS BUS protocol'
<table>
<tr><td>Range</td><td>0 ... 7</td></tr>
</table>
<table>
<tr><th>Instance</th><th>Use</th></tr>
<tr><td>0</td><td>-</td></tr>
<tr><td>1</td><td>ACC bus master</td></tr>
<tr><td rowspan="5">2</td><td>Profibus slave (option A-SPB)</td></tr>
<tr><td>EtherCAT slave (option A-SEC)</td></tr>
<tr><td>CAN / ACC bus slave (option A-SCN)</td></tr>
<tr><td>EtherNet/IP (option A-SIP)</td></tr>
<tr><td>Profinet IO Device (option A-SPN)</td></tr>
<tr><td>3</td><td>I/O option</td></tr>
<tr><td>4</td><td>1st Ethernet X20</td></tr>
<tr><td>5</td><td>EtherCAT master (option A-MEC)</td></tr>
<tr><td>6</td><td>Reserved</td></tr>
<tr><td>7</td><td>2nd Ethernet X60</td></tr>
</table>
</td></tr>
<tr><td>uiAxisNo</td><td>UINT</td><td>Axis ID number
'uiAxisNo' = 0: Selection of communication modules on the same physical system (PLC internal)</td></tr>
</table>

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>FuiGetNetStatus</td><td>UINT</td><td>Network status
<table>
<tr><td>Bit</td><td>Meaning</td></tr>
<tr><td>0</td><td>Module ready for operation</td></tr>
<tr><td>1</td><td>Network ready (preoperational mode)</td></tr>
<tr><td>2</td><td>Error</td></tr>
<tr><td>3</td><td>Warning</td></tr>
<tr><td>4</td><td>Operational mode</td></tr>
<tr><td>5</td><td>Reserved</td></tr>
<tr><td>6</td><td>Reserved</td></tr>
<tr><td>7</td><td>Bit 0 valid (module ready)</td></tr>
<tr><td>8</td><td>Link in (e.g. X85, X185, etc.)</td></tr>
<tr><td>9</td><td>Link out (e.g. X86, X186, X20, X60, X137)</td></tr>
<tr><td>10...15</td><td>Not currently used</td></tr>
</table></td></tr>
</table>

4.2.10 FwGetTargetInfo (F)

The 'FwGetTargetInfo' function reads in information from the target system.

User interface

Input variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>enSelect</td><td>ENUM</td><td>EN_TARGET_INFO
Selection of target system information
<table>
<tr><td>Default</td><td>TAR_VERSION</td></tr>
</table>
<table>
<tr><td>Range</td><td>Meaning</td></tr>
<tr><td>TAR_VERSION</td><td>Date of target system version</td></tr>
<tr><td>ADDR_SWITCH</td><td>Not currently used</td></tr>
<tr><td>ACTIVE_BUS</td><td>'Bus active' information</td></tr>
<tr><td>LIFE_CYCLE</td><td>Not currently used</td></tr>
<tr><td>MAGIC_LOW</td><td>Not currently used</td></tr>
</table></td></tr>
</table>

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>FwGetTargetInfo</td><td>WORD</td><td>Target system information
<table>
<tr><th>Value</th><th>Meaning</th></tr>
<tr><td>16#yyww</td><td>enSelect = TAR_VERSION
yy - year, ww = calendar week
(e.g. 16#1345: calendar week 45 of year 2013)</td></tr>
<tr><td>2#xxxx.xxx0</td><td>enSelect = ACTIVE BUS
Bit 1 ... bit 7 are assigned to the corresponding instances 1 ... 7 according to ID2'SERCOS cycle time'. The corresponding bit is TRUE if the bus cycle is active.
<table>
<tr><th>Instance</th><th>Use</th></tr>
<tr><td>0</td><td>-</td></tr>
<tr><td>1</td><td>ACC bus master</td></tr>
<tr><td rowspan="5">2</td><td>Profibus slave (option A-SPB)</td></tr>
<tr><td>EtherCAT slave (option A-SEC)</td></tr>
<tr><td>CAN / ACC bus slave (option A-SCN)</td></tr>
<tr><td>EtherNet/IP (option A-SIP)</td></tr>
<tr><td>Profinet IO Device (option A-SPN)</td></tr>
<tr><td>3</td><td>I/O option</td></tr>
<tr><td>4</td><td>1st Ethernet X20</td></tr>
<tr><td>5</td><td>EtherCAT master (option A-MEC)</td></tr>
<tr><td>6</td><td>Reserved</td></tr>
<tr><td>7</td><td>2nd Ethernet X60</td></tr>
</table>
This information is of interest if the bus cycle time does not match the PGT cycle time</td></tr>
</table>
</td></tr>
</table>

4.2.11 PID_TO_KPKIKD (FB)

The 'PID_TO_KPKIKD' function block calculates the proportional time, the reset time, the derivative time, and the sampling time for the controller parameters Kp, Ki, and Kd from the proportional component. The 'PID_CTRL' function block requires these values as input values.

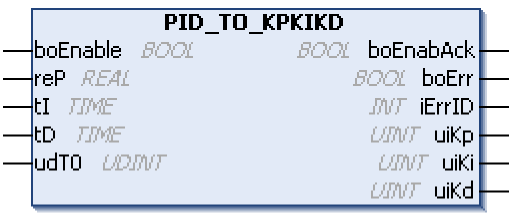
The function block is called in the asynchronous program level PLC_PRG.

Thus:

'PID_CTRL' combines with 'PID_TO_KPKIKD' to form a PID controller whose standard control parameters are independent of the sampling time.
This separation makes it possible for the two modules to work with different time frequencies in applications with limited processing time.

The time-consuming calculation of Kp, Ki, Kd with 'PID_TO_KPKIKD' can be performed at asynchronous program level PLC_PRG while the runtime-optimized control algorithm is being processed at synchronous program level FPLC_PRG.

User interface



Input variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnable</td><td>BOOL</td><td>Enable signal: With a positive edge, the initialisation of the block starts.
As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC.
In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.</td></tr>
<tr><td>reP</td><td>REAL</td><td>Proportional component
Internal normalization 1/256:
reP = 1 -> Kp = 256 -> P gain of PID_CTRL = 1</td></tr>
<tr><td>tI</td><td>TIME</td><td>Integration time constant: reset time Tn<table><tr><td>Unit</td><td>ms</td></tr><tr><td>Default</td><td>4294967295</td></tr></table></td></tr>
<tr><td>tD</td><td>TIME</td><td>Differentiation time constant: derivative time Tv<table><tr><td>Unit</td><td>ms</td></tr></table></td></tr>
<tr><td>udT0</td><td>UDINT</td><td>Sampling time for processing the PID algorithm<table><tr><td>Unit</td><td>0.001 ms</td></tr><tr><td>Default</td><td>1000</td></tr></table></td></tr>
</table>

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state<table><tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr><tr><td>TRUE</td><td>Error</td></tr></table></td></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output<table><tr><td colspan="2">iErrID = 0</td><td>No error</td></tr><tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr><tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr></table><table><tr><th>Value</th><th>Meaning</th></tr><tr><td>1</td><td>Kp rounded to 0</td></tr><tr><td>2</td><td>Kp rounded to 1</td></tr><tr><td>3</td><td>Kp limited to 32767</td></tr><tr><td>4</td><td>Ki rounded to 0</td></tr><tr><td>5</td><td>Ki rounded to 1</td></tr><tr><td>6</td><td>Ki limited to 32767</td></tr><tr><td>7</td><td>Kd rounded to 0</td></tr><tr><td>8</td><td>Kd rounded to 1</td></tr><tr><td>9</td><td>Kd limited to 32767</td></tr></table></td></tr>
<tr><td>uiKp</td><td>UINT</td><td>Proportional gain (P) of the PID controller<table><tr><td>Unit</td><td>1/256</td></tr></table></td></tr>
<tr><td>uiKi</td><td>UINT</td><td>Integration gain (I) of the PID controller.<table><tr><td>Unit</td><td>1/256</td></tr></table></td></tr>
<tr><td>uiKd</td><td>UINT</td><td>Differential gain (D) of the PID controller<table><tr><td>Unit</td><td>1/256</td></tr></table></td></tr>
</table>

4.2.12 TIME_TO_COUNT (FB)

The 'TIME_TO_COUNT' function block converts the time difference measured with the TimeStamp blocks into a position reference, for example (see AmkDevAccess documentation).

User interface

Input variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnable</td><td>BOOL</td><td>Enable signal: With a positive edge, the initialisation of the block starts.
As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC.
In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.</td></tr>
<tr><td>boTimeAck</td><td>BOOL</td><td>Signal identifying a new time value.</td></tr>
<tr><td>diTime</td><td>DINT</td><td>Measured time value; time difference compared with position to be determined<table><tr><td>Unit</td><td>ns</td></tr></table></td></tr>
<tr><td>diInVal</td><td>DINT</td><td>Input value; e.g. path increments
Used to calculate 'diCount' and 'diOffset'</td></tr>
</table>

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state<table><tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr><tr><td>TRUE</td><td>Error</td></tr></table></td></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output<table><tr><td colspan="2">iErrID = 0</td><td>No error</td></tr><tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr><tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr></table><table><tr><td>Value</td><td>Meaning</td></tr><tr><td>1</td><td>'diOffset' too high; the value is limited to the maximum</td></tr></table></td></tr>
<tr><td>boRefPulse</td><td>BOOL</td><td>Homing pulse
Derived directly from 'boTimeAck'</td></tr>
<tr><td>diCount</td><td>DINT</td><td>Counter value, which represents the sum of the input value differences per cycle
Thus:</td></tr>
<tr><td>diOffset</td><td>DINT</td><td>Offset of the counter value to the homing pulse<table><tr><td>Unit</td><td>Incr</td></tr></table>Thus:
where
T_{PGT} [ns] = ID2 x 1000
ΔdiInVal = $diInVal_k$ - $diInVal_{k-1}$</td></tr>
</table>

Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device. Based on 'stDevice.uiCycleTime', the time T_{PGT} is also determined

4.3 FastFunctions

The function blocks in the 'FastFunctions' group must be called in the 'FPLC_PRG' program block which is called in the PLC as a real-time task with equidistant sampling. These function blocks generate output values whose increment by sampling time is used as the cyclic setpoint for drive control.

CAM_CONT	Camshaft control
CAM_CONT_1	Camshaft control with higher accuracy of position assignment
CAM_PROF	Table-based function interpolator
CAM_PROF_1	Table-based function interpolator with additional output of the 1st and 2nd derivation of the table function
PID_CTRL	PID controller
PM_CORRECT	Printing mark correction
PM_DETECT	Printing mark detection and correction value generation
POS	Fast positioning module
POS_1	Fast positioning module with with extended functionality
POS_AJ	Fast positioning module with variable specification of the position, velocity, acceleration and jerk
RATIO_ABS	Multiplication and division
RATIO_INC	Multiplication and division based on exact increments
RATIO_INC_1	Multiplication and division based on exact increments
VGEN	Velocity setpoint
VGEN_A	Velocity setpoint with limited acceleration
VGEN_AJ	Velocity setpoint with limited acceleration and jerk

4.3.1 CAM_CONT (FB)

The 'CAM_CONT' function block is a cam switch.

It controls a binary output variable (cam) as a function of the 'diInVal' input variable. The input variable can be a position value or a temporal value, for example.
The signal states of the binary output are defined with a cam table.
The switching points are defined based on the setting of the cam on / off position ('diOn' / 'diOff') in the cam table.
Up to 16 cams can be distributed on the track at will.
The switching points can be changed in the cam table "online", i.e. while the function block is activated ('boEnable' = TRUE).
Each block instance constitutes a cam track.

User interface

Input variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnable</td><td>BOOL</td><td>Enable signal: With a positive edge, the initialisation of the block starts.
As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC.
In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.</td></tr>
<tr><td>enMode</td><td>ENUM</td><td>EN_CAM_CONT_MODE
Selection mode between incremental and absolute input evaluation
<table><tr><td>Default</td><td>CAM_CONT_INC</td></tr></table>
<table><tr><th>Range</th><th>Meaning</th></tr><tr><td>CAM_CONT_INC</td><td>Incremental input value evaluation</td></tr><tr><td>CAM_CONT_ABS</td><td>Absolute input value evaluation</td></tr></table></td></tr>
<tr><td>diInVal</td><td>DINT</td><td>Input value of the camshaft control (position)</td></tr>
<tr><td>udModulo</td><td>UDINT</td><td>Modulo value
In mode 'enMode' = CAM_CONT_INC, this is the value at which cam table evaluation restarts at "0"
<table><tr><td>Range</td><td>0 … $+2^{31}-1$</td></tr><tr><td>Default</td><td>20000</td></tr></table></td></tr>
<tr><td>tFilter</td><td>TIME</td><td>Filter time constant
Attenuates the impact of changes in velocity in the context of dead-time compensation
<table><tr><td>Default</td><td>t#1 ms</td></tr></table></td></tr>
<tr><td>tDelay</td><td>TIME</td><td>Dead-time constant
To calculate the offset of the binary information depending on the current velocity in the context of dead-time compensation
<table><tr><td>Default</td><td>t#0 ms (dead-time compensation not active)</td></tr></table></td></tr>
<tr><td>uiHyst</td><td>UINT</td><td>Hysteresis value
(H), applied to the on and off edges (X_{on}, X_{off}) of a cam signal
<table><tr><td>Default</td><td>0 (hysteresis not active)</td></tr></table>
In conjunction with dead-time compensation, the hysteresis must be set higher than the dead-time compensation path Xdead

Thus:
$X_{dead} = T_{dead} * n * G / 60000$

where
<table><tr><td>X_{dead}</td><td>Dead-time compensation path [incr]</td></tr><tr><td>T_{dead}</td><td>Dead time [ms]</td></tr><tr><td>n</td><td>Speed [rpm]</td></tr><tr><td>G</td><td>Encoder resolution [incr/rev]</td></tr></table>
In incremental input value evaluation mode ('enMode' = CAM_CONT_INC), the following must be true:
<table><tr><td>$H < udModulo -(X_{off} - X_{on})$</td><td>for $X_{off} > X_{on}$</td></tr><tr><td>$H < X_{on} - X_{off})$</td><td>for $X_{off} < X_{on}$</td></tr></table></td></tr>
</table>

Name	Type	Description
pstTab	POINTER	POINTER TO ST_CONT_TAB Pointer to the cam table

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table></td></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
<table>
<tr><td>Value</td><td>Meaning</td></tr>
<tr><td>1</td><td>Modulo value limited to maximum</td></tr>
<tr><td>2</td><td>Filter time constant set to 1</td></tr>
<tr><td>3</td><td>Filter time constant limited to maximum</td></tr>
<tr><td>4</td><td>Dead-time constant set to 0</td></tr>
<tr><td>5</td><td>Dead-time constant set to 1</td></tr>
<tr><td>6</td><td>Dead-time constant limited to maximum</td></tr>
<tr><td>7</td><td>Modulo value = 0 when mode CAM_CONT_INC</td></tr>
</table></td></tr>
<tr><td>boOutVal</td><td>BOOL</td><td>Cam Output signal</td></tr>
</table>

Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

Description

The camshaft control has the following properties:

- Incremental or absolute mode
- Filter in the context of dead-time compensation
- Dead-time compensation
- Hysteresis

Mode

- **Set incremental input value** ('enMode' = CAM_CONT_INC):
  The 'diInVal' input variable is processed as a 32-bit signed fixed-point number (32-bit integer value).
  In response to every call, the block generates the input value differences from two consecutive items of input information and adds these up to a positive 32-bit value. The internal counter works modulo; in other words, it counts up to a configurable final value 'udModulo' and then starts again at zero.
- **Set absolute input value** ('enMode' = CAM_CONT_ABS):
  The 'diInVal' input variable is processed as a 32-bit signed fixed-point number (32-bit integer value). Overshoot at the end of the travel range is limited.

Filter

To attenuate the impact of changes in velocity for dead-time compensation, multiple speed values are averaged. The 'tFilter' filter time constant determines the number of velocity values for which averaging is performed (number = 'tFilter' [ms]/stDevice.uiCycleTime [ms]).

Dead-time compensation

For dead-time compensation the binary information is offset leading based on the current velocity. The 'tDelay' dead-time constant accounts for the time taken to calculate the offset.

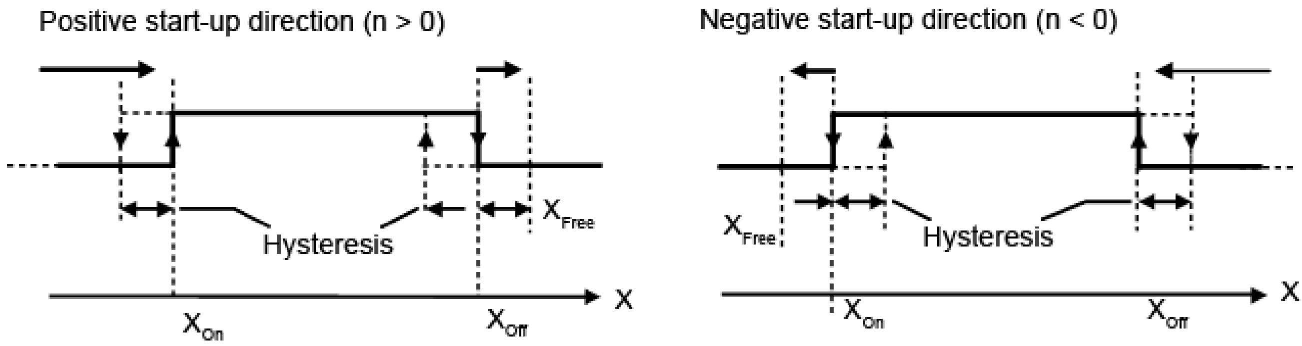
Hysteresis

The hysteresis ensures that the binary output always adopts a stable state, even if the input value of the block is moving around a rising or falling cam edge at the time.

The generation of the hysteresis (X_{on}, X_{off}) is illustrated in the figure below:

- Positive approach direction (n > 0; X increasing)
- Negative approach direction (n < 0; X decreasing)

Abbildung 2: CAM_CONT: Hysteresis generation



A "positive approach direction" results in the following behavior at the binary output (cam):

- Cam information "0" is output starting from a position $X < X_{on}$.
- Cam information "1" is output as of position $X \geq X_{on}$.
- Cam information "1" is retained during reverse rotation to position $X \geq X_{on}$–H.
  Cam information "0" is output during further reverse rotation to position $X < X_{on}$–H.
- Cam information "0" is output during forward rotation starting from position $X \geq X_{off}$.
  - Cam information "0" is retained in the event of reverse rotation to position $X \geq X_{off}$–H before position $X = X_{free}$= X_{off}+H is reached.
    Cam information "1" is output during further reverse rotation.
  - In the event of reverse rotation after position $X \geq X_{free}$ has been reached, the cam signal is generated according to the "negative approach direction".

A "negative approach direction" results in the following behavior:

- Cam information "0" is output starting from a position $X \geq X_{off}$.
- Cam information "1" is output as of position $X < X_{off}$.
- Cam information "1" is retained during forward rotation to position $X < X_{off}$+H.
  Cam information "0" is output during further forward rotation.
- Cam information "0" is output during reverse rotation starting from position $X < X_{on}$.
  - Cam information "0" is retained in the event of reverse rotation to position $X < X_{on}$+H prior to overshooting position $X = X_{free} = X_{on}$–H.
    Cam information "1" is output during further forward rotation.
  - In the event of reverse rotation after position $X < X_{free}$ has been reached, the cam signal is generated according to the "positive approach direction".

Switchover between hysteresis generation of positive (negative) approach direction takes place once a cam has completed its rotation and position X_{free} has been reached or overshot.

4.3.2 CAM_CONT_1 (FB)

The 'CAM_CONT_1' function block is a cam switch with higher accuracy and is based on the function block CAM_CONT.

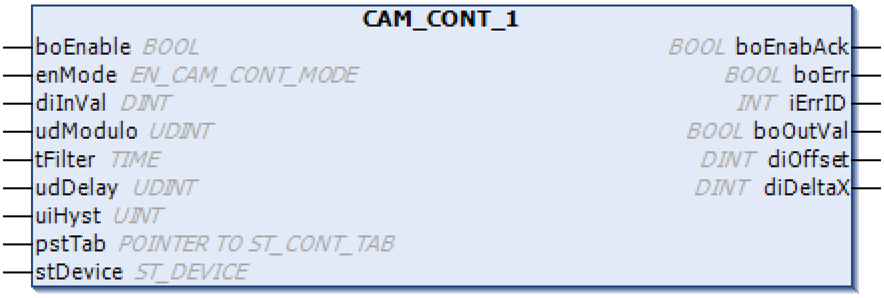
It controls a binary output variable (cam) as a function of the 'diInVal' input variable. The input variable can be a position value or a temporal value, for example.
The signal states of the binary output are defined with a cam table.
The switching points are defined based on the setting of the cam on / off position ('diOn' / 'diOff') in the cam table.
Up to 16 cams can be distributed on the track at will.
The switching points can be changed in the cam table "online", i.e. while the function block is activated ('boEnable' = TRUE).
Each block instance constitutes a cam track.

Extensions between CAM_CONT_1 and CAM_CONT:

- Additional output information:
  diOffset: DINT (offset of boOutVal in relation to the current scanning position, at time "k * T0")
  diDeltaX: DINT (Actual approximation of the speed Δx / T0 = "diInVal (k) - diInVal (k-1)")
  with: T0 = sampling time; K = actual time index
- Higher resolution of the delay time (udDelay)
  udDelay : UDINT [0.001 ms]

These extensions allow a more accurate time output together with, for example, the 'Timestamp' outputs of the I/O extension of the Ax PLC modules, the I/ option of the iSA module or the external EL2252 EtherCat modules of the cam switch (see AmkDevAccess Library: FB CAM_CONT_TS).

User interface



Input variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnable</td><td>BOOL</td><td>Enable signal: With a positive edge, the initialisation of the block starts.
As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC.
In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.</td></tr>
<tr><td>enMode</td><td>ENUM</td><td>EN_CAM_CONT_MODE
Selection mode between incremental and absolute input evaluation<table><tr><td>Default</td><td>CAM_CONT_INC</td></tr></table><table><tr><th>Range</th><th>Meaning</th></tr><tr><td>CAM_CONT_INC</td><td>Incremental input value evaluation</td></tr><tr><td>CAM_CONT_ABS</td><td>Absolute input value evaluation</td></tr></table></td></tr>
<tr><td>diInVal</td><td>DINT</td><td>Input value of the camshaft control (position)</td></tr>
</table>

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>udModulo</td><td>UDINT</td><td>Modulo value
In mode 'enMode' = CAM_CONT_INC, this is the value at which cam table evaluation restarts at "0"<table><tr><td>Range</td><td>0 … +2^{31}-1</td></tr><tr><td>Default</td><td>20000</td></tr></table></td></tr>
<tr><td>tFilter</td><td>TIME</td><td>Filter time constant
Attenuates the impact of changes in velocity in the context of dead-time compensation<table><tr><td>Default</td><td>t#1 ms</td></tr></table></td></tr>
<tr><td>udDelay</td><td>UDINT</td><td>Dead-time constant
To calculate the offset of the binary information depending on the current velocity in the context of dead-time compensation<table><tr><td>Resolution</td><td>t#0.001 ms</td></tr><tr><td>Default</td><td>0 (dead-time compensation not active)</td></tr></table></td></tr>
<tr><td>uiHyst</td><td>UINT</td><td>Hysteresis value
(H), applied to the on and off edges (X_{on}, X_{off}) of a cam signal<table><tr><td>Default</td><td>0 (hysteresis not active)</td></tr></table>In conjunction with dead-time compensation, the hysteresis must be set higher than the dead-time compensation path Xdead

Thus:
$X_{dead} = T_{dead} * n * G / 60000$

where<table><tr><td>X_{dead}</td><td>Dead-time compensation path [incr]</td></tr><tr><td>T_{dead}</td><td>Dead time [ms]</td></tr><tr><td>n</td><td>Speed [rpm]</td></tr><tr><td>G</td><td>Encoder resolution [incr/rev]</td></tr></table>In incremental input value evaluation mode ('enMode' = CAM_CONT_INC), the following must be true:<table><tr><td>$H < udModulo - (X_{off} - X_{on})$</td><td>for $X_{off} > X_{on}$</td></tr><tr><td>$H < X_{on} - X_{off})$</td><td>for $X_{off} < X_{on}$</td></tr></table></td></tr>
<tr><td>pstTab</td><td>POINTER</td><td>POINTER TO ST_CONT_TAB
Pointer to the cam table</td></tr>
</table>

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state<table><tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr><tr><td>TRUE</td><td>Error</td></tr></table></td></tr>
</table>

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
<table>
<tr><th>Value</th><th>Meaning</th></tr>
<tr><td>1</td><td>Modulo value limited to maximum</td></tr>
<tr><td>2</td><td>Filter time constant set to 1</td></tr>
<tr><td>3</td><td>Filter time constant limited to maximum</td></tr>
<tr><td>4</td><td>Dead-time constant set to 0</td></tr>
<tr><td>5</td><td>Dead-time constant set to 1</td></tr>
<tr><td>6</td><td>Dead-time constant limited to maximum</td></tr>
<tr><td>7</td><td>Modulo value = 0 when mode CAM_CONT_INC</td></tr>
</table></td></tr>
<tr><td>boOutVal</td><td>BOOL</td><td>Cam Output signal</td></tr>
<tr><td>diOffset</td><td>DINT</td><td>Offset of 'boOutVal' to the actual sampling positio, at the time 'k*T_0'
T_0 Sampling time [ms]
k Acutal time index</td></tr>
<tr><td>diDeltaX</td><td>DINT</td><td>Actual approximation for the velocity $\Delta X/T_0$ = 'diInVal(k) - diInVal(k-1)'
T_0 Sampling time [ms]
k Acutal time index</td></tr>
</table>

Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

Description

The camshaft control has the following properties:

- Incremental or absolute mode
- Filter in the context of dead-time compensation
- Dead-time compensation
- Hysteresis

Mode

- **Set incremental input value** ('enMode' = CAM_CONT_INC):
  The 'diInVal' input variable is processed as a 32-bit signed fixed-point number (32-bit integer value).
  In response to every call, the block generates the input value differences from two consecutive items of input information and adds these up to a positive 32-bit value. The internal counter works modulo; in other words, it counts up to a configurable final value 'udModulo' and then starts again at zero.
- **Set absolute input value** ('enMode' = CAM_CONT_ABS):
  The 'diInVal' input variable is processed as a 32-bit signed fixed-point number (32-bit integer value). Overshoot at the end of the travel range is limited.

Filter

To attenuate the impact of changes in velocity for dead-time compensation, multiple speed values are averaged. The 'tFilter' filter time constant determines the number of velocity values for which averaging is performed (number = 'tFilter' [ms]/stDevice.uiCycleTime [ms]).

Dead-time compensation

For dead-time compensation the binary information is offset leading based on the current velocity. The 'tDelay' dead-time constant accounts for the time taken to calculate the offset.

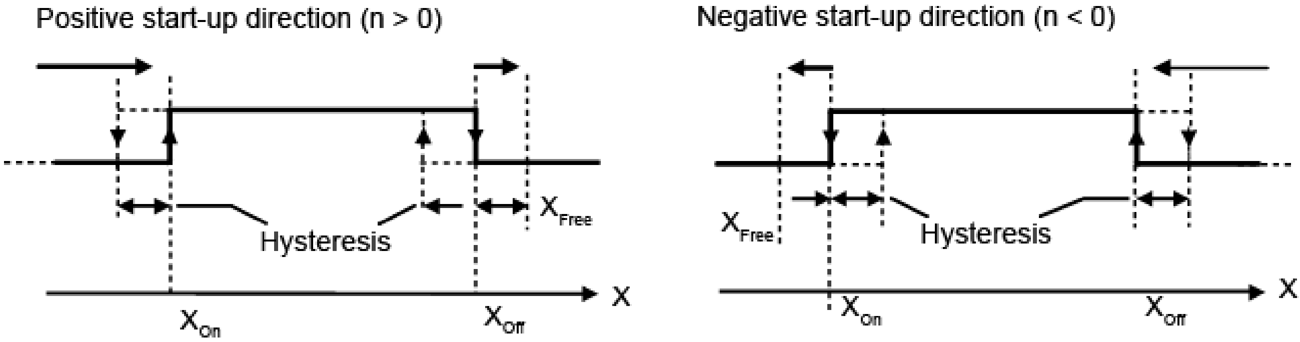
Hysteresis

The hysteresis ensures that the binary output always adopts a stable state, even if the input value of the block is moving around a rising or falling cam edge at the time.

The generation of the hysteresis (X_{on}, X_{off}) is illustrated in the figure below:

- Positive approach direction (n > 0; X increasing)
- Negative approach direction (n < 0; X decreasing)

Abbildung 3: CAM_CONT: Hysteresis generation



A "positive approach direction" results in the following behavior at the binary output (cam):

- Cam information "0" is output starting from a position $X < X_{on}$.
- Cam information "1" is output as of position $X \geq X_{on}$.
- Cam information "1" is retained during reverse rotation to position $X \geq X_{on}–H$.
  Cam information "0" is output during further reverse rotation to position $X < X_{on}–H$.
- Cam information "0" is output during forward rotation starting from position $X \geq X_{off}$.
  - Cam information "0" is retained in the event of reverse rotation to position $X \geq X_{off}–H$ before position $X = X_{free} = X_{off}+H$ is reached.
    Cam information "1" is output during further reverse rotation.
  - In the event of reverse rotation after position $X \geq X_{free}$ has been reached, the cam signal is generated according to the "negative approach direction".

A "negative approach direction" results in the following behavior:

- Cam information "0" is output starting from a position $X \geq X_{off}$.
- Cam information "1" is output as of position $X < X_{off}$.
- Cam information "1" is retained during forward rotation to position $X < X_{off}+H$.
  Cam information "0" is output during further forward rotation.
- Cam information "0" is output during reverse rotation starting from position $X < X_{on}$.
  - Cam information "0" is retained in the event of reverse rotation to position $X < X_{on}+H$ prior to overshooting position $X = X_{free} = X_{on}–H$.
    Cam information "1" is output during further forward rotation.
  - In the event of reverse rotation after position $X < X_{free}$ has been reached, the cam signal is generated according to the "positive approach direction".

Switchover between hysteresis generation of positive (negative) approach direction takes place once a cam has completed its rotation and position X_{free} has been reached or overshot.

4.3.3 CAM_PROF (FB)

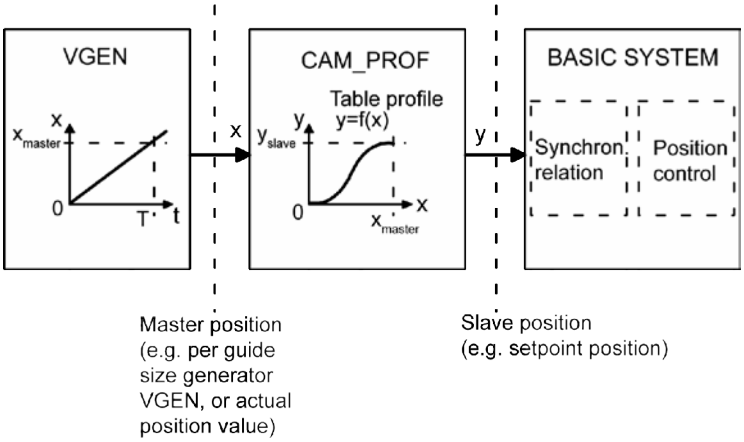
The 'CAM_PROF' function block provides a table-based function interpolator.

The function interpolator assigns an output value 'diOutVal' to an input value 'diInVal' based on tables.

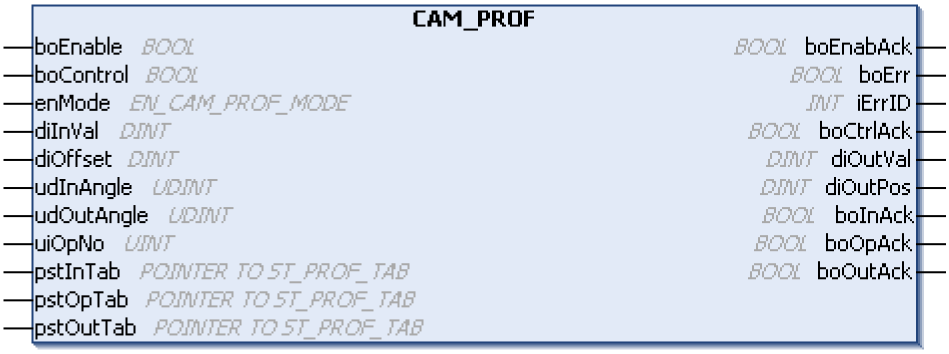
- For Y and XY tables, interpolation points are used to describe the assignment y = f(x).
  The interpolation between the points is linear.
- In the context of the XYVA format, the assignment y = f(x) is described section by section with 5th order polynomials.

Input value can be any internal or external value, e.g. the actual position of a master or a defined incremental number for each sampling time. The output value corresponds to the position setpoint of a slave drive, for example.

Abbildung 4: CAM_PROF: Principle of the table-based function interpolator



User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.
boControl	BOOL	Start / Stop Table control based on table interpolator 'enMode'

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>enMode</td><td>ENUM</td><td>EN_CAM_PROF_MODE
Selection mode of the required movement sequence
<table>
<tr><td>Default</td><td>CAM_PROF_CONT</td></tr>
</table>
<table>
<tr><th>Range</th><th>Meaning</th></tr>
<tr><td>CAM_PROF_
CONT</td><td>Continuous movement
without taking 'boControl' into account</td></tr>
<tr><td>CAM_PROF_
PIN_POUT</td><td>Input / output mode
with control of table transition by 'boControl'</td></tr>
<tr><td>CAM_PROF_
START_STOP</td><td>Start / stop mode
with control of table transition by 'boControl' and automatic stop at end of table</td></tr>
<tr><td>CAM_PROF_IM_
START_STOP</td><td>Immediate start / stop mode
with control of table transition by 'boControl' and automatic stop at end of table</td></tr>
<tr><td>CAM_PROF_
CONT_R</td><td>Continuous movement
without taking 'boControl' into account
the movement sequence starts relative to the current position</td></tr>
<tr><td>CAM_PROF_
PIN_POUT_R</td><td>Input / output mode
with control of table transition by 'boControl'
the movement sequence starts relative to the current position</td></tr>
<tr><td>CAM_PROF_
START_STOP_R</td><td>Start / stop mode
with control of table transition by 'boControl' and automatic stop at end of table
the movement sequence starts relative to the current position</td></tr>
<tr><td>CAM_PROF_IM_
START_STOP_R</td><td>Immediate start / stop mode
with control of table transition by 'boControl' and automatic stop at end of table
the movement sequence starts relative to the current position</td></tr>
</table></td></tr>
<tr><td>diInVal</td><td>DINT</td><td>x Input value (table x axis)
<table>
<tr><td>Unit</td><td>incr</td></tr>
</table></td></tr>
<tr><td>diOffset</td><td>DINT</td><td>Offset of the counter value to the homing pulse
<table>
<tr><td>Unit</td><td>Incr</td></tr>
</table></td></tr>
<tr><td>udInAngle</td><td>UDINT</td><td>Phasing in angle
Maximum permissible position on the x axis up to which phasing in is permitted
(relevant for 'enMode' = CAM_PROF_PIN_POUT / CAM_PROF_START_STOP / CAM_PROF_PIN_POUT_R / CAM_PROF_START_STOP_R)
<table>
<tr><td>Unit</td><td>incr</td></tr>
<tr><td>Default</td><td>1000</td></tr>
</table></td></tr>
<tr><td>udOutAngle</td><td>UDINT</td><td>Phasing out angle
Maximum permissible position on the x axis up to which phasing out is permitted
(relevant for 'enMode' = CAM_PROF_PIN_POUT / CAM_PROF_PIN_POUT_R)
<table>
<tr><td>Unit</td><td>incr</td></tr>
<tr><td>Default</td><td>1000</td></tr>
</table></td></tr>
</table>

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>uiOpNo</td><td>UINT</td><td>Number of operating table cycles to be processed [1]
(relevant for 'enMode' = CAM_PROF_START_STOP / CAM_PROF_IM_START_STOP / CAM_PROF_START_STOP_R / CAM_PROF_IM_START_STOP_R)
<table><tr><td>Default</td><td>3</td></tr></table></td></tr>
<tr><td>pstInTab</td><td>POINTER</td><td>POINTER TO ST_PROF_TAB
Pointer to phasing-in table.
(for all 'enMode' except CAM_PROF_CONT / CAM_PROF_CONT_R)
<table>
<tr><th>Range</th><th>Meaning</th></tr>
<tr><td>pointer to 0</td><td>Table not necessary</td></tr>
<tr><td>pointer to ST_PROF_YTAB</td><td>Y table
(Siehe 'ST_PROF_YTAB (ST)' auf Seite 143.)</td></tr>
<tr><td>pointer to ST_PROF_XYTAB</td><td>XY table
(Siehe 'ST_PROF_XYTAB (ST)' auf Seite 142.)</td></tr>
<tr><td>pointer to ST_PROF_XYVATAB</td><td>XYVA table
(Siehe 'ST_PROF_XYVATAB (ST)' auf Seite 217.)</td></tr>
<tr><td>pointer to ST_PROF_TAB</td><td>either Y, XY, or XYVA table
(Siehe 'ST_PROF_TAB (ST)' auf Seite 141.)</td></tr>
</table></td></tr>
<tr><td>pstOpTab</td><td>POINTER</td><td>POINTER TO ST_PROF_TAB
Reference to operating table (table-supported cam)
<table>
<tr><th>Range</th><th>Meaning</th></tr>
<tr><td>pointer to ST_PROF_YTAB</td><td>Y table
(Siehe 'ST_PROF_YTAB (ST)' auf Seite 143.)</td></tr>
<tr><td>pointer to ST_PROF_XYTAB</td><td>XY table
(Siehe 'ST_PROF_XYTAB (ST)' auf Seite 142.)</td></tr>
<tr><td>pointer to ST_PROF_XYVATAB</td><td>XYVA table
(Siehe 'ST_PROF_XYVATAB (ST)' auf Seite 217.)</td></tr>
<tr><td>pointer to ST_PROF_TAB</td><td>either Y, XY, or XYVA table
(Siehe 'ST_PROF_TAB (ST)' auf Seite 141.)</td></tr>
</table></td></tr>
<tr><td>pstOutTab</td><td>POINTER</td><td>POINTER TO ST_PROF_TAB
Pointer to phasing-out table.
(for all 'enMode' except CAM_PROF_CONT / CAM_PROF_CONT_R)
<table>
<tr><th>Range</th><th>Meaning</th></tr>
<tr><td>pointer to 0</td><td>Table not necessary</td></tr>
<tr><td>pointer to ST_PROF_YTAB</td><td>Y table
(Siehe 'ST_PROF_YTAB (ST)' auf Seite 143.)</td></tr>
<tr><td>pointer to ST_PROF_XYTAB</td><td>XY table
(Siehe 'ST_PROF_XYTAB (ST)' auf Seite 142.)</td></tr>
<tr><td>pointer to ST_PROF_XYVATAB</td><td>XYVA table
(Siehe 'ST_PROF_XYVATAB (ST)' auf Seite 217.)</td></tr>
<tr><td>pointer to ST_PROF_TAB</td><td>either Y, XY, or XYVA table
(Siehe 'ST_PROF_TAB (ST)' auf Seite 141.)</td></tr>
</table></td></tr>
</table>

Output variables

Name	Type	Description
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table></td></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Warning:
<table>
<tr><th>Value</th><th>Meaning</th></tr>
<tr><td>1</td><td>Offset too high</td></tr>
<tr><td>2</td><td>Input angle too high</td></tr>
<tr><td>3</td><td>Output angle too high</td></tr>
</table>
Error:
<table>
<tr><th>Value</th><th>Meaning</th></tr>
<tr><td>1</td><td>Illegal mode</td></tr>
<tr><td>2</td><td>Phasing in table required</td></tr>
<tr><td>3</td><td>Operating table required</td></tr>
<tr><td>4</td><td>Phasing out table required</td></tr>
<tr><td>5</td><td>Illegal element number in phasing in table</td></tr>
<tr><td>6</td><td>Illegal element number in operating table</td></tr>
<tr><td>7</td><td>Illegal element number in phasing out table</td></tr>
<tr><td>8</td><td>Illegal number of master increments in phasing in table</td></tr>
<tr><td>9</td><td>Illegal number of master increments in operating table</td></tr>
<tr><td>10</td><td>Illegal number of master increments in phasing out table</td></tr>
<tr><td>11</td><td>Illegal number of operating tables</td></tr>
<tr><td>12</td><td>Illegal x-value sequence in phasing in table</td></tr>
<tr><td>13</td><td>Illegal x-value sequence in operating table</td></tr>
<tr><td>14</td><td>Illegal x-value sequence in phasing out table</td></tr>
<tr><td>15</td><td>Illegal phasing in table type</td></tr>
<tr><td>16</td><td>Illegal operating table type</td></tr>
<tr><td>17</td><td>Illegal phasing out table type</td></tr>
<tr><td>18</td><td>Illegal starting value for phasing in table (≠ 0)</td></tr>
<tr><td>19</td><td>Illegal starting value for operating table (≠ 0)</td></tr>
<tr><td>20</td><td>Illegal starting value for phasing out table (≠ 0)</td></tr>
</table></td></tr>
<tr><td>boCtrlAck</td><td>BOOL</td><td>Acknowledgement of table control
<table>
<tr><td>FALSE</td><td>Output of output values inactive</td></tr>
<tr><td>TRUE</td><td>Output of output values active, 'boControl' applies</td></tr>
</table></td></tr>
<tr><td>diOutVal</td><td>DINT</td><td>y Output value as sum of output increments</td></tr>
<tr><td>diOutPos</td><td>DINT</td><td>y Output position
Table display y axis: $0 \le y \le y_{max}$ (table final value)</td></tr>
<tr><td>boInAck</td><td>BOOL</td><td>Acknowledge end of phasing in table
Pulse at end of phasing in table; 'boInAck' = TRUE for 2 sampling time points</td></tr>
<tr><td>boOpAck</td><td>BOOL</td><td>Acknowledge end of operating table
Pulse at end of operating table; 'boOpAck' = TRUE for 2 sampling time points</td></tr>
<tr><td>boOutAck</td><td>BOOL</td><td>Acknowledge end of phasing out table
Pulse at end of phasing out table; 'boOutAck' = TRUE for 2 sampling time points</td></tr>
</table>

4.3.3.1 Table types

The 'CAM_PROF' function interpolator can process three types of table

- Y tables – with equidistant X axis resolution
- XY tables – with freely definable X axis resolution
- XYVA tables – for section-by-section definition of the curve with 5th order polynomials.

The reference to the table structure is made with the pointer variables 'pstInTab', 'pstOpTab', and 'pstOutTab'.
The transition between the tables is controlled with the binary input signal 'boControl'.
The assignment of the table can be changed by exchanging the address (this corresponds to table switchover).

4.3.3.1.1 Y table

Y tables are based on the 'ST_PROF_YTAB' structure. They support the definition of a function y=f(x) with equidistant x axis resolution.

In the table structure, only the y values of the function y=f(x) are described.

The corresponding x values are generated in the 'CAM_PROF' block with 'uiNoElement'+1 equidistant points.

Thus:

$$A = \frac{\text{udMasterInc}}{\text{uiNoElement}}$$

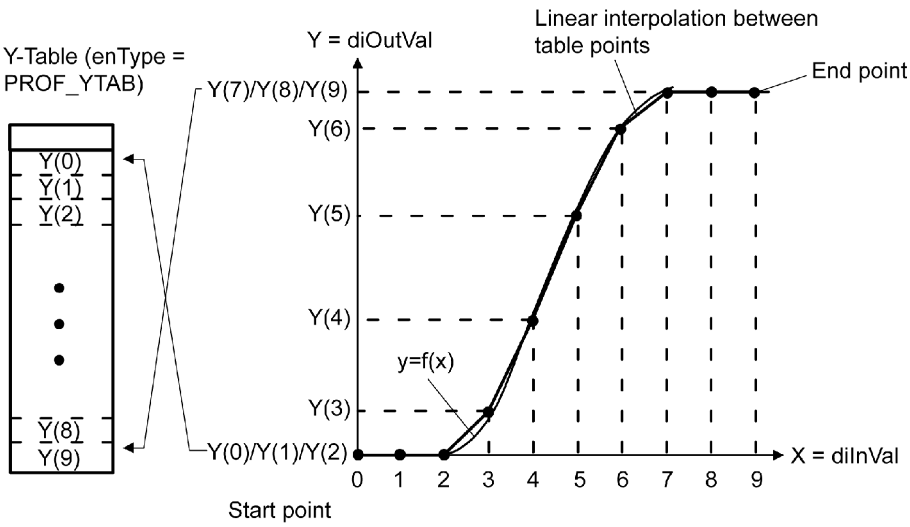
where A: equidistant spacing

Tabelle 1: CAM_PROF: Table structure of the Y table

Header information:	**enType = PROF_YTAB**	**uiNoElement**
	udMasterInc	
Interpolation points:	diY[0] = 0	
	diY[1]	
	...	
	diY[360]	

Limiting to 'uiNoElement' = 360 is negated by mode 'enType' = PROF_YTAB_NL

Abbildung 5: CAM_PROF: Principle of the function interpolator with Y table



Advantages of the Y table

- High information density, as only the y values are saved
- Transparent presentation

Disadvantages of the Y table

- Points also have to be defined in sections with linear function characteristic

4.3.3.1.2 XY table

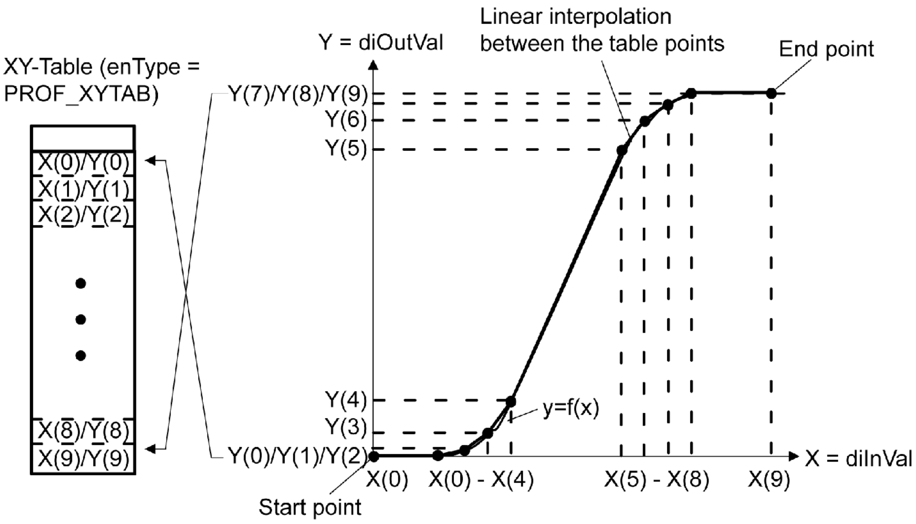
XY tables are based on the 'ST_PROF_XYTAB' structure. They support the definition of a function y=f(x) with any x axis resolution.
In the table structure, the x and y values of the function y=f(x) are described by 'uiNoElement'+1 pair of values.

Tabelle 2: CAM_PROF: Table structure of the XY table

Header information:	**enType = PROF_XYTAB**	**uiNoElement**
	udMasterInc (not used)	
Interpolation points:	stElement[0].diX = 0	
	stElement[0].diY = 0	
	stElement[1].diX	
	stElement[1].diY	
	...	
	stElement[180].diX	
	stElement[180].diY	

Limiting to 'uiNoElement' = 180 is negated by mode 'enType' = PROF_XYTAB_NL

Abbildung 6: CAM_PROF: Principle of the function interpolator with XY table



Advantages of the XY table

- Since the point spacing can be freely defined, the point density can be adapted to the curvature of the curve
- Only the start and end points need to be specified to describe linear sections

Disadvantages of the XY table

- Fewer points, since both x and y values must be specified
- Lack of transparency in presentation based on pairs of points

4.3.3.1.3 XYVA table

XY tables are based on the 'ST_PROF_XYVATAB' structure. They support the definition of a function y=f(x) based on 'uiNoElement' 5th order polynomials.

A pointer of the table structure references the dX, dY, dV, and dA values which describe the function y=f(x) section by section.

dX and dY describe the interpolation point of the function, dV describes the velocity (value of first derivation), and dA describes the acceleration (value of second derivation) in this point.

The table interpolation point structure 'stCam_A': ARRAY [0...3] OF SMC_CAMXYVA and the structure 'stCam': MC_CAM_REF are generated automatically based on the cam disk editor under CODESYS.

The 'CAMXYVA_TO_PROF' function block from the AmkSupport library converts the CODESYS structures into the AMK structure 'ST_PROF_XYVATAB'. (

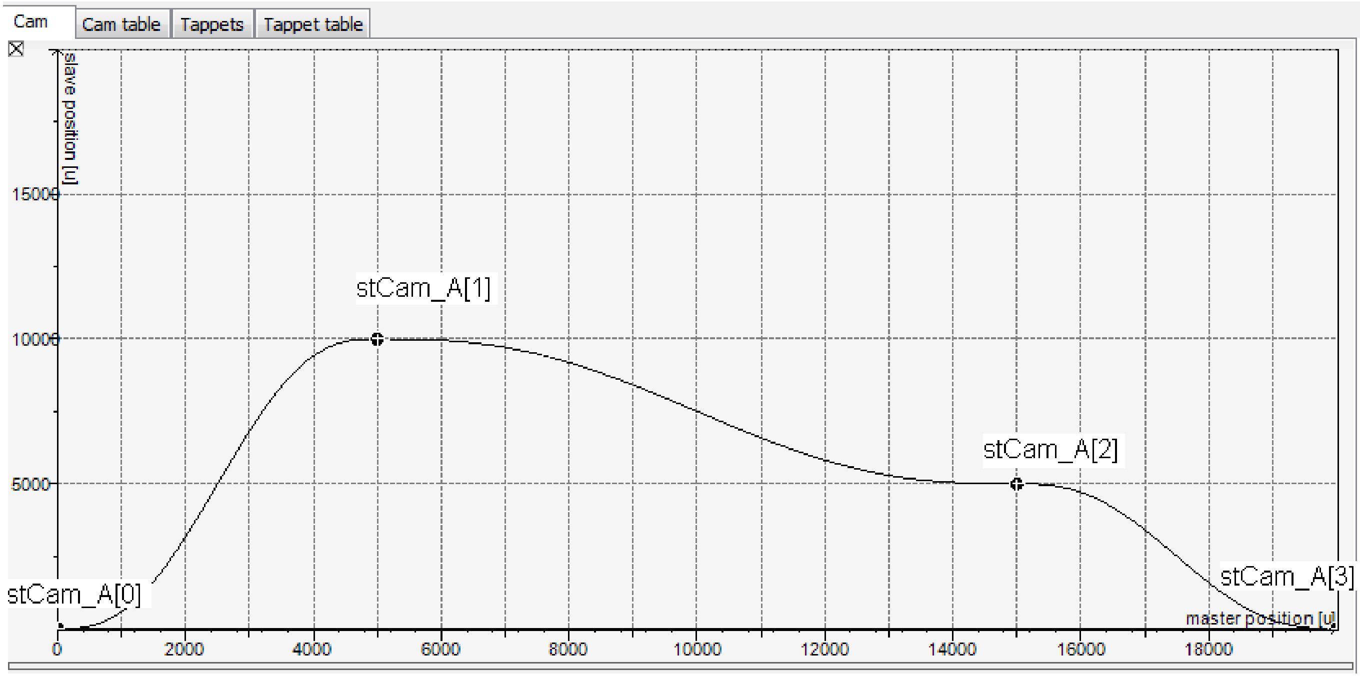
Tabelle 3: CAM_PROF: Table structure of the XYVA table

Header information:	**enType = PROF_XYVATAB**	**uiNoElement**
	udMasterInc (not used)	
Pointer to interpolation point table:	pstCamXYVA	

Table structure of the interpolation point table

XYVA interpolation point table:	
	stCam[0].dX = 0
	stCam[0].dY = 0
	stCam[0].dV
	stCam[0].dA
	...
	stCam[N].dX
	stCam[N].dY
	stCam[N].dV
	stCam[N].dA

Abbildung 7: CAM_PROF: Principle of the function interpolator with XYVA tables



Cam Cam-table Tappets Tappet table

	X	Y	V	A	J	Segme...	min(Position)	max(Position)	max(|Velocity|)	max(|Acceleration|)
	0	0	0	0	0					
						Poly5	0	10000	3.75	0.0023094010767585032
	5000	10000	0	0	0					
						Poly5	5000	10000	0.9375	0.00028867513459481290
	15000	5000	0	0	0					
						Poly5	0	5000	1.875	0.0011547005383792516
	20000	0	0	0	0					

The figure shows the functional principle of the XYVA interpolator based on 5th order polynomials defined section by section. The example includes N = 3 polynomials with 4 sampling points stCam_A[0] ... stCam_A[3].

CODESYS structures 'stCAM' and 'stCAM_A'

```
stCAM: MC_CAM_REF   :=   (nElements = 4,
                          byType = 3,
                          xStart = 0.000000,
                          xEnd = 20000.000000,
                          nTappets = 0,
                          strCamName = "stCAM");

stCAM_A: ARRAY[0…3] OF SMC_CAMXYVA  :=   (dX = 0.000000, dY = 0.000000, dV = 0.000000, dA = 0.000000),
                                          (dX = 5000.000000, dY = 10000.000000, dV = 0.000000, dA = 0.000000),
                                          (dX = 15000.000000, dY = 5000.000000, dV = 0.000000, dA = 0.000000),
                                          (dX = 20000.000000, dY = 0.000000, dV = 0.000000, dA = 0.000000),
```

Advantages of the XYVA table

- Freely definable points with specification of 1st and 2nd derivation at start and end of section

Disadvantages of the XYVA table

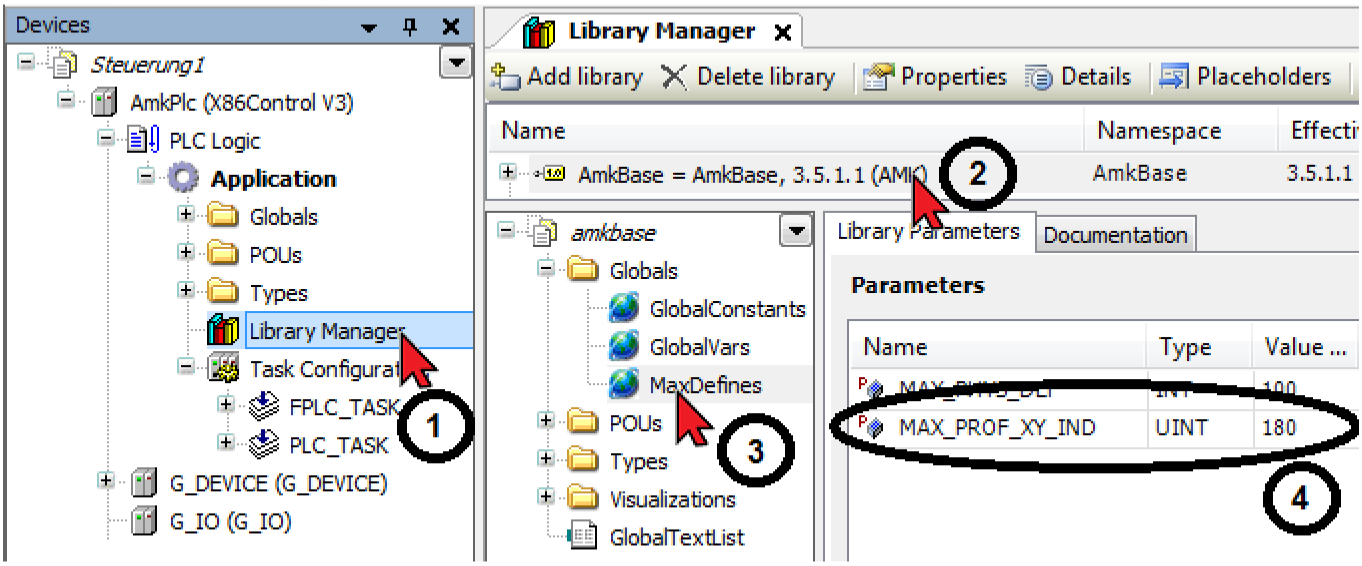
- Increased processing overhead for online determination of 5th order polynomials

4.3.3.2 Number of table interpolation points

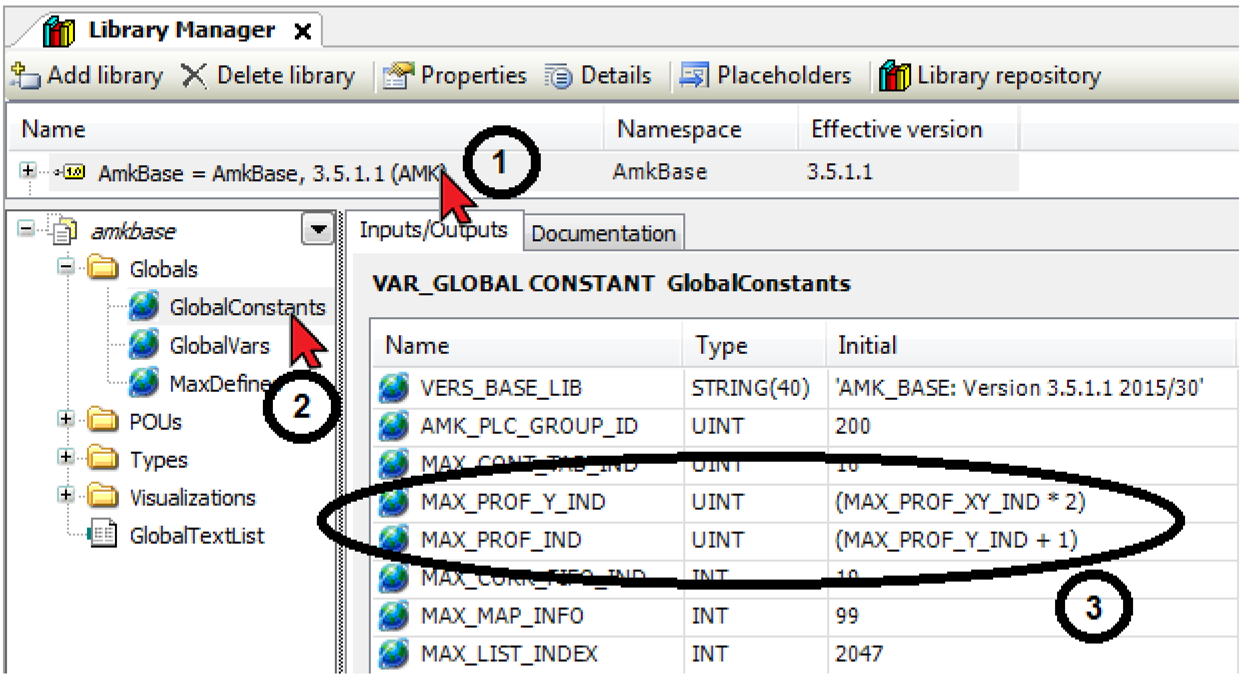
The number of table interpolation points are increased by the parameter MAX_PROF_XY_IND. (Library AmkBase → folder Globals → MaxDefines)

Default values:
181 XY-table sections (0 ... 180)
361 Y-table sections (0 ... 360)

Change the number of table interpolation points with CODESYS V3



The sizes MAX_PROF_Y_IND and MAX_PROF_IND are determined automatically.



If the maximum number of table interpolation points are increased,
using the parameter 'MAX_PROF_XY_IND', further the table types with the extension NL must be used.
_NL stands for 'not limited'

Default values: **181 XY table sections** **361 Y table sections**	**Advanced:** **> 181 XY table sections** **> 361 Y table sections**
'PROF_YTAB' 'PROF_XYTAB'	'PROF_YTAB_**NL**' 'PROF_XYTAB_**NL**'

4.3.3.3 Operating modes

The following operating modes are defined according to the mode selected in 'enMode':

- CAM_PROF_CONT, CAM_PROF_CONT_R
  Table-based continuous movement of a drive
- CAM_PROF_PIN_POUT, CAM_PROF_PIN_POUT_R
  Table-based continuous movement of a drive, which the option of input / output via special input / phasing out tables
- CAM_PROF_START_STOP, CAM_PROF_START_STOP_R
  Table-based movement of a drive after starting at the zero point of the table and with automatic stop at the end of the table
- CAM_PROF_IM_START_STOP, CAM_PROF_IM_START_STOP_R
  Immediate table-based movement of a drive after starting, with automatic stop at the end of the table

The relative movement modes ending '_R' are equivalent to the corresponding absolute modes but support the inclusion of a movement sequence in a table. This means that the interpolation in the table starts relative to the current master position (x = diInVal) and slave position (y = diOutVal).

The 'diOffset' variable sets the current master position x.
Once 'boEnable' has been activated, the assigned slave position y to which the slave axis is being moved can be read at 'diOutPos'.
The axes adopt the start position (x,y). The movement coupled via the table is resumed at this point.

Continuous movement

In 'CAM_PROF_CONT' mode, table interpolation is activated on a positive edge at 'boEnable'.
From this point on, incremental changes at the input variable 'diInVal' (x) generate incremental changes at the output variable 'diOutVal' (y).
The assignment y = f(x) is made according to the table definitions.
Only the 'pstOpTab' operating table is required for this mode.
The control signal 'boControl' is not evaluated.

4.3.3.3.1 Phasing in / phasing out

In 'CAM_PROF_PIN_POUT' mode, a positive edge at 'boEnable' activates input increment acquisition.
From this point in time, incremental changes at the input variable 'dInVal' generate changes in the master reference point (x).
The input, working, and phasing out tables assigned with 'pstInTab', 'pstOpTab', and 'pstOutTab' are evaluated. The operating table is a minimum requirement.
The control signal 'boControl' is evaluated to control input and output behavior.
The assignment y = f(x) is made according to the corresponding table definition.
A positive edge at 'boControl' and 'x ≤ udInAngle' activates the table interpolator.
Incremental changes according to the phasing in table are output at the output variable 'diOutVal' (y).
The transition to the operating table occurs at the end of the phasing in table. The operating table is processed until a negative edge occurs on 'boControl' and 'x ≤ udOutAngle'.
After this, there is a transition to the phasing out table with automatic stop of interpolation at the end of the phasing out table.

Incremental changes at the input variable 'dInVal' continue to generate changes in the master reference point (x). In other words, the reference to the master axis is retained.
If there is no phasing in table or no phasing out table ('pstInTab' = 0 or 'pstOutTab' = 0), the operating table is used instead.

4.3.3.3.2 Start with auto-stop

In 'CAM_PROF_START_STOP' mode, a positive edge at 'boEnable' activates input increment acquisition.
From this point in time, incremental changes at the input variable 'dInVal' generate changes in the master reference point (x).

The input, working, and phasing out tables assigned with 'pstInTab', 'pstOpTab', and 'pstOutTab' are evaluated.
The operating table is a minimum requirement.
The control signal 'boControl' is evaluated to control startup behavior.
The assignment y = f(x) is made according to the corresponding table definition.

A positive edge at 'boControl' and 'x ≤ udInAngle' activates the table interpolator.
Incremental changes according to the phasing in table are output at the output variable 'diOutVal' (y).
The transition to the operating table occurs at the end of the phasing in table.
The operating table is processed n times (with n = 'uiOpNo').
After this, there is a transition to the phasing out table with automatic stop of interpolation at the end of the phasing out table.

Incremental changes at the input variable 'dInVal' continue to generate changes in the master reference point (x). The reference to the master axis is retained.

If there is no phasing in table ('pstInTab' = 0), the process starts directly with the operating table.
If there is no phasing out table ('pstOutTab' = 0), the process stops at the end of the last pass of the operating table!

4.3.3.3.3 Immediate start with auto-stop

In 'CAM_PROF_IM_ START_STOP' mode, table interpolation is activated on a positive edge at 'boEnable'.
From this point on, incremental changes at the input variable 'diInVal' (x) generate incremental changes at the output variable 'diOutVal' (y).

The input, working, and phasing out tables assigned with 'pstInTab', 'pstOpTab', and 'pstOutTab' are evaluated.
The operating table is a minimum requirement.
The control signal 'boEnable' is evaluated to control startup behavior.
The assignment y = f(x) is made according to the corresponding table definition.

There is no direct reference to the master axis. Instead, a reference is generated at the time of the positive edge at 'boControl'.
After this, travel continues with reference to the master increment inputs.

Incremental changes according to the phasing in table are output at the output variable 'diOutVal' (y).
The transition to the operating table occurs at the end of the phasing in table.
The operating table is processed n times (with n = 'uiOpNo').
After this, there is a transition to the phasing out table with automatic stop of interpolation at the end of the phasing out table.

Incremental changes at the input variable 'dInVal' continue to generate changes in the master reference point (x). The reference to the master axis is retained.

If there is no phasing in table ('pstInTab' = 0), the process starts directly with the operating table.
If there is no phasing out table ('pstOutTab' = 0), the process stops at the end of the last pass of the operating table!

4.3.3.4 Online table switchover

The 'CAM_PROF' function block supports switchover between tables by changing the value of the pointer variables 'pstInTab', 'pstOpTab', and 'pstOutTab'. This can be done online while the table interpolator is active:

- Synchronous table changeover, switchover at table zero point
- Non-synchronous table changeover, switchover away from table zero point.

It is possible to switch between entirely different tables:

- The interpolation points and their number ('uiNoElement') can differ.
- The table type ('enType') can differ.
- The x axis resolution ('udMasterInc' or 'stElement[uiNoElement].diX') can differ.

If the tables have different x axis resolutions, the current master position must be converted to the assigned position of the new table. Thus:

$$X_{new} = X_{curr} \times \frac{X_{max\_new}}{X_{max\_curr}}$$

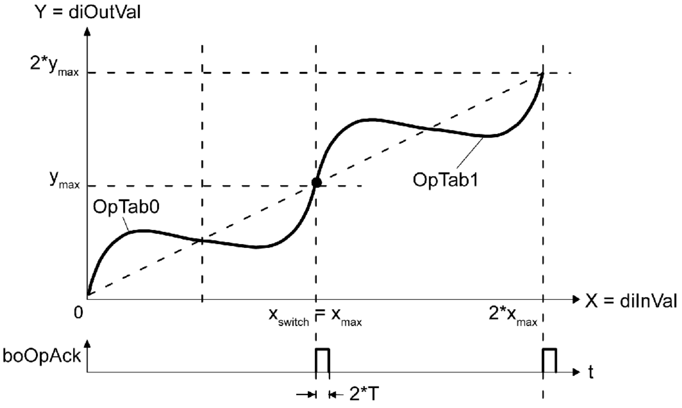
X_{curr}	x master position in the current table
X_{max_curr}	x axis resolution of the current table
X_{new}	x master position in the new table
X_{max_new}	x axis resolution of the new table

Since the 'CAM_PROF' block works with table pointers ('pstInTab', 'pstOpTab', 'pstOutTab'), these pointers must always reference a correct table structure.
Block processing is aborted if the corresponding pointer variable is found to have a value of 0 (no table) when attempting to access a table!

4.3.3.4.1 Synchronous table changeover

The binary output signals 'boInAck', 'boOpAck', and 'boOutAck' are provided for synchronous table changeover.
These signals generate a TRUE signal at the end of the corresponding table; this signal remains pending for two sampling cycles. Table changeover is synchronous if it takes place during this time, i.e. at the zero point of the table.
As shown in the figure below, the process is similar to that when changing between phasing in and operating tables (or between operating and phasing out tables).
If the maximum x and y final values of the tables are identical, the reference to the master is retained.

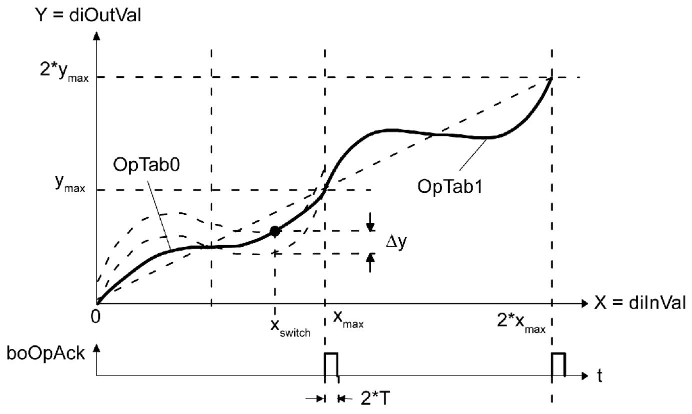
Abbildung 8: CAM_PROF: Synchronous table switchover



4.3.3.4.2 Non-synchronous table changeover

The binary output signals 'boInAck', 'boOpAck', and 'boOutAck' do not have to be evaluated for non-synchronous table changeover.
Table changeover is non-synchronous if it takes place at a point in time at which the assigned binary signal is set to FALSE, i.e. not at the zero point of the table.
As shown in the figure below, the y point on the table being phased in is shifted to the current y point of the previous table (by Δy).
A defined reference to the master cannot be retained even if the maximum x and y final values of the tables are identical.

Abbildung 9: CAM_PROF: Non-synchronous table changeover



4.3.4 CAM_PROF_1 (FB)

The 'CAM_PROF_1' function block provides a table-based function interpolator and is based on the function block CAM_PROF.
Functional description: Siehe 'CAM_PROF (FB)' auf Seite 78.

CAM_PROF_1 further outputs the following output value:

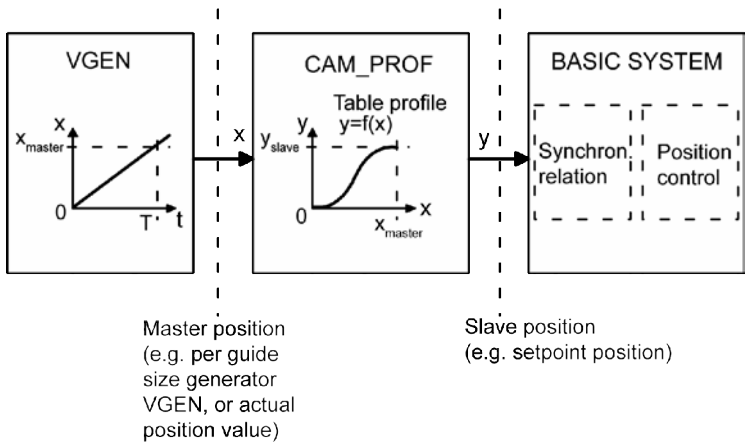
- 1. derivation, y '= f'(x), of the table-based function
- 2. derivation, y ' '= f ''(x), of the table-based function
- Actual table input (X) position (master position) in the current cycle
- Actual table output (Y) position (slave position) in the current cycle

The function interpolator assigns an output value 'diOutVal' to an input value 'diInVal' based on tables.

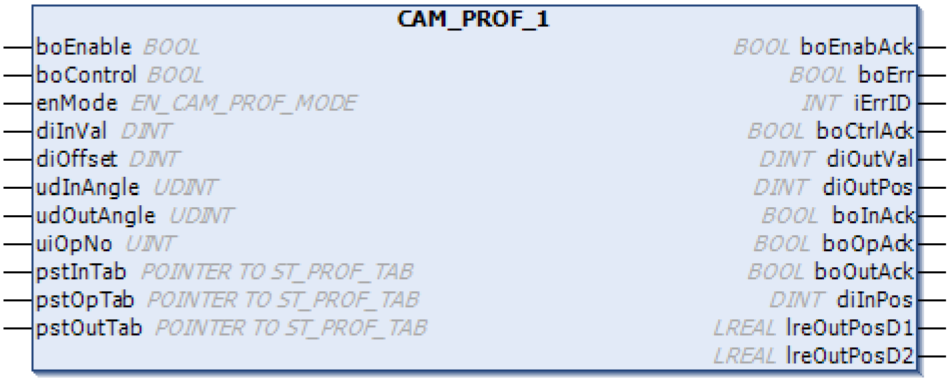
- For Y and XY tables, interpolation points are used to describe the assignment y = f(x).
  The interpolation between the points is linear.
- In the context of the XYVA format, the assignment y = f(x) is described section by section with 5th order polynomials.

Input value can be any internal or external value, e.g. the actual position of a master or a defined incremental number for each sampling time. The output value corresponds to the position setpoint of a slave drive, for example.

Abbildung 10: CAM_PROF_1: Principle of the table-based function interpolator



User interface



Input variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnable</td><td>BOOL</td><td>Enable signal: With a positive edge, the initialisation of the block starts.
As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC.
In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.</td></tr>
<tr><td>boControl</td><td>BOOL</td><td>Start / Stop
Table control based on table interpolator 'enMode'</td></tr>
<tr><td>enMode</td><td>ENUM</td><td>EN_CAM_PROF_MODE
Selection mode of the required movement sequence
<table>
<tr><td>Default</td><td>CAM_PROF_CONT</td></tr>
</table>
<table>
<tr><th>Range</th><th>Meaning</th></tr>
<tr><td>CAM_PROF_ CONT</td><td>Continuous movement
without taking 'boControl' into account</td></tr>
<tr><td>CAM_PROF_ PIN_POUT</td><td>Input / output mode
with control of table transition by 'boControl'</td></tr>
<tr><td>CAM_PROF_ START_STOP</td><td>Start / stop mode
with control of table transition by 'boControl' and automatic stop at end of table</td></tr>
<tr><td>CAM_PROF_IM_ START_STOP</td><td>Immediate start / stop mode
with control of table transition by 'boControl' and automatic stop at end of table</td></tr>
<tr><td>CAM_PROF_ CONT_R</td><td>Continuous movement
without taking 'boControl' into account
the movement sequence starts relative to the current position X_{act}/Y_{act}</td></tr>
<tr><td>CAM_PROF_ PIN_POUT_R</td><td>Input / output mode
with control of table transition by 'boControl'
the movement sequence starts relative to the current position X_{act}/Y_{act}</td></tr>
<tr><td>CAM_PROF_ START_STOP_R</td><td>Start / stop mode
with control of table transition by 'boControl' and automatic stop at end of table
the movement sequence starts relative to the current position X_{act}/Y_{act}</td></tr>
<tr><td>CAM_PROF_IM_ START_STOP_R</td><td>Immediate start / stop mode
with control of table transition by 'boControl' and automatic stop at end of table
the movement sequence starts relative to the current position X_{act}/Y_{act}</td></tr>
</table></td></tr>
</table>

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>diInVal</td><td>DINT</td><td>x Input value (table x axis)<table><tr><td>Unit</td><td>incr</td></tr></table></td></tr>
<tr><td>diOffset</td><td>DINT</td><td>Offset of the counter value to the homing pulse<table><tr><td>Unit</td><td>Incr</td></tr></table></td></tr>
<tr><td>udInAngle</td><td>UDINT</td><td>Phasing in angle
Maximum permissible position on the x axis up to which phasing in is permitted
(relevant for 'enMode' = CAM_PROF_PIN_POUT / CAM_PROF_START_STOP / CAM_PROF_PIN_POUT_R / CAM_PROF_START_STOP_R)<table><tr><td>Unit</td><td>incr</td></tr><tr><td>Default</td><td>1000</td></tr></table></td></tr>
<tr><td>udOutAngle</td><td>UDINT</td><td>Phasing out angle
Maximum permissible position on the x axis up to which phasing out is permitted
(relevant for 'enMode' = CAM_PROF_PIN_POUT / CAM_PROF_PIN_POUT_R)<table><tr><td>Unit</td><td>incr</td></tr><tr><td>Default</td><td>1000</td></tr></table></td></tr>
<tr><td>uiOpNo</td><td>UINT</td><td>Number of operating table cycles to be processed [1]
(relevant for 'enMode' = CAM_PROF_START_STOP / CAM_PROF_IM_START_STOP / CAM_PROF_START_STOP_R / CAM_PROF_IM_START_STOP_R)<table><tr><td>Default</td><td>3</td></tr></table></td></tr>
<tr><td>pstInTab</td><td>POINTER</td><td>POINTER TO ST_PROF_TAB
Pointer to phasing-in table.
(for all 'enMode' except CAM_PROF_CONT / CAM_PROF_CONT_R)<table><tr><th>Range</th><th>Meaning</th></tr><tr><td>pointer to 0</td><td>Table not necessary</td></tr><tr><td>pointer to ST_PROF_YTAB</td><td>Y table
(Siehe 'ST_PROF_YTAB (ST)' auf Seite 143.)</td></tr><tr><td>pointer to ST_PROF_XYTAB</td><td>XY table
(Siehe 'ST_PROF_XYTAB (ST)' auf Seite 142.)</td></tr><tr><td>pointer to ST_PROF_XYVATAB</td><td>XYVA table
(Siehe 'ST_PROF_XYVATAB (ST)' auf Seite 217.)</td></tr><tr><td>pointer to ST_PROF_TAB</td><td>either Y, XY, or XYVA table
(Siehe 'ST_PROF_TAB (ST)' auf Seite 141.)</td></tr></table></td></tr>
<tr><td>pstOpTab</td><td>POINTER</td><td>POINTER TO ST_PROF_TAB
Reference to operating table (table-supported cam)<table><tr><th>Range</th><th>Meaning</th></tr><tr><td>pointer to ST_PROF_YTAB</td><td>Y table
(Siehe 'ST_PROF_YTAB (ST)' auf Seite 143.)</td></tr><tr><td>pointer to ST_PROF_XYTAB</td><td>XY table
(Siehe 'ST_PROF_XYTAB (ST)' auf Seite 142.)</td></tr><tr><td>pointer to ST_PROF_XYVATAB</td><td>XYVA table
(Siehe 'ST_PROF_XYVATAB (ST)' auf Seite 217.)</td></tr><tr><td>pointer to ST_PROF_TAB</td><td>either Y, XY, or XYVA table
(Siehe 'ST_PROF_TAB (ST)' auf Seite 141.)</td></tr></table></td></tr>
</table>

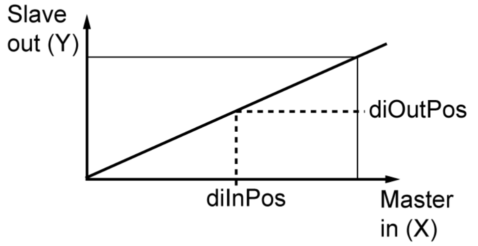
<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>pstOutTab</td><td>POINTER</td><td>POINTER TO ST_PROF_TAB
Pointer to phasing-out table.
(for all 'enMode' except CAM_PROF_CONT / CAM_PROF_CONT_R)
<table>
<tr><th>Range</th><th>Meaning</th></tr>
<tr><td>pointer to 0</td><td>Table not necessary</td></tr>
<tr><td>pointer to ST_PROF_YTAB</td><td>Y table
(Siehe 'ST_PROF_YTAB (ST)' auf Seite 143.)</td></tr>
<tr><td>pointer to ST_PROF_XYTAB</td><td>XY table
(Siehe 'ST_PROF_XYTAB (ST)' auf Seite 142.)</td></tr>
<tr><td>pointer to ST_PROF_XYVATAB</td><td>XYVA table
(Siehe 'ST_PROF_XYVATAB (ST)' auf Seite 217.)</td></tr>
<tr><td>pointer to ST_PROF_TAB</td><td>either Y, XY, or XYVA table
(Siehe 'ST_PROF_TAB (ST)' auf Seite 141.)</td></tr>
</table></td></tr>
</table>

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table></td></tr>
</table>

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Warning:
<table>
<tr><th>Value</th><th>Meaning</th></tr>
<tr><td>1</td><td>Offset too high</td></tr>
<tr><td>2</td><td>Input angle too high</td></tr>
<tr><td>3</td><td>Output angle too high</td></tr>
</table>
Error:
<table>
<tr><th>Value</th><th>Meaning</th></tr>
<tr><td>1</td><td>Illegal mode</td></tr>
<tr><td>2</td><td>Phasing in table required</td></tr>
<tr><td>3</td><td>Operating table required</td></tr>
<tr><td>4</td><td>Phasing out table required</td></tr>
<tr><td>5</td><td>Illegal element number in phasing in table</td></tr>
<tr><td>6</td><td>Illegal element number in operating table</td></tr>
<tr><td>7</td><td>Illegal element number in phasing out table</td></tr>
<tr><td>8</td><td>Illegal number of master increments in phasing in table</td></tr>
<tr><td>9</td><td>Illegal number of master increments in operating table</td></tr>
<tr><td>10</td><td>Illegal number of master increments in phasing out table</td></tr>
<tr><td>11</td><td>Illegal number of operating tables</td></tr>
<tr><td>12</td><td>Illegal x-value sequence in phasing in table</td></tr>
<tr><td>13</td><td>Illegal x-value sequence in operating table</td></tr>
<tr><td>14</td><td>Illegal x-value sequence in phasing out table</td></tr>
<tr><td>15</td><td>Illegal phasing in table type</td></tr>
<tr><td>16</td><td>Illegal operating table type</td></tr>
<tr><td>17</td><td>Illegal phasing out table type</td></tr>
<tr><td>18</td><td>Illegal starting value for phasing in table (≠ 0)</td></tr>
<tr><td>19</td><td>Illegal starting value for operating table (≠ 0)</td></tr>
<tr><td>20</td><td>Illegal starting value for phasing out table (≠ 0)</td></tr>
</table></td></tr>
<tr><td>boCtrlAck</td><td>BOOL</td><td>Acknowledgement of table control
<table>
<tr><td>FALSE</td><td>Output of output values inactive</td></tr>
<tr><td>TRUE</td><td>Output of output values active, 'boControl' applies</td></tr>
</table></td></tr>
<tr><td>diOutVal</td><td>DINT</td><td>y Output value as sum of output increments</td></tr>
<tr><td>diOutPos 1)</td><td>DINT</td><td>y Output position
<table>
<tr><td>Unit</td><td>Incr</td></tr>
</table>
Y-table position to display the table ordinate: $0 \le y \le y_{max}$ (y_{max} = table Y-end position); with y = f(x).</td></tr>
<tr><td>boInAck</td><td>BOOL</td><td>Acknowledge end of phasing in table
Pulse at end of phasing in table; 'boInAck' = TRUE for 2 sampling time points</td></tr>
<tr><td>boOpAck</td><td>BOOL</td><td>Acknowledge end of operating table
Pulse at end of operating table; 'boOpAck' = TRUE for 2 sampling time points</td></tr>
<tr><td>boOutAck</td><td>BOOL</td><td>Acknowledge end of phasing out table
Pulse at end of phasing out table; 'boOutAck' = TRUE for 2 sampling time points</td></tr>
</table>

Name	Type	Description
diInPos 1)	DINT	x Input position Unit: Incr X-table position, to display the table abscissa: $0 \le x \le x_{max}$ (x_{max} = table X-end position); with y = f(x).
lreOutPosD1	LREAL	1. derivation, y '= f'(x), of the table-based function
lreOutPosD2	DINT	2. derivation, y ' '= f ''(x), of the table-based function

1)


Further information: Siehe 'CAM_PROF (FB)' auf Seite 78.

4.3.4.1 Mathematical consideration of functional relationships

For a mathematical consideration of the functional relationships, the following variables are important:

Variable (input / output)	Mathematical identifier	Meaning	Dimension
diInPos (I)		Abscissa (x-axis)	[x]
diOutPos (O)		Ordinate (y-axis)	[y]
lreOutPosD1 (O)		1. derivative (after x)	[y] / [x]
lreOutPosD2 (O)		2. derivative (after x)	[y] / [x]^2

The functional relationship 'y = f (x)' is stored in the form of a table:

- As a point sequence (Y- / XY- table)
- As a spline parameter sequence (XYVA- table)

If, for example, a path-to-path relationship is given with 'y = f (x)' (e.g. if the position 'y' of the slave drive is described as a function of the position 'x' of the master drive), the following time dependencies apply:

Input variables (x-axis):

Position:

Velocity:

Acceleration:

Output variables (y-axis):

Position:

Velocity:

Acceleration:

For a constant velocity the following is valid:

Input variables (x-axis):

Output variables (y-axis):

That means:

- v_y is proportional to the first derivative f '(x)
- a_y is proportional to the second derivative f ''(x)

For the specific case of a constant velocity in the x-axis (e.g. the master drive moves at a constant velocity v_0) the values of the output variables are:

- lreOutPosD1 is proportional to the velocity of the y-axis (slave axis) and
- lreOutPosD2 is proportional to the acceleration of the y-axis (slave axis)

4.3.4.2 Technical realization of derivations

With the CAM_PROF_1 the derivations for **Y- / XY-tables** can be calculated in sections and approximately (e. g. by numerical differentiation)

A requirement for the numerical differentiation of the function sequences (Y- / XY-tables) described as a point sequence is:

- The point sequences are continuously differentiable at least twice
- The resolution of the point sequences is sufficiently high

With the CAM_PROF_1 the derivations for **XYVA-table**s, with the knowledge of the corresponding function 'y = f (x)', but in the form of a mathematical derivation of the derived functions

The advantage of the closed calculation of the derivatives of the polynomial described functional sequences (XYVA-tables) lies in the exact calculation of this (for any position of the function sequences).

4.3.4.2.1 Y- / XY-tables

For 'Y / XY tables', with linear interpolation, the following two different cases can be distinguished:

For a 'Y-table' yTab (of the type ST_PROF_YTAB), for each segment 'k' (with k = 1 to N):

1. Derivation (lreOutPosD1)

2. Derivation (lreOutPosD2)

with

For an 'XY-table' xyTab (of type ST_PROF_XYTAB), for each segment 'k' (with k = 1 to N):

1. Derivation (lreOutPosD1)

In the last point of segment N (theoretical):

2. Derivation (lreOutPosD2)

with:

and

4.3.4.2.2 'XYVA-tables'

For 'XYVA tables', with section-defined polynomials of the fifth order, the following definitions are given.

For an 'XYVA-table' xyvaTab (of type ST_PROF_XYVATAB) for each segment 'k' (with k = 1 to xyvaTab.uiNoElement):

Basic function (diOutPos)

with

and

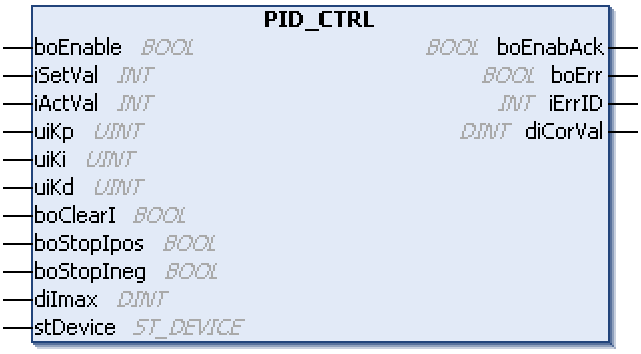
1. Derivation (lreOutPosD1)

2. Derivation (lreOutPosD2)

4.3.5 PID_CTRL (FB)

The 'PID_CTRL' (PID controller) function block supports configurable drive control of internal and external variables.

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.
iSetVal	INT	Setpoint
iActVal	INT	Actual value
uiKp	UINT	Proportional gain (P) of the PID controller Unit: 1/256
uiKi	UINT	Integration gain (I) of the PID controller. Unit: 1/256
uiKd	UINT	Differential gain (D) of the PID controller Unit: 1/256
boClearI	BOOL	Delete accumulated integral action

Name	Type	Description
boStopIpos	BOOL	Integrator stop on positive increase in integral action
boStopIneg	BOOL	Integrator stop on negative increase in integral action
diImax	DINT	Maximum permissible integral action Range: 0 ... 2^{31}-1 Unit: 1/256 Default: 256000

Output variables

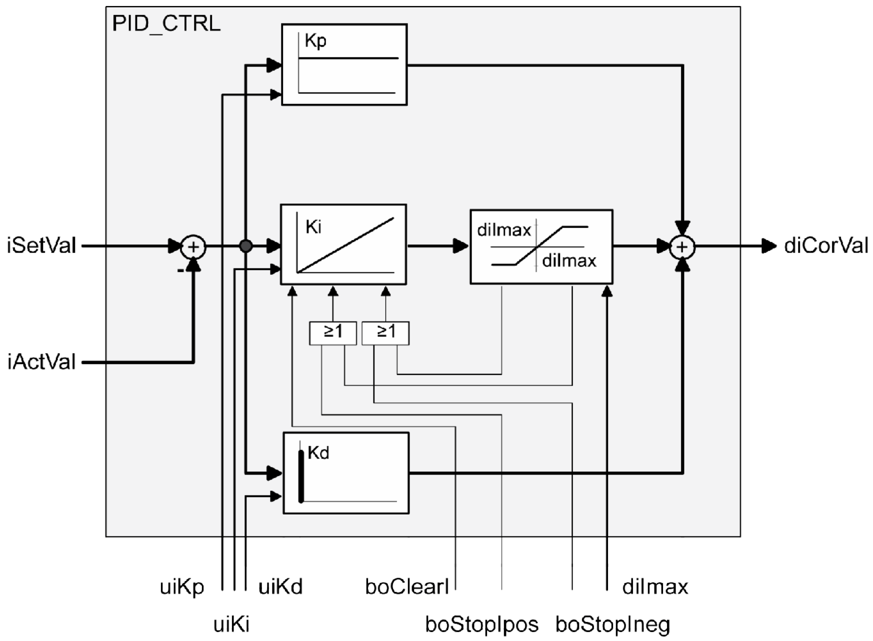
Name	Type	Description
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled
boErr	BOOL	The function block is in an error state FALSE: No error (permitted commanding or warning) TRUE: Error
iErrID	INT	Error identity number: Diagnostic number is output iErrID = 0: No error iErrID ≠ 0, boErr = TRUE: Error iErrID ≠ 0, boErr = FALSE: Warning
diCorVal	DINT	Command variable

Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

Description

Abbildung 11: PID_CTRL: Principle of the PID controller



To optimize processing time, no floating-point operations are used for the PID algorithm. Also, Kp, Ki, and Kd are normalized to 1/256.

Thus:

where

y(k) = diCorVal
Δx(k) = iSetVal - iActVal

'diImax' limits the integral action. The integrator stops automatically when this limit is reached.
The integrator can be controlled externally with the 'boStopIpos' and 'boStopIneg' input variables, e.g. by applying a limit with 'ADD_LIMIT'.
The 'boClearI' input variable provides a means of externally deleting the accumulated integral action.

The amplification factors Kp, Ki, and Kd are calculated from the proportional component P, the reset time Tn, the derivative time Tv, and the sampling time T0 with the 'PID_TO_KPKIKD' function block.

'ADD_LIMIT' queries an additive command variable (feed-forward control) with limiting of the result.

4.3.6 PM_CORRECT (FB)

The 'PM_CORRECT' function block supports controlled output of correction values.
Controlled output means linear interpolation of the correction value across a definable 'diInVal' range.

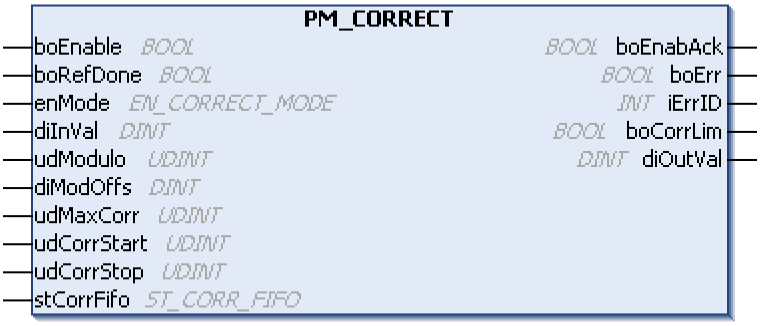
The function block takes the correction values from the FIFO structure 'stCorrFifo', the content of which is written by the 'PM_DETECT' block, for example. 'stCorrFifo' supports spacing of the printing mark sensors greater than one format.

Further properties:

- Definition of interpolation behavior
  Parameter to determine the interpolation range and the maximum permissible correction per format
- Indication whether the correction value is being limited

Combined with the 'PM_DETECT' function block, 'PM_CORRECT' facilitates efficient printing mark control.

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.
boRefDone	BOOL	Homing cycle completed Acknowledgement signal to indicate that a homing cycle has been completed. • In mode 'enMode' = DETECT_AUTO, 'boRefDone' = FALSE when the block is activated or on a positive edge change at 'boRefStart'. Once the first mark has been detected, 'boRefDone' = TRUE is set. • In mode 'enMode' = DETECT_MANUAL, this variable is of no significance. 'boRefDone' = TRUE always applies.

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>enMode</td><td>ENUM</td><td>EN_CORRECT_MODE
Selection mode of operating mode; 'enMode' can be changed online
<table><tr><td>Default</td><td>CORRECT_SET2OUT</td></tr></table>
<table>
<tr><th>Range</th><th>Meaning</th></tr>
<tr><td>CORRECT_ SET2OUT</td><td>Only the correction value is output</td></tr>
<tr><td>CORRECT_ ADD2OUT</td><td>The interpolation of the correction value is output added to the input value 'diInVal'</td></tr>
<tr><td>CORRECT_ SET2OUT_NB</td><td>Only the correction value is output
Avoidance of reversal of direction</td></tr>
<tr><td>CORRECT_ ADD2OUT_NB</td><td>The interpolation of the correction value is output added to the input value 'diInVal'
Avoidance of reversal of direction</td></tr>
</table></td></tr>
<tr><td>diInVal</td><td>DINT</td><td>Input value referenced by the subsequent input variables</td></tr>
<tr><td>udModulo</td><td>UDINT</td><td>Modulo format
Describes the setpoint distance between two consecutive printing marks.
The value is saved when the block is activated (positive edge at 'boEnable'). A subsequent change does not affect the active block.
<table>
<tr><td>Range</td><td>0 ... 1000000000</td></tr>
<tr><td>Default</td><td>2000</td></tr>
</table></td></tr>
<tr><td>diModOffs</td><td>DINT</td><td>Modulo offset
Set PM position within modulo format. This information is required in order to set the offset following an automatic homing cycle.
The 'diModOffs' variable is normally generated from the 'diModVal' output of the 'PM_DETECT' block.
If 'diModOffs' is greater than 'udModulo', the modulo residual ('diModOffs' % 'udModulo') applies.</td></tr>
<tr><td>udMaxCorr</td><td>UDINT</td><td>Maximum permissible correction value to which the correction value output per modulo format is limited.
The value can be changed when the block is active
<table>
<tr><td>Range</td><td>0 ... 999999999</td></tr>
<tr><td>Default</td><td>2000</td></tr>
</table></td></tr>
<tr><td>udCorrStart</td><td>UDINT</td><td>Correction starting value at which the output of correction values commences.
The value can be changed when the block is active
<table>
<tr><td>Range</td><td>0 ... 999999999</td></tr>
<tr><td>Default</td><td>15000</td></tr>
</table></td></tr>
<tr><td>udCorrStop</td><td>UDINT</td><td>Correction stop value at which the output of correction values ends.
The value can be changed when the block is active
<table>
<tr><td>Range</td><td>0 ... 999999999</td></tr>
<tr><td>Default</td><td>19999</td></tr>
</table></td></tr>
</table>

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Ramp function is active</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table></td></tr>
</table>

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Warning
<table>
<tr><td>Range</td><td>Meaning</td></tr>
<tr><td>1</td><td>Excess velocity of input values (Δ diInVal) so that the permissible range can be detected correctly, for example</td></tr>
<tr><td>2</td><td>Read from correction FIFO not possible because, for example, mark detection has not yet commenced</td></tr>
<tr><td>3</td><td>Impermissible direction of rotation prevented
only for 'enMode' = CORRECT_SET2OUT_NB, CORRECT_ADD2OUT_NB</td></tr>
</table>
Error
<table>
<tr><td>Range</td><td>Meaning</td></tr>
<tr><td>1</td><td>Illegal FIFO index</td></tr>
<tr><td>2</td><td>Illegal mode</td></tr>
<tr><td>3</td><td>Illegal modulo format; 'udModulo' > 1000000000</td></tr>
<tr><td>4</td><td>Illegal "maximum valid correction value":
'udMaxCorr' ≥ 'udModulo'</td></tr>
<tr><td>5</td><td>Illegal correction starting value:
'udCorrStart' ≥ 'udModulo'</td></tr>
<tr><td>6</td><td>Illegal correction stopping value:
'udCorrStop' ≥ 'udModulo'</td></tr>
<tr><td>7</td><td>Illegal correction starting / stopping value combination</td></tr>
</table>
</td></tr>
<tr><td>boCorrLim</td><td>BOOL</td><td>Correction limiting
Display a limit of the correction value according to 'udMaxCorr'.
The variable is set to true for one cycle after 'udCorrStart' and before 'udCorrStop'</td></tr>
<tr><td>diOutVal</td><td>DINT</td><td>Output value

Output of the correction value in the form of a linear interpolation covering the range 'udCorrStart' through 'udCorrStop' ('enMode' = 'CORRECT_SET2OUT' or 'CORRECT_SET2OUT_NB').
Outputs the correction value in the form of a linear interpolation covering the range 'udCorrStart' through 'udCorrStop' additively linked to the input value 'diInVal'
('enMode' = 'CORRECT_ADD2OUT' or 'CORRECT_ADD2OUT_NB').

</td></tr>
</table>

Input and output variables

Name	Type	Description
stCorrFifo	STRUCT	ST_CORR_FIFO Correction value FIFO Transfer of detected correction values with 'PM_DETECT' function block The correction values are output at the 'diOutVal' output in the form of a linear interpolation, distributed across the range from 'udCorrStart' to 'udCorrStop'

4.3.7 PM_DETECT (FB)

The 'PM_DETECT' function block is used to detect a printing mark and calculate correction values.

The correction values are calculated from the difference between the setpoint position and the position at the time at which a printing mark is detected.
The correction values are saved in a FIFO structure.

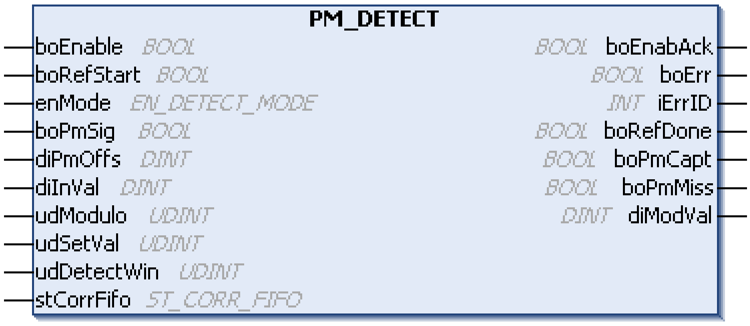
Thus the spacing between the printing mark sensors can be greater than one format.

Further properties:

- Automatic homing to the first printing mark
  Definition of the coordinate reference
- Allowance of a validity range for the printing mark
- Signaling of printing mark detection

Combined with the 'PM_CORRECT' function block, 'PM_DETECT' facilitates efficient printing mark control (PMC). Siehe 'AmkPmc - Printing mark control specific to AMK' auf Seite 233.;

User interface



Input variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnable</td><td>BOOL</td><td>Enable signal: With a positive edge, the initialisation of the block starts.
As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC.
In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.</td></tr>
<tr><td>boRefStart</td><td>BOOL</td><td>Start of a new homing cycle; alignment with next printing mark
Only applies in conjunction with 'enMode' = DETECT_AUTO.
When the block is activated in 'DETECT_AUTO' mode, a homing cycle is started without 'boRefStart' being evaluated.
A positive edge change at 'boRefStart' triggers repeat homing without the block having to be reactivated (positive edge at 'boEnable').</td></tr>
<tr><td>enMode</td><td>ENUM</td><td>EN_DETECT_MODE
Selection mode of the operating mode
<table>
<tr><td>Default</td><td>DETECT_AUTO</td></tr>
</table>
<table>
<tr><th>Range</th><th>Meaning</th></tr>
<tr><td>DETECT_AUTO</td><td>Automatic homing with reference to the first printing mark</td></tr>
<tr><td>DETECT_
MANUAL</td><td>Manual homing
The printing mark must be aligned manually with the sensor prior to enabling with 'boEnable'.</td></tr>
</table></td></tr>
<tr><td>boPmSig</td><td>BOOL</td><td>Printing mark signal (PM signal)
Signal indicating a printing mark inside the modulo format
(The signal must remain pending for at least 1 sampling time)</td></tr>
<tr><td>diPmOffs</td><td>DINT</td><td>Printing mark offset (PM offset)
Describes the deviation between the time-discrete input value 'diInVal' (kT0) and the actual input value 'diInVal'(TboPmSig) at the time of the edge change on the printing mark signal
The following applies:</td></tr>
<tr><td>diInVal</td><td>DINT</td><td>Input value referenced by the subsequent input variables</td></tr>
</table>

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>udModulo</td><td>UDINT</td><td>Modulo format
Describes the setpoint distance between two consecutive printing marks.
The value is saved when the block is activated (positive edge at 'boEnable'). A subsequent change does not affect the active block.<table><tr><td>Range</td><td>0 ... 1000000000</td></tr><tr><td>Default</td><td>2000</td></tr></table></td></tr>
<tr><td>udSetVal</td><td>UDINT</td><td>PM setpoint
Describes the expected distance between printing mark and printing mark sensor.
The value is saved when the block is activated (positive edge at 'boEnable'). A subsequent change does not affect the active block.
If udSetVal ≥ udModulo, n correction values "0" are entered in 'stCorrFifo'.

This corresponds to a slip in the correction value of n formats. In other words, the mark sensor is positioned n formats upstream of the tool position.<table><tr><td>Default</td><td>1000</td></tr></table></td></tr>
<tr><td>udDetectWin</td><td>UDINT</td><td>Permissible range
A 'boPmSig' flag signal is permitted within this range. The value can be changed online when the block is active<table><tr><td>Range</td><td>0 ... 999999999</td></tr><tr><td>Default</td><td>5000</td></tr></table></td></tr>
</table>

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state<table><tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr><tr><td>TRUE</td><td>Error</td></tr></table></td></tr>
</table>

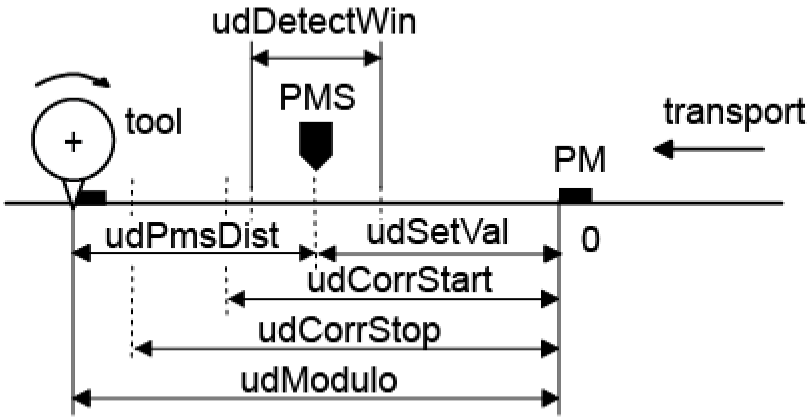
<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Warning
<table>
<tr><th>Value</th><th>Meaning</th></tr>
<tr><td>1</td><td>Excess velocity of input values (Δ 'diInVal') so that the permissible range can be detected correctly, for example</td></tr>
<tr><td>2</td><td>Write to correction FIFO not possible, e.g. because correction is not being carried out</td></tr>
<tr><td>3</td><td>Illegal direction of rotation, e.g. if the direction of rotation changes compared with the reaching of the first mark; a correction value is not entered in the FIFO</td></tr>
</table>
Error
<table>
<tr><th>Value</th><th>Meaning</th></tr>
<tr><td>1</td><td>Illegal FIFO index</td></tr>
<tr><td>2</td><td>Illegal mode</td></tr>
<tr><td>3</td><td>Illegal modulo format
'udModulo' > 1000000000</td></tr>
<tr><td>4</td><td>Illegal PM setpoint
'udSetVal'/'udModulo' > MAX_CORR_FIFO_IND_1</td></tr>
<tr><td>5</td><td>Illegal validity range
'udDetecWin' ≥ 'udModulo'</td></tr>
</table>
</td></tr>
<tr><td>boRefDone</td><td>BOOL</td><td>Homing cycle completed
Acknowledgement signal to indicate that a homing cycle has been completed.
In mode 'enMode' = DETECT_AUTO, 'boRefDone' = FALSE when the block is activated or on a positive edge change at 'boRefStart'.
Once the first mark has been detected, 'boRefDone' = TRUE is set.
In mode 'enMode' = DETECT_MANUAL, this variable is of no significance. 'boRefDone' = TRUE always applies.</td></tr>
<tr><td>boPmCapt</td><td>BOOL</td><td>Printing mark detected
Signal indicating that a printing mark has been detected inside the permissible range.
'boPmCapt' is TRUE for one cycle only.
At the same time, the calculated correction value is entered in the FIFO correction value.</td></tr>
<tr><td>boPmMiss</td><td>BOOL</td><td>Printing mark missing
Signal indicating that a mark has not been detected inside the permissible range.
'boPmMiss' is TRUE for one cycle only.
At the same time, the value "0" is entered in the FIFO correction value.</td></tr>
<tr><td>diModVal</td><td>DINT</td><td>Modulo value
Displays the current modulo position (0 ≤ diModVal < udModulo).
The sign depends on the direction of rotation.</td></tr>
</table>

Input and output variables

Name	Type	Description
stCorrFifo	STRUCT	ST_CORR_FIFO Correction value FIFO Transfer of detected correction values with 'PM_CORRECT' function block 'diCorrVal'[...] = 'diModVal' - 'udSetVal'

Description

Abbildung 12: PM_DETECT: Input variables



The figure illustrates the relationship of some of the input variables used by the 'PM_DETECT' and 'PM_CORRECT' blocks.

Where: (udSetVal % udModulo) < udCorrStart < udCorrStop < udModulo

If udPmsDist > udModulo, then: udSetVal = [udModulo • (n+1)] -X
where

X = udPmsDist % udModulo

$$n = \frac{\text{udPmsDist} - 1}{\text{udModulo}}$$

The 'FudPmcSetVal' function calculates this value

Siehe 'FudPmcSetVal (F)' auf Seite 233.

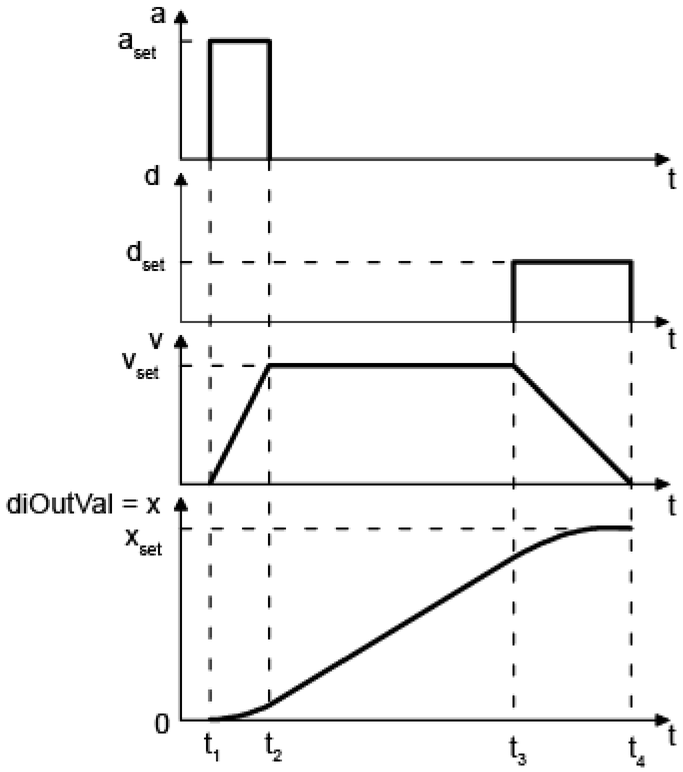
(See document Software description AmkPmc library , Part no. 205009).

4.3.8 POS (FB)

The 'POS' function block supports fast positioning controlled via binary inputs.

The movement sequence is defined with the position ('diPosition'), velocity ('udVelocity'), acceleration ('udAccel') and deceleration ('udDecel') parameters, along with the selected operating mode ('enMode').
All parameters, with the exception of the position, can be changed during the positioning process.

Abbildung 13: POS / POS_1: Principle of positioning



a_{set} Acceleration value 'udAccel'

d_{set} Deceleration setpoint 'udDecel'

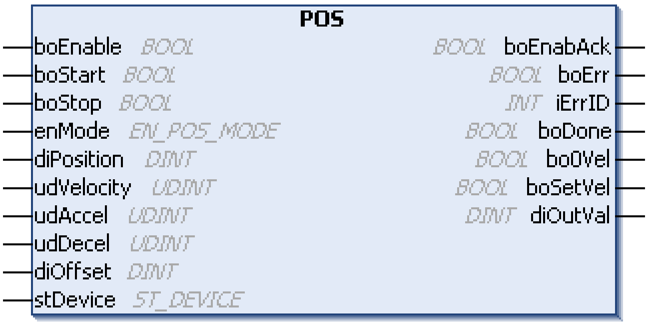
v_{set} Setpoint of positioning velocity 'udVelocity'

x_{set} Position setpoint 'diPosition'

Temporal behavior

t = t1	The output values are output with the active edge of 'boStart' (interpolation).
t1 ≤ t ≤ t2	The velocity is increased proportional to the acceleration value.
t = t2	The velocity setpoint is reached
t2 ≤ t ≤ t3	Constant velocity phase The positioning operation runs at setpoint velocity until the deceleration phase commences.
t = t3	Stop time Start of deceleration in order to come to a standstill in time t4 at the defined end point with velocity 0. This point in time is dependent upon the selected operating mode and possibly on the stop signal 'boStop'.
t3 ≤ t ≤ t4	Deceleration phase The velocity is reduced proportional to the deceleration value.
t = t4	End point The predefined position is reached at this point in time

User interface



Input variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnable</td><td>BOOL</td><td>Enable signal: With a positive edge, the initialisation of the block starts.
As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC.
In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.</td></tr>
<tr><td>boStart</td><td>BOOL</td><td>With a positive edge, the execution of the block starts.</td></tr>
<tr><td>boStop</td><td>BOOL</td><td>With a positive edge, the execution of the block is aborted or completed.</td></tr>
<tr><td>enMode</td><td>ENUM</td><td>EN_POS_MODE
Selection mode of the selected movement sequence (operating mode)
(Siehe 'Operating modes of the POS / POS_1 function blocks' auf Seite 112.)
<table>
<tr><td>Default</td><td>POS_REL</td></tr>
</table>
<table>
<tr><th>Range</th><th>Meaning</th></tr>
<tr><td>POS_REL</td><td>Relative positioning</td></tr>
<tr><td>POS_REL_ RETRIG</td><td>Relative positioning, retriggerable</td></tr>
<tr><td>POS_MODULO</td><td>Modulo positioning</td></tr>
<tr><td>POS_ INTERPOSED</td><td>Positioning with override function with retraction</td></tr>
<tr><td>POS_ INTERPOSED_ NB</td><td>Positioning with override function without retraction</td></tr>
</table></td></tr>
<tr><td>diPosition</td><td>DINT</td><td>Setpoint position
Definition of the final position
<table>
<tr><td>Unit</td><td>incr</td></tr>
<tr><td>Default</td><td>600000</td></tr>
</table></td></tr>
<tr><td>udVelocity</td><td>UDINT</td><td>Setpoint velocity
Definition of the final velocity
<table>
<tr><td>Range</td><td>0 ... 300000000</td></tr>
<tr><td>Unit</td><td>incr/s</td></tr>
<tr><td>Default</td><td>200000</td></tr>
</table></td></tr>
<tr><td>udAccel</td><td>UDINT</td><td>Acceleration with which the target velocity is run
<table>
<tr><td>Range</td><td>0 ... 400000000</td></tr>
<tr><td>Unit</td><td>incr/s^2</td></tr>
<tr><td>Default</td><td>100000</td></tr>
</table></td></tr>
</table>

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>udDecel</td><td>UDINT</td><td>Deceleration with which a lower target velocity is achieved
<table>
<tr><td>Range</td><td>0 ... 400000000</td></tr>
<tr><td>Unit</td><td>incr/s^2</td></tr>
<tr><td>Default</td><td>1000000</td></tr>
</table></td></tr>
<tr><td>diOffset</td><td>DINT</td><td>Offset of the counter value to the homing pulse
<table>
<tr><td>Unit</td><td>Incr</td></tr>
</table></td></tr>
</table>

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table></td></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Warning
<table>
<tr><th>Range</th><th>Meaning</th></tr>
<tr><td>1</td><td>Setpoint velocity = 0</td></tr>
<tr><td>2</td><td>Illegal setpoint velocity; limited to minimum or maximum value</td></tr>
<tr><td>3</td><td>Acceleration = 0</td></tr>
<tr><td>4</td><td>Illegal acceleration; limited to minimum or maximum value</td></tr>
<tr><td>5</td><td>Deceleration = 0</td></tr>
<tr><td>6</td><td>Illegal deceleration; limited to minimum or maximum value</td></tr>
<tr><td>7</td><td>Deceleration value corrected</td></tr>
</table>
Error
<table>
<tr><th>Range</th><th>Meaning</th></tr>
<tr><td>1</td><td>Illegal mode ('enMode')</td></tr>
<tr><td>2</td><td>Illegal offset in 'POS_INTERPOSED_NB' mode</td></tr>
</table></td></tr>
<tr><td>boDone</td><td>BOOL</td><td>Response that the function block has been completely executed.</td></tr>
<tr><td>bo0Vel</td><td>BOOL</td><td>When 'bo0Vel' is active, no setpoint is output.</td></tr>
<tr><td>boSetVel</td><td>BOOL</td><td>When 'boSetVel' is active, the target velocity has been reached.</td></tr>
<tr><td>diOutVal</td><td>DINT</td><td>Output value</td></tr>
</table>

Input and output variables

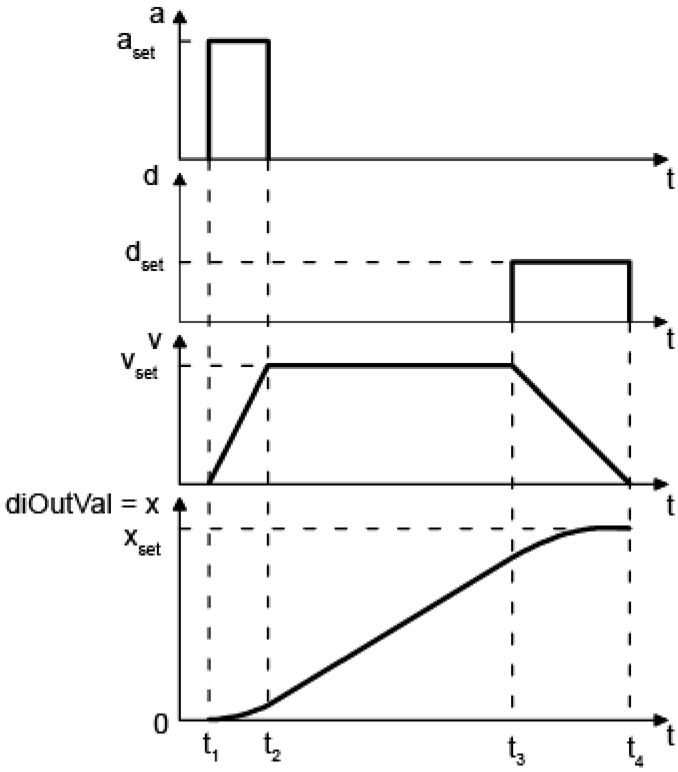
Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

4.3.9 POS_1 (FB)

The 'POS_1' function block supports fast positioning controlled via binary inputs.

The movement sequence is defined with the position ('diPosition'), velocity ('udVelocity'), acceleration ('udAccel') and deceleration ('udDecel') parameters, along with the selected operating mode ('enMode').
All parameters, with the exception of the position, can be changed during the positioning process.

Abbildung 14: POS / POS_1: Principle of positioning



a_{set} Acceleration value 'udAccel'

d_{set} Deceleration setpoint 'udDecel'

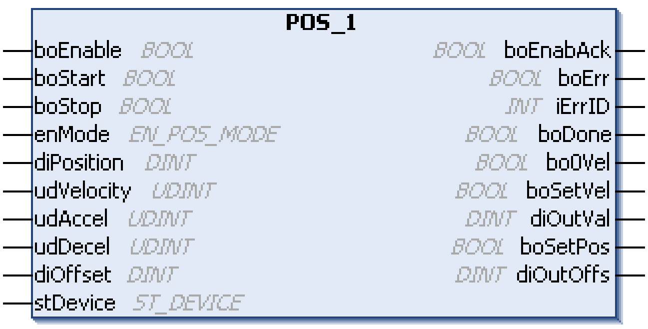
v_{set} Setpoint of positioning velocity 'udVelocity'

x_{set} Position setpoint 'diPosition'

Temporal behavior

t = t1	The output values are output with the active edge of 'boStart' (interpolation).
t1 ≤ t ≤ t2	The velocity is increased proportional to the acceleration value.
t = t2	The velocity setpoint is reached
t2 ≤ t ≤ t3	Constant velocity phase The positioning operation runs at setpoint velocity until the deceleration phase commences.
t = t3	Stop time Start of deceleration in order to come to a standstill in time t4 at the defined end point with velocity 0. This point in time is dependent upon the selected operating mode and possibly on the stop signal 'boStop'.
t3 ≤ t ≤ t4	Deceleration phase The velocity is reduced proportional to the deceleration value.
t = t4	End point The predefined position is reached at this point in time

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.
boStart	BOOL	With a positive edge, the execution of the block starts.
boStop	BOOL	With a positive edge, the execution of the block is aborted or completed.
enMode	ENUM	EN_POS_MODE Selection mode of the selected movement sequence (operating mode) (Siehe 'Operating modes of the POS / POS_1 function blocks' auf Seite 112.) Default: POS_REL Range / Meaning: POS_REL: Relative positioning POS_REL_RETRIG: Relative positioning, retriggerable POS_REL_RETRIG_EXT: Relative positioning, retriggerable, enhanced function POS_MODULO: Modulo positioning POS_INTERPOSED: Positioning with override function with retraction POS_INTERPOSED_NB: Positioning with override function without retraction
diPosition	DINT	Setpoint position Definition of the final position Unit: incr Default: 600000
udVelocity	UDINT	Setpoint velocity Definition of the final velocity Range: 0 ... 300000000 Unit: incr/s Default: 200000
udAccel	UDINT	Acceleration with which the target velocity is run Range: 0 ... 400000000 Unit: incr/s^2 Default: 100000
udDecel	UDINT	Deceleration with which a lower target velocity is achieved Range: 0 ... 400000000 Unit: incr/s^2 Default: 1000000
diOffset	DINT	Offset of the counter value to the homing pulse Unit: Incr

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table></td></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Warning
<table>
<tr><td>Range</td><td>Meaning</td></tr>
<tr><td>1</td><td>Setpoint velocity = 0</td></tr>
<tr><td>2</td><td>Illegal setpoint velocity; limited to minimum or maximum value</td></tr>
<tr><td>3</td><td>Acceleration = 0</td></tr>
<tr><td>4</td><td>Illegal acceleration; limited to minimum or maximum value</td></tr>
<tr><td>5</td><td>Deceleration = 0</td></tr>
<tr><td>6</td><td>Illegal deceleration; limited to minimum or maximum value</td></tr>
<tr><td>7</td><td>Deceleration value corrected</td></tr>
<tr><td>8</td><td>Retrigger not possible</td></tr>
<tr><td>9</td><td>Retriggered movement not until after the end of the previous positioning</td></tr>
</table>
Error
<table>
<tr><td>Range</td><td>Meaning</td></tr>
<tr><td>1</td><td>Illegal mode ('enMode')</td></tr>
<tr><td>2</td><td>Illegal offset in POS_INTERPOSED_NB mode</td></tr>
</table></td></tr>
<tr><td>boDone</td><td>BOOL</td><td>Response that the function block has been completely executed.</td></tr>
<tr><td>bo0Vel</td><td>BOOL</td><td>When 'bo0Vel' is active, no setpoint is output.</td></tr>
<tr><td>boSetVel</td><td>BOOL</td><td>When 'boSetVel' is active, the target velocity has been reached.</td></tr>
<tr><td>diOutVal</td><td>DINT</td><td>Output value</td></tr>
<tr><td>boSetPos</td><td>BOOL</td><td>In retrigger mode the signal for a cycle becomes active when the retrigger position has been reached.</td></tr>
<tr><td>diOutOffs</td><td>DINT</td><td>Offset value before the retrigger is started; only in retrigger mode</td></tr>
</table>

Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

4.3.10 Operating modes of the POS / POS_1 function blocks

The following operating modes are defined according to the 'enMode' variable:

- POS_REL — Relative positioning
- POS_REL_RETRIG — Relative positioning, retriggerable
- POS_REL_RETRIG_EXT — Relative positioning, retriggerable, enhanced function ('POS_1' only)
- POS_MODULO — Modulo positioning
- POS_INTERPOSED — Positioning with override function with retraction

- POS_INTERPOSED_NB Positioning with override function without retraction

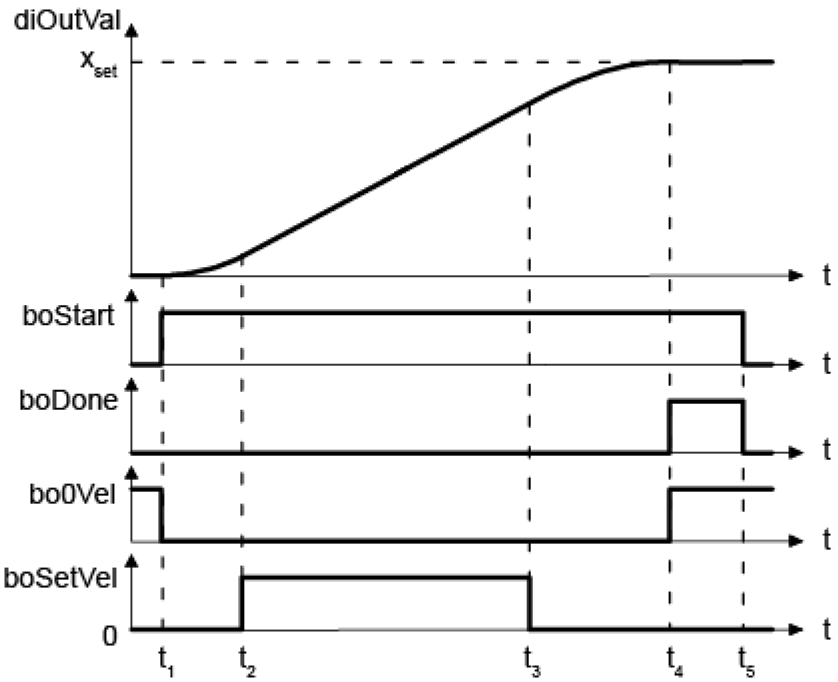
All input variables, with the exception of the position, can be changed during the positioning process.

- 'diPosition'
  The position value is saved at the start of positioning (positive edge at 'boStart'). If the position is required again during the positioning process, e.g. in 'POS_REL_RETRIG' or 'POS_MODULO' mode, it is always the saved position value that is referenced.
  However, a new position value can be transferred in 'POS_REL_RETRIG_EXT' mode in the context of retriggering.
- 'diOffset'
  In positioning with override function ('POS_INTERPOSED' or 'POS_INTERPOSED_NB'), the 'diOffset' is added to the saved 'diPosition' when there is a positive edge at 'boStop'.
- Changes in value during the deceleration phase (see POS (FB), POS_1 (FB) Figure: Principle of positioning, t3 ≤ t ≤ t4) are ignored.
- In some modes (e.g. 'POS_INTERPOSED_NB'), the deceleration 'udDecel' is automatically increased to the smallest possible valid value if the deceleration setpoint is not compatible with predefined positioning behavior. In this case the corresponding warning is output (iErrID = 7: deceleration value corrected).

enMode = POS_REL

In 'relative positioning' mode, a drive is moved by a predefined position setpoint (x_{set}) relative to the position at that moment in time.

Abbildung 15: POS / POS_1: enMode = POS_REL

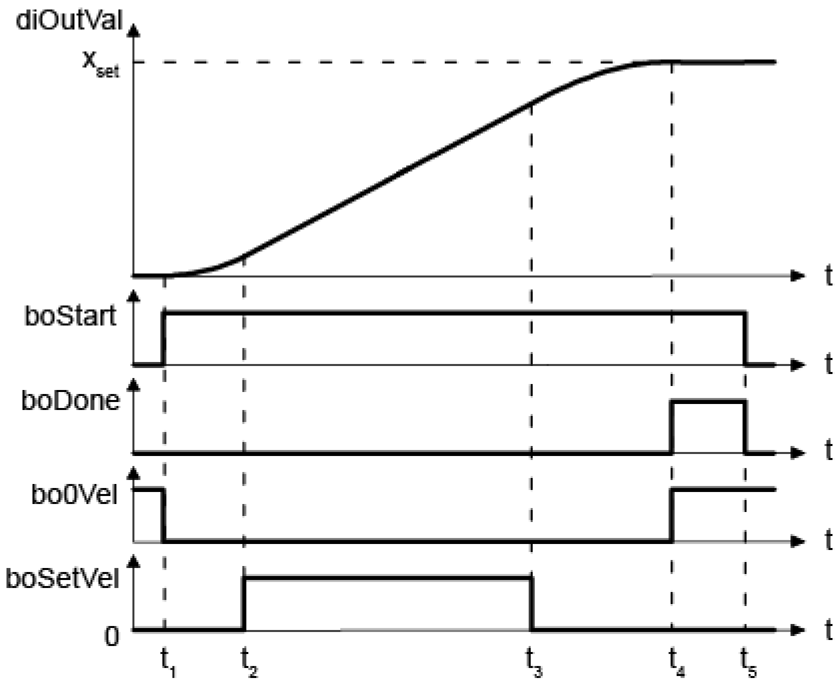


enMode = POS_REL_RETRIG

In 'relative retriggerable positioning' mode, a drive is moved by a predefined position setpoint (x_{set}) relative to the position at that moment in time.

It is also possible to retrigger the positioning process before the final position is reached.
This means that the current final position of the position setpoint (xset) is added if a new start trigger is detected during movement (positive edge at 'boStart').

Abbildung 16: POS / POS_1: enMode = POS_REL_RETRIG

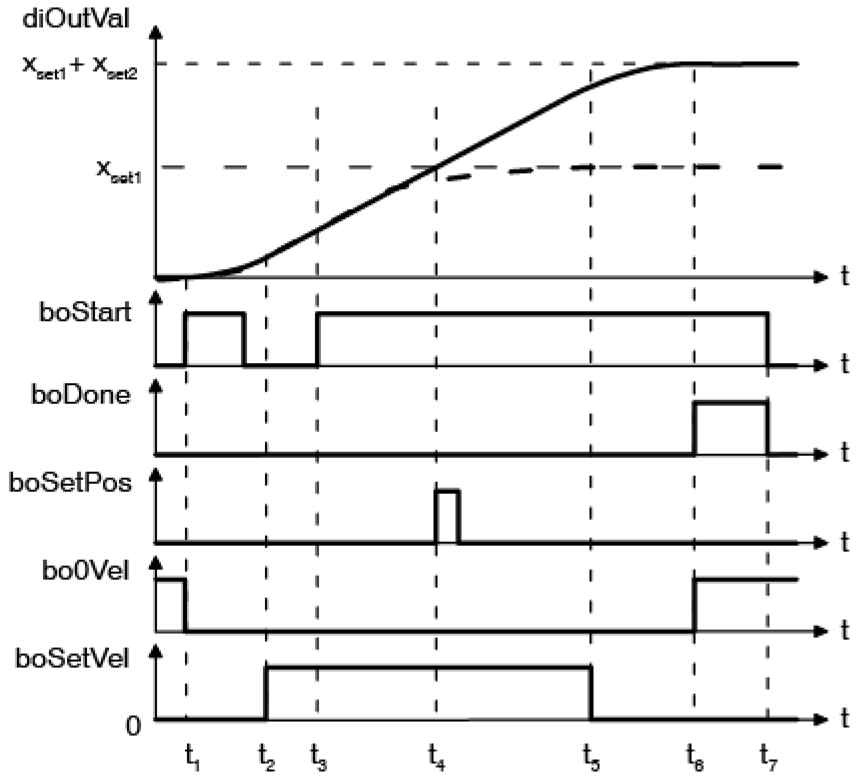


Retriggering is only possible if the deceleration phase has not yet been reached (t < t4)

enMode = POS_REL_RETRIG_EXT

'Relative, retriggerable positioning with enhanced function' operating mode corresponds to 'POS_REL_RETRIG' mode with the addition that the time of transition from the positioning started originally to the retriggered positioning is signaled at the output signal 'boSetPos'. To this end, this output is set to TRUE for one cycle when the transition occurs. Moreover, in the context of retriggering, a new relative setpoint position can be specified with 'diPosition'.

Abbildung 17: POS_1: 'enMode' = POS_REL_RETRIG_EXT

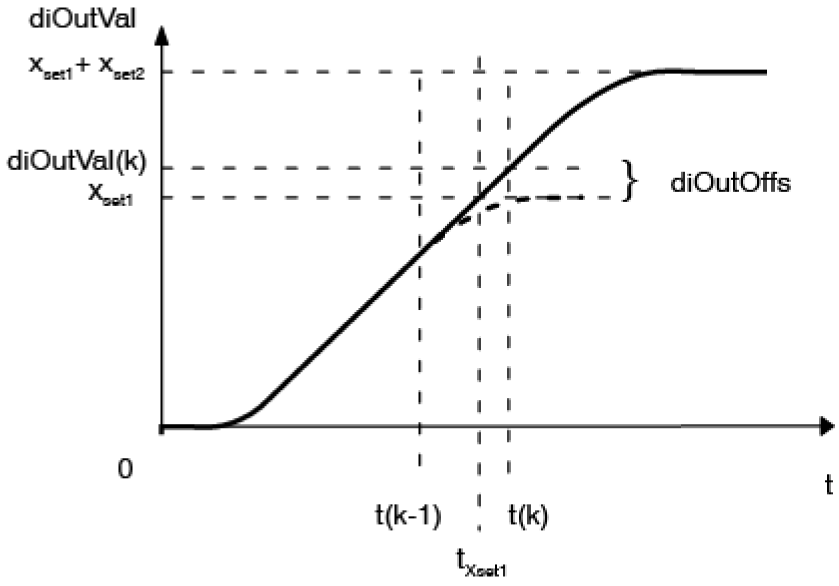


t = t1 Positioning in 'POS_REL_RETRIG_EXT' mode with 'diPosition' = X_{set1} is started ('boStart' = TRUE).
In the absence of retriggering, positioning would come to a stop in setpoint X_{set1} (represented by the dashed characteristic).

t = t3 Retriggering is triggered with 'diPosition' = X_{set2} prior to the start of the deceleration phase.
This produces the new setpoint position X_{set1}+X_{set2}. Selecting the same velocity 'udVelocity' produces the solid characteristic shown in the figure.

t = t4 'boSetPos' designates the transition from the original positioning started with 'diPosition'=X_{set1} to the retriggered position X_{set1}+X_{set2}

Abbildung 18: POS_1: 'enMode' = POS_REL_RETRIG_EXT, 'diOutOffs'



At point in time t = t(k), the current position deviates from the original setpoint X_{set1}. This difference is output through the 'diOutOffs' variable.

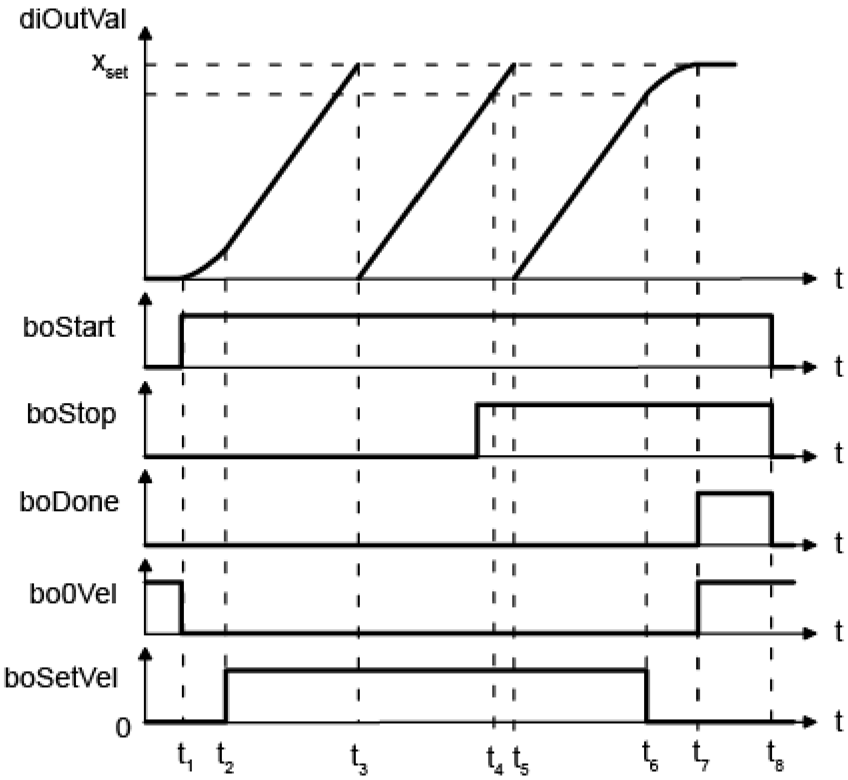
diOutOffs = diOutVal(k) – Xset1

For a movement in the positive direction this results in a positive value for 'diOutOffs'.

enMode = POS_MODULO

In 'modulo positioning' mode, a drive is moved continuously. The drive can come to a stop at a multiple of the predefined position 'diPosition'.

Abbildung 19: POS / POS_1: enMode = POS_MODULO



t = t1 Positioning starts with a positive edge at 'boStart'

t = t4 After a positive edge at 'boStop', the position moves to the next possible multiple of X_{set}.
ΔdiPosition = n • X_{set}; n - integer positive number

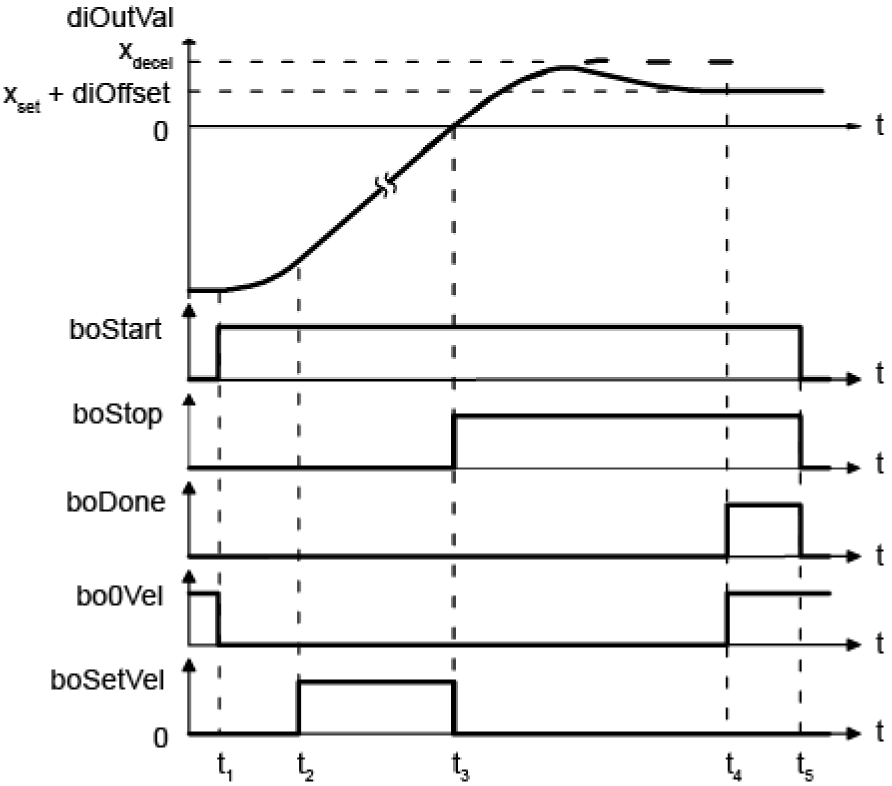
enMode = POS_INTERPOSED

In 'positioning with override function with retraction' mode, a drive is moved continuously. The drive can come to a stop at a predefined position 'diPosition'.
This position references the current position at the time when a positive edge change of the stop signal 'boStop' is detected (t = t3). The position value is corrected by a value pending at 'diOffset' at the time the drive comes to a stop (position value = 'diPosition' + 'diOffset').

'Positioning with override function with retraction' is used when

'diPosition' + 'diOffset' < X_{decel}

Abbildung 20: POS / POS_1: 'enMode' = POS_INTERPOSED



enMode = POS_INTERPOSED_NB

In 'positioning with override function without retraction' mode, a drive is moved continuously. The drive can come to a stop at a predefined position 'diPosition'.
This position references the current position at the time when a positive edge change of the stop signal 'boStop' is detected (t = t3). The position value is corrected by a value pending at 'diOffset' at the time the drive comes to a stop (position value = 'diPosition' + 'diOffset').

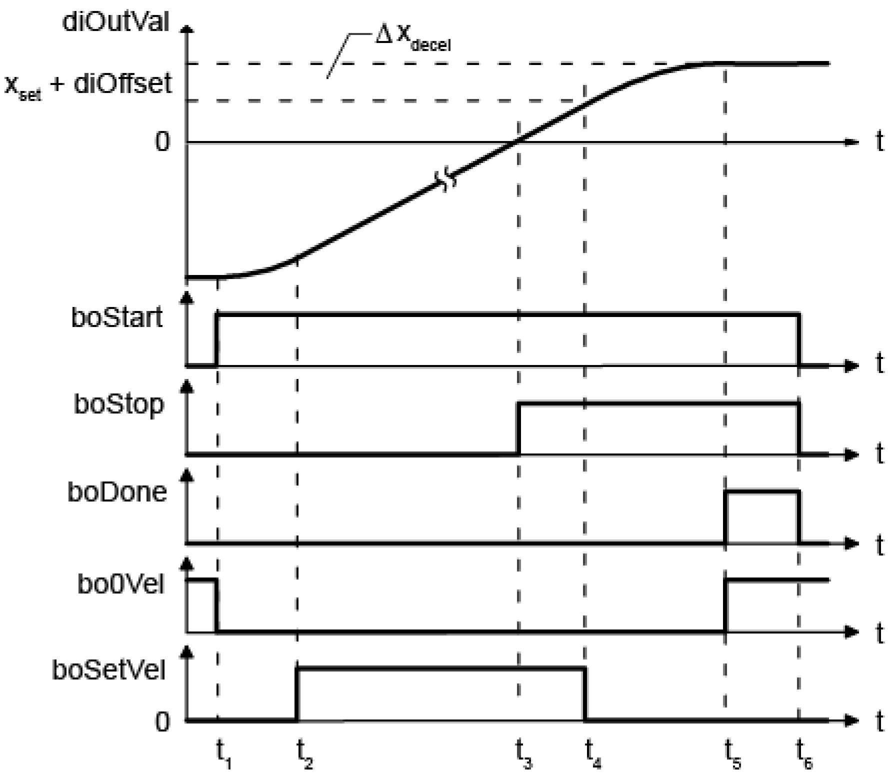
'Positioning with override function without retraction' is used when

'diPosition' + 'diOffset' ≥ X_{decel}

Positioning without retraction is achieved by the deceleration being increased automatically until the deceleration path X_{decel} = position value.

If: (diOffset + xset) < 0, the block generates an error message (iErrID = 2; illegal offset)

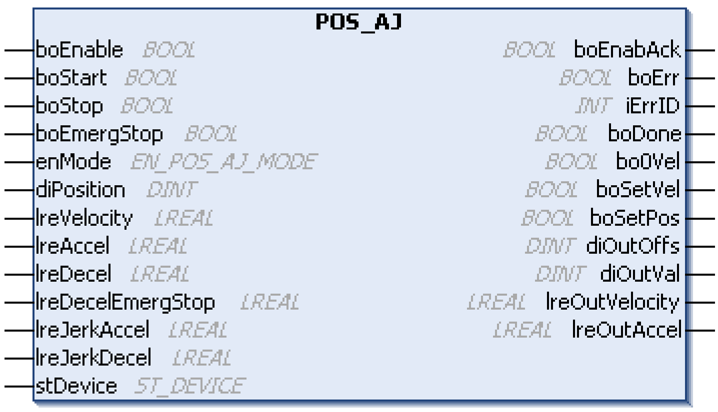
Abbildung 21: POS / POS_1: 'enMode' = POS_INTERPOSED_NB



4.3.11 POS_AJ (FB)

The 'POS_AJ' function block supports fast positioning controlled via binary inputs.
The movement sequence is defined with the position ('diPosition'), velocity ('lreVelocity'), acceleration ('lreAccel') and deceleration ('lreDecel'), jerk during acceleration ('lreJerkAccel'), and jerk during deceleration ('lreJerkDecel') parameters, along with the selected operating mode ('enMode').
All parameters can be changed during the positioning process.

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.
boStart	BOOL	With a positive edge, the execution of the block starts.

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boStop</td><td>BOOL</td><td>With a positive edge, the execution of the block is aborted or completed.</td></tr>
<tr><td>boEmergStop</td><td>BOOL</td><td>EMERGENCY STOP: The setpoint of the velocity is decreased to zero along the emergency-stop ramp. Once initiated, an emergency stop cannot be aborted.</td></tr>
<tr><td>enMode</td><td>ENUM</td><td>EN_POS_AJ_MODE
Selection mode of the selected movement sequence
<table><tr><td>Default</td><td>POS_AJ_REL</td></tr></table>
<table><tr><th>Range</th><th>Meaning</th></tr><tr><td>POS_AJ_REL</td><td>Relative positioning</td></tr><tr><td>POS_AJ_REL_ RETRIG</td><td>Relative positioning, retriggerable</td></tr></table></td></tr>
<tr><td>diPosition</td><td>DINT</td><td>Setpoint position
Definition of the final position
<table><tr><td>Unit</td><td>incr</td></tr><tr><td>Default</td><td>600000</td></tr></table></td></tr>
<tr><td>lreVelocity</td><td>LREAL</td><td>Setpoint velocity with which the final velocity is set
<table><tr><td>Range</td><td>$1.43\ 10^{-13}$ < |'lreVelocity'| < $1.43\ 10^{+13}$</td></tr><tr><td>Unit</td><td>incr/s</td></tr><tr><td>Default</td><td>200000</td></tr></table></td></tr>
<tr><td>lreAccel</td><td>LREAL</td><td>Acceleration for positioning and jog mode
<table><tr><td>Range</td><td>$1{,}43\ 10^{-13}$ < |'lreAccel'| < $1{,}43\ 10^{+16}$</td></tr><tr><td>Unit</td><td>incr/s^2</td></tr><tr><td>Default</td><td>1000000</td></tr></table></td></tr>
<tr><td>lreDecel</td><td>LREAL</td><td>Deceleration for positioning and jog mode
<table><tr><td>Range</td><td>$1{,}43\ 10^{-13}$ < |'lreDecel'| < $1{,}43\ 10^{+16}$</td></tr><tr><td>Unit</td><td>incr/s^2</td></tr><tr><td>Default</td><td>1000000</td></tr></table></td></tr>
<tr><td>lreDecelEmergStop</td><td>LREAL</td><td>Deceleration for emergency stop
<table><tr><td>Range</td><td>$1.43\ 10^{-13}$ < |'lreDecelEmergStop'| < $1.43\ 10^{+16}$</td></tr><tr><td>Unit</td><td>incr/s^2</td></tr><tr><td>Default</td><td>10000000</td></tr></table></td></tr>
<tr><td>lreJerkAccel</td><td>LREAL</td><td>Jerk during acceleration
<table><tr><td>Range</td><td>$1.43\ 10^{-13}$ < |'lreJerkAccel'| < $1.43\ 10^{+19}$</td></tr><tr><td>Unit</td><td>incr/s^3</td></tr><tr><td>Default</td><td>100000</td></tr></table></td></tr>
<tr><td>lreJerkDecel</td><td>LREAL</td><td>Jerk during deceleration
<table><tr><td>Range</td><td>$1.43\ 10^{-13}$ < |'lreJerkDecel'| < $1.43\ 10^{+19}$</td></tr><tr><td>Unit</td><td>incr/s^3</td></tr><tr><td>Default</td><td>1000000</td></tr></table></td></tr>
</table>

Output variables

Name	Type	Description
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state<table><tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr><tr><td>TRUE</td><td>Error</td></tr></table></td></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output<table><tr><td colspan="2">iErrID = 0</td><td>No error</td></tr><tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr><tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr></table>Warning<table><tr><th>Range</th><th>Meaning</th></tr><tr><td>1</td><td>Setpoint velocity = 0</td></tr><tr><td>2</td><td>Illegal setpoint velocity; limited to minimum or maximum value</td></tr><tr><td>3</td><td>Acceleration = 0</td></tr><tr><td>4</td><td>Illegal acceleration; limited to minimum or maximum value</td></tr><tr><td>5</td><td>Deceleration = 0</td></tr><tr><td>6</td><td>Illegal deceleration; limited to minimum or maximum value</td></tr><tr><td>7</td><td>Deceleration value corrected</td></tr><tr><td>8</td><td>Retrigger not possible</td></tr><tr><td>9</td><td>Retriggered movement not until after the end of the previous positioning</td></tr><tr><td>21</td><td>Emergency stop deceleration = 0
'lreDecelEmergStop' = 'lreDecel'</td></tr><tr><td>22</td><td>Excess emergency stop deceleration
'lreDecelEmergStop' = max. value</td></tr><tr><td>23</td><td>Jerk during acceleration = 0
'lreJerkAccel' = max. value</td></tr><tr><td>24</td><td>Excess jerk during acceleration
'lreJerkAccel' = max. value</td></tr><tr><td>25</td><td>Jerk during deceleration = 0
'lreJerkDecel' = max. value</td></tr><tr><td>26</td><td>Excess jerk during deceleration
'lreJerkDecel' = max. value</td></tr><tr><td>27</td><td>Illegal arguments during calculation of movement profile</td></tr><tr><td>28</td><td>Setpoint velocity adapted in the context of movement profile calculation</td></tr><tr><td>29</td><td>Final velocity adapted in the context of movement profile calculation during acceleration</td></tr><tr><td>30</td><td>Final velocity adapted in the context of movement profile calculation during deceleration</td></tr></table>Error<table><tr><th>Range</th><th>Meaning</th></tr><tr><td>1</td><td>Illegal mode 'enMode'</td></tr></table></td></tr>
<tr><td>boDone</td><td>BOOL</td><td>Response that the function block has been completely executed.</td></tr>
<tr><td>bo0Vel</td><td>BOOL</td><td>When 'bo0Vel' is active, no setpoint is output.</td></tr>
<tr><td>boSetVel</td><td>BOOL</td><td>When 'boSetVel' is active, the target velocity has been reached.</td></tr>
<tr><td>boSetPos</td><td>BOOL</td><td>In retrigger mode the signal for a cycle becomes active when the retrigger position has been reached.</td></tr>
<tr><td>diOutOffs</td><td>DINT</td><td>Offset value before the retrigger is started; only in retrigger mode<table><tr><td>Unit</td><td>incr</td></tr></table></td></tr>
</table>

Name	Type	Description
diOutVal	DINT	Output value position Unit | incr
lreOutVelocity	LREAL	Output value velocity Unit | incr/s
lreOutAccel	LREAL	Output value acceleration Unit | incr/s^2

Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

4.3.12 RATIO_ABS (FB)

The 'RATIO_ABS' function block is used to multiply and divide 32-bit values.

However, unlike 'RATIO_INC_1', for example, the input value 'dInVal' is treated as an absolute value (no input differences are generated).
Although the product 'diInVal' x 'diMultiplier' is generated as a 64-bit value initially, it must be possible for the result following division by 'udDivider' to be displayed as a 32-bit value.

Thus:

$$\text{diOutVal} = \text{diInVal} \times \frac{\text{diMultiplier}}{\text{udDivider}}$$

User interface

Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.
diInVal	DINT	Absolute Input value Input value differences are not generated
diMultiplier	DINT	Multiplier by which the input value differences are multiplied Default | 10000
udDivider	UDINT	Divisor by which the input value differences are divided Default | 10000

Output variables

Name	Type	Description
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled

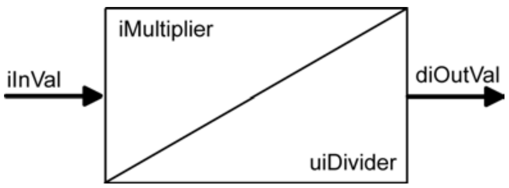
<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table></td></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Warning
<table>
<tr><th>Range</th><th>Meaning</th></tr>
<tr><td>2</td><td>Divisor = 0; set to 1</td></tr>
<tr><td>5</td><td>Output value cannot be displayed as DINT</td></tr>
</table></td></tr>
<tr><td>diOutVal</td><td>DINT</td><td>Output value</td></tr>
</table>

4.3.13 RATIO_INC (FB)

The 'RATIO_INC' function block performs multiplication and division to specific increments; the ratio of input increments to output increments is defined.

The calculation algorithm ensures that any remainder is not lost. Moreover, the overrun of the 16-bit input value is managed by working with incremental difference:

Abbildung 22: RATIO_INC: Principle



$$\frac{\Delta \text{diOutVal}}{\Delta \text{iInVal}} = \frac{\text{iMultiplier}}{\text{uiDivider}}$$

$$\text{diOutVal}(k) = \text{diOutVal}(k-1) + \frac{\text{iMultiplier}}{\text{uiDivider}} \times (\text{iInVal}(k) - \text{iInVal}(k-1))$$

k	Sampling point (in time)
iInVal(k-1) := iInVal(k)	for k = 0 (positive edge of 'boEnable')

User interface

Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.
iInVal	INT	Input value e.g. the low-order word of the 32-bit actual position

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>iMultiplier</td><td>INT</td><td>Multiplier by which the input value differences are multiplied<table><tr><td>Range</td><td>-32766 ... +32766</td></tr><tr><td>Unit</td><td>incr</td></tr><tr><td>Default</td><td>10000</td></tr></table></td></tr>
<tr><td>uiDivider</td><td>UINT</td><td>Divisor by which the input value differences are divided<table><tr><td>Range</td><td>1 ... +32767</td></tr><tr><td>Unit</td><td>incr</td></tr><tr><td>Default</td><td>10000</td></tr></table></td></tr>
</table>

Output variables

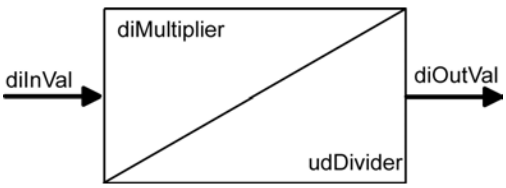
<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state<table><tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr><tr><td>TRUE</td><td>Error</td></tr></table></td></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output<table><tr><td colspan="2">iErrID = 0</td><td>No error</td></tr><tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr><tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr></table>Warning<table><tr><th>Range</th><th>Meaning</th></tr><tr><td>1</td><td>Multiplier too high;
limited to minimum or maximum value</td></tr><tr><td>2</td><td>Divisor = 0; set to 1</td></tr><tr><td>3</td><td>Divisor > 32767;
remainder not taken into account</td></tr></table></td></tr>
<tr><td>diOutVal</td><td>DINT</td><td>Output value
Sum of incoming increments, weighted with the 'iMultiplier'/'uiDivider' ratio</td></tr>
</table>

4.3.14 RATIO_INC_1 (FB)

The 'RATIO_INC_1' function block performs multiplication and division to specific increments; the ratio of input increments to output increments is defined.

'RATIO_INC_1' is equivalent to the 'RATIO_INC' function block, except that the 'diInVal', 'diMultiplier', and 'udDivider' input variables are 32-bit values.

Abbildung 23: RATIO_INC_1: Principle



k			Sampling point (in time)
diInVal(k-1)	:=	diInVal(k)	for k = 0 (positive edge of 'boEnable')

User interface

Input variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnable</td><td>BOOL</td><td>Enable signal: With a positive edge, the initialisation of the block starts.
As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC.
In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.</td></tr>
<tr><td>diInVal</td><td>DINT</td><td>Input value</td></tr>
<tr><td>diMultiplier</td><td>DINT</td><td>Multiplier by which the input value differences are multiplied<table><tr><td>Default</td><td>10000</td></tr></table></td></tr>
<tr><td>udDivider</td><td>UDINT</td><td>Divisor by which the input value differences are divided<table><tr><td>Default</td><td>10000</td></tr></table></td></tr>
</table>

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state<table><tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr><tr><td>TRUE</td><td>Error</td></tr></table></td></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output<table><tr><td colspan="2">iErrID = 0</td><td>No error</td></tr><tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr><tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr></table>Warning<table><tr><th>Range</th><th>Meaning</th></tr><tr><td>2</td><td>Divisor = 0; set to 1</td></tr><tr><td>4</td><td>Output difference cannot be displayed as DINT;
the difference is limited to the maximum DINT value</td></tr></table></td></tr>
<tr><td>diOutVal</td><td>DINT</td><td>Output value</td></tr>
</table>

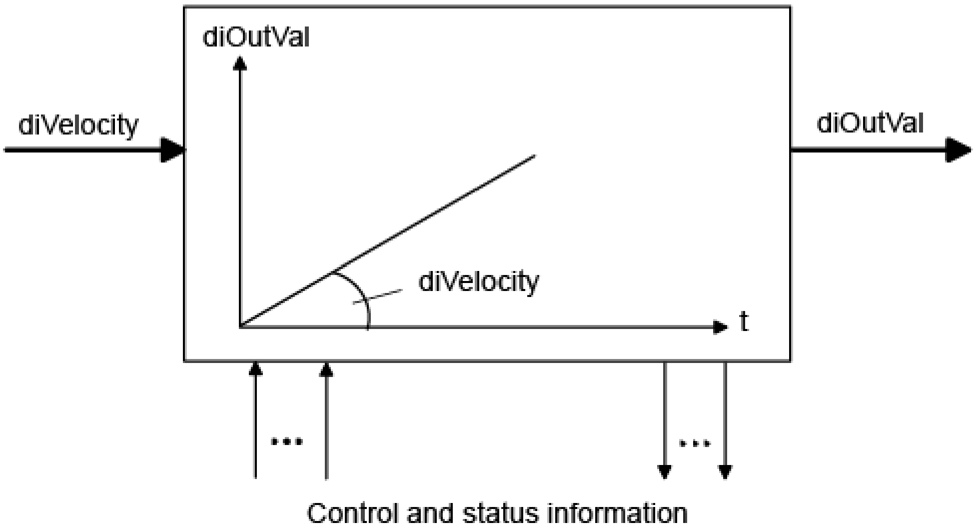
4.3.15 VGEN (FB)

The 'VGEN' function block is a velocity generator.
The output value is a position setpoint which changes in proportion with the velocity. It is also possible to output a defined number of increments.

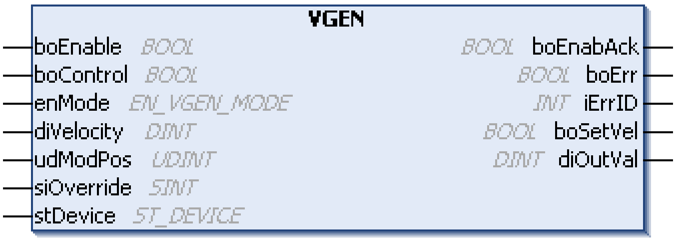
The following functions are supported:

- Generation of an increment increase in accordance with a definable velocity.
- Online changes to input parameters
- Modes for continuous and cyclic increment generation

Abbildung 24: VGEN: Block diagram



User interface



Input variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnable</td><td>BOOL</td><td>Enable signal: With a positive edge, the initialisation of the block starts.
As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC.
In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.</td></tr>
<tr><td>boControl</td><td>BOOL</td><td>Start / Stop
<table>
<tr><th colspan="2">Range</th><th>Meaning</th></tr>
<tr><td rowspan="2">enMode = VGEN_ CONT</td><td>FALSE</td><td>Velocity = 0</td></tr>
<tr><td>TRUE</td><td>Velocity = 'diVelocity'</td></tr>
<tr><td>enMode = VGEN_ CYCLE</td><td>FALSE -> TRUE</td><td>Velocity = 'diVelocity'
for a series of 'udModPos' increments</td></tr>
</table></td></tr>
<tr><td>enMode</td><td>ENUM</td><td>EN_VGEN_MODE
Selection mode of the operating mode
<table>
<tr><td>Default</td><td>VGEN_CONT</td></tr>
</table>
<table>
<tr><th>Range</th><th>Meaning</th></tr>
<tr><td>VGEN_CONT</td><td>Continuous increment increase,
as long as 'boControl' = TRUE</td></tr>
<tr><td>VGEN_CYCLE</td><td>Increment increase by the defined value 'udModPos' on a positive edge at 'boControl'</td></tr>
</table></td></tr>
</table>

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>diVelocity</td><td>DINT</td><td>Setpoint velocity
Definition of the final velocity<table><tr><td>Range</td><td>-300000000 ... 300000000</td></tr><tr><td>Unit</td><td>incr/s</td></tr><tr><td>Default</td><td>1000</td></tr></table></td></tr>
<tr><td>udModPos</td><td>UDINT</td><td>Modulo position
Number of increments to be output in mode 'enMode' = VGEN_CYCLE<table><tr><td>Unit</td><td>incr</td></tr><tr><td>Default</td><td>2000</td></tr></table></td></tr>
<tr><td>siOverride</td><td>SINT</td><td>Velocity output factor<table><tr><td>Range</td><td>-100 ... +100</td></tr><tr><td>Unit</td><td>%</td></tr><tr><td>Default</td><td>100</td></tr></table></td></tr>
</table>

Output variables

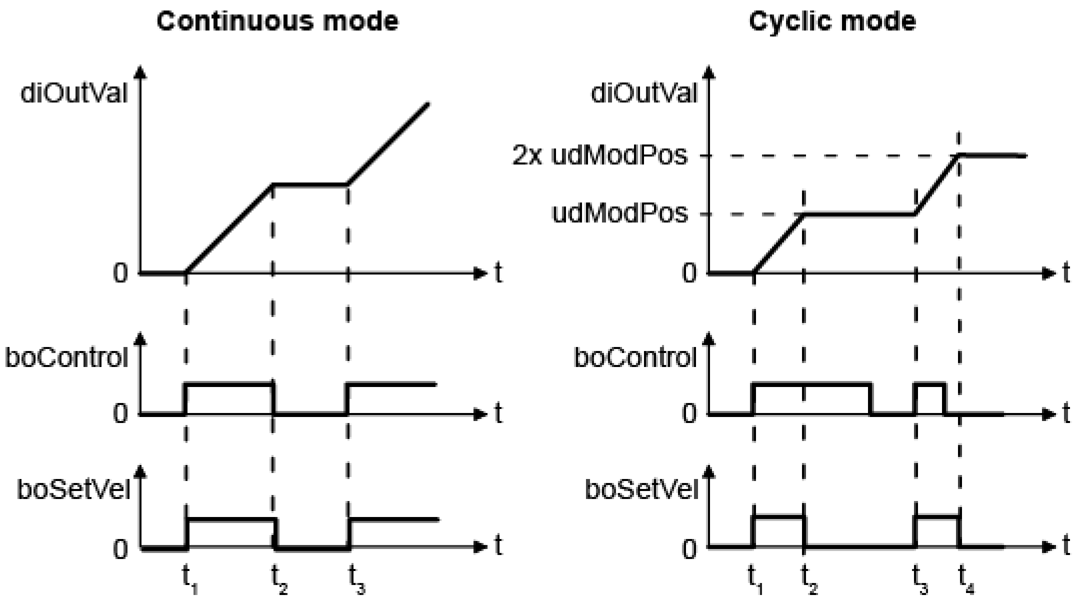
<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state<table><tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr><tr><td>TRUE</td><td>Error</td></tr></table></td></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output<table><tr><td colspan="2">iErrID = 0</td><td>No error</td></tr><tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr><tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr></table>Warning<table><tr><th>Range</th><th>Meaning</th></tr><tr><td>1</td><td>Illegal mode</td></tr></table>Error<table><tr><th>Range</th><th>Meaning</th></tr><tr><td>1</td><td>Illegal setpoint velocity
(limited to minimum or maximum value)</td></tr><tr><td>2</td><td>Illegal override
(limited to minimum or maximum value)</td></tr></table></td></tr>
<tr><td>boSetVel</td><td>BOOL</td><td>When 'boSetVel' is active, the target velocity has been reached.</td></tr>
<tr><td>diOutVal</td><td>DINT</td><td>Output value</td></tr>
</table>

Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

Description

Abbildung 25: VGEN: Principle of operation



'enMode' is used to differentiate between continuous and cyclic operation.

- Continuous operation: 'enMode' = VGEN_CONT
  Generation of a continuous increment increase, corresponding to a predefined setpoint velocity ('diVelocity' [incr/s]). The increment increase is controlled with the control signal ('boControl').

  'boControl' = TRUE t1 ≤ t ≤ t2; t3 < t

  'boControl' = FALSE t < t1; t2 ≤ t ≤ t3

- Cyclic operation: 'enMode' = VGEN_CYCLE
  Generation of a defined increment increase ('udModPos' [incr]) corresponding to a predefined setpoint velocity ('diVelocity' [incr/s]). The increment increase is controlled with the positive edge of the control signal ('boControl').

  'boControl' =FALSE -> TRUE (t = t1; t = t3): Increment increase udModPos t1 ≤ t ≤ t2; t3 ≤ t ≤ t4

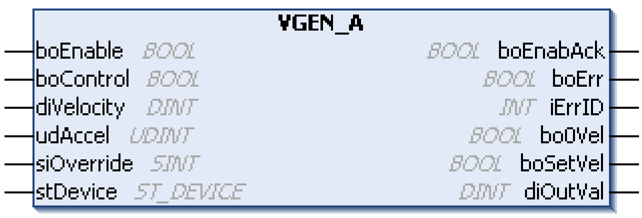
4.3.16 VGEN_A (FB)

The function block 'VGEN_A' is a velocity generator with definable acceleration.
The output value is a position with an increment difference proportional to the velocity 'diVelocity' and a change in increment difference proportional to the acceleration 'udAccel'.
The block can be used for direct control of a drive. It can also be used as an input value generator for other blocks ('CAM_PROF', 'CAM_CONT', etc.).

The following functions are supported:

- Generation of an increment increase in accordance with a definable velocity
- Specification of a defined acceleration / deceleration
- Online changes to input parameters.

User interface



Input variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnable</td><td>BOOL</td><td>Enable signal: With a positive edge, the initialisation of the block starts.
As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC.
In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.</td></tr>
<tr><td>boControl</td><td>BOOL</td><td>Start / Stop
Control of increment output
<table>
<tr><td>FALSE</td><td>Reduction in velocity proportional to 'udAccel' until velocity 0 is reached</td></tr>
<tr><td>TRUE</td><td>Increase in velocity proportional to 'udAccel' until velocity 'diVelocity' is reached</td></tr>
</table></td></tr>
<tr><td>diVelocity</td><td>DINT</td><td>Setpoint velocity
Definition of the final velocity
<table>
<tr><td>Range</td><td>-300000000 ... 300000000</td></tr>
<tr><td>Unit</td><td>incr/s</td></tr>
<tr><td>Default</td><td>500000</td></tr>
</table></td></tr>
<tr><td>udAccel</td><td>UDINT</td><td>Acceleration with which the target velocity is run
<table>
<tr><td>Range</td><td>-4000000000 ... 4000000000</td></tr>
<tr><td>Unit</td><td>incr/s^2</td></tr>
<tr><td>Default</td><td>100000</td></tr>
</table></td></tr>
<tr><td>siOverride</td><td>SINT</td><td>Velocity output factor
<table>
<tr><td>Range</td><td>-100 ... +100</td></tr>
<tr><td>Unit</td><td>%</td></tr>
<tr><td>Default</td><td>100</td></tr>
</table></td></tr>
</table>

Output variables

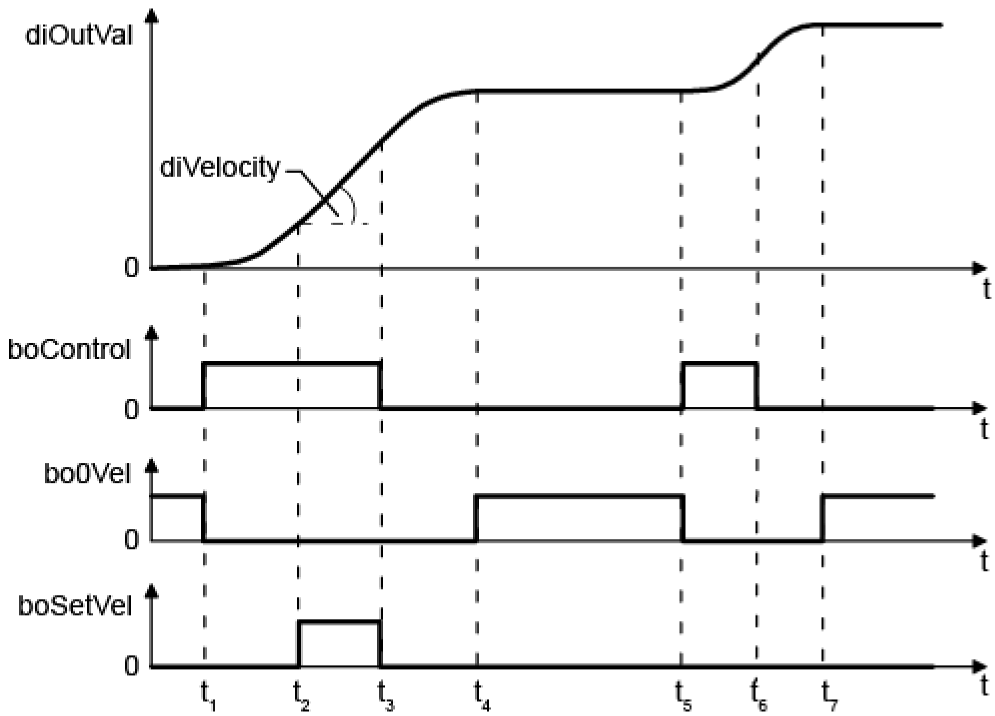
<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table></td></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error
<table>
<tr><th>Range</th><th>Meaning</th></tr>
<tr><td>1</td><td>Illegal setpoint velocity
(limited to minimum or maximum value)</td></tr>
<tr><td>2</td><td>Illegal acceleration
(limited to minimum or maximum value)</td></tr>
<tr><td>3</td><td>Illegal override
(limited to minimum or maximum value)</td></tr>
</table></td></tr>
<tr><td>bo0Vel</td><td>BOOL</td><td>When 'bo0Vel' is active, no setpoint is output.</td></tr>
<tr><td>boSetVel</td><td>BOOL</td><td>When 'boSetVel' is active, the target velocity has been reached.</td></tr>
</table>

Name	Type	Description
diOutVal	DINT	Output value

Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

Abbildung 26: VGEN_A: Principle of operation



t = t1	boControl = TRUE	The output of position setpoints (increments) commences once the control signal has been activated
t1 ≤ t ≤ t2		Acceleration phase In this range, the velocity is increased proportional to the acceleration 'udAccel' until the predefined setpoint velocity 'diVelocity' is reached.
t = t2		Setpoint velocity reached; 'boSetVel' = TRUE
t2 ≤ t ≤ t3		In this range, the velocity remains constant (constant increment difference).
t = t3	boControl = FALSE	Once the control signal has been deactivated, the difference of the output increments is reduced.
t3 ≤ t ≤ t4		Deceleration phase In this range, the velocity is reduced proportional to the acceleration 'udAccel' until velocity 0 is reached.
t = t4		Standstill reached; 'bo0Vel' = TRUE
t4 ≤ t ≤ t5		Standstill phase In this range, the output value does not change (velocity = 0)
t = t5	boControl = TRUE	Renewed output of position setpoints
t5 ≤ t ≤ t6		Renewed acceleration
t = t6	boControl = FALSE	Acceleration phase is interrupted before the setpoint velocity is reached
t6 ≤ t≤ t7		Deceleration phase starts immediately without setpoint velocity being reached
t = t7		Standstill reached; 'bo0Vel' = TRUE

4.3.17 VGEN_AJ (FB)

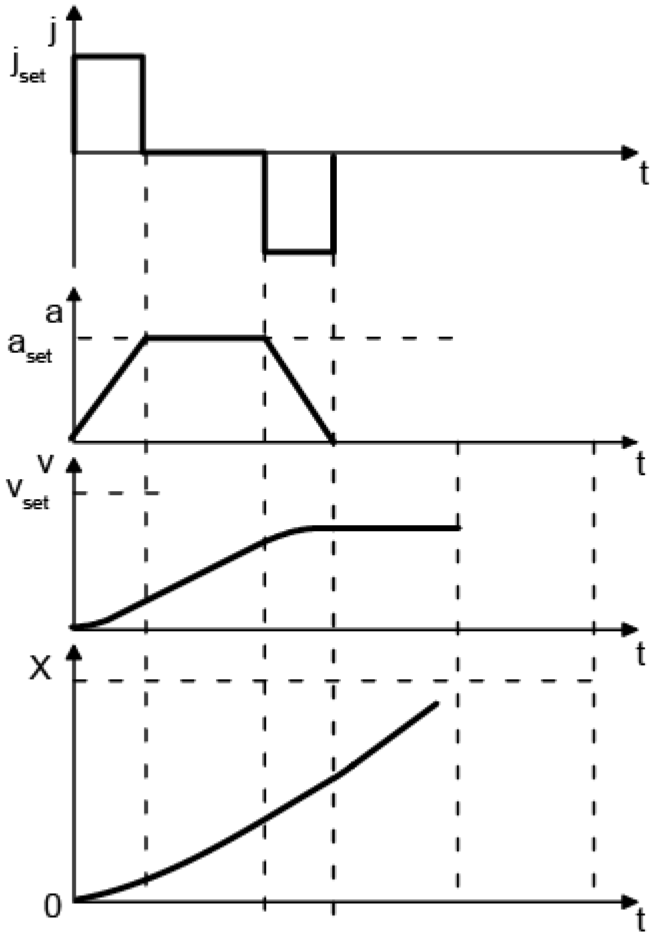
The block 'VGEN_AJ' is a velocity generator with definable values for acceleration and jerk.
The output value is a position with an increment difference proportional to the velocity 'diVelocity' a change in increment difference proportional to the acceleration 'udAccel', and a change in increment difference change proportional to the jerk 'udAccJerk' / 'udDecJerk' / 'udQDecelJerk'.

The block can be used for direct control of a drive. It can also be used as an input value generator for other blocks ('CAM_PROF', 'CAM_CONT', etc.).

The following functions are supported:

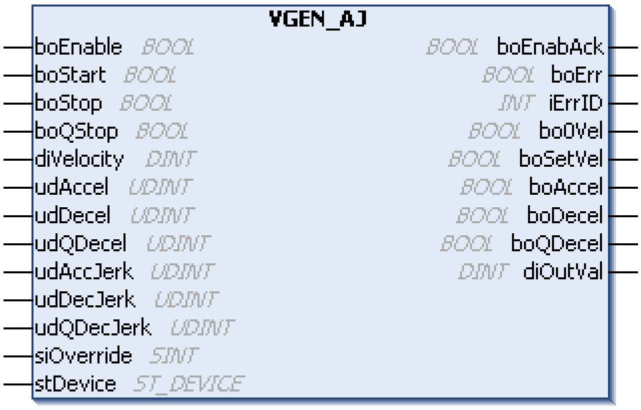
- Generation of an increment increase in accordance with a definable velocity.
- Specification of a defined acceleration (deceleration).
- Special quick stop mode with assigned deceleration values.
- Jerk default values for the various ramps.
- Velocity override
- Online changes to input parameters

Abbildung 27: VGEN_AJ: Principle of operation



The figure illustrates the graphical relationship between jerk j_{set} , acceleration, a_{set}, velocity v_{set}, and the resulting position characteristic X.

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.
boStart	BOOL	Acceleration ramp is generated with the 'udAccel' and 'udAccJerk' parameters until velocity 'diVelocity'
boStop	BOOL	Braking ramp is generated with the 'udDecel' and 'udDecJerk' parameters until velocity 0
boQStop	BOOL	Fast braking ramp is generated with the 'udQDecel' and 'udQDecJerk' parameters until velocity 0. The fast braking ramp cannot be interrupted.
diVelocity	DINT	Setpoint velocity Definition of the final velocity Range: -300000000 … 300000000 Unit: incr/s Default: 500000
udAccel	UDINT	Acceleration with which the target velocity is run Range: -4000000000 ... 4000000000 Unit: incr/s^2 Default: 100000
udDecel	UDINT	Deceleration with which a lower target velocity is achieved Range: -4000000000 ... 4000000000 Unit: incr/s^2 Default: 100000
udQDecel	UDINT	Fast deceleration with which a lower target velocity is achieved Range: -4000000000 ... 4000000000 Unit: incr/s^2 Default: 500000

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>udAccJerk</td><td>UDINT</td><td>Jerk during acceleration<table><tr><td>Range</td><td>-4000000000 ... 4000000000</td></tr><tr><td>Unit</td><td>incr/s^2</td></tr><tr><td>Default</td><td>100000</td></tr></table></td></tr>
<tr><td>udDecJerk</td><td>UDINT</td><td>Jerk during deceleration<table><tr><td>Range</td><td>-4000000000 ... 4000000000</td></tr><tr><td>Unit</td><td>incr/s^2</td></tr><tr><td>Default</td><td>100000</td></tr></table></td></tr>
<tr><td>udQDecJerk</td><td>UDINT</td><td>Jerk during fast deceleration<table><tr><td>Range</td><td>-4000000000 ... 4000000000</td></tr><tr><td>Unit</td><td>incr/s^2</td></tr><tr><td>Default</td><td>100000</td></tr></table></td></tr>
<tr><td>siOverride</td><td>SINT</td><td>Velocity output factor<table><tr><td>Range</td><td>-100 ... +100</td></tr><tr><td>Unit</td><td>%</td></tr><tr><td>Default</td><td>100</td></tr></table></td></tr>
</table>

Output variables

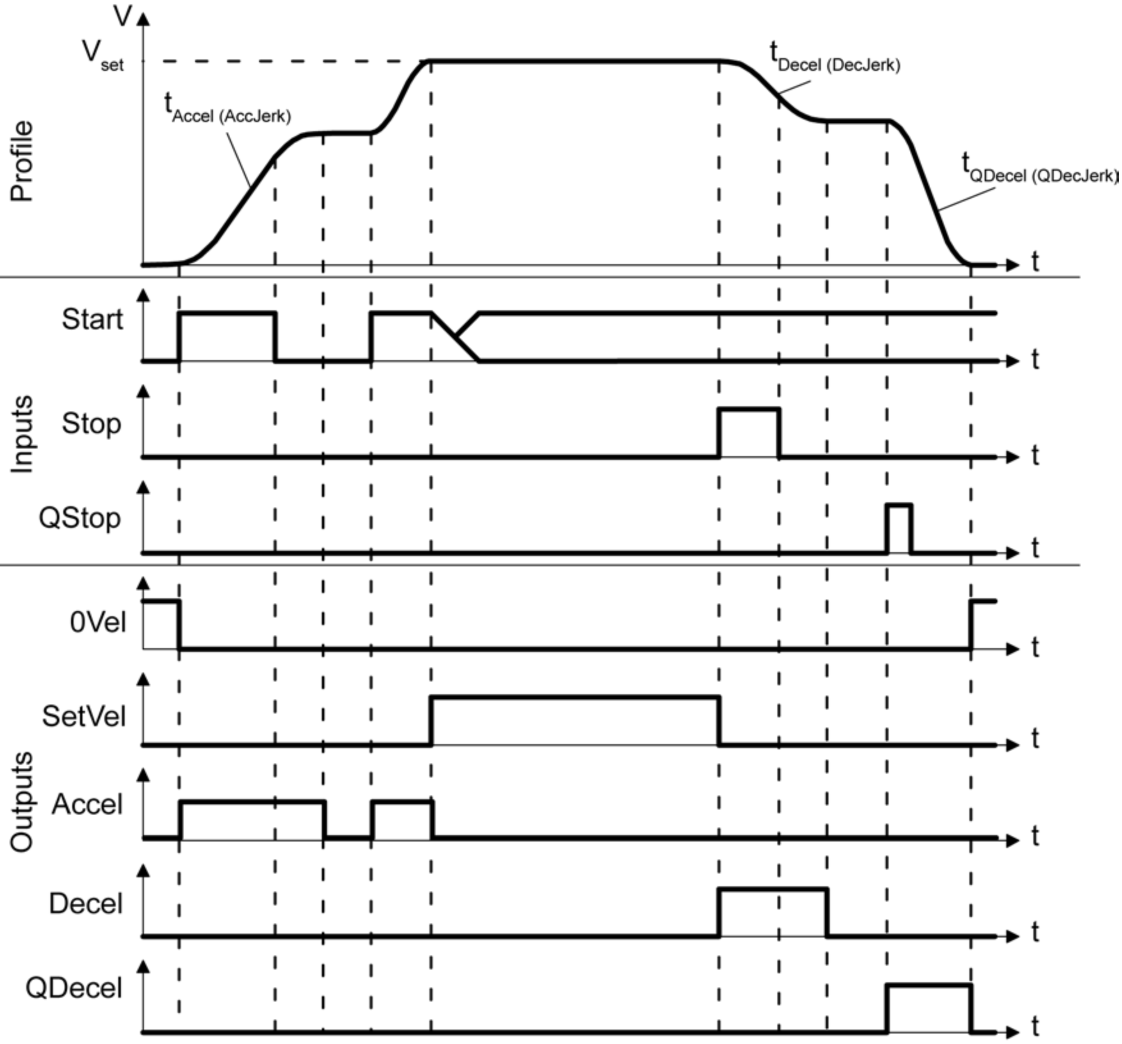
<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state<table><tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr><tr><td>TRUE</td><td>Error</td></tr></table></td></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output<table><tr><td colspan="2">iErrID = 0</td><td>No error</td></tr><tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr><tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr></table>Error<table><tr><th>Range</th><th>Meaning</th></tr><tr><td>1</td><td>Velocity too high</td></tr><tr><td>2</td><td>Acceleration set to 0 / deceleration set to 0</td></tr><tr><td>3</td><td>Acceleration too high / deceleration too high</td></tr><tr><td>4</td><td>Jerk set to 0</td></tr><tr><td>5</td><td>Jerk too high</td></tr><tr><td>6</td><td>Override too high</td></tr><tr><td>7</td><td>Jerk corrected (more than 20% of the setpoint)</td></tr></table></td></tr>
<tr><td>bo0Vel</td><td>BOOL</td><td>When 'bo0Vel' is active, no setpoint is output.</td></tr>
<tr><td>boSetVel</td><td>BOOL</td><td>When 'boSetVel' is active, the target velocity has been reached.</td></tr>
<tr><td>boAccel</td><td>BOOL</td><td>Acceleration phase active</td></tr>
<tr><td>boDecel</td><td>BOOL</td><td>Deceleration phase active</td></tr>
<tr><td>boQDecel</td><td>BOOL</td><td>Fast deceleration phase active</td></tr>
<tr><td>diOutVal</td><td>DINT</td><td>Output value</td></tr>
</table>

Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

Description

Abbildung 28: VGEN_AJ: Behavior of the velocity generator

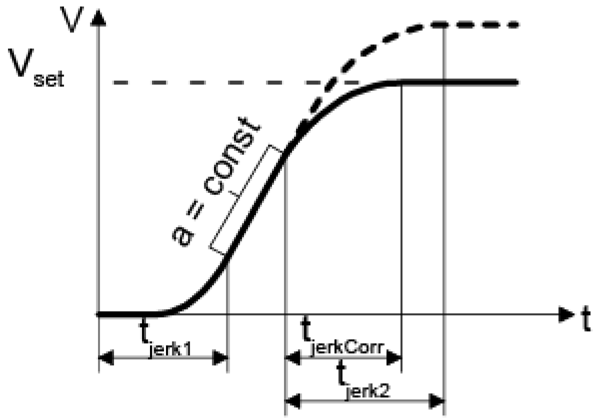


The figure shows how the velocity generator uses different values for the acceleration, deceleration, and fast deceleration ramps. Accordingly, the generator uses three different jerk values, one for each ramp.

All parameters can be changed during block runtime.
However, please note:

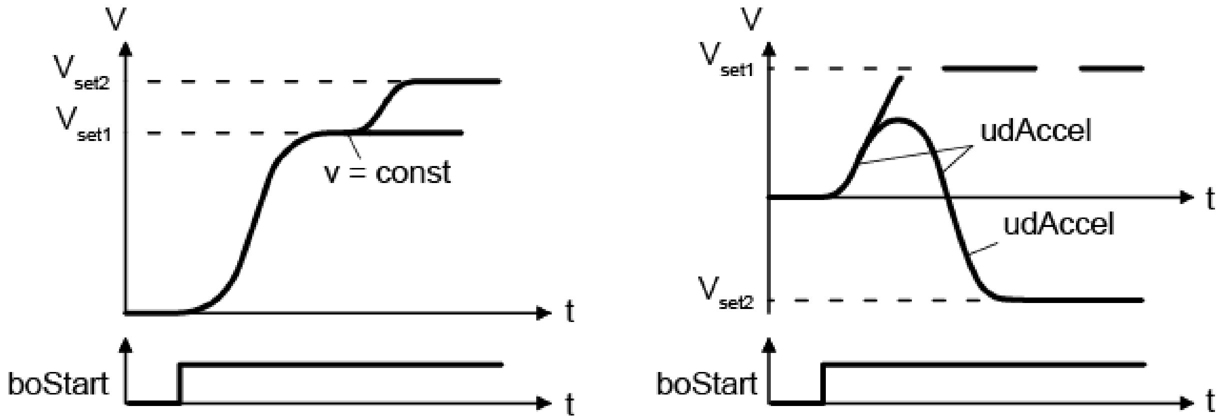
- If the jerk setpoint changes during the constant acceleration phase such that the velocity setpoint (or velocity 0) would be exceeded (or undershot), the jerk phase commences immediately. The minimum possible jerk is calculated for this.

Abbildung 29: VGEN_AJ: Response to change in jerk



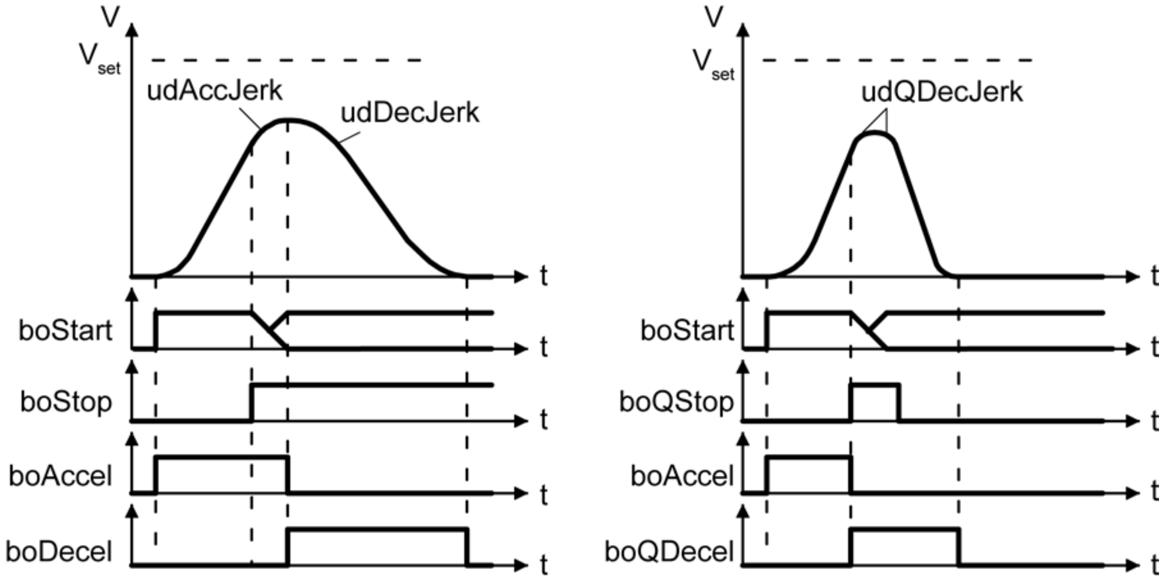
- If the jerk changes after the second jerk phase following constant acceleration / deceleration has commenced, this change is not taken into account.
- If the setpoint velocity or the velocity override changes during the acceleration ramp, the changed value is applied if: 'boStart' = TRUE, 'boStop' = FALSE, 'boQStop' = FALSE.
  If the changed velocity requires negative acceleration, this is done with a ramp defined by 'udAccel' and 'udAccJerk'. Although it looks like a deceleration ramp, in this case the values of the acceleration ramp apply.

Abbildung 30: VGEN_AJ: Response to change in override velocity



- If the 'boStop' input becomes active during the acceleration ramp ('boStop' = TRUE), 'VGEN_AJ' responds initially by generating a jerk phase with 'udAccJerk' until the current acceleration becomes 0. After this, the block starts a deceleration ramp as defined by 'udDecel' and 'udDecJerk'.
  The procedure differs for a fast stop: 'VGEN_AJ' reduces the current acceleration immediately with 'udQDecJerk'. If the current acceleration is 0, the process continues with a deceleration ramp with 'udQDecel'.

  Abbildung 31: VGEN_AJ: Different response to normal stop and fast stop



4.4 System

The following blocks are called from other libraries; they are not usually used directly by the application programmer.

FboAddToBusSysTime	Get local time information
FboGetPlcVarPointers	Get controller-internal address range pointer
FdiGetDiffToBusSysTime	Get local time information

4.4.1 FboAddToBusSysTime (F)

The 'FboAddToBusSysTime' function adds an offset to the system-internal distributed bus clock time of a bus system instance. The result is displayed as a 64-bit time value in nanoseconds.

The function is not used directly by the user.

User interface

Input variables

Name	Type	Description
iInstance	INT	Bus instance number Range: 0 ... 7
diAddTime	DINT	Additive time value added to the 64-bit distributed clock time of the bus instance Unit: ns
pubOutputTime	POINTER	POINTER TO BYTE Pointer to the 64-bit structure in which the calculated time value is written

Output variables

Name	Type	Description
FboAddToBusSysTime	BOOL	Function return value (not used)

4.4.2 FboGetPlcVarPointers (F)

The 'FboGetPlcVarPointers' function queries address range pointers inside the controller. The result is recorded in a structure.

The function provides the basis for the AMK concept for automatic bus configuration. It is used in the context of the AmkDevAccBase library and is not used directly by the user.

User interface

Input variables

Name	Type	Description
iInstance	INT	Bus instance number Range: -1 ... 7 -1: controller-internal 0 ... 7: bus system instance
pstPlcVarPointers	POINTER	POINTER TO ST_PLC_VAR_POINTERS Pointer to the structure which takes up the PLC variables

Output variables

Name	Type	Description
FboGetPlcVarPointers	BOOL	Function return value Range / Meaning FALSE: Unable to retrieve pointers TRUE: Pointers retrieved

4.4.3 FdiGetDiffToBusSysTime (F)

The 'FdiGetDiffToBusSysTime' function queries the difference between a 64-bit time value and the system-internal distributed bus clock time.
The result is returned in nanoseconds.

User interface

Input variables

Name	Type	Description
iInstance	INT	Bus instance number Range: 0 ... 7
pubInputTime	POINTER	POINTER TO BYTE Pointer to the 64-bit structure based on the value of which the difference in relation to the distributed bus clock time is generated

Output variables

Name	Type	Description
FdiGetDiffToBusSysTime	DINT	Return value Difference between 'pubInputTime' and the system-internal distributed bus clock time Unit: ns

4.5 Types

4.5.1 Structures

4.5.1.1 Basic

4.5.1.1.1 ST_LOCAL_TIME_INFO (ST)

Structure elements

Name	Type	Description
diLocToUtcDiff	DINT	Difference between local time and UTC (Coordinated Universal Time) Unit: ms
boIsDst	BOOL	Summertime flag

Structure definition

```
TYPE ST_LOCAL_TIME_INFO:
        STRUCT
                diLocToUtcDiff:DINT;
                boIsDst:BOOL;
        END_STRUCT
END_TYPE
```

4.5.1.2 CamContactor

4.5.1.2.1 ST_CONT (ST)

Structure elements

Name	Type	Description
diOn	DINT	Cam activation point 'diInVal' value at and above which 'boOutVal' = TRUE is set (Siehe 'CAM_CONT (FB)' auf Seite 71.)
diOff	DINT	Cam deactivation point 'diInVal' value at and above which 'boOutVal' = FALSE is set (Siehe 'CAM_CONT (FB)' auf Seite 71.)

Structure definition

```
TYPE ST_CONT:
    STRUCT
        diOn:DINT;
        diOff:DINT;
    END_STRUCT
END_TYPE
```

4.5.1.2.2 ST_CONT_TAB (ST)

The cam table is based on the 'ST_CONT_TAB structure', which permits the definition of up to 16 (MAX_CONT_TAB_IND) cam on and off points.

Structure elements

Name	Type	Description
uiActCams	UINT	The number of array elements (cam activation and deactivation points) currently taken into account in the 'stCam' substructure array Range: 1 ... uiMaxCams Default: 1
uiMaxCams	UINT	Maximum permissible number of array elements of the 'stCam' substructure array Default: 16 (constant)
diRes	DINT	(Not used in the context of the specific function for AMK)
stCam	ARRAY	ARRAY [1..MAX_CONT_TAB_IND] OF ST_CONT Array of cam activation and deactivation points

Structure definition

```
MAX_CONT_TAB_IND:UINT:=16;            (* highest valid index for ST_CONT_TAB.stCam[1...] *)

TYPE ST_CONT_TAB:
    STRUCT
        uiActCams:UINT := 1;
        uiMaxCams:UINT := MAX_CONT_TAB_IND;
        diRes:DINT;
        stCAM:ARRAY[1..MAX_CONT_TAB_IND] OF ST_CONT;
    END_STRUCT
END_TYPE
```

4.5.1.3 CamProfile

4.5.1.3.1 ST_PROF_TAB (ST)

Up to 361 DINT type table elements can be defined with the 'ST_PROF_TAB' structure.

Modes 'enTabType' = PROF_YTAB_NL and 'enTabType' = PROF_XYTAB_NL lift this restriction on table elements.

Structure elements

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>enType</td><td>ENUM</td><td>EN_PROF_TAB_TYPE
Table type, to differentiate between X and XY tables
<table>
<tr><td>Default</td><td>PROF_YTAB</td></tr>
</table>
<table>
<tr><th>Range</th><th>Meaning</th></tr>
<tr><td>PROF_YTAB</td><td>Equidistant X positions, Y positions defined by table value</td></tr>
<tr><td>PROF_XYTAB</td><td>X and Y positions defined by table values</td></tr>
<tr><td>PROF_YTAB_NL</td><td>Equidistant X positions, Y positions defined by table value, not limited</td></tr>
<tr><td>PROF_XYTAB_ NL</td><td>X and Y positions defined by table values, not limited</td></tr>
<tr><td>PROFXYVATAB</td><td>Polynomial table: X and Y positions, velocity, acceleration defined by table values</td></tr>
</table></td></tr>
<tr><td>uiNoElement</td><td>UINT</td><td>Element number of the last table element calculated,
number of table interpolation points</td></tr>
<tr><td>udMasterInc</td><td>UDINT</td><td>Increments of the master drive which produce a table cycle
Max. table X value</td></tr>
<tr><td>diElement[0]
....
diElement[360]</td><td>ARRAY</td><td>ARRAY [0..MAX_PROF_IND] OF DINT
Table elements</td></tr>
</table>

Structure definition

```
MAX_PROF_Y_IND:UINT:=360;                (* highest valid index for diElement[x] *)

TYPE ST_PROF_TAB:
    STRUCT
        enType:EN_PROF_TAB_TYPE ;
        uiNoElement:UINT;
        udMasterInc:UDINT;
        diElement:ARRAY[0…MAX_PROF_Y_IND] OF DINT;
    END_STRUCT
END_TYPE
```

4.5.1.3.2 ST_PROF_XY (ST)

Structure elements

Name	Type	Description
diX	DINT	X value of table interpolation point
diY	DINT	Y value of table interpolation point

Structure definition

```
TYPE ST_PROF_XY:
    STRUCT
        diX:DINT;
        diY:DINT;
    END_STRUCT
END_TYPE
```

4.5.1.3.3 ST_PROF_XYTAB (ST)

The 'ST_PROF_XYTAB' structure defines an XY table whose x axis can be split at will. The table structure contains the x and y values of the function y = f(x).

Structure elements

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>enType</td><td>ENUM</td><td>EN_PROF_TAB_TYPE
Table type, to differentiate between X and XY tables
<table>
<tr><td>Default</td><td>PROF_XYTAB</td></tr>
</table>
<table>
<tr><th>Range</th><th>Meaning</th></tr>
<tr><td>PROF_XYTAB</td><td>X and Y positions defined by table values</td></tr>
<tr><td>PROF_XYTAB_NL</td><td>X and Y positions defined by table values, not limited</td></tr>
</table></td></tr>
<tr><td>uiNoElement</td><td>UINT</td><td>Element number of the last table element calculated,
number of table interpolation points
<table>
<tr><td>Range</td><td>1 ... 180</td></tr>
<tr><td>Default</td><td>180</td></tr>
</table></td></tr>
<tr><td>udMasterInc</td><td>UDINT</td><td>Increments of the master drive which produce a table cycle
Max. table X value
(not used for XY tables)</td></tr>
<tr><td>stElement</td><td>ARRAY</td><td>ARRAY [0..MAX_PROF_XY_IND] OF ST_PROF_XY
Table elements, X and Y positions
<table>
<tr><th>Range</th><th>Meaning</th></tr>
<tr><td>stElement[0]</td><td>X / Y value at zero point of table</td></tr>
<tr><td>diX</td><td>X value at zero point of table, always 0</td></tr>
<tr><td>diY</td><td>Y value at zero point of table, always 0</td></tr>
<tr><td>stElement[1]</td><td>1st X / Y value of the table</td></tr>
<tr><td>diX</td><td>X value of the table</td></tr>
<tr><td>diY</td><td>Y value of the table</td></tr>
<tr><td>stElement[2]</td><td>2nd X / Y value of the table</td></tr>
<tr><td>diX</td><td>X value of the table</td></tr>
<tr><td>diY</td><td>Y value of the table</td></tr>
<tr><td>...</td><td>...</td></tr>
<tr><td>stElement[180]</td><td>180th X / Y value of the table</td></tr>
<tr><td>diX</td><td>X value of the table (where diX > stElement[179].diX)</td></tr>
<tr><td>diY</td><td>Y value of the table</td></tr>
</table></td></tr>
</table>

Structure definitions

Structure definition

```
MAX_PROF_XY_IND:UINT:=180;          (* highest valid index for 'stElement[0…]' *)

TYPE ST_PROF_YTAB:
    STRUCT
        enType:EN_PROF_TAB_TYPE:=PROF_XYTAB;
        uiNoElement:UINT:=MAX_PROF_Y_IND;
        udMasterInc:UDINT:=20000;
        stElement:ARRAY[0…MAX_PROF_XY_IND] OF ST_PROF_XY;
    END_STRUCT
END_TYPE
```

If the table type 'enType' = PROF_XYTAB_NL is selected, the value for MAX_PROF_XY_IND can be redefined at program level. This enables the original limit of up to 180 XY table sections to be increased.
(Siehe 'Number of table interpolation points' auf Seite 87.)

4.5.1.3.4 ST_PROF_YTAB (ST)

The 'ST_PROF_YTAB' structure defines an XY table whose x axis is split equally. The table structure contains the y values of the function y = f(x).

The corresponding x values are generated in the 'CAM_PROF' block with 'uiNoElement'+1 equidistant points.
Thus:

$$A = \frac{udMasterInc}{uiNoElement}$$

where A: equidistant spacing

Structure elements

Name	Type	Description
enType	ENUM	EN_PROF_TAB_TYPE Table type, to differentiate between X and XY tables Default: PROF_YTAB Range / Meaning: **PROF_YTAB**: Equidistant X positions, Y positions defined by table value **PROF_YTAB_NL**: Equidistant X positions, Y positions defined by table value, not limited
uiNoElement	UINT	Element number of the last table element calculated, number of table interpolation points Range: 1 ... 360 Default: 360
udMasterInc	UDINT	Increments of the master drive which produce a table cycle Max. table X value Number of incoming increments at 'dilnVal' necessary for the table to be run through once

Name	Type	Description
diY	ARRAY	ARRAY [0..MAX_PROF_Y_IND] OF DINT Y value of table interpolation point Range: Meaning diY[0]: 0: Y value at zero point of table diY[1]: 1st Y value of table diY[2]: 2nd Y value of table ... diY[360]: 360th Y value of table

Structure definition

```
MAX_PROF_Y_IND:UINT:=360;                    (* highest valid index for ST_PROF_TAB.diY[0…] *)

TYPE ST_PROF_YTAB:
    STRUCT
        enType:EN_PROF_TAB_TYPE:=PROF_YTAB ;
        uiNoElement:UINT:=MAX_PROF_Y_IND;
        udMasterInc:UDINT:=20000;
        diY:ARRAY[0…MAX_PROF_Y_IND] OF DINT;
    END_STRUCT
END_TYPE
```

If the table type 'enType' = PROF_XYTAB_NL is selected, the value for 'MAX_PROF_Y_IND' can be redefined at program level. This enables the original limit of up to 360 Y table sections to be increased.
(Siehe 'Number of table interpolation points' auf Seite 87.)

4.5.1.4 Device

4.5.1.4.1 LogicalDevice

4.5.1.4.1.1 ST_DEVICE (ST)

The device description structure 'ST_DEVICE' combines information that is required to access a device (e.g. a drive) via a bus (e.g. EtherCAT or ACC).

Variables of this type are created during controller configuration in CODESYS V3. They essentially serve as "symbolic pointers" (identifier / handle) for subsequent assignment to an actual device.
They are used during the course of programming to link a variable of a physical device to this name. All variables of a device to which the same 'ST_DEVICE' is assigned must also be assigned to the same device.
However, the assignment itself is not made until later during bus configuration in AIPEX PRO, when this symbolic variable name is linked to a real device.

A real advantage of this procedure is the fact that programming is almost entirely independent of the physical characteristics of the device configuration:
A program function can, for example, be assigned to another drive without the program itself being changed.

Structure elements

Name	Type	Description
iPhysInd	INT	Index for the reference to the physically assigned device Range: 0 ... MAX_PHYS_DEF Default: 0
uiCycleTime	UINT	Cycle time (usually corresponding to ID2 'SERCOS cycle time') Unit: 0.001 ms

Name	Type	Description
stDmt	STRUCT	ST_DMT Device mapping table

Structure definition

```
TYPE ST_DEVICE:
    STRUCT
        iPhysInd:INT;
        uiCycleTime:UINT;
        stDmt:ST_DMT;
    END_STRUCT
END_TYPE
```

From the point of view of application programming, no other information about the 'ST_DEVICE' or 'ST_DMT' structures is required.
Therefore, there is no need for a more detailed description here.

4.5.1.4.2 PhysicalDevice

4.5.1.4.2.1 ST_NET_NO (ST)

The 'ST_NET_NO' structure describes the fieldbus address.

Structure elements

Name	Type	Description
usSubmNo	USINT	Submodule number
usBaseNo	USINT	Base number
usResNo	USINT	Reserved
usCrossNo	USINT	Cross communication number

Structure definition

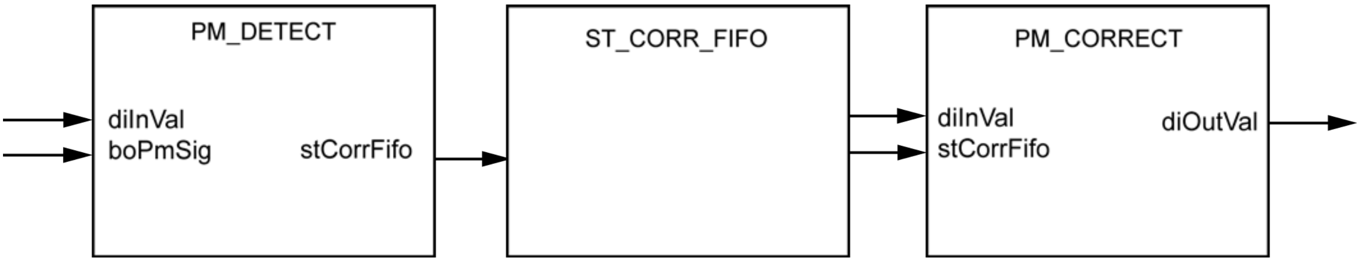
```
TYPE ST_NET_NO:
    STRUCT
        usSubmNo:USINT;
        usBaseNo:USINT;
        usResNo:USINT;
        usCrossNo:USINT;
    END_STRUCT
END_TYPE
```

4.5.1.5 PmControl

4.5.1.5.1 ST_CORR_FIFO (ST)

The 'ST_CORR_FIFO' is used to transfer correction values between the 'PM_DETECT' and 'PM_CORRECT' blocks.

Abbildung 32: ST_CORR_FIFO: Fundamental structure of printing mark control



The header information includes the following variables:

- 'uiInIndex'; to specify the write position (per PM_DETECT).
- 'uiOutIndex'; to specify the read position (with PM_CORRECT).

The array 'diCorrVal'[0 ... MAX_CORR_FIFO_IND] contains the correction values.

Header information	uiInIndex
	uiOutIndex
Correction values	diCorrVal[0]
	diCorrVal[1]
	...
	diCorrVal[MAX_CORR_FIFO_IND]

The corresponding index is incremented after every read or write operation. If the array limit 'MAX_CORR_FIFO_IND' is exceeded, the index is set to 0.

The following applies:

Before reading: uiInIndex = uiOutIndex -> FIFO is empty

Before writing: (uiInIndex + 1) MOD (MAX_CORR_FIFO_IND + 1) = uiOutIndex -> FIFO is full

Structure elements

Name	Type	Description
uiInIndex	UINT	Write index Range: 0 ... MAX_CORR_FIFO_IND
uiOutIndex	UINT	Read index Range: 0 ... MAX_CORR_FIFO_IND
diCorrVal	ARRAY	ARRAY [0..MAX_CORR_FIFO_IND] OF DINT Current correction value enter most recently in the FIFO structure 'stCorrFifo' Array for saving correction values

Structure definition

```
MAX_CORR_FIFO_IND: UNIT:=19;                (* maximum valid index for diCorrVal *)


TYPE ST_CORR_FIFO:
      STRUCT
            uiInIndex:UNIT;
            uiOutIndex:UNIT;
            diCorrVal:ARRAY[0…MAX_CORR_FIFO_IND] OF DINT;
      END_STRUCT
END_TYPE
```

4.5.1.6 System

4.5.1.6.1 ST_PLC_VAR_POINTERS (ST)

Structure elements

Name	Type	Description
pbyInAsync	POINTER	POINTER TO BYTE Pointer to the asynchronous input variable range
pbyOutAsync	POINTER	POINTER TO BYTE Pointer to the asynchronous output variable range
pbyInSync	POINTER	POINTER TO BYTE Pointer to the synchronous input variable range
pbyOutSync	POINTER	POINTER TO BYTE Pointer to the synchronous output variable range

Structure definition

```
TYPE ST_PLC_VAR_POINTERS:
	STRUCT
		pbyInAsync: POINTER TO BYTE;
		pbyOutAsync: POINTER TO BYTE;
		pbyInSync: POINTER TO BYTE;
		pbyOutSync: POINTER TO BYTE;
	END_STRUCT
END_TYPE
```

5 AmkSupport - Support functions specific to AMK

AmkSupport is an internal AMK library which contains support functions specific to AMK to support special hardware and technologies. It is divided into:

Basic	Basic functions
Convert	Conversion functions
FifoHandling	FIFO functions
SequencialPos	Sequential positioning functions

5.1 Basic

MIN_MAX	Extreme value determination with reset

5.1.1 General

5.1.1.1 MIN_MAX (FB)

The 'MIN_MAX' function block provides the extreme values (minimum, maximum) of the input variables in the two output variables.

User interface

Input variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boReset</td><td>BOOL</td><td>Reset signal<table><tr><td>FALSE</td><td>Extreme value generation in progress</td></tr><tr><td>TRUE</td><td>'diMinVal' := 'diMaxVal' := 'diActVal'</td></tr></table></td></tr>
<tr><td>diActVal</td><td>DINT</td><td>Input value, actual value
Minimum and maximum since the last reset are provided as output values</td></tr>
</table>

Output variables

Name	Type	Description
diMinVal	DINT	Minimum output value since last reset
diMaxVal	DINT	Maximum output value since last reset

5.2 Convert (conversion blocks)

Counter

COUNT_TO_DI	Generate DINT pulse encoder information

Polynomial

CAMXYVA_TO_PROF	Convert 3S structures into AMK structure
XYVA_TO_PROF	Conversion of table interpolation points

Visu

PROF_TO_VISU	Calculate visualization tables for graphical representation of X / XY table characteristics

5.2.1 Counter

5.2.1.1 COUNT_TO_DI (FB)

The 'COUNT_TO_DI' function block converts the counter values 'boRefPulse', 'diCount', 'diOffset' into an AMK counter value / pulse encoder information.

'COUNT_TO_DI' is the inverse of the block 'DI_TO_COUNT'. (See document Software description AmkBase Bibliothek, Part no. 204986)

User interface

Input variables

Name	Type	Description
boRefPulse	BOOL	Homing pulse Displays a detected zero pulse How long is the signal pending for?
diCount	DINT	32-bit Counter value generated from the value changes in the current 16-bit counter status read during each cycle
diOffset	DINT	Offset of the counter value to the homing pulse Unit: Incr

Output variables

Name	Type	Description
diOutVal	DINT	Pulse encoder information AMK 32-bit data format Low word = $diOutVal_{LW}$: generated on the homing pulse from the counter value corrected by the offset High word = $diOutVal_{HW}$: current 16-bit counter reading

5.2.2 Polynominal

5.2.2.1 CAMXYVA_TO_PROF (FB)

The 'CAMXYVA_TO_PROF' function block converts the CODESYS structures 'MC_CAM_REF' and 'ARRAY[0...N] OF SMC_CAMXYVA' into the AMK structure 'ST_PROF_XYVATAB' and generates a pointer to this structure for the 'CAM_PROF' block (See document Software description AmkBase Bibliothek, Part no. 204986).

The definition of these structures is based on 3S libraries which are only integrated with the full Softmotion license.

AmkCamEditor is an AMK library which contains copies of the structures required for the table function of the cam disk editor. The AmkCamEditor library is, therefore, an absolute necessity when working with XYVA tables or the cam disk editor with polynomial tables!

User interface

Input variables

Name	Type	Description
pstMcCamRef	POINTER	POINTER TO MC_CAM_REF Pointer to the 3S header structure generated by the CAM editor 'MC_CAM_REF'
pstSmcCamXYVA	POINTER	POINTER TO SMC_CAMXYVA Pointer to the 3S interpolation point array generated by the CAM editor ARRAY [0..N] OF SMC_CAMXYVA

Output variables

Name	Type	Description
pstProfXYVATab	POINTER	POINTER TO ST_PROF_XYVATAB Pointer to the AMK-specific XYVA table structure ST_PROF_XYVATAB expected by the 'CAM_PROF' AMK function block

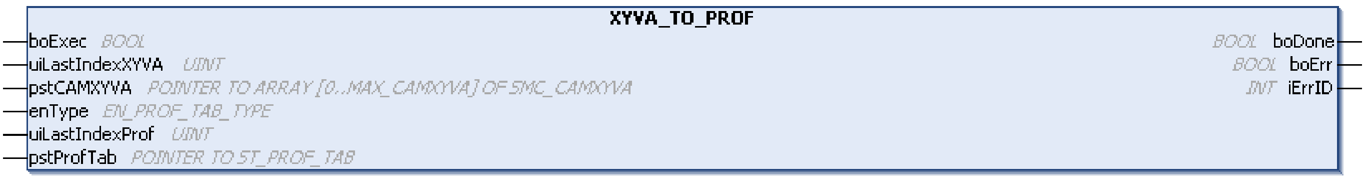
5.2.2.2 XYVA_TO_PROF (FB)

The 'XYVA_TO_PROF' function block converts a 3S polynomial XYVA interpolation point table 'ARRAY[0..MAX_CAMXYVA] OF SMC_CAMXYVA' into an AMK Y or XY table (See document Software description AmkBase Bibliothek, Part no. 204986).

However, unlike the 'CAMXYVA_TO_PROF' block, the description of the curve is converted in full. A table based on the Y or XY format is calculated from a table in XYVA format based on the 'SMC_CAMXYVA' format.

The 'XYVA_TO_PROF' block thus supports offline conversion of the XYVA format and the use of the result with a 'CAM_PROF' which does not support the XYVA format, or supports the display of a table in XYVA format in a graph through the 'PROF_TO_VISU' block.

User interface



Input variables

Name	Type	Description
boExec	BOOL	Function execution: With a positive edge, the execution of the block starts. As long as 'boExec' = TRUE, the block is processed by the PLC. In the state 'boExec' = FALSE execution of the block is ended.
uiLastIndexXYVA	UINT	Last array index of the XYVA input table 'SMC_CAMXYVA' Range: 1.. MAX_CAMXYVA where MAX_CAMXYVA:UINT := 31
pstCAMXYVA	POINTER	POINTER TO ARRAY [0..MAX_CAMXYVA] OF SMC_CAMXYVA Pointer to the XYVA input table specific to 3S

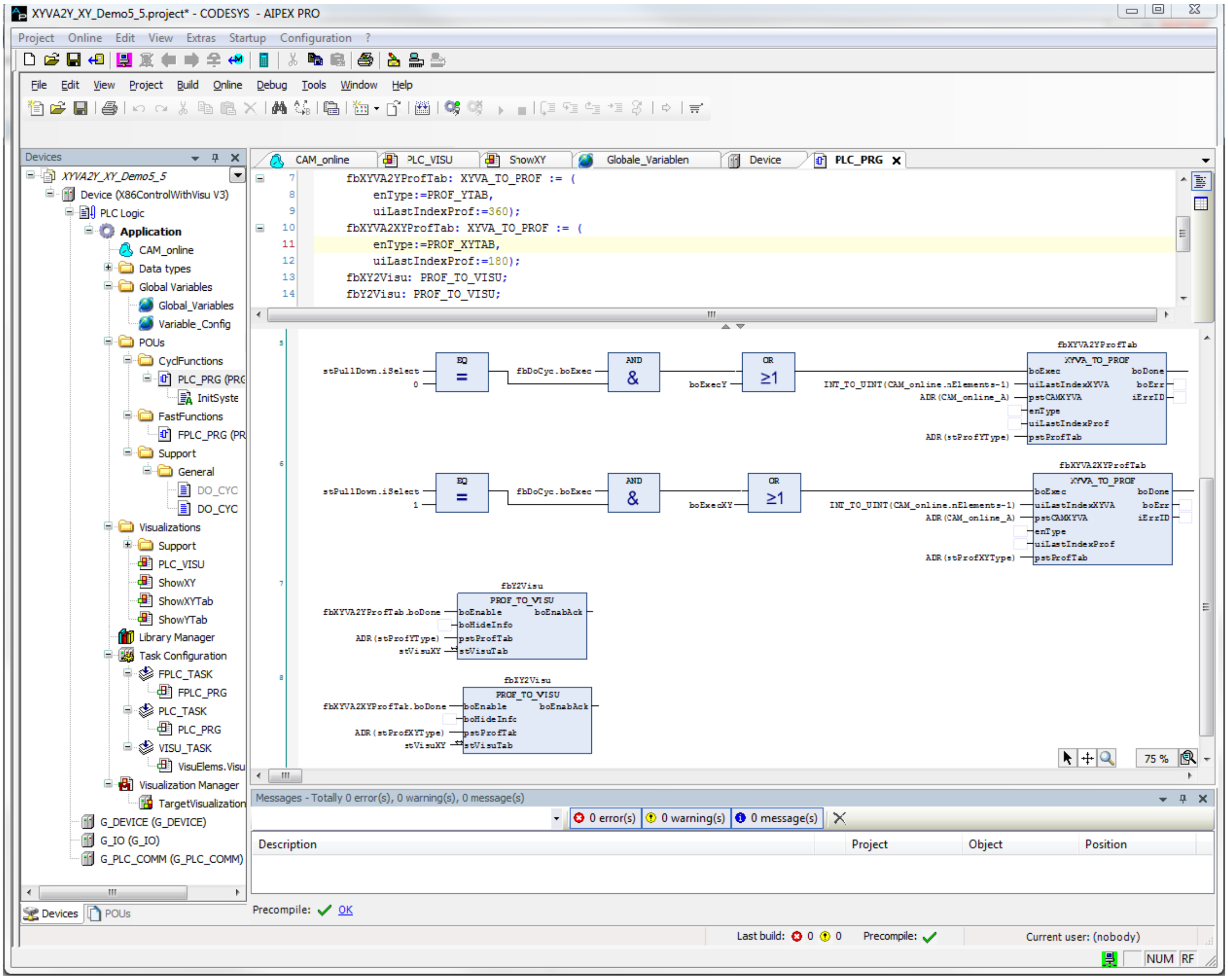
<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>enType</td><td>ENUM</td><td>EN_PROF_TAB_TYPE
Table type
Selection of the phasing out table type
See document Software description AmkBase Bibliothek, Part no. 204986
<table>
<tr><td>Default</td><td>PROF_YTAB</td></tr>
</table>
<table>
<tr><th>Range</th><th>Meaning</th></tr>
<tr><td>PROF_YTAB</td><td>Y table</td></tr>
<tr><td>PROF_XYTAB</td><td>XY table</td></tr>
</table></td></tr>
<tr><td>uiLastIndexProf</td><td>UINT</td><td>Last array index of the Y / XY output table 'CAM_PROF'
<table>
<tr><td rowspan="2">Range</td><td>1 .. MAX_PROF_Y_IND
if 'enType' := PROF_YTAB;</td></tr>
<tr><td>1 .. MAX_PROF_XY_IND
if 'enType' := PROF_XYTAB</td></tr>
</table></td></tr>
<tr><td>pstProfTab</td><td>POINTER</td><td>POINTER TO ST_PROF_TAB
Pointer to the Y / XY output table</td></tr>
</table>

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boDone</td><td>BOOL</td><td>Response that the function block has been completely executed.</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table></td></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
<table>
<tr><th>Range</th><th>Meaning</th></tr>
<tr><td>Type</td><td>EN_XYVA_CONV_ERR</td></tr>
<tr><td>1</td><td>XYVA_CONV_ILL_TYPE
Invalid table type</td></tr>
<tr><td>2</td><td>XYVA_CONV_ILL_IND_PROF
Invalid index of output table</td></tr>
<tr><td>3</td><td>XYVA_CONV_ILL_INDXYVA
Invalid index of input table</td></tr>
<tr><td>4</td><td>XYVA_CONV_ILL_START
Starting point of XYVA table is not {0,0}</td></tr>
<tr><td>5</td><td>XYVA_CONV_ILL_END_IND
Invalid index of input or output table at end of conversion, possibly due to the value for 'uiLastIndexProf' being too low</td></tr>
</table></td></tr>
</table>

Description

Abbildung 33: XYVA_TO_PROF: Conversion for online visualization



5.2.3 Visu

5.2.3.1 PROF_TO_VISU (FB)

The 'PROF_TO_VISU' function block converts a Y or XY table into a structure that is suitable for displaying the curve characteristic in a graph ('ST_VISU_TAB').

User interface

Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.
boHideInfo	BOOL	Suppress display of X axis information in ViXY
pstProfTab	POINTER	POINTER TO ST_PROF_TAB Pointer to the Y / XY output table

Output variables

Name	Type	Description
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled

Input and output variables

Name	Type	Description
stVisuTab	STRUCT	ST_VISU_TAB Structure of the visualization table Basis for the ViXY and ViCursor visualization blocks

Description

ViXY visualization supports online display of the curve characteristic in a graph (an XY diagram).
The curve characteristic is approximated with MAX_VISU_XY := 80 linear partial segments.
ViCursor visualization allows the cursor of the XY diagram to be manipulated.

ViXY and ViCursor are linked by referencing visualizations with the 'ST_VISU_TAB' structure.

Abbildung 34: PROF_TO_VISU: Online visualization

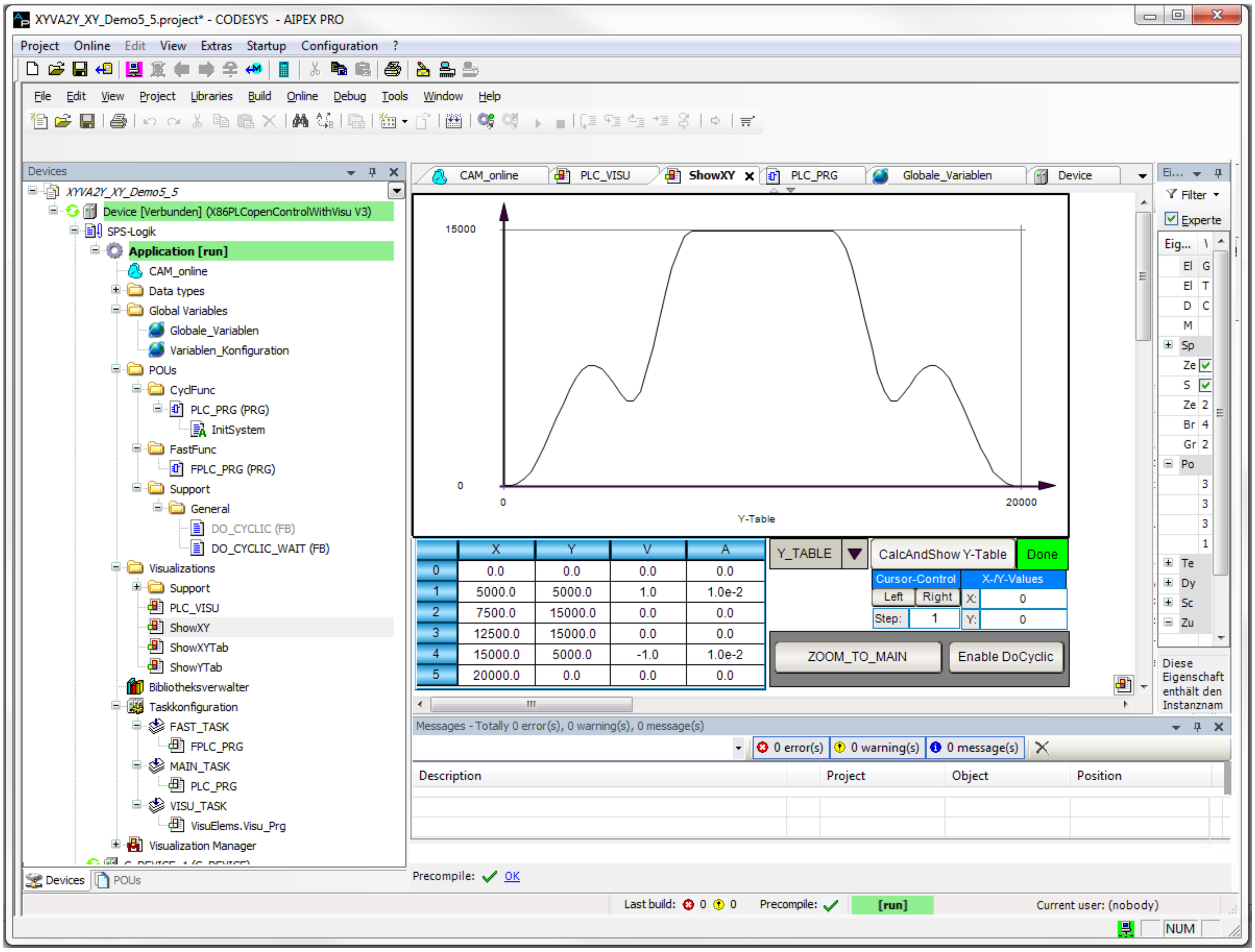
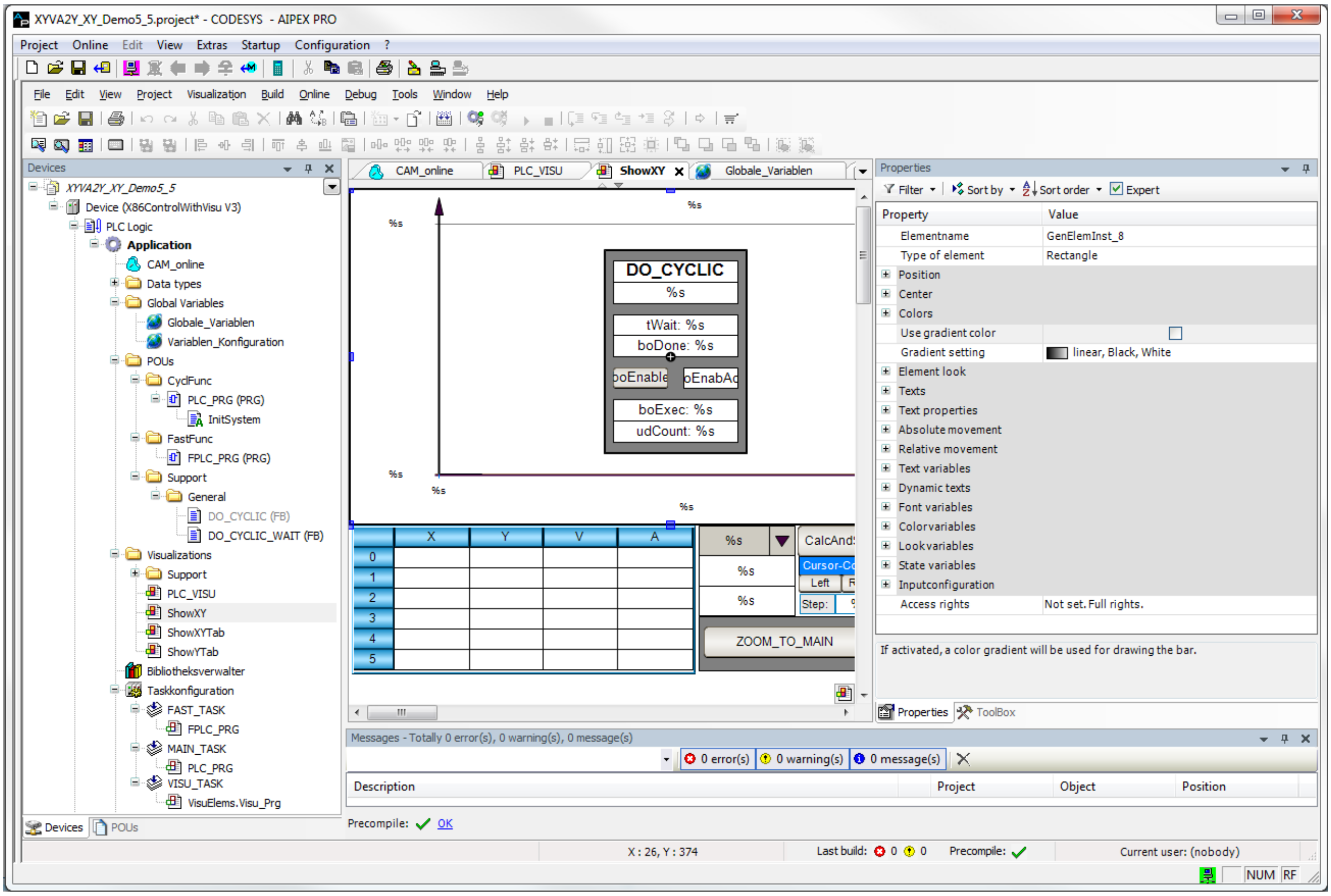
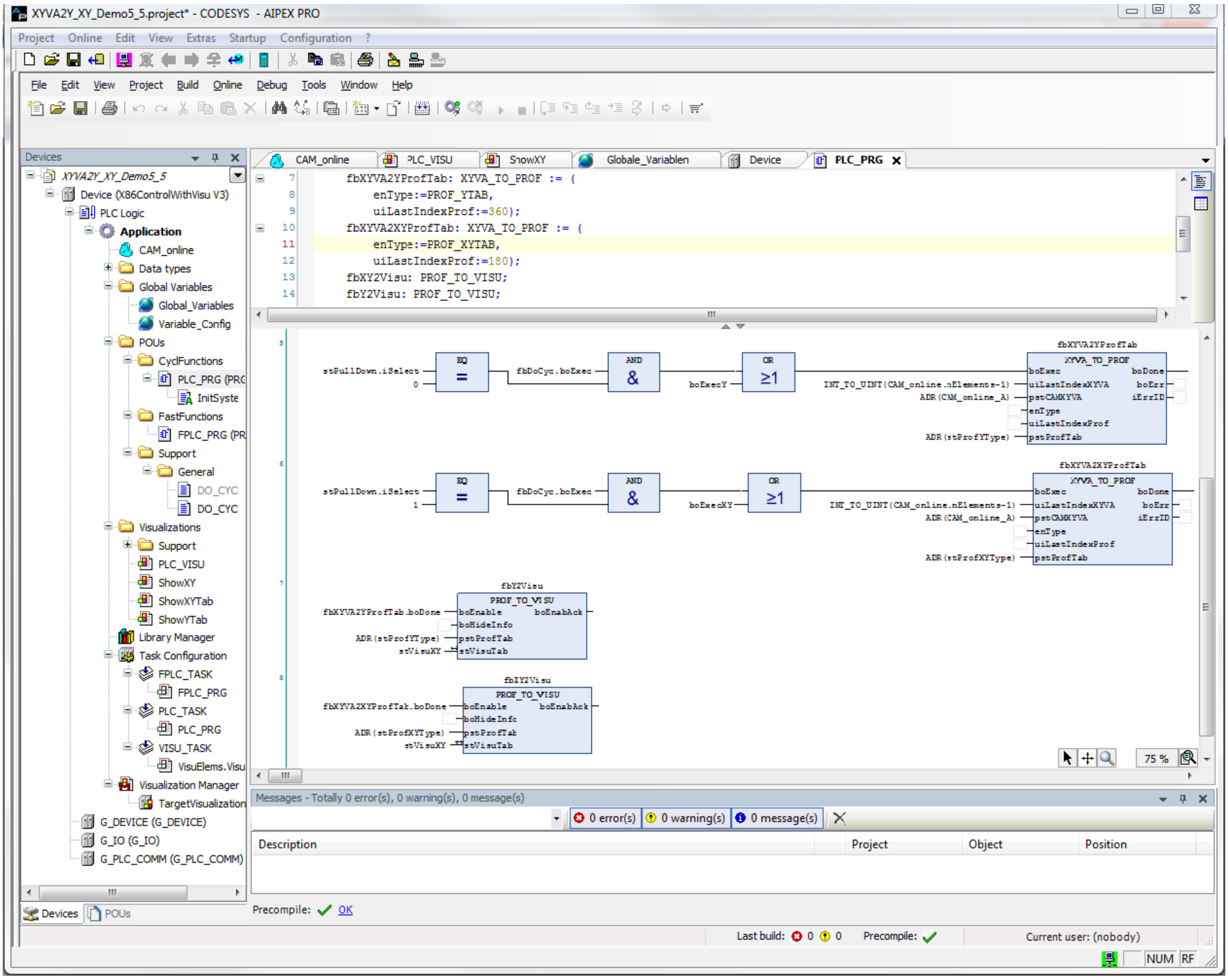


Abbildung 35: PROF_TO_VISU: Referenced visualization



To display an XYVA table in a graph, the table can be converted to a Y or XY table first with the 'XYVA_TO_PROF' block.

Abbildung 36: PROF_TO_VISU: Conversion for online visualization



5.3 FifoHandling (FIFO functions)

FIFO_HANDLER	FIFO block

5.3.1 FIFO_HANDLER (FB)

The 'FIFO_HANDLER' function block serves as a FIFO memory (FIFO stands for first in first out).

The block is characterized as follows:

- The information managed in the FIFO (a FIFO element) can be structured at will
- The FIFO is organized so that it can also be used for communication between two processes (thread save)
- The size of the FIFO can be specified as variable

The 'FifoInit()' action must be executed before the rest of the FIFO function can be used.

The values for 'uiEleSize', 'uiFifoSize', 'pbyFifo', and 'stFifoHeader' predefined in the context of 'FifoInit()' must not be changed again subsequently

'FifoReset()' is only possible if there has not yet been a FIFO overrun

User interface

Input variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boExec</td><td>BOOL</td><td>Function execution: With a positive edge, the execution of the block starts.
As long as 'boExec' = TRUE, the block is processed by the PLC.
In the state 'boExec' = FALSE execution of the block is ended.</td></tr>
<tr><td>enMode</td><td>ENUM</td><td>EN_FIFO_HANDLER_MODE
Selection mode of the function
The function is executed in the context of a [Fifoxxxxx] action
<table>
<tr><td>Default</td><td>FIFO_INIT</td></tr>
</table>
<table>
<tr><th>Range</th><th>Meaning</th></tr>
<tr><td>FIFO_INIT
[FifoInit]</td><td>Initialize FIFO
The relevant input variables are added to the 'stFifoHeader' variable and the memory made available for the FIFO is cleared (FifoClear).</td></tr>
<tr><td>FIFO_CLEAR
[FifoClear]</td><td>Clear FIFO
'uiEleNmb' and 'udEleInd' are cleared (:= 0).
The internal state 'stFifoHeader.enFifoState' = FIFO_ STATE_READY is also set.</td></tr>
<tr><td>FIFO_RESET
[FifoReset]</td><td>Reset output index
The action sets 'udEleInd' := 0 and 'uiEleNmb' to the number of elements already written to the FIFO.
This enables the elements already written to be read again.

The action can only be used if there has not yet been a FIFO overrun: 'stFifoHeader.enFifoState' = FIFO_ STATE_READY</td></tr>
<tr><td>FIFO_READ
[FifoRead]</td><td>Read out FIFO element
Read the first FIFO element written and not yet read out. This current element is then written to the address referenced in 'pbyEle' with 'uiEleSize' bytes.
'uiEleNmb' is decremented, 'udEleInd' is incremented.</td></tr>
<tr><td>FIFO_WRITE
[FifoWrite]</td><td>Write FIFO element
The element referenced by 'pbyEle' is written. 'uiEleSize' bytes are written.
'uiEleNmb' is incremented.</td></tr>
</table></td></tr>
<tr><td>uiEleSize</td><td>UINT</td><td>Size (in bytes) of the element to be written / read</td></tr>
<tr><td>pbyEle</td><td>POINTER</td><td>POINTER TO BYTE
enMode = FIFO_READ:
Pointer to the address starting from which the element read out is saved
enMode = FIFO_WRITE:
Pointer to the address starting from which data is transferred to the FIFO</td></tr>
<tr><td>uiFifoSize</td><td>UINT</td><td>Size (in bytes) of the memory made available for the FIFO organization</td></tr>
</table>

Name	Type	Description
pbyFifo	POINTER	POINTER TO BYTE Pointer to the address starting from which memory capacity is made available for the FIFO organization

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boDone</td><td>BOOL</td><td>Response that the function block has been completely executed.</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table></td></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
<table>
<tr><th>Range</th><th>Meaning</th></tr>
<tr><td>1</td><td>Illegal mode</td></tr>
<tr><td>2</td><td>Invalid element size</td></tr>
<tr><td>3</td><td>Element pointer not initialized</td></tr>
<tr><td>4</td><td>Invalid FIFO range size</td></tr>
<tr><td>5</td><td>FIFO range pointer not initialized</td></tr>
<tr><td>6</td><td>Illegal FIFO header information</td></tr>
<tr><td>7</td><td>FIFO is not initialized</td></tr>
<tr><td>8</td><td>Reset function illegal</td></tr>
<tr><td>9</td><td>FIFO full</td></tr>
<tr><td>10</td><td>FIFO empty</td></tr>
</table></td></tr>
<tr><td>uiEleNmb</td><td>UINT</td><td>Number of elements written to the FIFO and not yet read back.</td></tr>
<tr><td>udEleInd</td><td>UDINT</td><td>FIFO position from which data is currently being read with 'FifoRead()'.</td></tr>
</table>

Input and output variables

Name	Type	Description
stFifoHeader	STRUCT	ST_FIFO_HEADER FIFO header information Organization of the FIFO

5.4 Sequencial positioning

POS_SEQUENCER	Sequencial positioning

5.4.1 POS_SEQUENCER (FB)

The 'POS_SEQUENCER' function block organizes a sequence of position overrides.

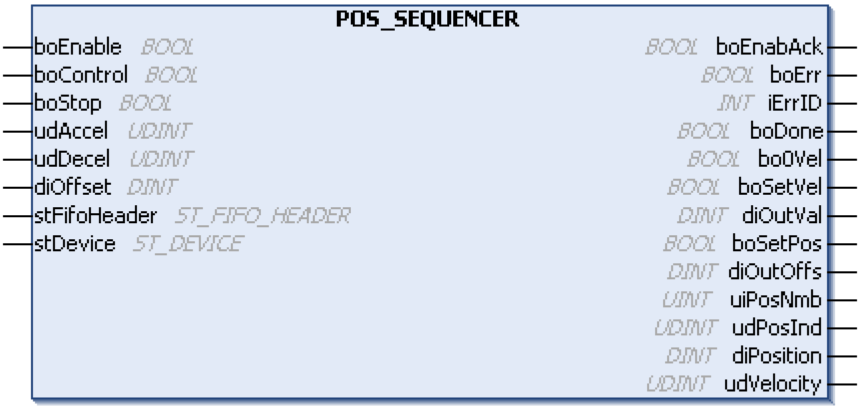
It uses the 'POS_1' block in mode enMode = POS_REL_RETRIG_EXT

(See document Software description AmkBase Bibliothek , Part no. 204986).

A set of position and velocity values describes the positioning sequence. This data is transferred by a FIFO prior to the 'POS_SEQUENCER' block being activated.

An element based on the 'ST_POS_ELE' structure must be used to specify the position and velocity value pairs.

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.
boControl	BOOL	Start / Stop Positive edge: Starts the sequence Negative edge: Reset; the sequence is restarted.
boStop	BOOL	With a positive edge, the execution of the block is aborted or completed.
udAccel	UDINT	Acceleration with which the target velocity is run Range: 0 ... 400000000 Unit: incr/s^2
udDecel	UDINT	Deceleration with which a lower target velocity is achieved Range: 0 ... 4000000000 Unit: incr/s^2
diOffset	DINT	Offset of the counter value to the homing pulse Unit: Incr

Output variables

Name	Type	Description
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled
boErr	BOOL	The function block is in an error state FALSE: No error (permitted commanding or warning) TRUE: Error

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Warning
<table>
<tr><th>Range</th><th>Meaning</th></tr>
<tr><td>11</td><td>Setpoint velocity = 0</td></tr>
<tr><td>12</td><td>Illegal setpoint velocity
limited to minimum or maximum value</td></tr>
<tr><td>13</td><td>Acceleration = 0</td></tr>
<tr><td>14</td><td>Illegal acceleration
limited to minimum or maximum value</td></tr>
<tr><td>15</td><td>Deceleration = 0</td></tr>
<tr><td>16</td><td>Illegal deceleration
limited to minimum or maximum value</td></tr>
<tr><td>17</td><td>Deceleration value corrected</td></tr>
</table>
Error
<table>
<tr><th>Range</th><th>Meaning</th></tr>
<tr><td>1</td><td>Illegal mode</td></tr>
<tr><td>2</td><td>Invalid element size</td></tr>
<tr><td>3</td><td>Element pointer not initialized</td></tr>
<tr><td>4</td><td>Invalid FIFO range size</td></tr>
<tr><td>5</td><td>FIFO range pointer not initialized</td></tr>
<tr><td>6</td><td>Illegal FIFO header information</td></tr>
<tr><td>7</td><td>FIFO is not initialized</td></tr>
<tr><td>8</td><td>Illegal reset function</td></tr>
<tr><td>9</td><td>FIFO full</td></tr>
<tr><td>10</td><td>FIFO empty</td></tr>
<tr><td>11</td><td>Illegal mode</td></tr>
<tr><td>12</td><td>Illegal offset in POS_INTERPOSED_NB mode</td></tr>
</table></td></tr>
<tr><td>boDone</td><td>BOOL</td><td>Response that the function block has been completely executed.</td></tr>
<tr><td>bo0Vel</td><td>BOOL</td><td>Standstill reached, velocity = 0
When 'bo0Vel' is active, no setpoint is output.</td></tr>
<tr><td>boSetVel</td><td>BOOL</td><td>Setpoint velocity reached, velocity = 'diVelocity'
When 'boSetVel' is active, the target velocity has been reached.</td></tr>
<tr><td>diOutVal</td><td>DINT</td><td>Output value</td></tr>
<tr><td>boSetPos</td><td>BOOL</td><td>In retrigger mode the signal for a cycle becomes active when the retrigger position has been reached.</td></tr>
<tr><td>diOutOffs</td><td>DINT</td><td>Offset value before the retrigger is started; only in retrigger mode</td></tr>
<tr><td>uiPosNmb</td><td>UINT</td><td>Number of pairs of values written to the FIFO and not yet read back</td></tr>
<tr><td>udPosInd</td><td>UDINT</td><td>FIFO index from which data is currently being read with 'FifoRead()'.</td></tr>
<tr><td>diPosition</td><td>DINT</td><td>Setpoint position
Definition of the final position</td></tr>
<tr><td>udVelocity</td><td>UDINT</td><td>Setpoint velocity
Definition of the final velocity</td></tr>
</table>

Input and output variables

Name	Type	Description
stFifoHeader	STRUCT	ST_FIFO_HEADER FIFO header information Organization of the FIFO
stDevice	STRUCT	The device description structure assigns the block a device.

Description

Example for transferring a position / velocity pair with the 'FifoWrite()' action:

Declaration

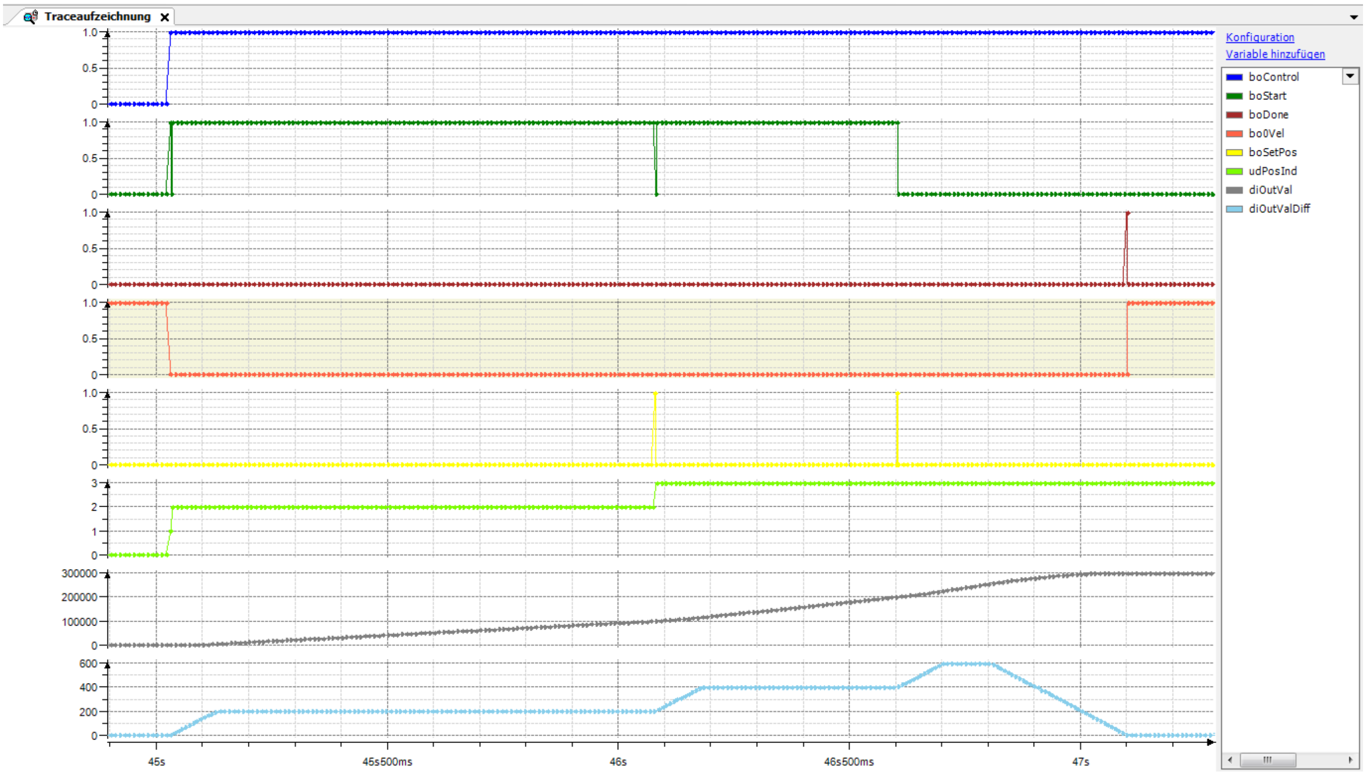
```
fbFifoHandler: AmkSupport.FIFO_HANDLER;
stPosEle: AmkSupport.ST_POS_ELE;
```

Program

```
stPosEle.diPosition:=100000;
stPosEle.udVelocity:=100000;
fbFifoHandler.FifoWrite(
                uiEleSize:= SIZEOF(stPosEle),
                pbyEle:= ADR(stPosEle),
                uiFifoSize:= SIZEOF(FPLC_PRG.arr_stPosFiFo),
                pbyFifo:= ADR(FPLC_PRG.arr_stPosFiFo),
                stFifoHeader:= FPLC_PRG.stPosFifoHeader);
```

For a positioning sequence with three position / velocity pairs, for example, this results in the following signal characteristic:

Abbildung 37: POS_SEQUENCER: signal characteristic, positioning sequence



5.5 Data types

ST_FIFO_HEADER	The FIFO header information is used by the 'FIFO_HANDLER' function block to organize a FIFO
ST_POS_ELE	Specification of a positioning element comprising position and velocity value

5.5.1 FifoHandling

5.5.1.1 ST_FIFO_HEADER (ST)

The FIFO header information is used by the 'FIFO_HANDLER' function block to organize a FIFO.

The FIFO header information must be created in the application. However, its content is used exclusively in the context of the internal organization of the FIFO; it does not have to be evaluated from the point of view of the application.

Structure elements

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>uiInIndex</td><td>UINT</td><td>Write index: is incremented with FifoWrite().</td></tr>
<tr><td>uiOutIndex</td><td>UINT</td><td>Read index: is incremented with FifoRead().</td></tr>
<tr><td>uiMaxIndex</td><td>UINT</td><td>Maximum permissible index.</td></tr>
<tr><td>enFifoState</td><td>ENUM</td><td>EN_FIFO_STATE
FIFO state
<table>
<tr><td>Default</td><td>FIFO_STATE_INIT</td></tr>
</table>
<table>
<tr><th>Range</th><th>Meaning</th></tr>
<tr><td>FIFO_STATE_ INIT</td><td>FIFO not yet initialized</td></tr>
<tr><td>FIFO_STATE_ READY</td><td>FIFO ready for use</td></tr>
<tr><td>FIFO_STATE_ READY_ TURNOVER</td><td>FIFO overrun has occurred</td></tr>
</table>
</td></tr>
<tr><td>uiEleSize</td><td>UINT</td><td>Size (in bytes) of the element to be written / read
For FifoInit(), the value is taken from the corresponding input variable. After this, the input variable must not change again.</td></tr>
<tr><td>pbyFifo</td><td>POINTER</td><td>POINTER TO BYTE
Pointer to the address starting from which memory capacity is made available for the FIFO organization</td></tr>
</table>

Structure definition

```
TYPE ST_FIFO_HEADER:
    STRUCT
        uiInIndex: UINT;
        uiOutIndex: UINT;
        uiMaxIndex: UINT;
        enFifoState: EN_FIFO_STATE;
        uiEleSize: UINT;
        pbyFifo: POINTER TO BYTE;
    END_STRUCT
END_TYPE
```

5.5.2 Sequencial Positioning

5.5.2.1 ST_POS_ELE (ST)

Specification of a positioning element comprising position and velocity value based on the 'ST_POS_ELE' structure.

Structure elements

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>diPosition</td><td>DINT</td><td>Setpoint position
Definition of the final position
Overall increment increase of the output value<table><tr><td>Unit</td><td>incr</td></tr><tr><td>Default</td><td>600000</td></tr></table></td></tr>
<tr><td>udVelocity</td><td>UDINT</td><td>Setpoint velocity
Definition of the final velocity
Increment difference of the output value by time<table><tr><td>Range</td><td>0 ... 300000000</td></tr><tr><td>Unit</td><td>incr/s</td></tr><tr><td>Default</td><td>200000</td></tr></table></td></tr>
</table>

Structure definition

```
TYPE ST_POS_ELE:
    STRUCT
        diPosition: DINT;
        udVelocity: UDINT;
    END_STRUCT
END_TYPE
```

6 AmkSystem - System functions specific to AMK

AmkSystem is an internal AMK library for system-wide AMK communication. It is divided into:

ID_Access	ID access functions
Support	Support functions

The 'ID_Access' blocks facilitate access to the drive parameters. They are based on the base blocks 'ID_READ_1' and 'ID_WRITE_1', which facilitate system-wide communication via independent standard communication channels (e.g. ACC: SDO transfer, SERCOS: service channel, etc.).

The 'ST_DEVICE' device description structure, which is made available in the context of automatic bus configuration, serves the purpose of addressing the AMK subsystems.

Read or write access to other AMK subsystems can only be initiated by assemblies with bus master function!

The Support blocks provide support functions. They consist of special support blocks which have been provided either for internal or internal and external support tasks.

Only the 'FstNetNoOfDevice' function is of interest for the user.

6.1 ID_Access (ID access functions)

AllElementsOfOneID

Read all elements of SERCOS-based parameters

READ_ID_ALL	Read all parameter elements

ElementaryAccess

Element parameter access

READ_ID_DINT	Read parameter value
READ_ID_DINT_TMP	Read parameter value
READ_ID_LIST	Read parameter values from a list
READ_LIST_512	Read parameter values from a 512-byte list
READ_SDO	SDO read access
WRITE_ID_DINT	Write parameter value
WRITE_ID_DINT_TMP	Write parameter value
WRITE_ID_LIST	Write parameter values in a list
WRITE_LIST_512	Write parameter values in a 512-byte list
WRITE_SDO	SDO write access

HigherAccess

Simplified parameter access

READ_ID_DINT_ONCE	Read parameter value
WRITE_ID_DINT_ONCE	Write parameter value

MoreIds

Multiple parameter access

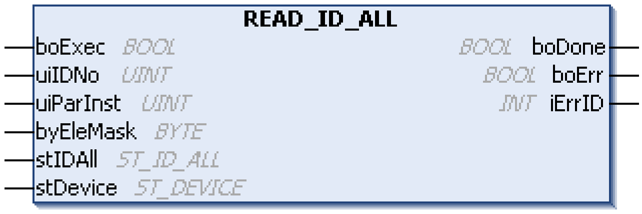
READ_ALL_IDS	Read all elements of all parameters
READ_N_IDS_DINT	Read N parameter values
WRITE_N_IDS_DINT	Write N parameter values

6.1.1 AllElementsOfOneID

6.1.1.1 READ_ID_ALL (FB)

The 'READ_ID_ALL' function block reads elements of a parameter stored in the AMK subsystem.

User interface



Input variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boExec</td><td>BOOL</td><td>Function execution: With a positive edge, the execution of the block starts.
As long as 'boExec' = TRUE, the block is processed by the PLC.
In the state 'boExec' = FALSE execution of the block is ended.</td></tr>
<tr><td>uiIDNo</td><td>UINT</td><td>ID number to be read out</td></tr>
<tr><td>uiParInst</td><td>UINT</td><td>Instance or Parameter set number or instance number</td></tr>
<tr><td>byEleMask</td><td>BYTE</td><td>Element mask
Selection of the parameter element to be read.
<table>
<tr><th>Range</th><th>Meaning</th></tr>
<tr><td>0</td><td>Not used</td></tr>
<tr><td>1</td><td>Not used</td></tr>
<tr><td>2</td><td>Name</td></tr>
<tr><td>3</td><td>Attribute</td></tr>
<tr><td>4</td><td>Unit</td></tr>
<tr><td>5</td><td>Minimum</td></tr>
<tr><td>6</td><td>Maximum</td></tr>
<tr><td>7</td><td>Value</td></tr>
</table></td></tr>
</table>

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boDone</td><td>BOOL</td><td>Response that the function block has been completely executed.</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table></td></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Range: Siehe 'Error bit information' auf Seite 532.</td></tr>
</table>

Input and output variables

Name	Type	Description
stIDAll	STRUCT	ST_ID_ALL Parameter information Accommodates the element information
stDevice	STRUCT	The device description structure assigns the block a device.

6.1.1.2 READ_ID_LIST_ALL (FB)

The 'READ_ID_LIST_ALL' function block reads elements of a parameter stored in the AMK subsystem. The function block also reads list parameters.

User interface

Input variables

Name	Type	Description
boExec	BOOL	Function execution: With a positive edge, the execution of the block starts. As long as 'boExec' = TRUE, the block is processed by the PLC. In the state 'boExec' = FALSE execution of the block is ended.
uiIDNo	UINT	ID number to be read out
uiParInst	UINT	Instance or Parameter set number or instance number
stNetNo	STRUCT	ST_NET_NO Network address The network address can be identified with the 'FstNetNoOfDevice' function from the 'ST_DEVICE' structure, for example.
uiSize	UINT	Maximum data length available to accommodate the information to be read. uiSize ≤ SIZEOF(variable) referenced by 'pbyData'!
pbyData	POINTER	POINTER TO READ DATA Pointer referencing the structure / variable which is receiving the information read.

Output variables

Name	Type	Description
boDone	BOOL	Response that the function block has been completely executed.
boErr	BOOL	The function block is in an error state <table><tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr><tr><td>TRUE</td><td>Error</td></tr></table>

Name	Type	Description
iErrID	INT	Error identity number: Diagnostic number is output iErrID = 0 — No error iErrID ≠ 0 — boErr = TRUE — Error iErrID ≠ 0 — boErr = FALSE — Warning Range: Siehe 'Error bit information' auf Seite 532.
boList	BOOL	Identifier for a list parameter FALSE — The data to be read is in 'stIDAll.diData' TRUE — List parameter: The list to be read is transferred to the list structure referenced by 'pbyData'

Input and output variables

Name	Type	Description
stIDAll	STRUCT	ST_ID_ALL Parameter information Accommodates the element information

6.1.2 ElementaryAccess

6.1.2.1 READ_ID_DINT (FB)

The 'READ_ID_DINT' function block reads the value of a parameter stored in the AMK subsystem.

User interface

Input variables

Name	Type	Description
boExec	BOOL	Function execution: With a positive edge, the execution of the block starts. As long as 'boExec' = TRUE, the block is processed by the PLC. In the state 'boExec' = FALSE execution of the block is ended.
uiIDNo	UINT	ID number to be read out
uiParInst	UINT	Instance or Parameter set number or instance number

Output variables

Name	Type	Description
boDone	BOOL	Response that the function block has been completely executed.
boErr	BOOL	The function block is in an error state FALSE — No error (permitted commanding or warning) TRUE — Error

Name	Type	Description
iErrID	INT	Error identity number: Diagnostic number is output iErrID = 0 — No error iErrID ≠ 0, boErr = TRUE — Error iErrID ≠ 0, boErr = FALSE — Warning Range: Siehe 'Error bit information' auf Seite 532.
diIDVal	DINT	Parameter value read from database

Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

6.1.2.2 READ_ID_DINT_TMP (FB)

The 'READ_ID_DINT_TMP' function block reads the value of a parameter organized temporarily in the AMK subsystem.

User interface

Input variables

Name	Type	Description
boExec	BOOL	Function execution: With a positive edge, the execution of the block starts. As long as 'boExec' = TRUE, the block is processed by the PLC. In the state 'boExec' = FALSE execution of the block is ended.
uiIDNo	UINT	ID number to be read out
uiParInst	UINT	Instance or Parameter set number or instance number

Output variables

Name	Type	Description
boDone	BOOL	Response that the function block has been completely executed.
boErr	BOOL	The function block is in an error state FALSE — No error (permitted commanding or warning) TRUE — Error
iErrID	INT	Error identity number: Diagnostic number is output iErrID = 0 — No error iErrID ≠ 0, boErr = TRUE — Error iErrID ≠ 0, boErr = FALSE — Warning Range: Siehe 'Error bit information' auf Seite 532.
diIDVal	DINT	Parameter value read from database

Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

6.1.2.3 READ_ID_LIST (FB)

The 'READ_ID_LIST' function block reads in values of a list parameter from the database of an AMK subsystem.

User interface

Input variables

Name	Type	Description
boExec	BOOL	Function execution: With a positive edge, the execution of the block starts. As long as 'boExec' = TRUE, the block is processed by the PLC. In the state 'boExec' = FALSE execution of the block is ended.
uiIDNo	UINT	ID number to be read out
uiParInst	UINT	Instance or Parameter set number or instance number
uiSize	UINT	Maximum data length available to accommodate the information to be read. uiSize ≤ SIZEOF(variable) referenced by 'pbyData'!
pbyData	POINTER	POINTER TO READ DATA Pointer referencing the structure / variable which is receiving the information read.

Output variables

Name	Type	Description
boDone	BOOL	Response that the function block has been completely executed.
boErr	BOOL	The function block is in an error state FALSE — No error (permitted commanding or warning) TRUE — Error
iErrID	INT	Error identity number: Diagnostic number is output iErrID = 0 — No error iErrID ≠ 0 — boErr = TRUE — Error iErrID ≠ 0 — boErr = FALSE — Warning Range: Siehe 'Error bit information' auf Seite 532.

Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

6.1.2.4 READ_LIST_512 (FB)

The 'READ_LIST_512' function block reads in values of a list parameter that is up to 512 bytes in size from the database of an AMK subsystem.

User interface

Input variables

Name	Type	Description
boExec	BOOL	Function execution: With a positive edge, the execution of the block starts. As long as 'boExec' = TRUE, the block is processed by the PLC. In the state 'boExec' = FALSE execution of the block is ended.
uiIDNo	UINT	ID number to be read out
uiParInst	UINT	Instance or Parameter set number or instance number

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boDone</td><td>BOOL</td><td>Response that the function block has been completely executed.</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state<table><tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr><tr><td>TRUE</td><td>Error</td></tr></table></td></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output<table><tr><td colspan="2">iErrID = 0</td><td>No error</td></tr><tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr><tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr></table>Range: Siehe 'Error bit information' auf Seite 532.</td></tr>
</table>

Input and output variables

Name	Type	Description
stList512	STRUCT	ST_LIST_512 List 512 Accommodates list information
stDevice	STRUCT	The device description structure assigns the block a device.

6.1.2.5 READ_SDO (FB)

The function block 'READ_SDO' reads a value from a CAN object.

User interface

Input variables

Name	Type	Description
boExec	BOOL	Function execution: With a positive edge, the execution of the block starts. As long as 'boExec' = TRUE, the block is processed by the PLC. In the state 'boExec' = FALSE execution of the block is ended.
uiIndex	UINT	Index of the SDO whose value is being read
usSubIndex	USINT	Subindex of the SDO whose value is being read
uiSize	UINT	Maximum data length available to accommodate the information to be read. uiSize ≤ SIZEOF(variable) referenced by 'pbyData'!
pbyData	POINTER	POINTER TO READ DATA Pointer referencing the structure / variable which is receiving the information read.

Output variables

<table>
<tr><th>Name</th><th>Type</th><th colspan="3">Description</th></tr>
<tr><td>boDone</td><td>BOOL</td><td colspan="3">Response that the function block has been completely executed.</td></tr>
<tr><td rowspan="3">boErr</td><td rowspan="3">BOOL</td><td colspan="3">The function block is in an error state</td></tr>
<tr><td>FALSE</td><td colspan="2">No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td colspan="2">Error</td></tr>
<tr><td rowspan="5">iErrID</td><td rowspan="5">INT</td><td colspan="3">Error identity number: Diagnostic number is output</td></tr>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
<tr><td colspan="3">Range: Siehe 'Error bit information' auf Seite 532.</td></tr>
<tr><td>uiOutSize</td><td>UINT</td><td colspan="3">Current data length entered (read) in the structure referenced by the 'pbyData' pointer.</td></tr>
</table>

Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

6.1.2.6 WRITE_ID_DINT (FB)

The 'WRITE_ID_DINT' function block writes the value of a parameter stored in the AMK subsystem.

User interface

Input variables

Name	Type	Description
boExec	BOOL	Function execution: With a positive edge, the execution of the block starts. As long as 'boExec' = TRUE, the block is processed by the PLC. In the state 'boExec' = FALSE execution of the block is ended.
uiIDNo	UINT	ID number to be read out
uiParInst	UINT	Instance or Parameter set number or instance number
diIDVal	DINT	Parameter value written to database

Output variables

Name	Type	Description
boDone	BOOL	Response that the function block has been completely executed.
boErr	BOOL	The function block is in an error state FALSE: No error (permitted commanding or warning) TRUE: Error
iErrID	INT	Error identity number: Diagnostic number is output iErrID = 0: No error iErrID ≠ 0, boErr = TRUE: Error iErrID ≠ 0, boErr = FALSE: Warning Range: Siehe 'Error bit information' auf Seite 532.

Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

6.1.2.7 WRITE_ID_DINT_TMP (FB)

The 'WRITE_ID_DINT_TMP' function block writes the value of a parameter organized temporarily in the AMK subsystem.

User interface

Input variables

Name	Type	Description
boExec	BOOL	Function execution: With a positive edge, the execution of the block starts. As long as 'boExec' = TRUE, the block is processed by the PLC. In the state 'boExec' = FALSE execution of the block is ended.
uiIDNo	UINT	ID number to be read out
uiParInst	UINT	Instance or Parameter set number or instance number
diIDVal	DINT	Parameter value written to database

Output variables

Name	Type	Description
boDone	BOOL	Response that the function block has been completely executed.
boErr	BOOL	The function block is in an error state FALSE | No error (permitted commanding or warning) TRUE | Error
iErrID	INT	Error identity number: Diagnostic number is output iErrID = 0 | No error iErrID ≠ 0 | boErr = TRUE | Error iErrID ≠ 0 | boErr = FALSE | Warning Range: Siehe 'Error bit information' auf Seite 532.

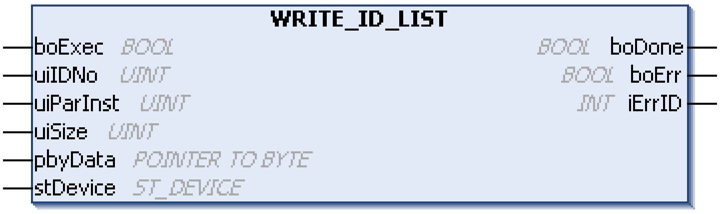
Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

6.1.2.8 WRITE_ID_LIST (FB)

The 'WRITE_ID_LIST' function block writes values of a list parameter to the database of an AMK subsystem.

User interface



Input variables

Name	Type	Description
boExec	BOOL	Function execution: With a positive edge, the execution of the block starts. As long as 'boExec' = TRUE, the block is processed by the PLC. In the state 'boExec' = FALSE execution of the block is ended.
uiIDNo	UINT	ID number to be written
uiParInst	UINT	Instance or Parameter set number or instance number
uiSize	UINT	Maximum data length of the information to be written. uiSize ≤ SIZEOF(variable) referenced by 'pbyData'!
pbyData	POINTER	POINTER TO WRITE DATA Pointer referencing the structure / variable which contains the information to be written.

Output variables

Name	Type	Description
boDone	BOOL	Response that the function block has been completely executed.

Name	Type	Description
boErr	BOOL	The function block is in an error state FALSE: No error (permitted commanding or warning) TRUE: Error
iErrID	INT	Error identity number: Diagnostic number is output iErrID = 0: No error iErrID ≠ 0, boErr = TRUE: Error iErrID ≠ 0, boErr = FALSE: Warning Range: Siehe 'Error bit information' auf Seite 532.

Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

6.1.2.9 WRITE_LIST_512 (FB)

The 'WRITE_LIST_512' function block writes values of a list parameter that is up to 512 bytes in size to the database of an AMK subsystem.

User interface

Input variables

Name	Type	Description
boExec	BOOL	Function execution: With a positive edge, the execution of the block starts. As long as 'boExec' = TRUE, the block is processed by the PLC. In the state 'boExec' = FALSE execution of the block is ended.
uiIDNo	UINT	ID number to be written
uiParInst	UINT	Instance or Parameter set number or instance number

Output variables

Name	Type	Description
boDone	BOOL	Response that the function block has been completely executed.
boErr	BOOL	The function block is in an error state FALSE: No error (permitted commanding or warning) TRUE: Error
iErrID	INT	Error identity number: Diagnostic number is output iErrID = 0: No error iErrID ≠ 0, boErr = TRUE: Error iErrID ≠ 0, boErr = FALSE: Warning Range: Siehe 'Error bit information' auf Seite 532.

Input and output variables

Name	Type	Description
stList512	STRUCT	ST_LIST_512 List 512 Accommodates list information
stDevice	STRUCT	The device description structure assigns the block a device.

6.1.2.10 WRITE_SDO (FB)

The function block 'WRITE_SDO' is used to write a value to a CAN object.

User interface

Input variables

Name	Type	Description
boExec	BOOL	Function execution: With a positive edge, the execution of the block starts. As long as 'boExec' = TRUE, the block is processed by the PLC. In the state 'boExec' = FALSE execution of the block is ended.
uiIndex	UINT	Index of the SDO whose value is being written
usSubIndex	USINT	Subindex of the SDO whose value is being written
uiSize	UINT	Maximum data length available to accommodate the information to be read. uiSize ≤ SIZEOF(variable) referenced by 'pbyData'!
pbyData	POINTER	POINTER TO WRITE DATA Pointer referencing the structure / variable which contains the information to be written.

Output variables

Name	Type	Description
boDone	BOOL	Response that the function block has been completely executed.
boErr	BOOL	The function block is in an error state FALSE: No error (permitted commanding or warning) TRUE: Error
iErrID	INT	Error identity number: Diagnostic number is output iErrID = 0: No error iErrID ≠ 0, boErr = TRUE: Error iErrID ≠ 0, boErr = FALSE: Warning Range: Siehe 'Error bit information' auf Seite 532.

Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

6.1.3 HigherAccess

6.1.3.1 READ_ID_DINT_ONCE (FB)

The 'READ_ID_DINT_ONCE' function block reads in the value of a parameter from the database of an AMK subsystem. The handshake 'boExec' / 'boDone' does not have to be organized.

User interface

Input variables

Name	Type	Description
uiIDNo	UINT	ID number to be read out
uiParInst	UINT	Instance or Parameter set number or instance number

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boDone</td><td>BOOL</td><td>Response that the function block has been completely executed.</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table></td></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Range: Siehe 'Error bit information' auf Seite 532.</td></tr>
<tr><td>diData</td><td>DINT</td><td>Parameter value read from database</td></tr>
</table>

Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

Actions

Name	Description
Start	Read: The process is started with the start action and acknowledged with 'boDone' = TRUE • The acknowledgement not revoked until the next start action is underway • The input parameters must be specified before the start action is triggered

6.1.3.2 WRITE_ID_DINT_ONCE (FB)

The 'WRITE_ID_DINT_ONCE' function block writes the value of a parameter to the database of an AMK subsystem. The handshake 'boExec' / 'boDone' does not have to be organized.

User interface

Input variables

Name	Type	Description
uiIDNo	UINT	ID number to be written
uiParInst	UINT	Instance or Parameter set number or instance number
diData	DINT	Parameter value written to database

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boDone</td><td>BOOL</td><td>Response that the function block has been completely executed.</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state<table><tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr><tr><td>TRUE</td><td>Error</td></tr></table></td></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output<table><tr><td colspan="2">iErrID = 0</td><td>No error</td></tr><tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr><tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr></table>Range: Siehe 'Error bit information' auf Seite 532.</td></tr>
</table>

Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

Actions

Name	Description
Start	Write: The process is started with the start action and acknowledged with 'boDone' = TRUE • The acknowledgement not revoked until the next start action is underway • The input parameters must be specified before the start action is triggered

6.1.4 MoreIds

6.1.4.1 READ_ALL_IDS (FB)

The 'READ_ALL_IDS' function block reads all elements of all parameters listed in ID17 'ID-no. list all operational data' from the database of an AMK subsystem.

User interface

Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed. • The parameters to be read according to ID17 'ID-no. list all operational data' are identified.
boExec	BOOL	Function execution: With a positive edge, the execution of the block starts. As long as 'boExec' = TRUE, the block is processed by the PLC. In the state 'boExec' = FALSE execution of the block is ended. • 'uiIDActIndex' is incremented on a positive edge • The value of the current parameter according to 'uiIDActIndex' is read in.
uiParInst	UINT	Instance or Parameter set number or instance number
stNetNo	STRUCT	ST_NET_NO Network address The network address can be identified with the 'FstNetNoOfDevice' function from the 'ST_DEVICE' structure, for example.
uiSize	UINT	Maximum data length available to accommodate the information to be read. uiSize ≤ SIZEOF(variable) referenced by 'pbyData'!
pbyData	POINTER	POINTER TO READ DATA Pointer referencing the structure / variable which is receiving the information read.

Output variables

Name	Type	Description
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled
boDone	BOOL	Response that the function block has been completely executed.
boErr	BOOL	The function block is in an error state FALSE: No error (permitted commanding or warning) TRUE: Error
iErrID	INT	Error identity number: Diagnostic number is output iErrID = 0: No error iErrID ≠ 0, boErr = TRUE: Error iErrID ≠ 0, boErr = FALSE: Warning Range: Siehe 'Error bit information' auf Seite 532.

Name	Type	Description
uiIDNo	UINT	ID number to be read out
uiIDActIndex	UINT	Index pointing to the parameter number currently being read. Range | Meaning 0 ... udIDMaxIndex: 0 | Initial value 1 | First parameter to be read 2 | Second parameter to be read ... | ...
uiIDMaxIndex	UINT	Maximum index of the last parameter in ID17 'ID-no. list all operational data'
boList	BOOL	Identifier for a list parameter FALSE | The data to be read is in 'stIDAll.diData' TRUE | List parameter: The list to be read is transferred to the list structure referenced by 'pbyData'

Input and output variables

Name	Type	Description
stIDAll	STRUCT	ST_ID_ALL Parameter information Accommodates the element information

6.1.4.2 READ_N_IDS_DINT (FB)

The 'READ_N_IDS_DINT' function block reads a defined number of parameter values from the database or temporary values of an AMK subsystem.

User interface

Input variables

Name	Type	Description
boExec	BOOL	Function execution: With a positive edge, the execution of the block starts. As long as 'boExec' = TRUE, the block is processed by the PLC. In the state 'boExec' = FALSE execution of the block is ended.
enMode	ENUM	EN_ACCESS_N_IDS Selection mode Definition of remanent or temporary parameters Default | ACCESS_N_IDS_DINT_TMP Range | Meaning ACCESS_N_IDS_DINT_TMP | Read temporary parameter value ACCESS_N_IDS_DINT_REM | Read remanent parameter value
uiN	UINT	Number of parameters to be read Range | 1 ... MAX_INDEX_FOR_ID_VALUES

Output variables

Name	Type	Description
boDone	BOOL	Response that the function block has been completely executed.
boErr	BOOL	The function block is in an error state FALSE: No error (permitted commanding or warning) TRUE: Error
iErrID	INT	Error identity number: Diagnostic number is output iErrID = 0: No error iErrID ≠ 0, boErr = TRUE: Error iErrID ≠ 0, boErr = FALSE: Warning Range: Siehe 'Error bit information' auf Seite 532.
uiErrIndex	UINT	An error occurred when reading in the value cited Range / Meaning 0: No error 1 ... MAX_ INDEX_FOR_ID_ VALUES: Index no. of the parameter affected by an error

Input and output variables

Name	Type	Description
stIdValues	STRUCT	ST_N_ID_VALUES List of parameters that can be read / written.
stDevice	STRUCT	The device description structure assigns the block a device.

6.1.4.3 WRITE_N_IDS_DINT (FB)

The 'WRITE_N_IDS_DINT' function block writes a defined number of parameter values to the database or temporary values of an AMK subsystem.

User interface

Input variables

Name	Type	Description
boExec	BOOL	Function execution: With a positive edge, the execution of the block starts. As long as 'boExec' = TRUE, the block is processed by the PLC. In the state 'boExec' = FALSE execution of the block is ended.

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>enMode</td><td>ENUM</td><td>EN_ACCESS_N_IDS
Selection mode
Definition of remanent or temporary parameters
<table>
<tr><td>Default</td><td>ACCESS_N_IDS_DINT_TMP</td></tr>
</table>
<table>
<tr><th>Range</th><th>Meaning</th></tr>
<tr><td>ACCESS_N_IDS_DINT_TMP</td><td>Read temporary parameter value</td></tr>
<tr><td>ACCESS_N_IDS_DINT_REM</td><td>Read remanent parameter value</td></tr>
</table></td></tr>
<tr><td>uiN</td><td>UINT</td><td>Number of parameters to be written
<table>
<tr><td>Range</td><td>1 ... MAX_INDEX_FOR_ID_VALUES</td></tr>
</table></td></tr>
</table>

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boDone</td><td>BOOL</td><td>Response that the function block has been completely executed.</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table></td></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Range: Siehe 'Error bit information' auf Seite 532.</td></tr>
<tr><td>uiErrIndex</td><td>UINT</td><td>An error occurred when writing the value cited
<table>
<tr><th>Range</th><th>Meaning</th></tr>
<tr><td>0</td><td>No error</td></tr>
<tr><td>1 ... MAX_INDEX_FOR_ID_VALUES</td><td>Index no. of the parameter affected by an error</td></tr>
</table></td></tr>
</table>

Input and output variables

Name	Type	Description
stIdValues	STRUCT	ST_N_ID_VALUES List of parameters that can be read / written.
stDevice	STRUCT	The device description structure assigns the block a device.

6.2 Support functions

ForInternalUse
Internal library functions

CLEAR_DINT LOCK_EXEC	The 'ForInternalUse' groups blocks that are required in the context of the internal function. These blocks are not relevant for the application in the context of application programming. Therefore, they are not described in more detail here.

General
General functions

FstNetNoOfDevice	Identification of network number

6.2.1 General

6.2.1.1 FstNetNoOfDevice (F)

The 'FstNetNoOfDevice' identifies the network address based on the device structure variable 'stDevice'.

User interface

Input variables

Name	Type	Description
pstDevice	POINTER	POINTER TO ST_DEVICE Pointer to the device structure variable

Output variables

Name	Type	Description
FstNetNoOfDevice	STRUCT	ST_NET_NO Return value 'stNetNo' structure which is assigned to the device structure

6.3 Types

6.3.1 Structures

6.3.1.1 AllElementsOfOneID

6.3.1.1.1 ST_ID_ALL (ST)

The 'ST_ID_ALL' groups all the elements of a parameter.

Structure elements

Name	Type	Description
diData	DINT	Parameter value
diMin	DINT	Minimum permissible value
diMax	DINT	Maximum permissible value
udAttr	UDINT	Parameter attribute (according to SERCOS standard) Bit 0 ... 15 Scaling Bit 16 ... 18 Data length Bit 19 Function Bit 20 ... 23 Data type Bit 24 ... 27 Decimal places Bit 28 ... 30 Write-protected Bit 31 Not used
stUnit	STRUCT	ST_ID_UNIT Parameter unit displayed as a list with an ASCII string

Name	Type	Description
stName	STRUCT	ST_ID_NAME Parameter name displayed as a list with an ASCII string

Structure definition

```
TYPE ST_ID_ALL:
    STRUCT
        diData:DINT;
        diMin:DINT;
        diMax:DINT;
        udAttr:UDINT;
        stUnit:ST_ID_UNIT;
        stName:ST_ID_NAME;
    END_STRUCT
END_TYPE
```

6.3.1.1.2 ST_ID_NAME (ST)

Parameter name, shown as list with ASCII string.

Structure elements

Name	Type	Description
uiActLen	UINT	Current list length
uiMaxLen	UINT	Maximum list length
strName	STRING	Parameter name

Structure definition

```
TYPE ST_ID_NAME:
    STRUCT
        uiActLen:UINT;
        uiMaxLen:UINT;
        strName: STRING(ID_NAME_SIZE);
    END_STRUCT
END_TYPE
```

6.3.1.1.3 ST_ID_UNIT (ST)

Parameter unit, shown as list with ASCII string.

Structure elements

Name	Type	Description
uiActLen	UINT	Current list length
uiMaxLen	UINT	Maximum list length
strUnit	STRING	Parameter unit

Structure definition

```
TYPE ST_ID_NAME:
    STRUCT
        uiActLen:UINT;
        uiMaxLen:UINT;
        strUnit: STRING(ID_UNIT_SIZE);
    END_STRUCT
END_TYPE
```

6.3.1.2 ElementaryAccess

6.3.1.2.1 ST_LIST_512 / ST_LIST_1024 / ST_LIST_2048 / ST_LIST_4096 (ST)

The 'ST_LIST_512' / '_1024' / '_2048' / '_4096' structures provide memory capacity for list parameters:

Structure	Memory capacity [bytes]	Header data [words]	User data [words]
ST_LIST_512	512	2	254
ST_LIST_1024	1024	2	510
ST_LIST_2048	2048	2	1022
ST_LIST_4096	4096	2	2046

Structure elements

Name	Type	Description
uiActLen	UINT	Current list length
uiMaxLen	UINT	Maximum list length
uiListEle	ARRAY	ST_LIST_512 : ARRAY [2..255] OF UINT ST_LIST_1024 : ARRAY [2..511] OF UINT ST_LIST_2048 : ARRAY [2..1023] OF UINT ST_LIST_4096 ARRAY [2..2047] OF UINT List elements user data

Structure definitions

```
TYPE ST_LIST_512:
    STRUCT
        uiActLen:UINT;
        uiMaxLen:UINT;
        uiListEle:ARRAY [2…255] OF UINT;
    END_STRUCT
END_TYPE

TYPE ST_LIST_1024:
    STRUCT
        uiActLen:UINT;
        uiMaxLen:UINT;
        uiListEle:ARRAY [2…511] OF UINT;
    END_STRUCT
END_TYPE

TYPE ST_LIST_2048:
    STRUCT
        uiActLen:UINT;
        uiMaxLen:UINT;
        uiListEle:ARRAY [2…1023] OF UINT;
    END_STRUCT
END_TYPE
```

```
TYPE ST_LIST_4096:
    STRUCT
        uiActLen:UINT;
        uiMaxLen:UINT;
        uiListEle:ARRAY [2…2047] OF UINT;
    END_STRUCT
END_TYPE
```

6.3.1.2.2 ST_LIST_VAR_LEN (ST)

The 'ST_LIST_VAR_LEN' structure provides memory capacity for list parameters.

Structure elements

Name	Type	Description
uiActLen	UINT	Current list length
uiMaxLen	UINT	Maximum list length
uiListEle	ARRAY	ARRAY [2..MAX_LIST_INDEX] OF UINT List elements user data

Structure definition

```
VAR_GLOBAL CONSTANT
  MAX_LIST_INDEX : INT := 2047;          (* maximum index of list elements, constants defined in AmkBase.lib *)
END_VAR

TYPE ST_LIST_VAR_LEN:
    STRUCT
        uiActLen:UINT;
        uiMaxLen:UINT;
        uiListEle:ARRAY [2…MAX_LIST_INDEX] OF UINT;
    END_STRUCT
END_TYPE
```

6.3.1.3 MoreIds

6.3.1.3.1 ST_N_ID_VALUES (ST)

The 'ST_N_ID_VALUES' structure groups all parameter values to be read and written.

Structure elements

Name	Type	Description
arr_stIdValue	ARRAY	ARRAY [1..MAX_INDEX_FOR_ID_VALUES] OF ST_ID_VALUE

Structure definition

```
VAR_GLOBAL CONSTANT
  MAX_INDEX_FOR_ID_VALUES : UINT := 10;     (* number of parameters*)
END_VAR

TYPE ST_N_ID_VALUES:
    STRUCT
        arr_stIdValue: ARRAY[1..MAX_INDEX_FOR_ID_VALUES] OF ST_ID_VALUE;
    END_STRUCT
END_TYPE
```

6.3.1.3.2 ST_ID_VALUE (ST)

The 'ST_ID_VALUE' structure contains the variable that describe a parameter.

Structure elements

Name	Type	Description
uiIDNo	UINT	Parameter number (ID)
uiParInst	UINT	Instance or Parameter set number or instance number
diIDVal	DINT	Parameter value

Structure definition

```
TYPE ST_ID_VALUE:
    STRUCT
        uiIDNo: UINT;
        uiParInst: UINT;
        diIDVal:DINT;
    END_STRUCT
END_TYPE
```

7 AmkTabc - AMK table calculation blocks

AmkTabc is an AMK library containing blocks for calculating special table profiles. The basis for the library is provided by the 'TAB_CALC' block, which is contained in the AmkBase library.

The AmkTabc library is divided into the following table types:

OperatingTables	Operating tables
PhasingInTables	Phasing in tables
PhasingOutTables	Phasing out tables
PositioningProfiles	Positioning profiles
Support	Support functions

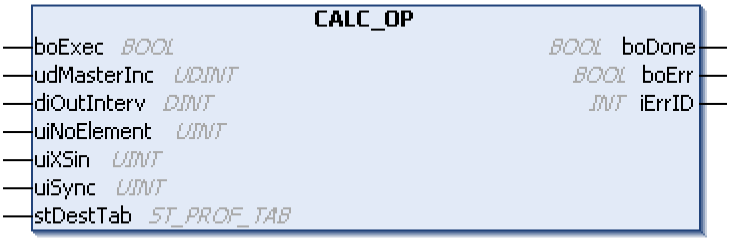
7.1 Operating tables

CALC_OP	Calculation of the operating table

7.1.1 CALC_OP (FB)

The 'CALC_OP' function block calculates the operating table based on a synchronous straight line to the start of a $\sin^2$ smoothing function with tangential merging.

User interface



Input variables

Name	Type	Description
boExec	BOOL	Function execution: With a positive edge, the execution of the block starts. As long as 'boExec' = TRUE, the block is processed by the PLC. In the state 'boExec' = FALSE execution of the block is ended.
udMasterInc	UDINT	Increments of the master drive which produce a table cycle Max. table X value Range: 0 ... 5000000 Unit: incr
diOutInterv	DINT	Output interface defining the output increments per table cycle Max. table Y value
uiNoElement	UINT	Element number of the last table element calculated, number of table interpolation points
uiXSin	UINT	Sine start x position [°] at which the change from synchronous straight line to $\sin^2$ smoothing function takes place
uiSync	UINT	Synchronous factor Ratio of output increments to input increments Incline of synchronous straight line; uiSync := 100 corresponds to an incline of 1 Range: 100 ... 32767 Unit: %

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boDone</td><td>BOOL</td><td>Response that the function block has been completely executed.</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table></td></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error
<table>
<tr><th>Value</th><th>Meaning</th></tr>
<tr><td>1</td><td>Incorrect number of elements
the maximum number is dependent on the table type 'enTabType'</td></tr>
<tr><td>2</td><td>Incorrect parameter set variant
dependent on the table type 'enTabKind'</td></tr>
<tr><td>3</td><td>'udMasterInc' value too high</td></tr>
<tr><td>4</td><td>'diOutInterv' value too high / too low</td></tr>
<tr><td>5</td><td>'diPar1' illegal value
dependent on 'enTabType' and 'enTabKind'</td></tr>
<tr><td>6</td><td>'diPar2' illegal value
dependent on 'enTabType' and 'enTabKind'</td></tr>
<tr><td>7</td><td>'diPar3' illegal value
dependent on 'enTabType' and 'enTabKind'</td></tr>
<tr><td>8</td><td>'diPar4' illegal value
dependent on 'enTabType' and 'enTabKind'</td></tr>
<tr><td>9</td><td>Illegal synchronous point</td></tr>
<tr><td>10</td><td>Illegal phasing in point</td></tr>
<tr><td>11</td><td>Illegal phasing out point</td></tr>
<tr><td>12</td><td>Illegal sine starting point</td></tr>
<tr><td>13</td><td>Velocity too low</td></tr>
<tr><td>14</td><td>Acceleration too low</td></tr>
</table></td></tr>
</table>

Input and output variables

Name	Type	Description
stDestTab	STRUCT	ST_PROF_TAB Profile table structure

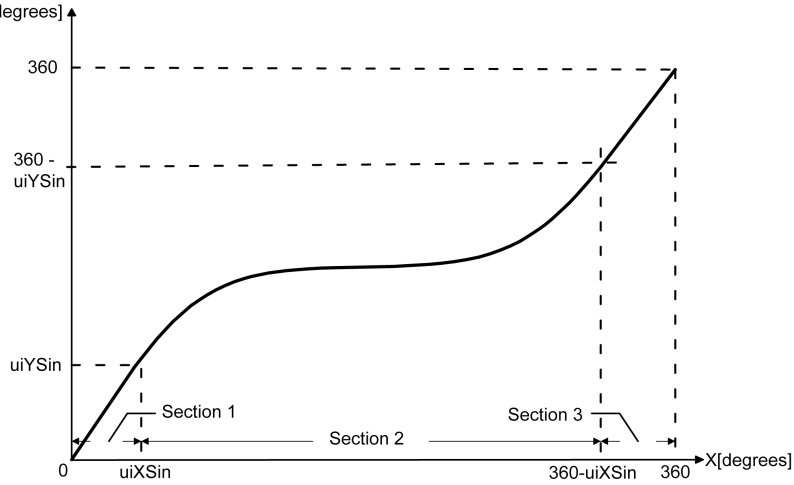
Description

The position setpoints for the periodic movement of an axis are stored in the operating table.

The output values of the operating table (y coordinates) are assigned to the incoming increments of a master axis (x coordinates). This assignment is made with the 'CAM_PROF' block. The table is calculated in the Y table format.

The output values for the operating table are calculated so that the process will start with linear movement with a variable synchronous factor 'uiSync'. Harmonic smoothing movement is calculated with a $\sin^2$ function.

Abbildung 38: CALC_OP: operating function



Section 1	0 ≤ x < uiXSin
Section 2	uiXSin ≤ x < 360-uiXSin
Section 3	360-uiXSin ≤ x < 360

7.2 Phasing in tables

A phasing in table phases a stationary axis into a movement sequence which is controlled by the 'CAM_PROF' block, for example.

CALC_IN_ALLDEF	Calculation of the phasing in table based on phasing in point and synchronous point
CALC_IN_INDEF	Calculation of the phasing in table based on phasing in point and synchronous ratio
CALC_IN_SYNCDEF	Calculation of the phasing in table based on phasing in y position and synchronous point

7.2.1 CALC_IN_ALLDEF (FB)

The 'CALC_IN_ALLDEF' function block calculates the phasing in table based on the phasing in point and the synchronous point. The phasing in table calculated is based on two parabolic partial sections with tangential transition and tangential merging into the synchronous straight line.

User interface

Input variables

Name	Type	Description
boExec	BOOL	Function execution: With a positive edge, the execution of the block starts. As long as 'boExec' = TRUE, the block is processed by the PLC. In the state 'boExec' = FALSE execution of the block is ended.

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>udMasterInc</td><td>UDINT</td><td>Increments of the master drive which produce a table cycle
Max. table X value<table><tr><td>Range</td><td>0 ... 5000000</td></tr><tr><td>Unit</td><td>incr</td></tr></table></td></tr>
<tr><td>diOutInterv</td><td>DINT</td><td>Output interface defining the output increments per table cycle
Max. table Y value<table><tr><td>Unit</td><td>incr</td></tr></table></td></tr>
<tr><td>uiNoElement</td><td>UINT</td><td>Element number of the last table element calculated,
number of table interpolation points<table><tr><td>Range</td><td>5 ... ((SIZEOF(ST_PROF_TAB)-8)/4)-1</td></tr></table></td></tr>
<tr><td>uiXIn</td><td>UINT</td><td>Phasing in point x coordinate
Phasing in starts from this position, transition to parabola<table><tr><td>Range</td><td>0 ... 360</td></tr><tr><td>Unit</td><td>°</td></tr></table></td></tr>
<tr><td>iYIn</td><td>INT</td><td>Phasing in point y coordinate
Phasing in starts from this position, transition to parabola<table><tr><td>Range</td><td>-360 ... 360</td></tr><tr><td>Unit</td><td>°</td></tr></table></td></tr>
<tr><td>uiXSync</td><td>UINT</td><td>Synchronous point x coordinate
The change to the synchronous straight line starts from this position<table><tr><td>Range</td><td>0 ... 360</td></tr><tr><td>Unit</td><td>°</td></tr></table></td></tr>
<tr><td>uiSync</td><td>UINT</td><td>Synchronous factor
Ratio of output increments to input increments
Incline of synchronous straight line; uiSync := 100 corresponds to an incline of 1<table><tr><td>Range</td><td>100 ... 32767</td></tr><tr><td>Unit</td><td>%</td></tr></table></td></tr>
</table>

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boDone</td><td>BOOL</td><td>Response that the function block has been completely executed.</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state<table><tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr><tr><td>TRUE</td><td>Error</td></tr></table></td></tr>
</table>

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error
<table>
<tr><th>Value</th><th>Meaning</th></tr>
<tr><td>1</td><td>Incorrect number of elements
the maximum number is dependent on the table type 'enTabType'</td></tr>
<tr><td>2</td><td>Incorrect parameter set variant
dependent on the table type 'enTabKind'</td></tr>
<tr><td>3</td><td>'udMasterInc' value too high</td></tr>
<tr><td>4</td><td>'diOutInterv' value too high / too low</td></tr>
<tr><td>5</td><td>'diPar1' illegal value
dependent on 'enTabType' and 'enTabKind'</td></tr>
<tr><td>6</td><td>'diPar2' illegal value
dependent on 'enTabType' and 'enTabKind'</td></tr>
<tr><td>7</td><td>'diPar3' illegal value
dependent on 'enTabType' and 'enTabKind'</td></tr>
<tr><td>8</td><td>'diPar4' illegal value
dependent on 'enTabType' and 'enTabKind'</td></tr>
<tr><td>9</td><td>Illegal synchronous point</td></tr>
<tr><td>10</td><td>Illegal phasing in point</td></tr>
<tr><td>11</td><td>Illegal phasing out point</td></tr>
<tr><td>12</td><td>Illegal sine starting point</td></tr>
<tr><td>13</td><td>Velocity too low</td></tr>
<tr><td>14</td><td>Acceleration too low</td></tr>
</table>
</td></tr>
</table>

Input and output variables

Name	Type	Description
stDestTab	STRUCT	ST_PROF_TAB Profile table structure

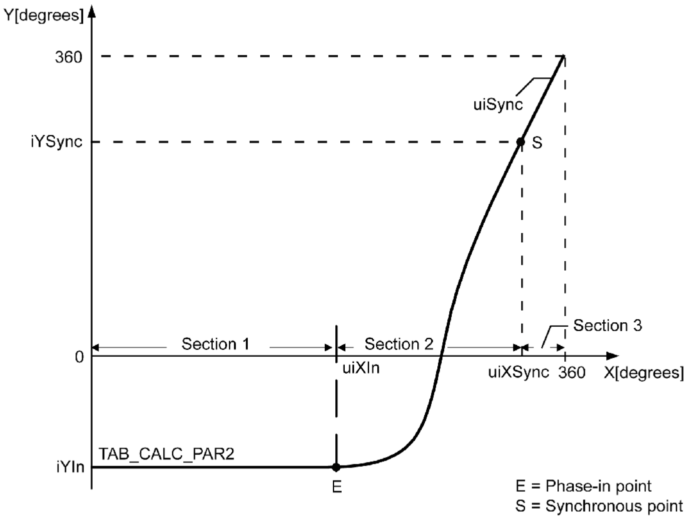
Description

The values for the phasing in table are calculated using two parabolas. This results in an extended definition range for the parameters of the phasing in process. Accordingly, the start value for the phasing in process can be less than 0 degrees.

The parabolas are calculated so that the acceleration value remains constant throughout the phasing in process.

The input parameters are the x and y coordinates of the phasing in point, the x coordinate of the synchronous point, and the synchronous factor.
The table is calculated in the Y table format.

Abbildung 39: CALC_IN_ALLDEF: Phasing in function with defined phasing in and synchronous points



Section 1	0 ≤ x < uiXIn	
Section 2	uiXIn ≤ x < uiXSync	Phasing in process: Two symmetrical parabolas $y = ax^2$ First parabola: vertex at 'uiXIn' Second parabola: vertex at 'uiXVer'
Section 3	uiXSync ≤ x < 360	

Calculation of the parabolas

The factor a of the second parabola $y = -a(x\text{-uiXVer})^2$ is calculated from the equation y' = uiSync/100 = -2a(x-uiXVer).
In the synchronous point S:

uiXVer: x coordinate of the vertex of the second parabola

The following dependencies and conditions apply:

-
-
-

7.2.2 CALC_IN_INDEF (FB)

The 'CALC_IN_INDEF' function block calculates the phasing in table based on the phasing in point and the synchronous ratio. The calculated phasing in table is based on a partial section of a parabola with tangential merging into the synchronous straight line.

User interface

Input variables

Name	Type	Description
boExec	BOOL	Function execution: With a positive edge, the execution of the block starts. As long as 'boExec' = TRUE, the block is processed by the PLC. In the state 'boExec' = FALSE execution of the block is ended.

Name	Type	Description
udMasterInc	UDINT	Increments of the master drive which produce a table cycle Max. table X value Range: 0 ... 5000000 Unit: incr
diOutInterv	DINT	Output interface defining the output increments per table cycle Max. table Y value Unit: incr
uiNoElement	UINT	Element number of the last table element calculated, number of table interpolation points Range: 5 ... ((SIZEOF(ST_PROF_TAB)-8)/4)-1
uiXIn	UINT	Phasing in point x coordinate Phasing in starts from this position, transition to parabola Range: 0 ... 360 Unit: °
iYIn	INT	Phasing in point y coordinate Phasing in starts from this position, transition to parabola Range: -360 ... 360 Unit: °
uiSync	UINT	Synchronous factor Ratio of output increments to input increments Incline of synchronous straight line; uiSync := 100 corresponds to an incline of 1 Range: 100 ... 32767 Unit: %

Output variables

Name	Type	Description
boDone	BOOL	Response that the function block has been completely executed.
boErr	BOOL	The function block is in an error state FALSE: No error (permitted commanding or warning) TRUE: Error

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error
<table>
<tr><th>Value</th><th>Meaning</th></tr>
<tr><td>1</td><td>Incorrect number of elements
the maximum number is dependent on the table type 'enTabType'</td></tr>
<tr><td>2</td><td>Incorrect parameter set variant
dependent on the table type 'enTabKind'</td></tr>
<tr><td>3</td><td>'udMasterInc' value too high</td></tr>
<tr><td>4</td><td>'diOutInterv' value too high / too low</td></tr>
<tr><td>5</td><td>'diPar1' illegal value
dependent on 'enTabType' and 'enTabKind'</td></tr>
<tr><td>6</td><td>'diPar2' illegal value
dependent on 'enTabType' and 'enTabKind'</td></tr>
<tr><td>7</td><td>'diPar3' illegal value
dependent on 'enTabType' and 'enTabKind'</td></tr>
<tr><td>8</td><td>'diPar4' illegal value
dependent on 'enTabType' and 'enTabKind'</td></tr>
<tr><td>9</td><td>Illegal synchronous point</td></tr>
<tr><td>10</td><td>Illegal phasing in point</td></tr>
<tr><td>11</td><td>Illegal phasing out point</td></tr>
<tr><td>12</td><td>Illegal sine starting point</td></tr>
<tr><td>13</td><td>Velocity too low</td></tr>
<tr><td>14</td><td>Acceleration too low</td></tr>
</table></td></tr>
<tr><td>uiXSync</td><td>UINT</td><td>Synchronous point x coordinate
The change to the synchronous straight line starts from this position
<table>
<tr><td>Unit</td><td>°</td></tr>
</table></td></tr>
</table>

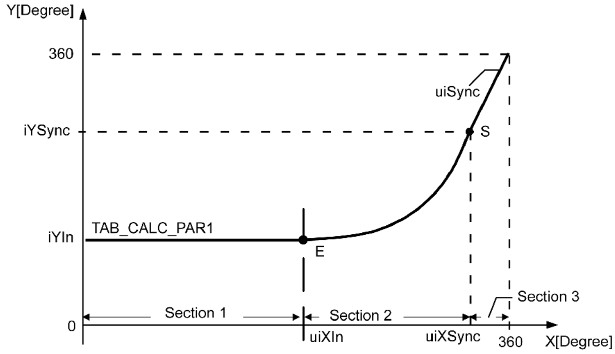
Input and output variables

Name	Type	Description
stDestTab	STRUCT	ST_PROF_TAB Profile table structure

Description

For the 'CALC_IN_INDEF' block, the phasing in curve is calculated with a parabola. Here, the input parameters are the x and y coordinates of the phasing in point, the x coordinate of the synchronous point, and the synchronous factor. The table is calculated in the Y table format.

Abbildung 40: CALC_IN_INDEF: Phasing in function with defined phasing in point



E = Phase-in point
S = Synchronous point

Section 1	$0 \le x <$ uiXIn	
Section 2	uiXIn $\le x <$ uiXSync	Phasing in process: parabola $y = ax^2$
Section 3	uiXSync $\le x < 360$	

Calculation of the parabola

The factor a of the parabola $y = -ax^2$ is calculated from the equation y' = uiSync/100 = 2ax .
In the synchronous point S:

Therefore, it follows that for a:

where

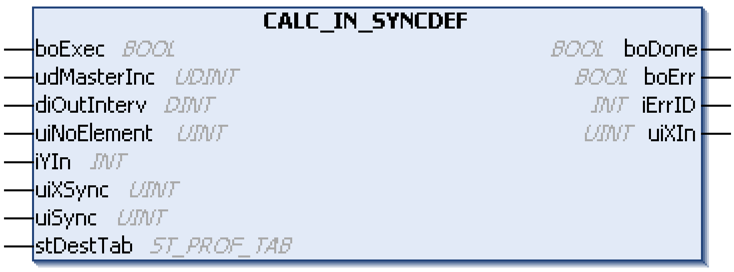
The following dependencies and conditions apply:

-
-

7.2.3 CALC_IN_SYNCDEF (FB)

The 'CALC_IN_SYNCDEF' function block calculates the phasing in table based on the y coordinates of the phasing in position and the synchronous point.
The phasing in table calculated is based on a parabolic partial section with tangential merging into the synchronous straight line.

User interface



Input variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boExec</td><td>BOOL</td><td>Function execution: With a positive edge, the execution of the block starts.
As long as 'boExec' = TRUE, the block is processed by the PLC.
In the state 'boExec' = FALSE execution of the block is ended.</td></tr>
<tr><td>udMasterInc</td><td>UDINT</td><td>Increments of the master drive which produce a table cycle
Max. table X value<table><tr><td>Range</td><td>0 ... 5000000</td></tr><tr><td>Unit</td><td>incr</td></tr></table></td></tr>
</table>

Name	Type	Description
diOutInterv	DINT	Output interface defining the output increments per table cycle Max. table Y value Unit: incr
uiNoElement	UINT	Element number of the last table element calculated, number of table interpolation points Range: 5 ... ((SIZEOF(ST_PROF_TAB)-8)/4)-1
iYIn	INT	Phasing in point y coordinate Phasing in starts from this position, transition to parabola Range: -360 ... 360 Unit: °
uiXSync	UINT	Synchronous point x coordinate The change to the synchronous straight line starts from this position Range: 0 ... 360 Unit: °
uiSync	UINT	Synchronous factor Ratio of output increments to input increments Incline of synchronous straight line; uiSync := 100 corresponds to an incline of 1 Range: 100 ... 32767 Unit: %

Output variables

Name	Type	Description
boDone	BOOL	Response that the function block has been completely executed.
boErr	BOOL	The function block is in an error state FALSE: No error (permitted commanding or warning) TRUE: Error

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error
<table>
<tr><td>Value</td><td>Meaning</td></tr>
<tr><td>1</td><td>Incorrect number of elements
the maximum number is dependent on the table type 'enTabType'</td></tr>
<tr><td>2</td><td>Incorrect parameter set variant
dependent on the table type 'enTabKind'</td></tr>
<tr><td>3</td><td>'udMasterInc' value too high</td></tr>
<tr><td>4</td><td>'diOutInterv' value too high / too low</td></tr>
<tr><td>5</td><td>'diPar1' illegal value
dependent on 'enTabType' and 'enTabKind'</td></tr>
<tr><td>6</td><td>'diPar2' illegal value
dependent on 'enTabType' and 'enTabKind'</td></tr>
<tr><td>7</td><td>'diPar3' illegal value
dependent on 'enTabType' and 'enTabKind'</td></tr>
<tr><td>8</td><td>'diPar4' illegal value
dependent on 'enTabType' and 'enTabKind'</td></tr>
<tr><td>9</td><td>Illegal synchronous point</td></tr>
<tr><td>10</td><td>Illegal phasing in point</td></tr>
<tr><td>11</td><td>Illegal phasing out point</td></tr>
<tr><td>12</td><td>Illegal sine starting point</td></tr>
<tr><td>13</td><td>Velocity too low</td></tr>
<tr><td>14</td><td>Acceleration too low</td></tr>
</table></td></tr>
<tr><td>uiXIn</td><td>UINT</td><td>Phasing in point x coordinate
Phasing in starts from this position, transition to parabola
<table>
<tr><td>Unit</td><td>°</td></tr>
</table></td></tr>
</table>

Input and output variables

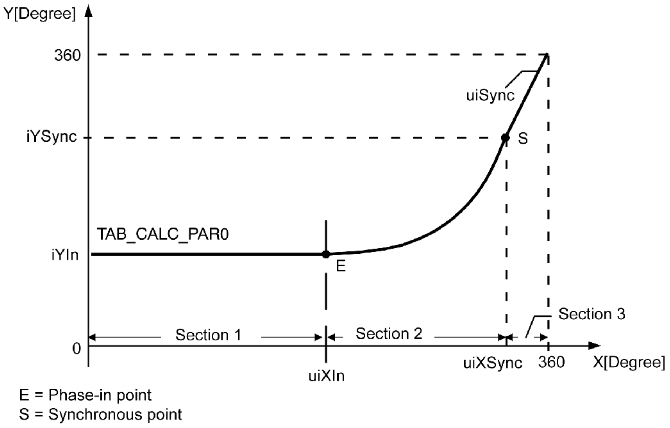
Name	Type	Description
stDestTab	STRUCT	ST_PROF_TAB Profile table structure

Description

The phasing in table is calculated with a parabola.

The input parameters are the y coordinates of the phasing in point, the x coordinate of the synchronous point, and the synchronous factor.
The table is calculated in the Y table format.

Abbildung 41: CALC_IN_SYNCDEF: Phasing in function with defined phasing in and synchronous points



Section 1	$0 \leq x < uiXIn$	
Section 2	$uiXIn \leq x < uiXSync$	Phasing in process: parabola $y = ax^2$
Section 3	$uiXSync \leq x < 360$	

Calculation of the parabola

The factor a of the parabola $y = -ax^2$ is calculated from the equation $y' = uiSync/100 = 2ax$.
In the synchronous point S:

Therefore, it follows that for a:

The following dependencies and conditions apply:

7.3 Phasing out tables

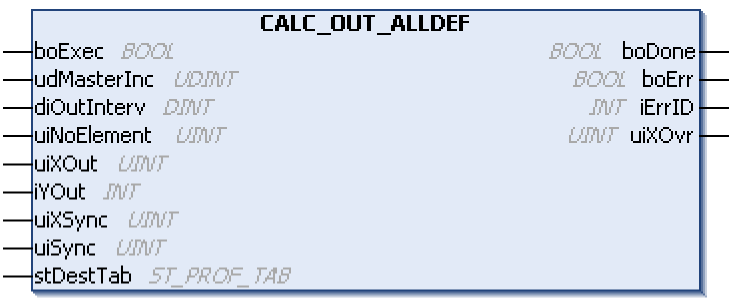
If an axis is moved in an operating table with the help of the 'CAM_PROF' block, for example, the phasing out table serves the purpose of phasing out from this sequence and stopping in a defined angular position.

CALC_OUT_ALLDEF	Calculation of the phasing out table based on phasing out point and synchronous point
CALC_OUT_OUTDEF	Calculation of the phasing out table based on phasing out point and synchronous ratio
CALC_OUT_SYNCDEF	Calculation of the phasing out table based on the y value of the phasing out position and synchronous point

7.3.1 CALC_OUT_ALLDEF (FB)

The 'CALC_OUT_ALLDEF' function block calculates the phasing out table based on the phasing out point and the synchronous point.
The phasing out table calculated is based on two parabolic partial sections with tangential transition and tangential exit out of the synchronous straight line.

User interface



Input variables

Name	Type	Description
boExec	BOOL	Function execution: With a positive edge, the execution of the block starts. As long as 'boExec' = TRUE, the block is processed by the PLC. In the state 'boExec' = FALSE execution of the block is ended.
udMasterInc	UDINT	Increments of the master drive which produce a table cycle Max. table X value Range: 0 ... 5000000 Unit: incr
diOutInterv	DINT	Output interface defining the output increments per table cycle Max. table Y value Unit: incr
uiNoElement	UINT	Element number of the last table element calculated, number of table interpolation points Range: 5 ... ((SIZEOF(ST_PROF_TAB)-8)/4)-1
uiXOut	UINT	Phasing out point x coordinate Phasing out ends with this position, transition from parabola to standstill Range: 0 ... 360 Unit: °
iYOut	INT	Phasing out point y coordinate Phasing out ends with this position, transition from parabola to standstill Range: -360 ... 360 Unit: °
uiXSync	UINT	Synchronous point x coordinate The synchronous straight line ends from this position Range: 0 ... 360 Unit: °
uiSync	UINT	Synchronous factor Ratio of output increments to input increments Incline of synchronous straight line; uiSync := 100 corresponds to an incline of 1 Range: 100 ... 32767 Unit: %

Output variables

Name	Type	Description
boDone	BOOL	Response that the function block has been completely executed.
boErr	BOOL	The function block is in an error state FALSE: No error (permitted commanding or warning) TRUE: Error

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error
<table>
<tr><th>Value</th><th>Meaning</th></tr>
<tr><td>1</td><td>Incorrect number of elements
the maximum number is dependent on the table type 'enTabType'</td></tr>
<tr><td>2</td><td>Incorrect parameter set variant
dependent on the table type 'enTabKind'</td></tr>
<tr><td>3</td><td>'udMasterInc' value too high</td></tr>
<tr><td>4</td><td>'diOutInterv' value too high / too low</td></tr>
<tr><td>5</td><td>'diPar1' illegal value
dependent on 'enTabType' and 'enTabKind'</td></tr>
<tr><td>6</td><td>'diPar2' illegal value
dependent on 'enTabType' and 'enTabKind'</td></tr>
<tr><td>7</td><td>'diPar3' illegal value
dependent on 'enTabType' and 'enTabKind'</td></tr>
<tr><td>8</td><td>'diPar4' illegal value
dependent on 'enTabType' and 'enTabKind'</td></tr>
<tr><td>9</td><td>Illegal synchronous point</td></tr>
<tr><td>10</td><td>Illegal phasing in point</td></tr>
<tr><td>11</td><td>Illegal phasing out point</td></tr>
<tr><td>12</td><td>Illegal sine starting point</td></tr>
<tr><td>13</td><td>Velocity too low</td></tr>
<tr><td>14</td><td>Acceleration too low</td></tr>
</table>
</td></tr>
<tr><td>uiXOvr</td><td>UINT</td><td>Maximum overshoot
Corresponds to the y position of the vertex of the first parabola in position 'uiXVer'</td></tr>
</table>

Input and output variables

Name	Type	Description
stDestTab	STRUCT	ST_PROF_TAB Profile table structure

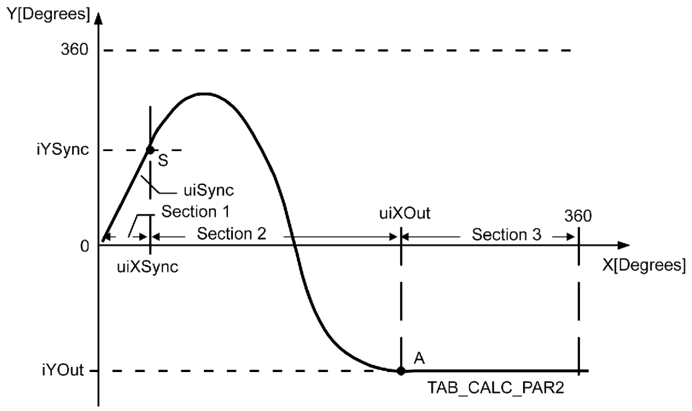
Description

The values of the phasing out table are calculated using two parabolas. This results in an extended definition range for the parameters of the phasing out process. Accordingly, the final value for the phasing out process can be less than 0 degrees.

The vertex of the first parabola is calculated so that the acceleration value remains constant throughout the phasing out process.

The input parameters are the x and y coordinates of the phasing out point, the x coordinate of the synchronous point, and the synchronous factor.
The table is calculated in the Y table format.

Abbildung 42: CALC_OUT_ALLDEF: Phasing out function with defined phasing out and synchronous points



A = Phase-out point
S = Synchronous point

Section 1	0 ≤ x < uiXSync	
Section 2	uiXSync ≤ x < uiXOut	Phasing out process: Two symmetrical parabolas $y = ax^2$ First parabola: vertex at 'uiXIn' Second parabola: vertex at 'uiXVer'
Section 3	uiXOut ≤ x < 360	y=iYOut

Calculation of the parabolas

The factor a of the first parabola $y = -a(x-uiXVer)^2$ is calculated from the equation y' = -2a(x-uiXVer).
In the synchronous point S:

uiXVer: x coordinate of the vertex of the first parabola

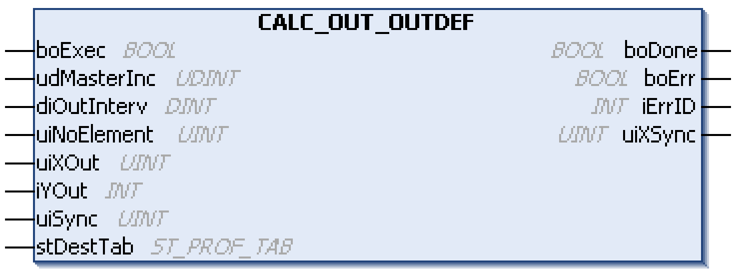
The following dependencies and conditions apply:

-
-

7.3.2 CALC_OUT_OUTDEF (FB)

The 'CALC_IN_OUTDEF' function block calculates the phasing out table based on the phasing out point and the synchronous point.
The calculated phasing out table is based on a partial section of a parabola with tangential exit out of the synchronous straight line.

User interface



Input variables

Name	Type	Description
boExec	BOOL	Function execution: With a positive edge, the execution of the block starts. As long as 'boExec' = TRUE, the block is processed by the PLC. In the state 'boExec' = FALSE execution of the block is ended.

Name	Type	Description
udMasterInc	UDINT	Increments of the master drive which produce a table cycle Max. table X value Range: 0 ... 5000000 Unit: incr
diOutInterv	DINT	Output interface defining the output increments per table cycle Max. table Y value Unit: incr
uiNoElement	UINT	Element number of the last table element calculated, number of table interpolation points Range: 5 ... ((SIZEOF(ST_PROF_TAB)-8)/4)-1
uiXOut	UINT	Phasing out point x coordinate Phasing out ends with this position, transition from parabola to standstill Range: 0 ... 360 Unit: °
iYOut	INT	Phasing out point y coordinate Phasing out ends with this position, transition from parabola to standstill Range: -360 ... 360 Unit: °
uiSync	UINT	Synchronous factor Ratio of output increments to input increments Incline of synchronous straight line; uiSync := 100 corresponds to an incline of 1 Range: 100 ... 32767 Unit: %

Output variables

Name	Type	Description
boDone	BOOL	Response that the function block has been completely executed.
boErr	BOOL	The function block is in an error state FALSE: No error (permitted commanding or warning) TRUE: Error

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error
<table>
<tr><th>Value</th><th>Meaning</th></tr>
<tr><td>1</td><td>Incorrect number of elements
the maximum number is dependent on the table type 'enTabType'</td></tr>
<tr><td>2</td><td>Incorrect parameter set variant
dependent on the table type 'enTabKind'</td></tr>
<tr><td>3</td><td>'udMasterInc' value too high</td></tr>
<tr><td>4</td><td>'diOutInterv' value too high / too low</td></tr>
<tr><td>5</td><td>'diPar1' illegal value
dependent on 'enTabType' and 'enTabKind'</td></tr>
<tr><td>6</td><td>'diPar2' illegal value
dependent on 'enTabType' and 'enTabKind'</td></tr>
<tr><td>7</td><td>'diPar3' illegal value
dependent on 'enTabType' and 'enTabKind'</td></tr>
<tr><td>8</td><td>'diPar4' illegal value
dependent on 'enTabType' and 'enTabKind'</td></tr>
<tr><td>9</td><td>Illegal synchronous point</td></tr>
<tr><td>10</td><td>Illegal phasing in point</td></tr>
<tr><td>11</td><td>Illegal phasing out point</td></tr>
<tr><td>12</td><td>Illegal sine starting point</td></tr>
<tr><td>13</td><td>Velocity too low</td></tr>
<tr><td>14</td><td>Acceleration too low</td></tr>
</table></td></tr>
<tr><td>uiXSync</td><td>UINT</td><td>Synchronous point x coordinate
The synchronous straight line ends from this position
<table>
<tr><td>Unit</td><td>°</td></tr>
</table></td></tr>
</table>

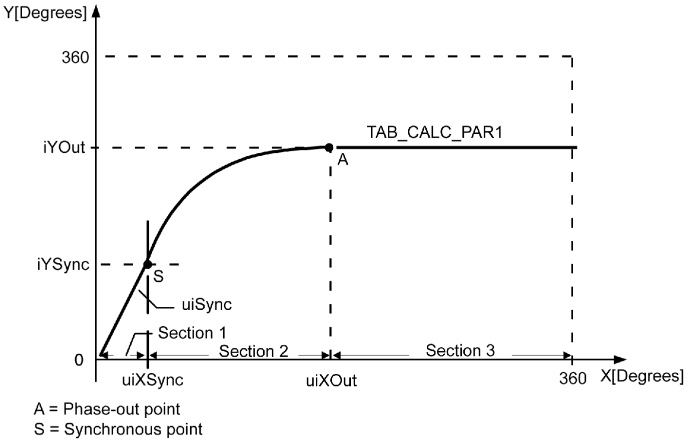
Input and output variables

Name	Type	Description
stDestTab	STRUCT	ST_PROF_TAB Profile table structure

Description

For the 'CALC_OUT_OUTDEF' block, the phasing out curve is calculated with a parabola. The input parameters are the x and y coordinates of the phasing out point and the synchronous factor. The table is calculated in the Y table format.

Abbildung 43: CALC_OUT_OUTDEF: Phasing out function with defined phasing out point



Section 1	$0 \leq x <$ uiXSync	
Section 2	uiXSync $\leq x <$ uiXOut	Phasing out process: Parabola $y = -ax^2$
Section 3	uiXOut $\leq x < 360$	

Calculation of the parabola

The factor a of the parabola $y = -ax^2$ is calculated from the equation y' = uiSync/100 = -2ax .
In the synchronous point S:

Therefore, it follows that for a:

where

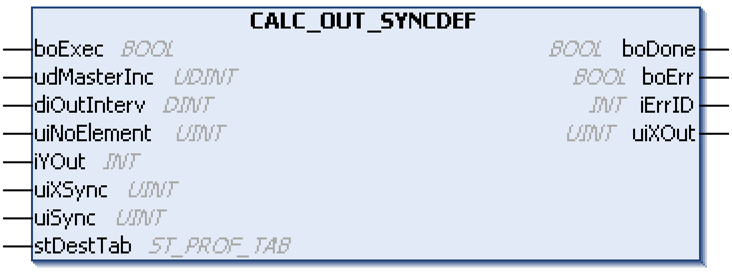
The following dependencies and conditions apply:

-
-

7.3.3 CALC_OUT_SYNCDEF (FB)

The 'CALC_OUT_SYNCDEF' function block calculates the phasing out table based on the y coordinate and the phasing out position.
The phasing out table calculated is based on a parabolic partial section with tangential exit out of the synchronous straight line.

User interface



Input variables

Name	Type	Description
boExec	BOOL	Function execution: With a positive edge, the execution of the block starts. As long as 'boExec' = TRUE, the block is processed by the PLC. In the state 'boExec' = FALSE execution of the block is ended.
udMasterInc	UDINT	Increments of the master drive which produce a table cycle Max. table X value Range: 0 ... 5000000 Unit: incr

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>diOutInterv</td><td>DINT</td><td>Output interface defining the output increments per table cycle
Max. table Y value<table><tr><td>Unit</td><td>incr</td></tr></table></td></tr>
<tr><td>uiNoElement</td><td>UINT</td><td>Element number of the last table element calculated,
number of table interpolation points<table><tr><td>Range</td><td>5 ... ((SIZEOF(ST_PROF_TAB)-8)/4)-1</td></tr></table></td></tr>
<tr><td>iYOut</td><td>INT</td><td>Phasing out point y coordinate
Phasing out ends with this position, transition from parabola to standstill<table><tr><td>Range</td><td>-360 ... 360</td></tr><tr><td>Unit</td><td>°</td></tr></table></td></tr>
<tr><td>uiXSync</td><td>UINT</td><td>Synchronous point x coordinate
The synchronous straight line ends from this position<table><tr><td>Unit</td><td>°</td></tr></table></td></tr>
<tr><td>uiSync</td><td>UINT</td><td>Synchronous factor
Ratio of output increments to input increments
Incline of synchronous straight line; uiSync := 100 corresponds to an incline of 1<table><tr><td>Range</td><td>100 ... 32767</td></tr><tr><td>Unit</td><td>%</td></tr></table></td></tr>
</table>

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boDone</td><td>BOOL</td><td>Response that the function block has been completely executed.</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state<table><tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr><tr><td>TRUE</td><td>Error</td></tr></table></td></tr>
</table>

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error
<table>
<tr><th>Value</th><th>Meaning</th></tr>
<tr><td>1</td><td>Incorrect number of elements
the maximum number is dependent on the table type 'enTabType'</td></tr>
<tr><td>2</td><td>Incorrect parameter set variant
dependent on the table type 'enTabKind'</td></tr>
<tr><td>3</td><td>'udMasterInc' value too high</td></tr>
<tr><td>4</td><td>'diOutInterv' value too high / too low</td></tr>
<tr><td>5</td><td>'diPar1' illegal value
dependent on 'enTabType' and 'enTabKind'</td></tr>
<tr><td>6</td><td>'diPar2' illegal value
dependent on 'enTabType' and 'enTabKind'</td></tr>
<tr><td>7</td><td>'diPar3' illegal value
dependent on 'enTabType' and 'enTabKind'</td></tr>
<tr><td>8</td><td>'diPar4' illegal value
dependent on 'enTabType' and 'enTabKind'</td></tr>
<tr><td>9</td><td>Illegal synchronous point</td></tr>
<tr><td>10</td><td>Illegal phasing in point</td></tr>
<tr><td>11</td><td>Illegal phasing out point</td></tr>
<tr><td>12</td><td>Illegal sine starting point</td></tr>
<tr><td>13</td><td>Velocity too low</td></tr>
<tr><td>14</td><td>Acceleration too low</td></tr>
</table></td></tr>
<tr><td>uiXOut</td><td>UINT</td><td>Phasing out point x coordinate
Phasing out ends with this position, transition from parabola to standstill
<table>
<tr><td>Range</td><td>0 ... 360</td></tr>
<tr><td>Unit</td><td>°</td></tr>
</table></td></tr>
</table>

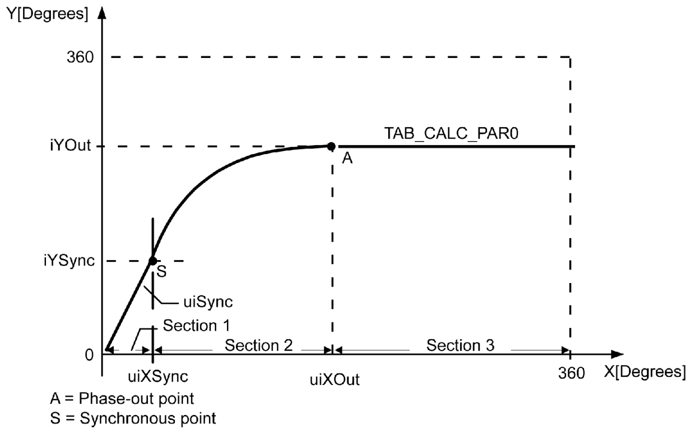
Input and output variables

Name	Type	Description
stDestTab	STRUCT	ST_PROF_TAB Profile table structure

Description

For the 'CALC_OUT_SYNCDEF' block, the phasing out curve is calculated with a parabola. The input parameters are the y coordinate of the phasing out point, the x coordinate of the synchronous point, and the synchronous factor. The table is calculated in the Y table format.

Abbildung 44: CALC_OUT_SYNCDEF: Phasing out function with defined synchronous points



Section 1	0 ≤ x < uiXSync	
Section 2	uiXSync ≤ x < uiXOut	Phasing out process: Parabola $y = -ax^2$
Section 3	uiXOut ≤ x < 360	

Calculation of the parabola

The factor a of the parabola $y = -ax^2$ is calculated from the equation y' = uiSync/100 = -2ax .
In the synchronous point S:

Therefore, it follows that for a:

where

The following dependencies and conditions apply:

-
-

7.4 Positioning profiles

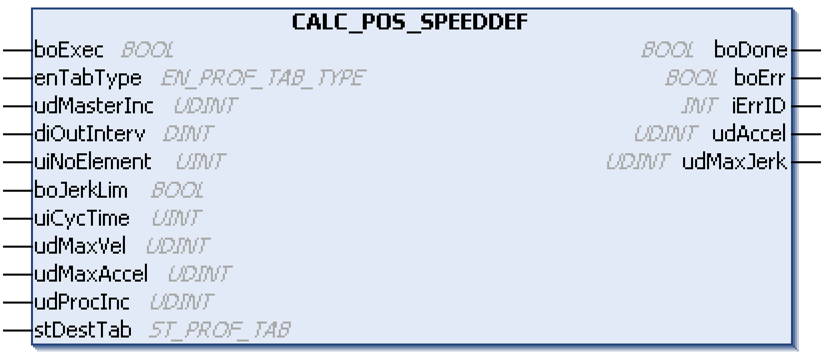
Positioning tables are used in conjunction with the 'CAM_PROF' block for a positioning operation with a defined travel profile.

CALC_POS_SPEEDDEF	Calculation of the positioning table based on the maximum velocity
CALC_POS_TIMEDEF	Calculation of the positioning table based on percentage acceleration

7.4.1 CALC_POS_SPEEDDEF (FB)

The 'CALC_POS_SPEEDDEF' function block calculates the positioning table. Positioning increments, positioning time, maximum acceleration, and maximum velocity are predefined.
'CALC_POS_SPEEDDEF' is based on the 'TAB_CALC' function block with 'enParSet' := TAB_CALC_PAR1

User interface



Input variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boExec</td><td>BOOL</td><td>Function execution: With a positive edge, the execution of the block starts.
As long as 'boExec' = TRUE, the block is processed by the PLC.
In the state 'boExec' = FALSE execution of the block is ended.</td></tr>
<tr><td>enTabType</td><td>ENUM</td><td>EN_PROF_TAB_TYPE
Table type, to differentiate between X and XY tables
<table>
<tr><td>Default</td><td>PROF_YTAB</td></tr>
</table>
<table>
<tr><th>Range</th><th>Meaning</th></tr>
<tr><td>PROF_YTAB</td><td>The y coordinate is saved as a table value,
the x coordinate must be equidistant.</td></tr>
<tr><td>PROF_YTAB_NL</td><td>ditto, the number of points is not limited</td></tr>
<tr><td>PROF_XYTAB</td><td>x and y coordinates are saved as table values</td></tr>
<tr><td>PROF_XYTAB_NL</td><td>ditto, the number of points is not limited</td></tr>
</table></td></tr>
<tr><td>udMasterInc</td><td>UDINT</td><td>Increments of the master drive which produce a table cycle
Max. table X value
<table>
<tr><td>Range</td><td>0 ... 5000000</td></tr>
<tr><td>Unit</td><td>incr</td></tr>
</table></td></tr>
<tr><td>diOutInterv</td><td>DINT</td><td>Output interface defining the output increments per table cycle
Max. table Y value
<table>
<tr><td>Unit</td><td>incr</td></tr>
</table></td></tr>
<tr><td>uiNoElement</td><td>UINT</td><td>Element number of the last table element calculated,
number of table interpolation points
<table>
<tr><td rowspan="2">Range</td><td>'enTabType' = PROF_YTAB; PROF_YTAB_NL:
5 ...</td></tr>
<tr><td>'enTabType' = PROF_XYTAB; PROF_XYTAB_NL:
5 ...</td></tr>
</table></td></tr>
<tr><td>boJerkLim</td><td>BOOL</td><td>Jerk limitation
<table>
<tr><td>FALSE</td><td>No jerk limitation; constant acceleration</td></tr>
<tr><td>TRUE</td><td>Constant jerk; linear increase in acceleration</td></tr>
</table></td></tr>
<tr><td>uiCycTime</td><td>UINT</td><td>Cycle time
Positioning takes place in this period of time, the master increments are input
<table>
<tr><td>Unit</td><td>ms</td></tr>
</table></td></tr>
<tr><td>udMaxVel</td><td>UDINT</td><td>Maximum velocity $|v_{max}|$
In the constant velocity range.
<table>
<tr><td>Range</td><td>1 … 65536000</td></tr>
<tr><td>Unit</td><td>0.0001 rpm</td></tr>
</table></td></tr>
<tr><td>udMaxAccel</td><td>UDINT</td><td>Maximum acceleration $|a_{max}|$
In the constant acceleration range
<table>
<tr><td>Range</td><td>1 … 65536000</td></tr>
<tr><td>Unit</td><td>0.001 rev/s^2</td></tr>
</table></td></tr>
<tr><td>udProcInc</td><td>UDINT</td><td>Process increments
Encoder resolution on process side y
<table>
<tr><td>Unit</td><td>incr/rev</td></tr>
</table></td></tr>
</table>

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boDone</td><td>BOOL</td><td>Response that the function block has been completely executed.</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table></td></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error
<table>
<tr><th>Value</th><th>Meaning</th></tr>
<tr><td>1</td><td>Incorrect number of elements
the maximum number is dependent on the table type 'enTabType'</td></tr>
<tr><td>2</td><td>Incorrect parameter set variant
dependent on the table type 'enTabKind'</td></tr>
<tr><td>3</td><td>'udMasterInc' value too high</td></tr>
<tr><td>4</td><td>'diOutInterv' value too high / too low</td></tr>
<tr><td>5</td><td>'diPar1' illegal value
dependent on 'enTabType' and 'enTabKind'</td></tr>
<tr><td>6</td><td>'diPar2' illegal value
dependent on 'enTabType' and 'enTabKind'</td></tr>
<tr><td>7</td><td>'diPar3' illegal value
dependent on 'enTabType' and 'enTabKind'</td></tr>
<tr><td>8</td><td>'diPar4' illegal value
dependent on 'enTabType' and 'enTabKind'</td></tr>
<tr><td>9</td><td>Illegal synchronous point</td></tr>
<tr><td>10</td><td>Illegal phasing in point</td></tr>
<tr><td>11</td><td>Illegal phasing out point</td></tr>
<tr><td>12</td><td>Illegal sine starting point</td></tr>
<tr><td>13</td><td>Velocity too low</td></tr>
<tr><td>14</td><td>Acceleration too low</td></tr>
</table></td></tr>
<tr><td>udAccel</td><td>UDINT</td><td>Acceleration phase
Acceleration as a proportion of the positioning cycle time
<table>
<tr><td>Unit</td><td>%</td></tr>
</table></td></tr>
<tr><td>udMaxJerk</td><td>UDINT</td><td>Maximum jerk during positioning
<table>
<tr><td>Unit</td><td>rev/s^3</td></tr>
</table></td></tr>
</table>

Input and output variables

Name	Type	Description
stDestTab	STRUCT	ST_PROF_TAB Profile table structure

Description

The 'CALC_POS_SPEEDDEF' function block supports positioning with a defined travel profile. The 'boJerkLim' can be set to select between positioning with jerk limitation and positioning without jerk limitation. A distinction can be made in the calculation between X and XY tables.

Abbildung 45: CALC_POS_SPEEDDEF: Positioning with jerk limitation

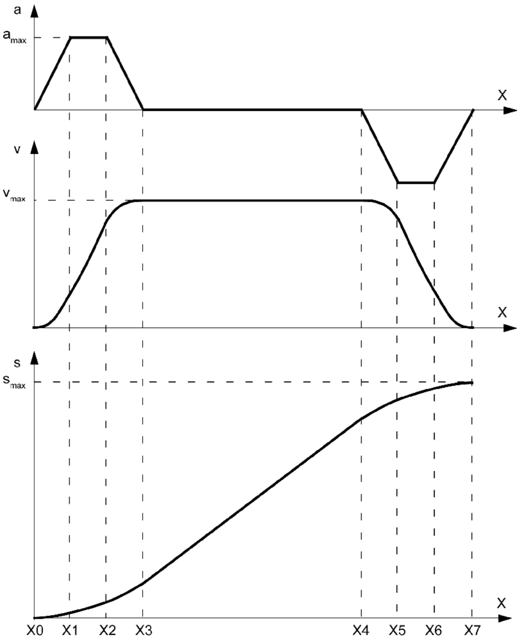
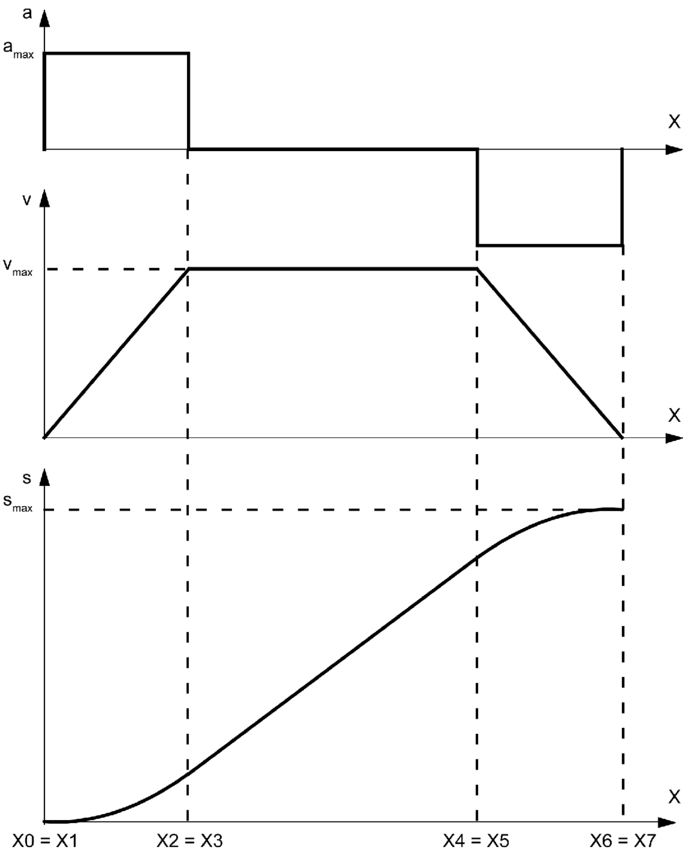


Abbildung 46: CALC_POS_SPEEDDEF: Positioning without jerk limitation



The following parameters must be specified prior to the table calculation:

- Encoder resolution of the axis to be traversed (udProcInd)
- Output interval s_{max} (diOutInterv)
- Cycle time T (uiCycTime)
- Maximum velocity v_{max} (udMaxVel)

So that the positioning operation can be completed in the specified cycle time, the following conditions apply for 'udMaxVel':

- Lower limit

  where udAccel = min = 1%
- Upper limit

  where udAccel = max = 50%

If 'udMaxVel' is lower than the permissible lower limit, the function block is terminated with an error message.
If the value is higher than the upper limit, 'udMaxVel' is set to the upper limit. In this case neither an error message nor a warning is output.

So that the acceleration can be completed in the specified cycle time, the following conditions apply for 'udMaxAccel':

- Lower limit
- Upper limit

Where the following applies for the acceleration time 'udAccel':

7.4.2 CALC_POS_TIMEDEF (FB)

The 'CALC_POS_TIMEDEF' function block calculates the positioning table based on the positioning increments, the positioning time, the max. acceleration, and the acceleration operation as a percentage of the positioning time.

User interface

Input variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boExec</td><td>BOOL</td><td>Function execution: With a positive edge, the execution of the block starts.
As long as 'boExec' = TRUE, the block is processed by the PLC.
In the state 'boExec' = FALSE execution of the block is ended.</td></tr>
<tr><td>enTabType</td><td>ENUM</td><td>EN_PROF_TAB_TYPE
Table type, to differentiate between X and XY tables
<table><tr><td>Default</td><td>PROF_YTAB</td></tr></table>
<table><tr><th>Range</th><th>Meaning</th></tr>
<tr><td>PROF_YTAB</td><td>The y coordinate is saved as a table value,
the x coordinate must be equidistant.</td></tr>
<tr><td>PROF_YTAB_NL</td><td>ditto, the number of points is not limited</td></tr>
<tr><td>PROF_XYTAB</td><td>x and y coordinates are saved as table values</td></tr>
<tr><td>PROF_XYTAB_NL</td><td>ditto, the number of points is not limited</td></tr></table></td></tr>
<tr><td>udMasterInc</td><td>UDINT</td><td>Increments of the master drive which produce a table cycle
Max. table X value
<table><tr><td>Range</td><td>0 ... 5000000</td></tr>
<tr><td>Unit</td><td>incr</td></tr></table></td></tr>
</table>

Name	Type	Description
diOutInterv	DINT	Output interface defining the output increments per table cycle Max. table Y value Unit: incr
uiNoElement	UINT	Element number of the last table element calculated, number of table interpolation points Range: 'enTabType' = PROF_YTAB; PROF_YTAB_NL: 5 ... 'enTabType' = PROF_XYTAB; PROF_XYTAB_NL: 5 ...
boJerkLim	BOOL	Jerk limitation FALSE: No jerk limitation; constant acceleration TRUE: Constant jerk; linear increase in acceleration
uiCycTime	UINT	Cycle time Positioning takes place in this period of time, the master increments are input Unit: ms
uiAccel	UINT	Acceleration phase Acceleration as a proportion of the positioning cycle time Unit: % Range: 1 … 50
udMaxAccel	UDINT	Maximum acceleration $\|a_{max}\|$ In the constant acceleration range Range: 1 … 65536000 Unit: 0.001 rev/s^2
udProcInc	UDINT	Process increments Encoder resolution on process side y Unit: incr/rev

Output variables

Name	Type	Description
boDone	BOOL	Response that the function block has been completely executed.
boErr	BOOL	The function block is in an error state FALSE: No error (permitted commanding or warning) TRUE: Error

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error
<table>
<tr><td>Value</td><td>Meaning</td></tr>
<tr><td>1</td><td>Incorrect number of elements
the maximum number is dependent on the table type 'enTabType'</td></tr>
<tr><td>2</td><td>Incorrect parameter set variant
dependent on the table type 'enTabKind'</td></tr>
<tr><td>3</td><td>'udMasterInc' value too high</td></tr>
<tr><td>4</td><td>'diOutInterv' value too high / too low</td></tr>
<tr><td>5</td><td>'diPar1' illegal value
dependent on 'enTabType' and 'enTabKind'</td></tr>
<tr><td>6</td><td>'diPar2' illegal value
dependent on 'enTabType' and 'enTabKind'</td></tr>
<tr><td>7</td><td>'diPar3' illegal value
dependent on 'enTabType' and 'enTabKind'</td></tr>
<tr><td>8</td><td>'diPar4' illegal value
dependent on 'enTabType' and 'enTabKind'</td></tr>
<tr><td>9</td><td>Illegal synchronous point</td></tr>
<tr><td>10</td><td>Illegal phasing in point</td></tr>
<tr><td>11</td><td>Illegal phasing out point</td></tr>
<tr><td>12</td><td>Illegal sine starting point</td></tr>
<tr><td>13</td><td>Velocity too low</td></tr>
<tr><td>14</td><td>Acceleration too low</td></tr>
</table>
</td></tr>
<tr><td>udMaxVel</td><td>UDINT</td><td>Maximum velocity $|v_{max}|$
In the constant velocity range.
<table><tr><td>Unit</td><td>0.0001 rpm</td></tr></table>
</td></tr>
<tr><td>udMaxJerk</td><td>UDINT</td><td>Maximum jerk during positioning
<table><tr><td>Unit</td><td>rev/s^3</td></tr></table>
</td></tr>
</table>

Input and output variables

Name	Type	Description
stDestTab	STRUCT	ST_PROF_TAB Profile table structure

Description

The 'CALC_POS_TIMEDEF' function block behaves in a similar way to the 'CALC_POS_SPEEDDEF' block.

The following parameters must be specified prior to the table calculation:

- Encoder resolution of the axis to be traversed (udProcInd)
- Output interval s_{max} (diOutInterv)
- Cycle time T (uiCycTime)
- Maximum acceleration (udMaxAccel)
- Acceleration phase (udAccel)

So that the acceleration can be completed in the specified cycle time, the following conditions apply for 'udMaxAccel':

- Lower limit
- Upper limit

Where the following applies for the acceleration time 'udMaxVel':

If 'udMaxAccel' is lower than the permissible lower limit, the function block is terminated with an error message.
If the value is higher than the upper limit, 'udMaxAccel' is set to the upper limit. In this case an error message is not output.

7.5 Support (support blocks)

Support blocks are used by the higher-level blocks in the library. They cannot be used directly by the user.

CALC_CHECK	Support block

7.5.1 CALC_CHECK (FB)

The 'CALC_CHECK' function block monitors the maximum permissible value for 'uiNoElement' based on the table type 'enTabType' and the table structure 'stDestTab'.

User interface

Input variables

Name	Type	Description
boExec	BOOL	Function execution: With a positive edge, the execution of the block starts. As long as 'boExec' = TRUE, the block is processed by the PLC. In the state 'boExec' = FALSE execution of the block is ended.
uiNoElement	UINT	Element number of the last table element calculated, number of table interpolation points Range: 'enTabType' = PROF_YTAB; PROF_YTAB_NL: 5 ... 'enTabType' = PROF_XYTAB; PROF_XYTAB_NL: 5 ...
enTabType	ENUM	EN_PROF_TAB_TYPE Table type, to differentiate between X and XY tables Default: PROF_YTAB Range / Meaning: PROF_YTAB: Y table PROF_YTAB_NL: Unlimited Y table PROF_XYTAB: XY table PROF_XYTAB_NL: Unlimited XY table

Output variables

Name	Type	Description
boDone	BOOL	Response that the function block has been completely executed.

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table></td></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error
<table>
<tr><td>Range</td><td>Meaning</td></tr>
<tr><td>1</td><td>Illegal value of uiNoElement</td></tr>
<tr><td>2</td><td>Illegal value of enTabType</td></tr>
</table></td></tr>
</table>

Input and output variables

Name	Type	Description
stTab	STRUCT	ST_PROF_TAB Profile table structure

Description

The function of the block is illustrated by the following IEC program:

```
FUNCTION_BLOCK CALC_CHECK
    VAR_INPUT
        boExec: BOOL;
        uiNoElement:UINT;
        enTabType:EN_PROF_TAB_TYPE;
    END_VAR
    VAR_OUTPUT
        boDone: BOOL;
        boErr: BOOL;
        iErrID: INT;
    END_VAR
    VAR_IN_OUT
        stTab: ST_PROF_TAB;
    END_VAR
    VAR
    END_VAR
```

```
IF boExec THEN
    CASE enTabType OF
        PROF_YTAB,PROF_YTAB_NL;
            IF uiNoElement>(SIZEOF(stTab)-8)/4-1 THEN
                    boErr:=TRUE;
                    iErrID:=1;
            ELSE
                    boDone:=TRUE;
            END_IF
        PROF_XYTAB,PROF_XYTAB_NL:
            IF uiNoElement>(SIZEOF(stTab)-8)/8-1 THEN
                    boErr:=TRUE;
                    iErrID:=1;
            ELSE
                    boDone:=TRUE;
            END_IF
        ELSE
            boErr:=TRUE;
            iErrID:=2; (* illegal enTabType; not supported by TAB_CALC *)
    END_CASE
ELSE
            boErr:=FALSE;
            boDone:=FALSE
            iErrID:=0;
END_IF
```

8 AmkCamEditor - Type definition specific to 3S

AmkCamEditor is an internal library which contains the listed type definitions which are specific to CamEditor. The definition of the data types specific to CamEditor is based on 3S libraries which are only integrated with the full Softmotion license. For this reason, the AMK CamEditor library contains copies of the structures required for the secondary function of the cam disk editor.

The library is, therefore, an absolute necessity when working with XYVA tables or the cam disk editor with polynomial tables.

Data types	Specific definitions
POUs	Specific table header information

8.1 Data types (specific definitions)

Cam profiles

ST_PROF_XYVATAB	XYVA table definition specific to AMK

Cam types

SMC_CAMXYVA	XYVA point information specific to 3S

8.1.1 CamProfile

8.1.1.1 ST_PROF_XYVATAB (ST)

The 'ST_PROF_XYVATAB' contains the table information specific to AMK and then references the 'SMC_CAMXYVA' structure which is specific to 3S and contains the interpolation points. (Siehe 'XYVA table' auf Seite 85.)

Structure elements

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>enType</td><td>ENUM</td><td>EN_PROF_TAB_TYPE
Table type
Designation for XYVA tables
<table><tr><td>Default</td><td>PROF_XYVATAB</td></tr></table>
<table><tr><td>Range</td><td>Meaning</td></tr><tr><td>PROF_
XYVATAB</td><td>X and Y positions, velocity, and acceleration defined by table values</td></tr></table></td></tr>
<tr><td>uiNoElement</td><td>UINT</td><td>Element number of the last table element calculated, number of table interpolation points of the 'SMC_CAMXYVA' structure
<table><tr><td>Range</td><td>8... MC_CAM_REF.nElements-1</td></tr><tr><td>Default</td><td>0</td></tr></table></td></tr>
<tr><td>udMasterInc</td><td>UDINT</td><td>Increments of the master drive which produce a table cycle
Max. table X value
(not used for XYVA tables)
<table><tr><td>Default</td><td>20000</td></tr></table></td></tr>
<tr><td>pstCamXYVA</td><td>POINTER</td><td>POINTER TO SMC_CAMXYVA
Pointer to the 'SMC_CAMXYVA' structure containing the dX, dY, dV, and dA values. The sections of the 5th order polynomial are defined with these values.</td></tr>
</table>

Structure definition

```
TYPE ST_PROF_XYVATAB:
        STRUCT
                enType:EN_PROF_TAB_TYPE:=PROF_XYVATAB;
                uiNoElement:UINT:=0;
                udMasterInc:UDINT:=20000;
                pstCamXYVA: POINTER TO SMC_CAMXYVA;
        END_STRUCT
END_TYPE
```

Based on the CODESYS cam disk editor, the 'stCam_A':ARRAY [0...3] OF SMC_CAMXYVA structure, which is specific to 3S, is generated automatically with table interpolation points and 'stCam':MC_CAM_REF.

Abbildung 47: ST_PROF_XYVATAB: Cam disk editor

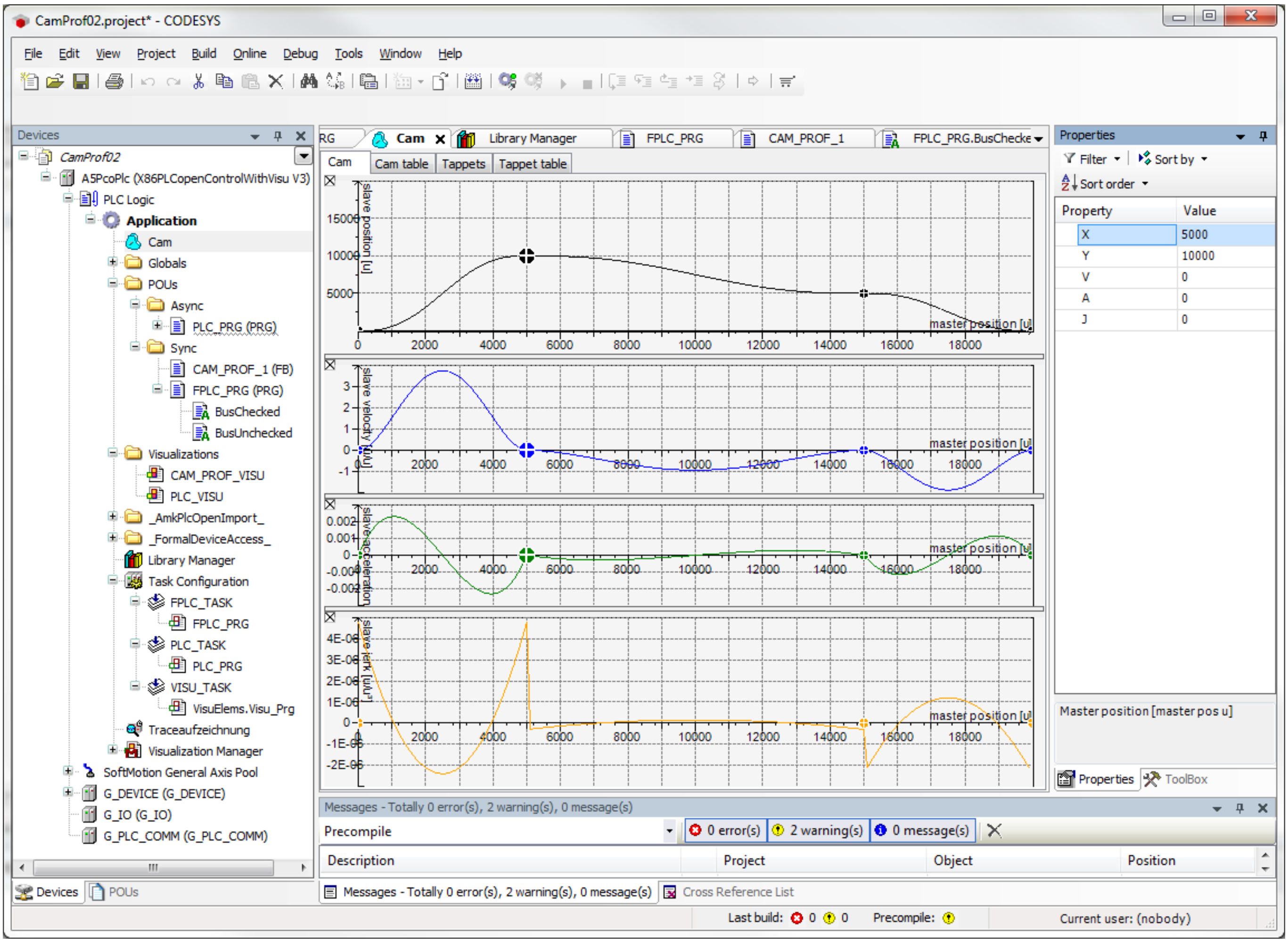
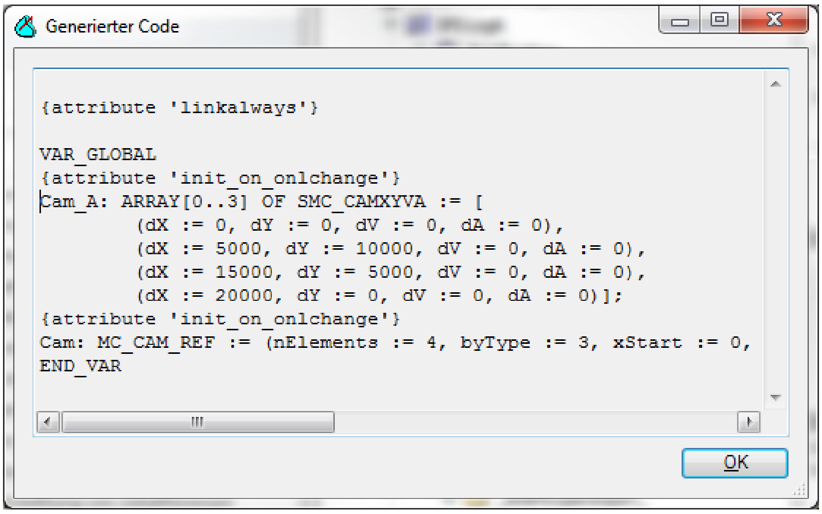


Abbildung 48: ST_PROF_XYVATAB: CAM structures specific to 3S



The definition of these structures is based on 3S libraries which are only integrated with the full Softmotion license.
AmkCamEditor is an AMK library which contains copies of the structures required for the table function of the cam disk editor. The AmkCamEditor library is, therefore, an absolute necessity when working with XYVA tables or the cam disk editor with polynomial tables!

8.1.2 CamTypes

8.1.2.1 SMC_CAMXYVA (ST)

The 'SMC_CAMXYVA' structure contains the type definitions specific to 3S which are needed by the CODESYS cam disk editor in order to write polynomial tables.
Moreover, the XYVA table supported by 'CAM_PROF' is based on an 'ARRAY OF SMC_CAMXYVA' which is referenced by a pointer in the context of the 'ST_PROF_XYVATAB' structure.

Structure elements

Name	Type	Description
dX	LREAL	x position, master
dY	LREAL	y position, slave
dV	LREAL	; slave velocity for a constant master velocity of 1
dA	LREAL	; slave acceleration for a constant master velocity of 1

Structure definition

```
TYPE SMC_CAMXYVA:
      STRUCT
            dX:LREAL;
            dY:LREAL;
            dV:LREAL;
            dA:LREAL;
      END_STRUCT
END_TYPE
```

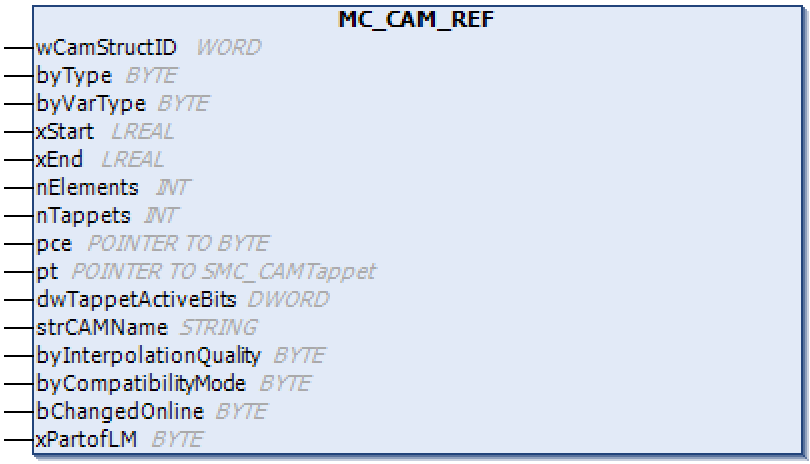
8.2 POUs (specific table header information)

MC_CAM_REF	Table header information specific to 3S

8.2.1 MC_CAM_REF (FB)

The 'MC_CAM_REF' function block contains the type definitions specific to 3S which are needed by the CODESYS cam disk editor in order to write polynomial tables.

User interface



Input variables

Name	Type	Description
byType	BYTE	Table type (Not used in the context of the specific function for AMK)
xStart	LREAL	Start of master position
xEnd	LREAL	End of master position
nElements	INT	Number of table elements is used, for example, by the 'CAMXYVA_TO_PROF' block to calculate 'uiNoElement' 'uiNoElement' = 'nElements' - 1
nTappets	INT	Number of cams (Not used in the context of the specific function for AMK)
strCAMName	STRING	CAM-Name (Not used in the context of the specific function for AMK)

9 AmkCom - Communication interface specific to AMK

AmkCom is an external library which contains communication blocks that are specific to AMK. It is divided into:

BasicFunctions	Basic functions
CommunicationProtocols	Communication protocols
DirectAccess	Direct access functions
ModbusExtensions	Modbus extensions

9.1 BasicFunctions

CLOSE_COM	Close serial communication interface
OPEN_COM	Initialize serial communication interface

9.1.1 CLOSE_COM (FB)

The 'CLOSE_COM' function block deactivates an active serial interface.

User interface

Input variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boExec</td><td>BOOL</td><td>Function execution: With a positive edge, the execution of the block starts.
As long as 'boExec' = TRUE, the block is processed by the PLC.
In the state 'boExec' = FALSE execution of the block is ended.</td></tr>
<tr><td>usComPort</td><td>USINT</td><td>Port selection
Differentiation between several serial interfaces
<table>
<tr><td>Range</td><td>Meaning</td></tr>
<tr><td>0</td><td>Serial default interface (Com2)</td></tr>
<tr><td>1</td><td>Serial interface 1 (Com1)</td></tr>
<tr><td>2</td><td>Serial interface 2 (Com2)</td></tr>
</table></td></tr>
</table>

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boDone</td><td>BOOL</td><td>Response that the function block has been completely executed.</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table></td></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error
<table>
<tr><td>Range</td><td>Meaning</td></tr>
<tr><td>1</td><td>Illegal value for 'usComPort'</td></tr>
<tr><td>-10</td><td>Serial interface assigned</td></tr>
</table></td></tr>
</table>

9.1.2 OPEN_COM (FB)

The 'OPEN_COM' function block activates the serial interface. The interface must be activated before it is accessed with one of the standard blocks from this library.

The following serial interfaces are supported:

- Virtual COM port (VCP)
  USB-serial converter RS232/RS485
- Communication device class, abstract control model (CDC-ACM)

User interface

Input variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boExec</td><td>BOOL</td><td>Function execution: With a positive edge, the execution of the block starts.
As long as 'boExec' = TRUE, the block is processed by the PLC.
In the state 'boExec' = FALSE execution of the block is ended.</td></tr>
<tr><td>usComPort</td><td>USINT</td><td>Port selection
Differentiation between several serial interfaces
<table>
<tr><th>Range</th><th>Meaning</th></tr>
<tr><td>11</td><td>1st virtual serial interface: VCP0 (A4/A5)</td></tr>
<tr><td>12</td><td>2nd virtual serial interface: VCP1 (A4/A5)</td></tr>
<tr><td>21</td><td>1st virtual serial interface: CDC-ACM (A4/A5)</td></tr>
<tr><td>22</td><td>2nd virtual serial interface: CDC-ACM (A4/A5)</td></tr>
</table></td></tr>
<tr><td>enMode</td><td>ENUM</td><td>EN_COM_MODE
Selection mode Communication
Differentiation between RS422, RS485, or RS232 modes
'enMode' is not relevant in the context of the USB-serial converter.
The mode is determined by the type of converter.
<table>
<tr><td>Default</td><td>RS422</td></tr>
</table>
<table>
<tr><th>Range</th><th>Meaning</th></tr>
<tr><td>RS422</td><td>Serial point-to-point interface, two pairs of conductors.</td></tr>
<tr><td>RS485</td><td>Serial bus interface, one pair of conductors</td></tr>
<tr><td>RS232</td><td>Serial point-to-point interface, two conductors and ground in accordance with RS232 standard</td></tr>
</table></td></tr>
<tr><td>stComSet</td><td>STRUCT</td><td>ST_COM_SET
Parameter structure
Setting of the interface parameters</td></tr>
</table>

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boDone</td><td>BOOL</td><td>Response that the function block has been completely executed.</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table></td></tr>
</table>

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error
<table>
<tr><th>Range</th><th>Meaning</th></tr>
<tr><td>1</td><td>Illegal port selection</td></tr>
<tr><td>2</td><td>Illegal communication mode</td></tr>
<tr><td>3</td><td>Illegal baud rate</td></tr>
<tr><td>4</td><td>Illegal number of stop bits</td></tr>
<tr><td>5</td><td>Illegal number of data bits</td></tr>
<tr><td>6</td><td>Parity bit monitoring required</td></tr>
<tr><td>7</td><td>Illegal parity bit monitoring</td></tr>
<tr><td>-10</td><td>Serial interface assigned</td></tr>
</table>
</td></tr>
</table>

9.2 CommunicationProtocols

MODBUS	Transfer of information via Modbus protocol

9.2.1 MODBUS (FB)

The 'MODBUS' function block implements a subset of the Modbus slave function. It facilitates communication, for example, with operator panels or other devices with a compatible Modbus master function.

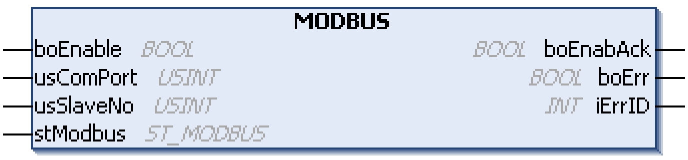
Tabelle 4: MODBUS: Supported Modbus function codes

Function code	Message frame designation
16#01 / 16#02	Read n bits
16#03 / 16#04	Read n words
16#05	Write 1 bit
16#06	Write 1 word
16#0F	Write n bits
16#10	Write n words

Variables that are defined in operator panels, for example, are mapped in the 'stModbus' structure:

- An RS422 (or RS232) interface can be selected to implement a point-to-point connection.
- An RS485 interface must be selected to implement a bus connection.

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.

Name	Type	Description
usComPort	USINT	Port selection Differentiation between several serial interfaces <table><tr><th>Range</th><th>Meaning</th></tr><tr><td>11</td><td>1st virtual serial interface: VCP0 (A4/A5)</td></tr><tr><td>12</td><td>2nd virtual serial interface: VCP1 (A4/A5)</td></tr><tr><td>21</td><td>1st virtual serial interface: CDC-ACM (A4/A5)</td></tr><tr><td>22</td><td>2nd virtual serial interface: CDC-ACM (A4/A5)</td></tr></table>
usSlaveNo	USINT	Modbus slave address <table><tr><td>Range</td><td>1 ... 32</td></tr><tr><td>Default</td><td>1</td></tr></table>

Output variables

Name	Type	Description
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled
boErr	BOOL	The function block is in an error state <table><tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr><tr><td>TRUE</td><td>Error</td></tr></table>
iErrID	INT	Error identity number: Diagnostic number is output <table><tr><td colspan="2">iErrID = 0</td><td>No error</td></tr><tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr><tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr></table>Error <table><tr><th>Range</th><th>Meaning</th></tr><tr><td>1</td><td>Serial interface not activated</td></tr><tr><td>2</td><td>Illegal port selection</td></tr><tr><td>3</td><td>Multiple instancing of the Modbus protocol</td></tr><tr><td>4</td><td>Illegal slave address</td></tr><tr><td>5</td><td>CRC error (CRC = cyclic redundancy check)</td></tr><tr><td>6</td><td>Send buffer full</td></tr><tr><td>7</td><td>Illegal data length in "read n words" message frame</td></tr><tr><td>8</td><td>Illegal data length in "write n words" message frame</td></tr><tr><td>9</td><td>Illegal data length in "read n bits" message frame</td></tr><tr><td>10</td><td>Illegal data length in "write 1 bit" message frame</td></tr><tr><td>11</td><td>Illegal address value in "read n words" message frame</td></tr><tr><td>12</td><td>Illegal address value in "write n words" message frame</td></tr><tr><td>13</td><td>Illegal address value in "read n bits" message frame</td></tr><tr><td>14</td><td>Illegal address value in "write 1 bit" message frame</td></tr><tr><td>15</td><td>Illegal function code</td></tr><tr><td>16</td><td>Receive buffer overrun</td></tr><tr><td>-10</td><td>Serial interface assigned</td></tr></table>

Input and output variables

Name	Type	Description
stModbus	STRUCT	ST_MODBUS Modbus structure Information exchange memory

9.3 DirectAccess

READ_COM	Receive characters via serial communication
WRITE_COM	Send characters via serial communication

9.3.1 READ_COM (FB)

The 'READ_COM' function block transfers characters to the communication buffer which are received via the serial interface. Before receiving can commence, the serial interface must be activated.

Once the read operation is underway, the characters received are written to the communication buffer until

- a defined number of characters has been received
- a defined end-of-text character has been read
- a configurable timeout has elapsed

The 'boExec' signal must remain set to TRUE until 'boDone' or 'boErr' indicates that the read operation is complete.

User interface

Input variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boExec</td><td>BOOL</td><td>Function execution: With a positive edge, the execution of the block starts.
As long as 'boExec' = TRUE, the block is processed by the PLC.
In the state 'boExec' = FALSE execution of the block is ended.</td></tr>
<tr><td>usComPort</td><td>USINT</td><td>Port selection
Differentiation between several serial interfaces
<table>
<tr><th>Range</th><th>Meaning</th></tr>
<tr><td>11</td><td>1st virtual serial interface: VCP0 (A4/A5)</td></tr>
<tr><td>12</td><td>2nd virtual serial interface: VCP1 (A4/A5)</td></tr>
<tr><td>21</td><td>1st virtual serial interface: CDC-ACM (A4/A5)</td></tr>
<tr><td>22</td><td>2nd virtual serial interface: CDC-ACM (A4/A5)</td></tr>
</table></td></tr>
<tr><td>uiMaxBytes</td><td>UINT</td><td>Maximum number of bytes
The read operation ends after a definable number of incoming characters
<table>
<tr><td>Range</td><td>0 ... 100</td></tr>
<tr><td>Default</td><td>100</td></tr>
</table></td></tr>
<tr><td>usEOT</td><td>USINT</td><td>EndOfText character
The read operation ends after the definable number of characters
(ASCII character, decimal representation)
<table>
<tr><td>Range</td><td>0
1 ... 255</td><td>No EOT monitoring
End read operation on receipt of 'usEOT'. The EOT character is not written to the communication buffer.</td></tr>
<tr><td>Default</td><td>13</td><td>ASCII control character for line return,
CR - carriage return</td></tr>
</table></td></tr>
</table>

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>tTimeOut</td><td>TIME</td><td>Timeout
The read operation ends after the timeout has elapsed<table>
<tr><td>Range</td><td>0 No timeout monitoring
1 ... 65535 Timeout time</td></tr>
<tr><td>Unit</td><td>ms</td></tr>
<tr><td>Default</td><td>5000 (t#5s)</td></tr>
</table></td></tr>
</table>

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boDone</td><td>BOOL</td><td>Response that the function block has been completely executed.</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table></td></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>Error<table>
<tr><th>Range</th><th>Meaning</th></tr>
<tr><td>1</td><td>Serial interface not activated</td></tr>
<tr><td>2</td><td>Illegal port selection</td></tr>
<tr><td>3</td><td>Illegal maximum number of bytes</td></tr>
<tr><td>4</td><td>Timeout has elapsed</td></tr>
<tr><td>5</td><td>Receive buffer full</td></tr>
<tr><td>6</td><td>Parity error in interface block</td></tr>
<tr><td>7</td><td>Framing error in interface block</td></tr>
<tr><td>8</td><td>Overrun error in interface block</td></tr>
<tr><td>-10</td><td>Serial interface assigned</td></tr>
</table></td></tr>
<tr><td>uiNoByte</td><td>UINT</td><td>Byte index of the communication buffer up to which data is being received<table>
<tr><td>Range</td><td>0 ... 99</td></tr>
</table></td></tr>
</table>

Input and output variables

Name	Type	Description
stComBuff	STRUCT	ST_COM_BUFF Communication buffer Buffers the characters received

9.3.2 WRITE_COM (FB)

The 'WRITE_COM' function block sends characters to the communication buffer via the serial interface.
Before sending can commence, the serial interface must be activated.

Once the send operation is underway, the characters received are sent from the communication buffer until

- a defined number of characters has been sent.

The 'boExec' signal must remain active (set to TRUE) until 'boDone' or 'boErr' indicates that the send operation is complete.

User interface

Input variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boExec</td><td>BOOL</td><td>Function execution: With a positive edge, the execution of the block starts.
As long as 'boExec' = TRUE, the block is processed by the PLC.
In the state 'boExec' = FALSE execution of the block is ended.</td></tr>
<tr><td>usComPort</td><td>USINT</td><td>Port selection
Differentiation between several serial interfaces
<table>
<tr><td>Range</td><td>Meaning</td></tr>
<tr><td>11</td><td>1st virtual serial interface: VCP0 (A4/A5)</td></tr>
<tr><td>12</td><td>2nd virtual serial interface: VCP1 (A4/A5)</td></tr>
<tr><td>21</td><td>1st virtual serial interface: CDC-ACM (A4/A5)</td></tr>
<tr><td>22</td><td>2nd virtual serial interface: CDC-ACM (A4/A5)</td></tr>
</table></td></tr>
<tr><td>uiNoByte</td><td>UINT</td><td>Byte index of the communication buffer up to which data is being sent
<table>
<tr><td>Range</td><td>0 ... 99</td></tr>
</table></td></tr>
</table>

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boDone</td><td>BOOL</td><td>Response that the function block has been completely executed.</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table></td></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error
<table>
<tr><td>Range</td><td>Meaning</td></tr>
<tr><td>1</td><td>Serial interface not activated</td></tr>
<tr><td>2</td><td>Illegal port selection</td></tr>
<tr><td>3</td><td>Illegal number of bytes (max. communication buffer index)</td></tr>
<tr><td>-10</td><td>Serial interface assigned</td></tr>
</table></td></tr>
</table>

Input and output variables

Name	Type	Description
stComBuff	STRUCT	ST_COM_BUFF Communication buffer Buffers the characters received

9.4 ModbusExtensions

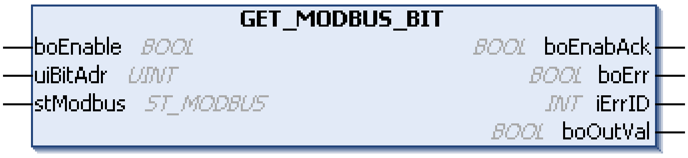
GET_MODBUS_BIT	Extract bit from 'stModbus' structure
SET_MODBUS_BIT	Add bit to 'stModbus' structure

9.4.1 GET_MODBUS_BIT (FB)

The 'GET_MODBUS_BIT' function block reads a single bit from the bit block of the Modbus structure.

The address of the Modbus variable can be set in advance; the position of the bit block does not have to be known in order to do this.

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.
uiBitAdr	UINT	Bit address Addressing of Modbus bit variables Range: 0 ... 511 (corresp. bit 0 ... bit 511)

Output variables

Name	Type	Description
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled
boErr	BOOL	The function block is in an error state FALSE: No error (permitted commanding or warning) TRUE: Error
iErrID	INT	Error identity number: Diagnostic number is output iErrID = 0: No error iErrID ≠ 0, boErr = TRUE: Error iErrID ≠ 0, boErr = FALSE: Warning Error Range: Meaning 1: Illegal bit address
boOutVal	BOOL	Output value Binary value of the Modbus bit variable in 'stModbus' according to the bit address

Input and output variables

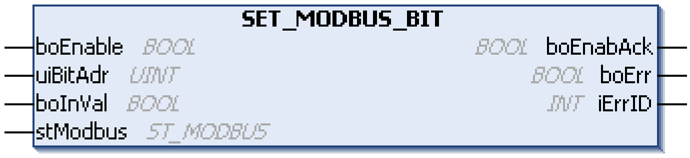
Name	Type	Description
stModbus	STRUCT	ST_MODBUS Modbus structure Information exchange memory

9.4.2 SET_MODBUS_BIT (FB)

The 'SET_MODBUS_BIT' function block writes a single bit to the bit block of the Modbus structure.

The address of the Modbus variable can be set in advance; the position of the bit block does not have to be known in order to do this.

User interface



Input variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnable</td><td>BOOL</td><td>Enable signal: With a positive edge, the initialisation of the block starts.
As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC.
In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.</td></tr>
<tr><td>uiBitAdr</td><td>UINT</td><td>Bit address
Addressing of Modbus bit variables<table><tr><td>Range</td><td>0 ... 511 (corresp. bit 0 ... bit 511)</td></tr></table></td></tr>
<tr><td>boInVal</td><td>BOOL</td><td>Input value
Binary value that is written to the Modbus bit variable in 'stModbus' according to the bit address</td></tr>
</table>

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state<table><tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr><tr><td>TRUE</td><td>Error</td></tr></table></td></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output<table><tr><td colspan="2">iErrID = 0</td><td>No error</td></tr><tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr><tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr></table>Error<table><tr><td>Range</td><td>Meaning</td></tr><tr><td>1</td><td>Illegal bit address</td></tr></table></td></tr>
</table>

Input and output variables

Name	Type	Description
stModbus	STRUCT	ST_MODBUS Modbus structure Information exchange memory

9.5 DataTypes

9.5.1 Structures

9.5.1.1 ST_COM_BUFF (ST)

The characters received or to be sent are stored in the 'ST_COM_BUFF' structure.

Structure elements

Name	Type	Description
usByte[0]	USINT	READ_COM: character received WRITE_COM: character to be sent
...		
usByte[99]	USINT	READ_COM: character received WRITE_COM: character to be sent

Structure definition

```
MAX_BYTE_IND: UINT:=99;                    (* Maximum index*)

TYPE ST_COM_BUFF:
    STRUCT
        usByte:ARRAY[0 … MAX_BYTE_IND] OF USINT;
    END_STRUCT
END_TYPE
```

9.5.1.2 ST_COM_SET (ST)

The 'ST_COM_SET' structure defines the parameters used for the activation of the interface.

Structure elements

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>uiBaudRate</td><td>UINT</td><td>Transfer rate
<table>
<tr><td>Range</td><td>1200, 2400, 4800, 9600, 19200, 38400, 57600</td></tr>
<tr><td>Unit</td><td>bit/s</td></tr>
<tr><td>Default</td><td>9600</td></tr>
</table></td></tr>
<tr><td>enParity</td><td>ENUM</td><td>EN_COM_PARITY
Parity
<table>
<tr><td>Default</td><td>PARITY_NO</td></tr>
</table>
<table>
<tr><th>Range</th><th>Meaning</th></tr>
<tr><td>PARITY_NO</td><td>No monitoring of the parity bit</td></tr>
<tr><td>PARITY_ODD</td><td>Odd monitoring of the parity bit</td></tr>
<tr><td>PARITY_EVEN</td><td>Even monitoring of the parity bit</td></tr>
</table></td></tr>
<tr><td>usDataBits</td><td>USINT</td><td>Number of data bits
For 'usDataBits' = 7, 'enParity' must = PARITY_ODD or PARITY_EVEN must be set.
<table>
<tr><td>Range</td><td>7, 8</td></tr>
<tr><td>Default</td><td>8</td></tr>
</table></td></tr>
<tr><td>usStopBits</td><td>USINT</td><td>Number of stop bits
<table>
<tr><td>Range</td><td>1, 2</td></tr>
<tr><td>Default</td><td>1</td></tr>
</table></td></tr>
</table>

Structure definition

```
TYPE ST_COM_SET:
        STRUCT
                uiBaudRate:UNIT;
                enParity:EN_COM_PARITY;
                usDataBits:USINT;
                usStopBits:USINT;
        END_STRUCT
END_TYPE
```

9.5.1.3 ST_MODBUS (ST)

The 'ST_MODBUS' structure creates a communication buffer for the variables content transferred via the Modbus protocol.

- Word or double-word variables are mapped in the word register block (uiRegBlock[0] ... uiRegBlock[255]).
- Bit variables are mapped in the bit block (byBitBlock[0] ... byBitBlock[63].

The 'GET_MODBUS_BIT' and 'SET_MODBUS_BIT' blocks support read and write access to single items of binary information in the bit block.

Structure elements

Name	Type	Description
uiRegBlock[0]	UINT	Word 0
uiRegBlock[1]	UINT	Word 1
...		...
uiRegBlock[255]	UINT	Word 255
byBitBlock[0]	BYTE	Bit0 ... Bit7
byBitBlock[1]	BYTE	Bit8 … Bit15
...		
byBitBlock[63]	BYTE	Bit504 … Bit511

Structure definition

```
MAX_REG_IND:UNIT:=255;          (* max. index of the Modbus register *)
MAX_BIT_IND:UNIT:=63;           (* max. index of the bits (in bytes) *)

TYPE ST_MODBUS:
        STRUCT
                uiRegBlock:ARRAY[0 … MAX_REG_IND] OF UNIT;
                uiBitBlock:ARRAY[0 … MAX_BIT_IND] OF BYTE;
        END_STRUCT
END_TYPE
```

9.5.1.4 ST_REC_TEL (ST)

The 'ST_REC_TEL' structure is a support structure for receiving message frames.

Structure elements

Name	Type	Description
usRecTelChar	ARRAY	ARRAY [0..MAX_CHAR_IND] OF USINT Characters received
pusActRecChar	POINTER	POINTER_TO_USINT Pointer to the current character
usTimeCount	USINT	Elapsed-time meter

Name	Type	Description
ub_MaxTime	USINT	Maximum time

Structure definition

```
MAX_CHAR_IND: UINT := 211                    (* maximum index *)

TYPE ST_REC_TEL:
    STRUCT
        usRecTelChar: ARRAY [0..MAX_CHAR_IND] OF USINT;
        pusActRecChar: POINTER TO USINT;
        usTimeCount: USINT;
        ub_MaxTime: USINT;
    END_STRUCT
END_TYPE
```

9.5.1.5 ST_TRANS_TEL (ST)

The 'ST_TRANS_TEL' structure is a support structure for sending message frames.

Structure elements

Name	Type	Description
usTransTelChar	ARRAY	ARRAY [0..MAX_CHAR_IND] OF USINT Characters sent
pusActTransChar	POINTER	POINTER_TO_USINT Pointer to the current character
uiTransCharNmb	UINT	Number of characters sent

Structure definition

```
MAX_CHAR_IND: UINT := 211                    (* maximum index *)

TYPE ST_TRANS_TEL:
    STRUCT
        usTransTelChar: ARRAY [0..MAX_CHAR_IND] OF USINT;
        pusActTransChar: POINTER TO USINT;
        uiTransCharNmb: UINT;
    END_STRUCT
END_TYPE
```

10 AmkPmc - Printing mark control specific to AMK

AmkPmc is an internal library which contains blocks that support printing mark control and are specific to AMK. It is divided into:

AdditionalFunctions	Additional functions
BasicFunctions	Basic functions
ExtendedFunctions	Extended functions

10.1 AdditionalFunctions

FudPmcSetVal	Calculation of the setpoint position for printing mark control
GET_CORR_VAL	Displays correction value
REF_RESET	Automatic homing to the printing mark
SET_OFFSET	Offset setting

10.1.1 FudPmcSetVal (F)

The FudPmcSetVal function calculates the setpoint position for the printing mark to be used for the remaining PMC blocks based on the 'udPmsDist' and 'udModulo' variables, the values of which are determined by the technology. Siehe 'Description' auf Seite 243.

The calculation also applies in particular if 'udPmsDist' > 'udModulo'; in other words, if the printing mark sensor is installed several formats upstream of the tool.

User interface

Input variables

Name	Type	Description
udPmsDist	UDINT	Distance between the printing mark sensor and the engagement point of the tool
udModulo	UDINT	Modulo format Describes the setpoint distance between two consecutive printing marks. The value is saved when the block is activated (positive edge at 'boEnable'). A subsequent change does not affect the active block.

Output variables

Name	Type	Description
FudPmcSetVal	UDINT	Printing mark setpoint Expected distance between printing mark and printing mark sensor

10.1.2 GET_CORR_VAL (FB)

The 'GET_CORR_VAL' function block displays the last correction value entered in the FIFO structure by the 'PM_DETECT' function block.

User interface

Output variables

Name	Type	Description
diCorrVal	DINT	Current correction value enter most recently in the FIFO structure 'stCorrFifo' • For stCorrFifo.uiInIndex = 0: diCorrVal := stCorrFifo.diCorrVal[MAX_CORR_FIFO_IND] • For stCorrFifo.uiInIndex ≠ 0: diCorrVal := stCorrFifo.diCorrVal[stCorrFifo.uiInIndex – 1]

Input and output variables

Name	Type	Description
stCorrFifo	STRUCT	ST_CORR_FIFO Correction value FIFO Transfer of correction values detected by 'PM_DETECT' to the 'PM_CORRECT' block

10.1.3 REF_RESET (FB)

The 'REF_RESET' function block triggers automatic homing of the 'PM_DETECT' and 'PMC_BASE' blocks.

User interface

Input variables

Name	Type	Description
boPmCapt	BOOL	Printing mark detected Signal indicating that a printing mark has been detected inside the permissible range. 'boPmCapt' is TRUE for one cycle only. At the same time, the calculated correction value is entered in the FIFO correction value.
boPmMiss	BOOL	Printing mark missing Signal indicating that a mark has not been detected inside the permissible range. 'boPmMiss' is TRUE for one cycle only. At the same time, the value "0" is entered in the FIFO correction value.
usPmMissNo	USINT	Number of missing printing marks until reset Maximum number of consecutive undetected printing marks until a homing cycle is started (boRefStart = TRUE) 'usPmMissNo' = 0: block inactive The input only applies in conjunction with enMode = DETECT_AUTO. Default: 5

Output variables

Name	Type	Description
boRefStart	BOOL	Start of a new homing cycle; alignment with next printing mark Only applies in conjunction with 'enMode' = DETECT_AUTO. When the block is activated in 'DETECT_AUTO' mode, a homing cycle is started without 'boRefStart' being evaluated. A positive edge change at 'boRefStart' triggers repeat homing without the block having to be reactivated (positive edge at 'boEnable').

10.1.4 SET_OFFSET (FB)

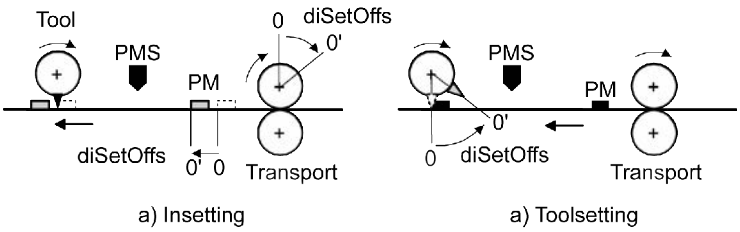
The 'SET_OFFSET' function block outputs an offset value that is proportional to the velocity.

In the context of printing mark control, the block is used for "soft" adjustment of the offset between printing mark position and tool reference point.

The 'diSetOffs' setpoint is broken down into component parts which are proportional to the velocity and output at 'diOffset' across multiple sampling points.

The offset velocity is proportional to the change in the input variable ('diInVal'(k)- 'diInVal'(k-1)). It can be changed online with 'uiOvrOffs'.

Abbildung 49: SET_OFFSET: Offset setting, principle of operation of 'diSetOffs'



Independent of the correction method (insetting / tool setting), a positive offset always means a positive shift in the printing mark, i.e. upstream of the tool setting point.

User interface

Input variables

Name	Type	Description
diInVal	DINT	Input value Determination of the output velocity of the setpoint offset
diSetOffs	DINT	Setpoint offset Is output after several sampling time points at 'diOffset'
uiOvrOffs	UINT	Offset override Set a velocity override on interpolation of the offset Range: 0 ... 100 Unit: % Default: 10

Output variables

Name	Type	Description
diOffset	DINT	Offset of the counter value to the homing pulse Unit: Incr The aim is 'diOffset' = 'diSetOffs'. However, in this context, the change in 'diOffset' is only made in increments of up to per sampling cycle.

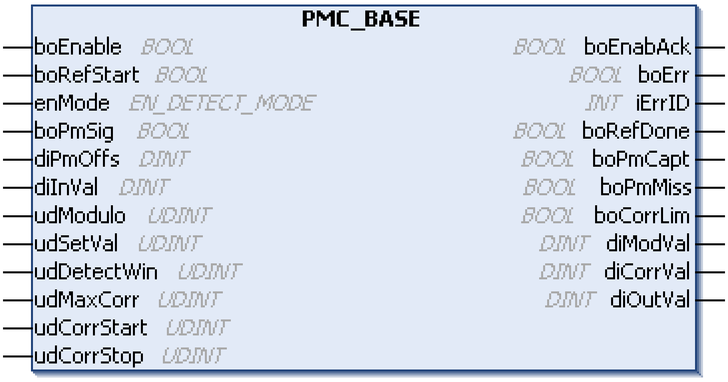
10.2 BasicFunctions

PMC_BASE	Base block for printing mark control

10.2.1 PMC_BASE (FB)

The 'PMC_BASE' function block serves basic printing mark control.
Using the 'stCorrFifo' FIFO, it combines the 'PM_DETECT' and 'PM_CORRECT' function blocks of the AmkBase library.

User interface



Input variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnable</td><td>BOOL</td><td>Enable signal: With a positive edge, the initialisation of the block starts.
As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC.
In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.</td></tr>
<tr><td>boRefStart</td><td>BOOL</td><td>Start of a new homing cycle; alignment with next printing mark
Only applies in conjunction with 'enMode' = DETECT_AUTO.
When the block is activated in 'DETECT_AUTO' mode, a homing cycle is started without 'boRefStart' being evaluated.
A positive edge change at 'boRefStart' triggers repeat homing without the block having to be reactivated (positive edge at 'boEnable').</td></tr>
<tr><td>enMode</td><td>ENUM</td><td>EN_DETECT_MODE
Selection mode operating mode
<table>
<tr><td>Default</td><td>DETECT_AUTO</td></tr>
</table>
<table>
<tr><th>Range</th><th>Meaning</th></tr>
<tr><td>DETECT_AUTO</td><td>Automatic homing with reference to the first printing mark</td></tr>
<tr><td>DETECT_
MANUAL</td><td>Manual homing
The printing mark must be aligned manually with the sensor prior to enabling with 'boEnable'</td></tr>
</table></td></tr>
<tr><td>boPmSig</td><td>BOOL</td><td>Printing mark signal (PM signal)
Signal indicating a printing mark inside the modulo format
(The signal must remain pending for at least 1 sampling time)</td></tr>
<tr><td>diPmOffs</td><td>DINT</td><td>Printing mark offset (PM offset)
Describes the deviation between the time-discrete input value 'diInVal' (kT0) and the actual input value 'diInVal'(TboPmSig) at the time of the edge change on the printing mark signal
The following applies:</td></tr>
<tr><td>diInVal</td><td>DINT</td><td>Input value</td></tr>
</table>

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>udModulo</td><td>UDINT</td><td>Modulo format
Describes the setpoint distance between two consecutive printing marks.
The value is saved when the block is activated (positive edge at 'boEnable'). A subsequent change does not affect the active block.<table><tr><td>Range</td><td>0 ... 1000000000</td></tr><tr><td>Default</td><td>20000</td></tr></table></td></tr>
<tr><td>udSetVal</td><td>UDINT</td><td>PM setpoint
Describes the expected distance between printing mark and printing mark sensor.
The value is saved when the block is activated (positive edge at 'boEnable'). A subsequent change does not affect the active block.
If udSetVal ≥ udModulo, n correction values "0" are entered in 'stCorrFifo'.

This corresponds to a slip in the correction value of n formats. In other words, the mark sensor is positioned n formats upstream of the tool position.<table><tr><td>Default</td><td>10000</td></tr></table></td></tr>
<tr><td>udDetectWin</td><td>UDINT</td><td>Permissible range
A 'boPmSig' flag signal is permitted within this range. The value can be changed online when the block is active<table><tr><td>Range</td><td>0 ... 999999999</td></tr><tr><td>Default</td><td>5000</td></tr></table></td></tr>
<tr><td>udMaxCorr</td><td>UDINT</td><td>Maximum permissible correction value to which the correction value output per modulo format is limited.
The value can be changed when the block is active<table><tr><td>Range</td><td>0 ... 999999999</td></tr><tr><td>Default</td><td>2000</td></tr></table></td></tr>
<tr><td>udCorrStart</td><td>UDINT</td><td>Correction starting value at which the output of correction values commences.
The value can be changed when the block is active<table><tr><td>Range</td><td>0 ... 999999999</td></tr><tr><td>Default</td><td>15000</td></tr></table></td></tr>
<tr><td>udCorrStop</td><td>UDINT</td><td>Correction stop value at which the output of correction values ends.
The value can be changed when the block is active<table><tr><td>Range</td><td>0 ... 999999999</td></tr><tr><td>Default</td><td>19999</td></tr></table></td></tr>
</table>

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state<table><tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr><tr><td>TRUE</td><td>Error</td></tr></table></td></tr>
</table>

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error
<table>
<tr><th>Range</th><th>Meaning</th></tr>
<tr><td>1 ... 9</td><td>Warnings and errors associated with the 'PM_DETECT' block</td></tr>
<tr><td>10 ... 19</td><td>Warnings and errors associated with the 'PM_CORRECT' block
with an offset of 10</td></tr>
<tr><td>20</td><td>Illegal value of 'udSetVal' or 'udDetectWin'
The ranges of the detection window and the correction range 'udCorrStart' ... 'udCorrStop' overlap.</td></tr>
</table></td></tr>
<tr><td>boRefDone</td><td>BOOL</td><td>Homing cycle completed
Acknowledgement signal to indicate that a homing cycle has been completed.
In mode 'enMode' = DETECT_AUTO, 'boRefDone' = FALSE when the block is activated or on a positive edge change at 'boRefStart'.
Once the first mark has been detected, 'boRefDone' = TRUE is set.
In mode 'enMode' = DETECT_MANUAL, this variable is of no significance. 'boRefDone' = TRUE always applies.</td></tr>
<tr><td>boPmCapt</td><td>BOOL</td><td>Printing mark detected
Signal indicating that a printing mark has been detected inside the permissible range.
'boPmCapt' is TRUE for one cycle only.
At the same time, the calculated correction value is entered in the FIFO correction value.</td></tr>
<tr><td>boPmMiss</td><td>BOOL</td><td>Printing mark missing
Signal indicating that a mark has not been detected inside the permissible range.
'boPmMiss' is TRUE for one cycle only.
At the same time, the value "0" is entered in the FIFO correction value.</td></tr>
<tr><td>boCorrLim</td><td>BOOL</td><td>Correction limiting
Display a limit of the correction value according to 'udMaxCorr'.
The variable is set to true for one cycle after 'udCorrStart' and before 'udCorrStop'</td></tr>
<tr><td>diModVal</td><td>DINT</td><td>Modulo value
Displays the current modulo position (0 ≤ diModVal < udModulo).
The sign depends on the direction of rotation.</td></tr>
<tr><td>diCorrVal</td><td>DINT</td><td>Current correction value enter most recently in the FIFO structure 'stCorrFifo'</td></tr>
<tr><td>diOutVal</td><td>DINT</td><td>Output value
Output of the correction value in the form of a linear interpolation covering the range 'udCorrStart' through 'udCorrStop' ('enMode' = 'CORRECT_SET2OUT' or 'CORRECT_SET2OUT_NB').
Outputs the correction value in the form of a linear interpolation covering the range 'udCorrStart' through 'udCorrStop' additively linked to the input value 'diInVal'
('enMode' = 'CORRECT_ADD2OUT' or 'CORRECT_ADD2OUT_NB').</td></tr>
</table>

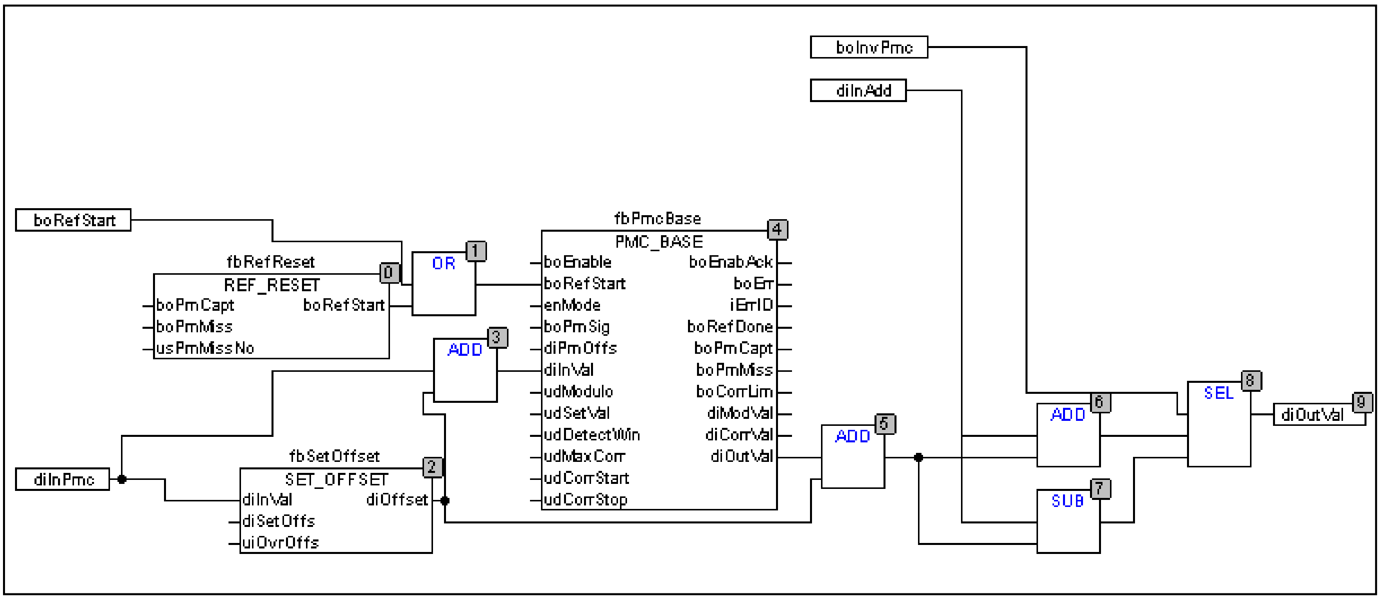
10.3 ExtendedFunctions

PMC	Printing mark control (overall function)

10.3.1 PMC (FB)

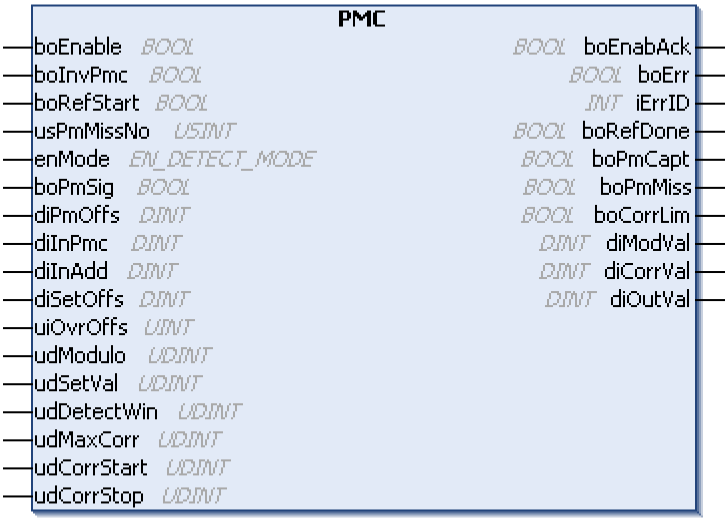
The 'PMC' function block combines the overall function of printing mark control.
It is based on the 'PMC_BASE', 'REF_RESET' and 'SET_OFFSET' blocks.

Abbildung 50: PMC: Structure



- The input variables of the 'PMC' block essentially correspond to the input variables of the integrated blocks. The presentation contains the additional input signals 'boRefStart', 'boInvPmc', 'diInPmc', and 'diInAdd' along with the logic operations in the context of the 'PMC' block.
- The output variables correspond to the output variables of the 'PMC_BASE' block. The 'diOutVal' output value is also mapped with a logic operation involving 'diOutVal' from the 'PMC_BASE' block, 'diOffset' from the 'SET_OFFSET' block, and 'diInAdd' from the 'PMC' block. 'boInvPmc' is used to invert the direction of effect of the PMC component of 'diOutVal', which is used in the context of the tool setting variant.

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boInvPmc</td><td>BOOL</td><td>Inversion of PMC direction of effect
Support for tool setting mode
<table><tr><td>Default</td><td>FALSE</td></tr></table>
<table><tr><th>Range</th><th>Meaning</th></tr><tr><td>FALSE</td><td>No inversion: insetting</td></tr><tr><td>TRUE</td><td>Inversion: tool setting</td></tr></table></td></tr>
<tr><td>boRefStart</td><td>BOOL</td><td>Start of a new homing cycle; alignment with next printing mark
Only applies in conjunction with 'enMode' = DETECT_AUTO.
When the block is activated in 'DETECT_AUTO' mode, a homing cycle is started without 'boRefStart' being evaluated.
A positive edge change at 'boRefStart' triggers repeat homing without the block having to be reactivated (positive edge at 'boEnable').</td></tr>
<tr><td>usPmMissNo</td><td>USINT</td><td>Number of missing printing marks until reset
Maximum number of consecutive undetected printing marks until a homing cycle is started (boRefStart = TRUE)
'usPmMissNo' = 0: block inactive
The input only applies in conjunction with enMode = DETECT_AUTO.
<table><tr><td>Default</td><td>5</td></tr></table></td></tr>
<tr><td>enMode</td><td>ENUM</td><td>EN_DETECT_MODE
Selection mode operating mode
<table><tr><td>Default</td><td>DETECT_AUTO</td></tr></table>
<table><tr><th>Range</th><th>Meaning</th></tr><tr><td>DETECT_AUTO</td><td>Automatic homing with reference to the first printing mark</td></tr><tr><td>DETECT_ MANUAL</td><td>Manual homing
The printing mark must be aligned manually with the sensor prior to enabling with 'boEnable'</td></tr></table></td></tr>
<tr><td>boPmSig</td><td>BOOL</td><td>Printing mark signal (PM signal)
Signal indicating a printing mark inside the modulo format
(The signal must remain pending for at least 1 sampling time)</td></tr>
<tr><td>diPmOffs</td><td>DINT</td><td>Printing mark offset (PM offset)
Describes the deviation between the time-discrete input value 'diInVal' (kT0) and the actual input value 'diInVal'(TboPmSig) at the time of the edge change on the printing mark signal
The following applies:</td></tr>
<tr><td>diInPmc</td><td>DINT</td><td>Input value for printing mark control
Reference point for the various input variables.</td></tr>
<tr><td>diInAdd</td><td>DINT</td><td>Additive input value
is added to output value 'diOutVal'; can be used for synchronous coupling, for example (master-slave)</td></tr>
<tr><td>diSetOffs</td><td>DINT</td><td>Setpoint offset
Is output after several sampling time points at 'diOffset'</td></tr>
<tr><td>uiOvrOffs</td><td>UINT</td><td>Offset override
Set a velocity override on interpolation of the offset
<table><tr><td>Range</td><td>0 ... 100</td></tr><tr><td>Unit</td><td>%</td></tr><tr><td>Default</td><td>10</td></tr></table></td></tr>
</table>

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>udModulo</td><td>UDINT</td><td>Modulo format
Describes the setpoint distance between two consecutive printing marks.
The value is saved when the block is activated (positive edge at 'boEnable'). A subsequent change does not affect the active block.<table><tr><td>Range</td><td>0 ... 1000000000</td></tr><tr><td>Default</td><td>20000</td></tr></table></td></tr>
<tr><td>udSetVal</td><td>UDINT</td><td>PM setpoint
Describes the expected distance between printing mark and printing mark sensor.
The value is saved when the block is activated (positive edge at 'boEnable'). A subsequent change does not affect the active block.
If udSetVal ≥ udModulo, n correction values "0" are entered in 'stCorrFifo'.

This corresponds to a slip in the correction value of n formats. In other words, the mark sensor is positioned n formats upstream of the tool position.<table><tr><td>Default</td><td>10000</td></tr></table></td></tr>
<tr><td>udDetectWin</td><td>UDINT</td><td>Permissible range
A 'boPmSig' flag signal is permitted within this range. The value can be changed online when the block is active<table><tr><td>Range</td><td>0 ... 999999999</td></tr><tr><td>Default</td><td>5000</td></tr></table></td></tr>
<tr><td>udMaxCorr</td><td>UDINT</td><td>Maximum permissible correction value to which the correction value output per modulo format is limited.
The value can be changed when the block is active<table><tr><td>Range</td><td>0 ... 999999999</td></tr><tr><td>Default</td><td>2000</td></tr></table></td></tr>
<tr><td>udCorrStart</td><td>UDINT</td><td>Correction starting value at which the output of correction values commences.
The value can be changed when the block is active<table><tr><td>Range</td><td>0 ... 999999999</td></tr><tr><td>Default</td><td>15000</td></tr></table></td></tr>
<tr><td>udCorrStop</td><td>UDINT</td><td>Correction stop value at which the output of correction values ends.
The value can be changed when the block is active<table><tr><td>Range</td><td>0 ... 999999999</td></tr><tr><td>Default</td><td>19999</td></tr></table></td></tr>
</table>

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state<table><tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr><tr><td>TRUE</td><td>Error</td></tr></table></td></tr>
</table>

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error
<table>
<tr><td>Range</td><td>Meaning</td></tr>
<tr><td>1 ... 9</td><td>Warnings and errors associated with the 'PM_DETECT' block</td></tr>
<tr><td>10 ... 19</td><td>Warnings and errors associated with the 'PM_CORRECT' block
with an offset of 10</td></tr>
<tr><td>20</td><td>Illegal value of 'udSetVal' or 'udDetectWin'
The ranges of the detection window and the correction range 'udCorrStart' ... 'udCorrStop' overlap.</td></tr>
</table></td></tr>
<tr><td>boRefDone</td><td>BOOL</td><td>Homing cycle completed
Acknowledgement signal to indicate that a homing cycle has been completed.
In mode 'enMode' = DETECT_AUTO, 'boRefDone' = FALSE when the block is activated or on a positive edge change at 'boRefStart'.
Once the first mark has been detected, 'boRefDone' = TRUE is set.
In mode 'enMode' = DETECT_MANUAL, this variable is of no significance. 'boRefDone' = TRUE always applies.</td></tr>
<tr><td>boPmCapt</td><td>BOOL</td><td>Printing mark detected
Signal indicating that a printing mark has been detected inside the permissible range.
'boPmCapt' is TRUE for one cycle only.
At the same time, the calculated correction value is entered in the FIFO correction value.</td></tr>
<tr><td>boPmMiss</td><td>BOOL</td><td>Printing mark missing
Signal indicating that a mark has not been detected inside the permissible range.
'boPmMiss' is TRUE for one cycle only.
At the same time, the value "0" is entered in the FIFO correction value.</td></tr>
<tr><td>boCorrLim</td><td>BOOL</td><td>Correction limiting
Display a limit of the correction value according to 'udMaxCorr'.
The variable is set to true for one cycle after 'udCorrStart' and before 'udCorrStop'</td></tr>
<tr><td>diModVal</td><td>DINT</td><td>Modulo value
Displays the current modulo position (0 ≤ diModVal < udModulo).
The sign depends on the direction of rotation.</td></tr>
<tr><td>diCorrVal</td><td>DINT</td><td>Current correction value enter most recently in the FIFO structure 'stCorrFifo'</td></tr>
<tr><td>diOutVal</td><td>DINT</td><td>Output value
Output of the correction value in the form of a linear interpolation covering the range 'udCorrStart' through 'udCorrStop' ('enMode' = 'CORRECT_SET2OUT' or 'CORRECT_SET2OUT_NB').
Outputs the correction value in the form of a linear interpolation covering the range 'udCorrStart' through 'udCorrStop' additively linked to the input value 'diInVal'
('enMode' = 'CORRECT_ADD2OUT' or 'CORRECT_ADD2OUT_NB').</td></tr>
</table>

10.3.1.1 Description

The aim of printing mark control is to hold the reference between the binary signal of the printing mark sensor and the position of a mechanical system, e.g. a current motor position.
Printing mark control enables a material to be cut in a defined position, even if this position shifts within certain limits.

Abbildung 51: PMC: Principle of a system with printing mark control

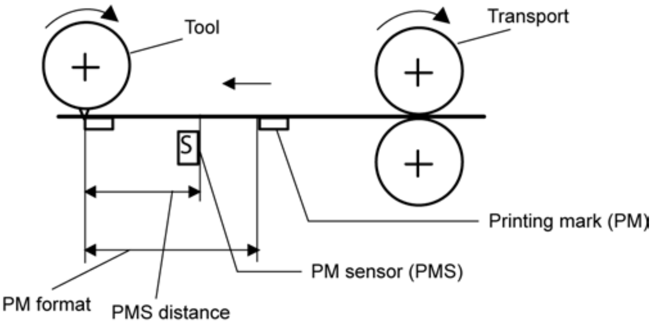
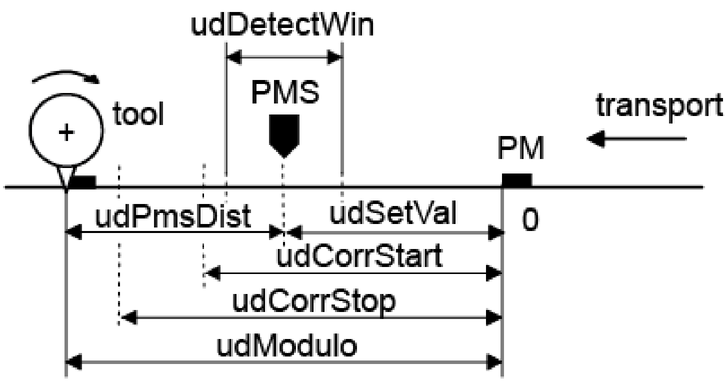


Abbildung 52: PMC: Relationship between input variables



Where:
(udSetVal MOD udModulo) < udCorrStart < udCorrStop < udModulo

The position of the printing mark can change for various reasons:
- Imprecise application of printing mark when printing the web
- Imprecise mapping ratio between longitudinal movement of the web and rotation of the tool
- Change in printing mark spacing due to physical factors, e.g. mechanical or thermal.

The relationship between the movement of the web and the movement of the tool is usually organized through position control; it is not part of printing mark control.
For overlaid printing mark control there are two options for correcting the changing position of the printing mark:
- Correction of the position of the transport system (insetting)
- Correction of the position of the tool (tool setting)

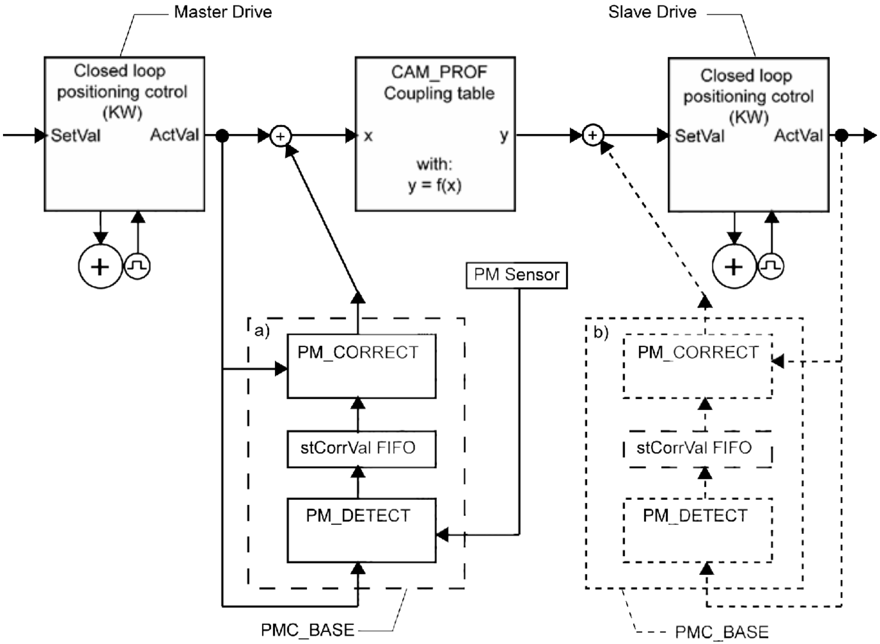
Printing mark control accesses two basic functions which are part of the AmkBase library:
- PM_DETECT: Detect printing marks and correction values in a FIFO structure:
- PM_CORRECT: Apply correction values from a FIFO structure and make correction

(See document Software description AmkBase Bibliothek , Part no. 204986)

There are two ways in which the correction can be made:

a. Printing mark control compares the position of the leading axis (master) with the printing mark signal. The correction is applied to the input variables of the 'CAM_PROF' function block.
b. Printing mark control compares the position of the following axis (slave) with the printing mark signal. The correction is applied to the output variables of the 'CAM_PROF' function block.

Abbildung 53: PMC: Interplay of the function blocks

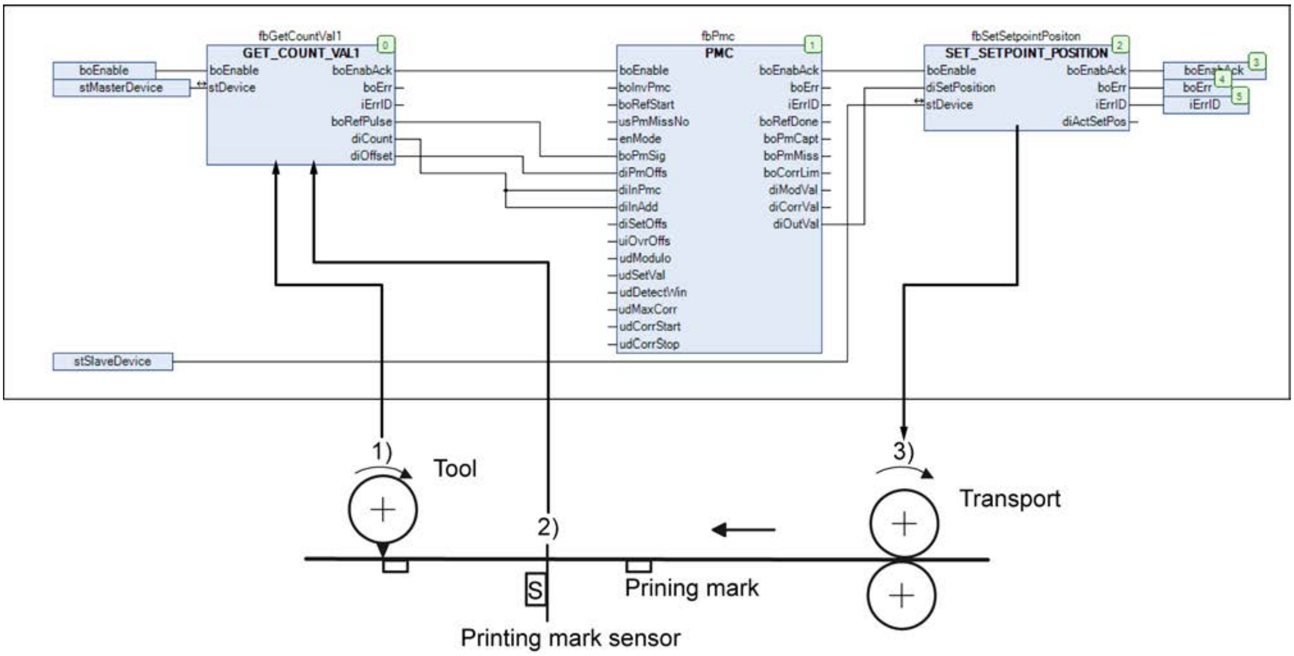


Insetting

In insetting mode:

- The master is the tool, transport is the slave.
- PMC.boInvPmc = FALSE: no inversion of the correction component
- The actual positions of the master (1) and the printing mark (2) are detected by the 'GET_COUNT_VAL1' function block, for example
  (prerequisite: ID32948 'Message 4x32' = 0x24; the sensor is detected by binary input BE3 of controller card KW-R06, for example).
- GET_COUNT_VAL1.diCount is used to guide the slave (PMC.diInAdd) and as the reference value for mark control (PMC.diInPmc).
- The logical operation linking the two position values is used with 'SET_SETPOINT_POSITION' as the position setpoint for the slave (3).
- If the slave drive is controlled independent of printing mark control, the 'PMC.diInAdd' input can remain open.

Abbildung 54: PMC: Insetting

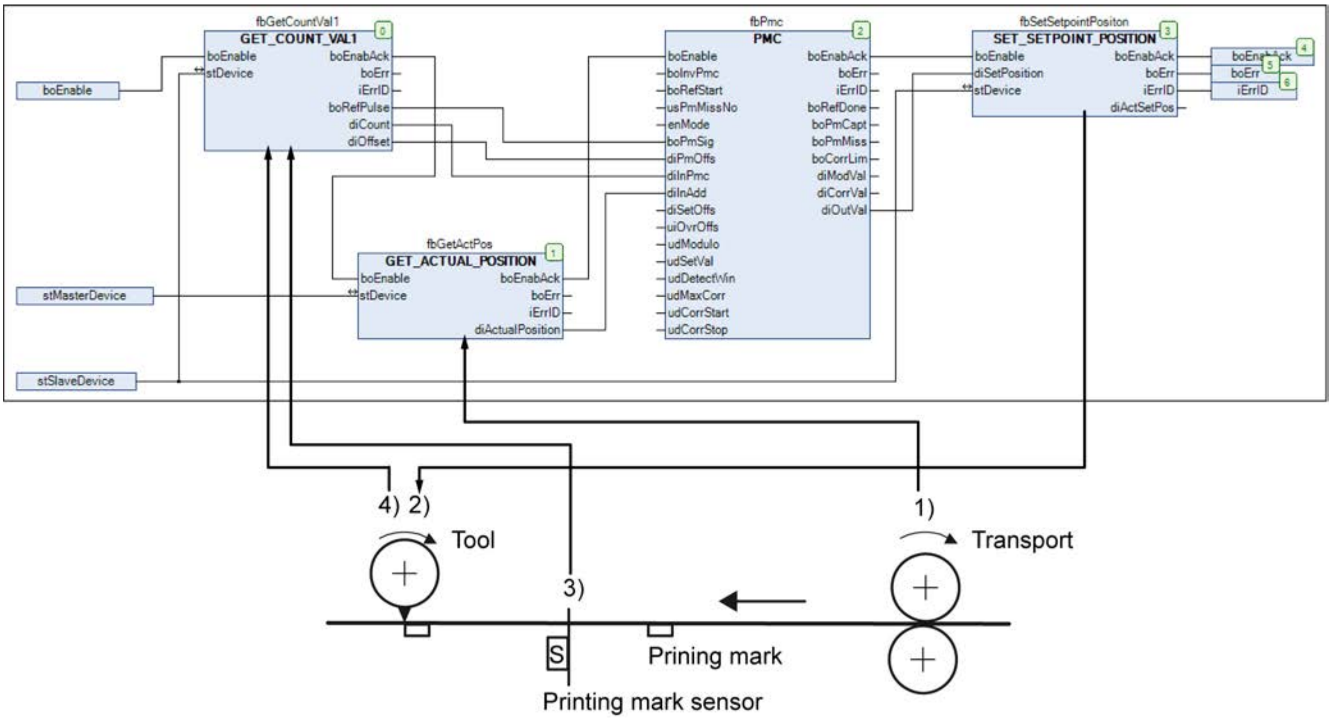


Tool setting

In tool setting mode:

- Transport is the master, the tool is the slave.
- PMC.boInvPmc = TRUE (inversion of the correction component).
- The actual position of the master (1) is detected by the 'GET_ACTUAL_POSITION' block.
- The printing mark (3) is detected by the 'GET_COUNT_VAL1' block further to a signal edge at a binary input, e.g. BE3 of controller card KW-R06.
  The temporal reference is converted to the actual position of the tool (4), which is also detected by the 'GET_COUNT_VAL1' block.
- GET_ACTUAL_POSITION.diActualPosition is used to control the slave (PMC.diInAdd).
- GET_COUNT_VAL1.diCount and GET_COUNT_VAL1.boRefPulse are used as reference values for mark control (PMC.diInPmc and PMC.boPmSig).
- The logic operation linking the two position values is used with 'SET_SETPOINT_POSITION' as the position setpoint for the slave (2).
- If the slave drive is controlled independent of printing mark control, the 'PMC.diInAdd' input can remain open.

Abbildung 55: PMC: Tool setting

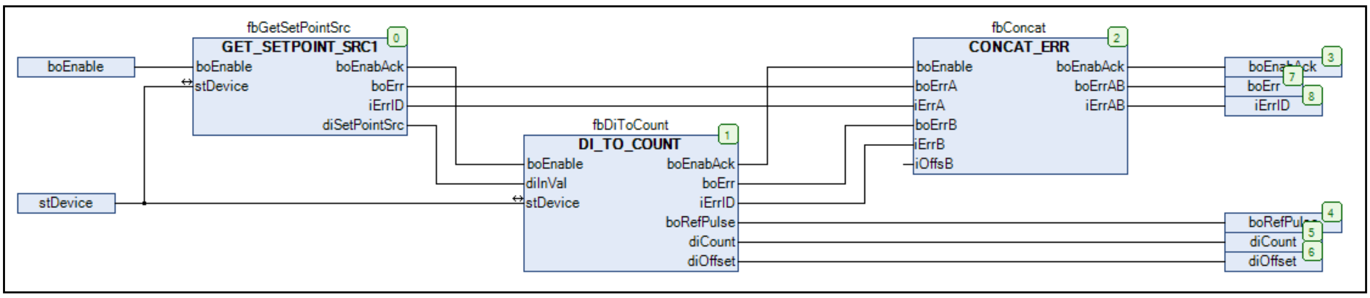


GET_COUNT_VAL1

The 'GET_COUNT_VAL1' function block consists of the 'GET_SETPOINT_SRC1', 'DI_TO_COUNT' and 'CONCAT_ERR' blocks. 'CONCAT_ERR' combines the error messages of the two previous blocks:

- The 'boErrAB' output is a logic OR operation of the 'boErrA' and 'boErrB' inputs.
  Accordingly, the meaning is:
  boErrAB = FALSE: no error; commanding permitted or warning
  boErrAB = TRUE: error
- The 'iErrAB' output maps the messages according to the following priority:
  - Error:
    boErrA = TRUE: iErrAB = iErrA
    boErrA = FALSE, boErrB = TRUE: iErrAB = iOffsB + iErrB
  - Warning:
    boErrA = FALSE, boErrB = FALSE, iErrA ≠ 0: iErrAB = iErrA
    boErrA = FALSE, boErrB = FALSE, iErrA = 0, iErrB ≠0: iErrAB = iOffsB + iErrB

Abbildung 56: PMC: GET_COUNT_VAL1



11 AmkBaseElems - Basic visualization function specific to AMK

The AmkBaseElems internal library provides the base function used for the implementation of simplified visualization input within all AMK libraries.

11.1 InternalVars

11.1.1 enSelectVisu

The essential task of the visualizations is to facilitate centrally organized visualization switching for various type-specific inputs. With its help it is possible to run visualizations alternately in touch-display mode (NumPad / KeyPad) as well as via keyboard input.

Declaration of 'enSelectVisu' global variables

VAR_GLOBAL **InternalVars**

Name	Type	Inherited from	Address	Initial	Comment
enSelectVisu	EN_VISU_INPUTMODE				select visu inputmode

11.2 Types

11.2.1 EN_VISU_INPUTMODE (EN)

Declaration of EN_VISU_INPUTMODE type

ENUM **EN_VISU_INPUTMODE**

Name	Type	Inherited from	Address	Initial	Comment
TOUCH	INT				use numpad / keypad
KEYBOARD	INT				use edit

For every variable type to be input, the assigned visualization is inserted within the CODESYS frame concept.

Variable type	Base visualization	NumPad / KeyPad
BYTE	ViByVal	N
DINT	ViDintVal	N
DWORD	ViDwVal	N
INT	ViIntVal	N
LREAL	ViLreVal	K
REAL	ViReVal	K
SINT	ViSintVal	N
STRING08	ViStr08Val	K
STRING15	ViStr15Val	K
STRING64	ViStr64Val	K
TIME	ViTimeVal	K
UDINT	ViUdVal	N
UINT	ViUintVal	N
USINT	ViUsintVal	N
WORD	ViWVal	N

In turn, the base visualization uses CODESYS frame switching.

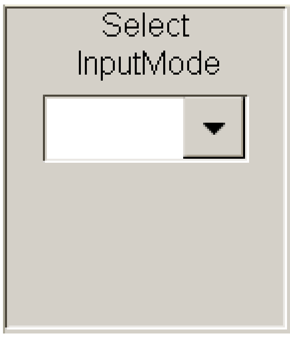
In this context, the following are activated based on the 'enSelectVisu' variable:

Variable

Name	Type	Description
enSelectVisu	ENUM	TOUCH – Input via InputType = NumPad / KeyPad KEYBOARD – Input via InputType = EDIT

The selection of the 'enSelectVisu' variable is made with the 'ViSelectVisuInputMode' visualization.

Figure: ViSelectVisuInputMode



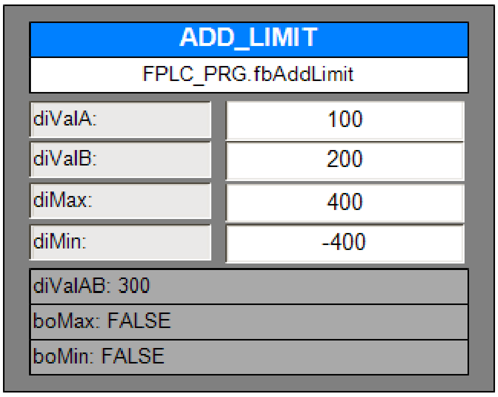
Alternatively, the 'AmkBaseElems.InternalVars.enSelectVisu' global variable can be used directly.

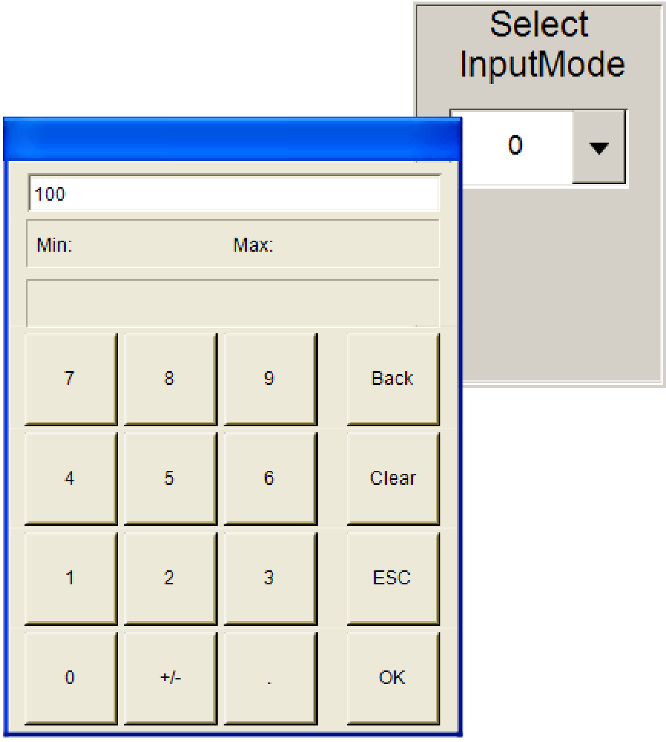
11.3 Examples

In the two examples, a value of 0 (TOUCH) or 1 (KEYBOARD) is assigned to the 'enSelectVisu' variable through the 'ViSelectVisuInputMode' visualization.

Example: variable input for 'enSelectVisu' = TOUCH

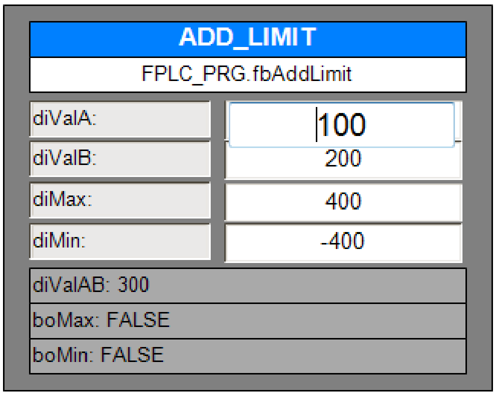
Input e.g. of variable 'diValA' by means of default value of 'enSelectVisu' = TOUCH (0). The NumPad opens so that the variable can be input.

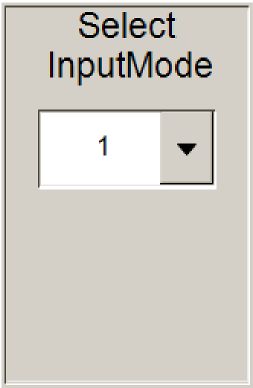




Example: variable input for 'enSelectVisu' = KEYBOARD

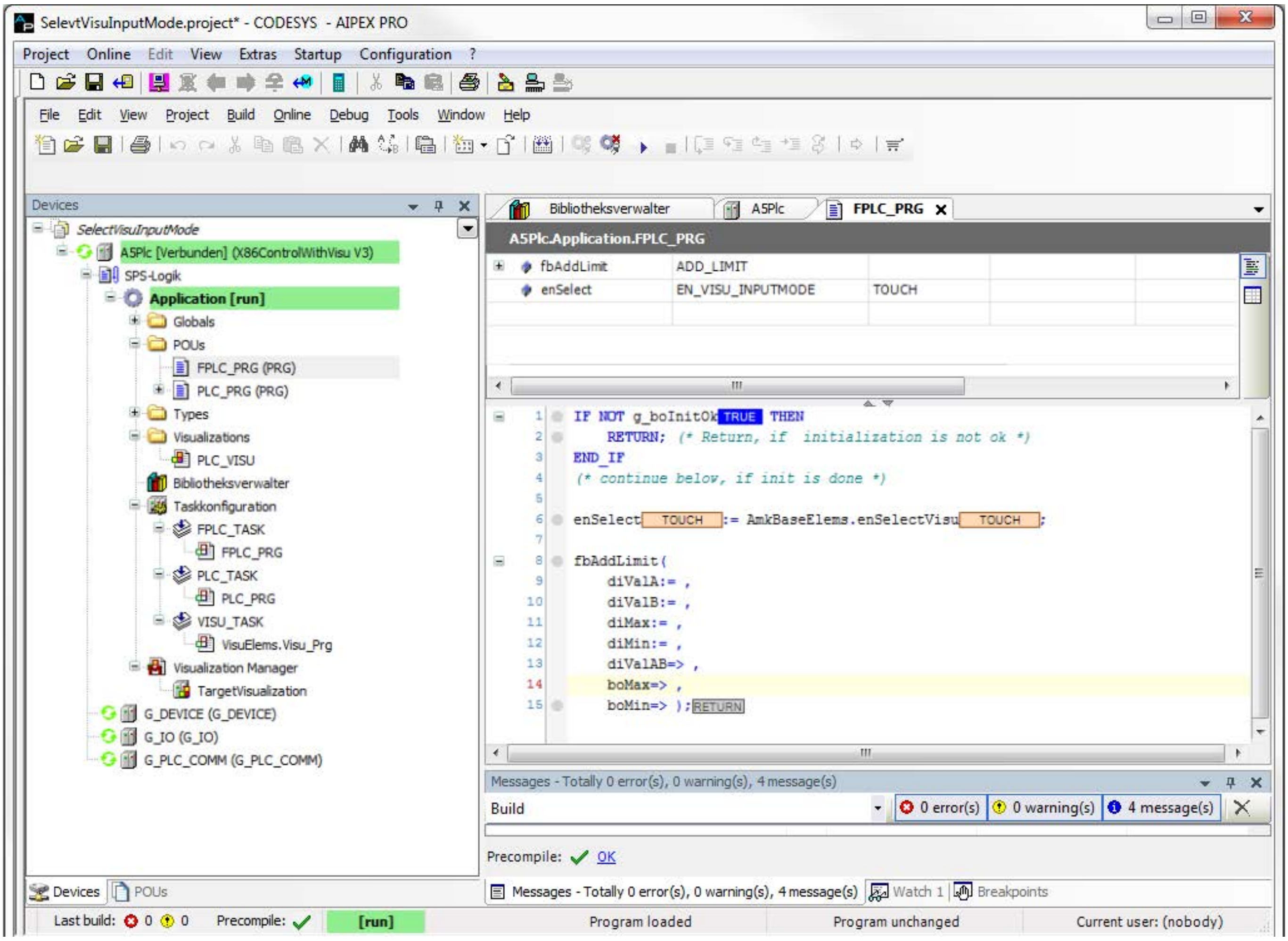
Input e.g. of variable 'diValA' by means of default value of 'enSelectVisu' = KEYBOARD (1). Setting this value enables the variable to be input via a keyboard.





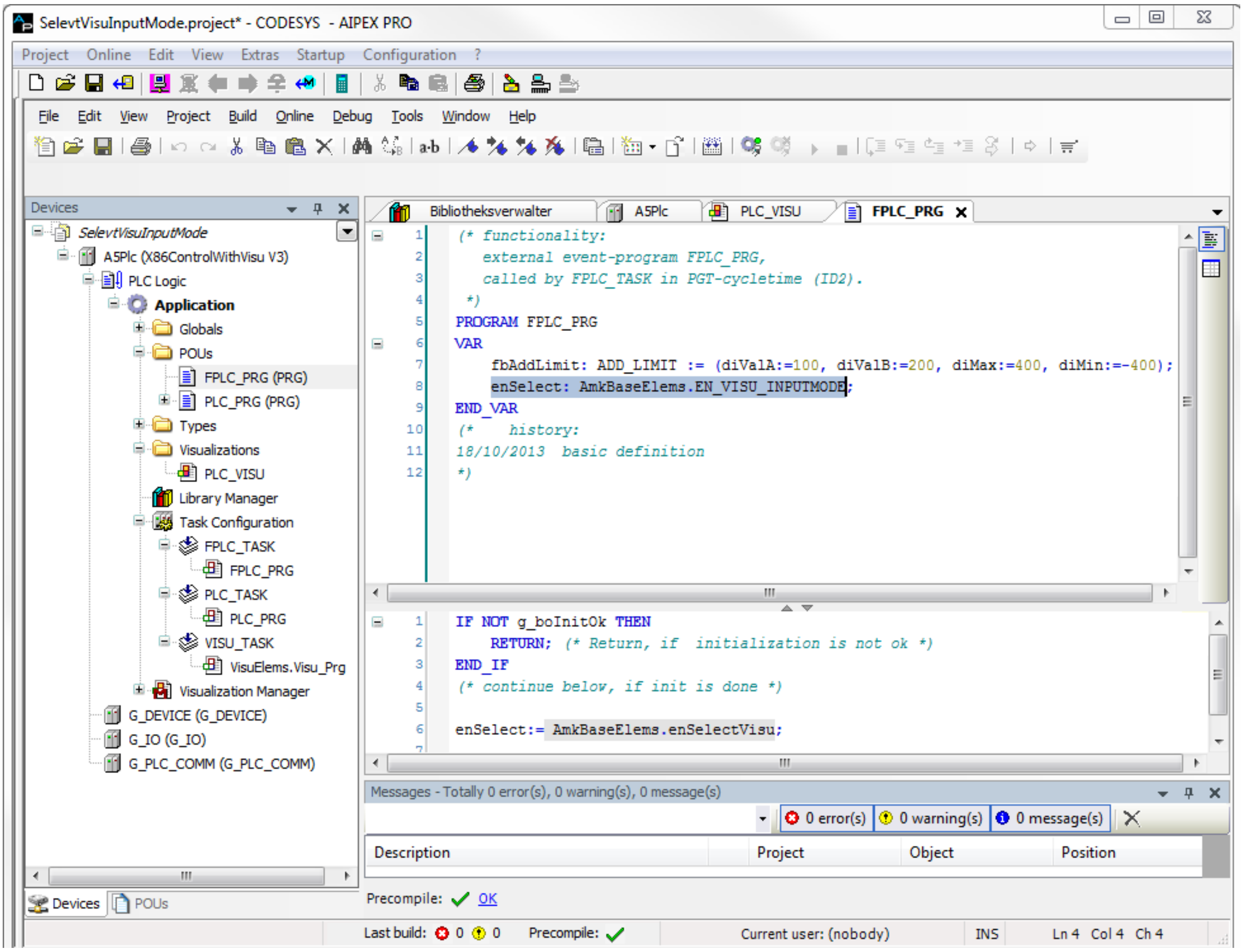
Example: variable access to 'enSelectVisu'

The variable is addressed by entering the full namespace 'AmkBaseElems.enSelectVisu'



Declaration of 'EN_VISU_INPUTMODE' type variables

The variable is addressed by entering the full namespace 'AmkBaseElems.EN_VISU_INPUTMODE'



12 AmkDevAccBase - Base device access function specific to AMK

'AmkDevAccBase' is an internal library which provides the base function used for the implementation of simplified device access in the AmkDevAccess library

The AmkDevAccBase library essentially contains the following blocks:

- DEVICE_ACCESS Maps the synchronous and asynchronous quantities that can be configured for a specific device.
- FDEV_ACCESS Formal mapping of the synchronous and asynchronous quantities that can be configured for specific devices and buses
- COMVAR_ACCESS Maps the synchronous and asynchronous quantities that can be configured manually (by setting the configured byte offset)
- PLCVAR_ACCESS Maps configurable synchronous and asynchronous quantities between multiple PLCs on the same bus system
- AFP_BASIC AFP (AMK fieldbus protocol) programmed in IEC for protocol-based communication with AMK devices, e.g. via the ACC bus (AMK CAN communication)

The "Basic - Functions" folder contains a number of other associated support functions which can in turn be used exclusively to implement these blocks.

The blocks in this library are intended solely for internal system development at AMK. They are used, for example, in the AmkDevAccess library and made available to users in a format customized to meet the requirements of their applications. Therefore, knowledge of these blocks is not necessary for applications.

13 AmkDevAccess - Device access function specific to AMK

AmkDevAccess is an internal library which provides a functional interface for access to basic device information. The blocks in the library are also a prerequisite for automatic bus configuration specific to AMK. They are divided into:

DeviceAccessAsync	Asynchronous device access blocks
DeviceAccessSync	Synchronous device access blocks
DeviceCmd	Device commanding
PlcVarAccess	PLC-PLC communication
Special	Blocks for specific buses and devices
Support	Support blocks

The blocks in the DeviceAccessAsync, DeviceAccessSync, DeviceCmd, and PlcVarAccess should be used in preference. They support programming which is not specific to a particular control system and bus system.

The blocks in the Special folder should only be used for applications in the AMK ACC bus system (AmkCanCommunication_ACC) or AMK-EtherCAT implementation (Sercos), since the functionality of the general blocks may not be sufficient.

The blocks in the Support folder are intended solely for AMK-internal system development; they should not be used in the application itself.

13.1 Blocks, specific devices and bus systems

13.1.1 Blocks for specific devices and bus systems in the AmkDevAccess library

The blocks listed in the tables are supported by the corresponding devices on the relevant bus systems (ACC_Bus, EtherCAT-Bus, 'local bus').

The 'local bus' connection provides 'AS' series controllers with access to internal controller information. A5x-MCxE series controllers can also access local IO.

Devices on the ACC bus

Block name (folder name)	**KE**	**KU/KW (R03)**	**KWZ**	**KWD**	**KWF**	**IDT4**	
Blocks that are not specific to devices or bus systems							
DeviceAccessAsync -Command - Control							
SET_CTRL_DC_BUSENABLE_x_UE	X	X	X	X	X	X	
SET_CTRL_ERR_RESET_x_FL	X	X	X	X	X	X	
SET_CTRL_INVERTER_ON_x_RF		X	X	X	X	X	
-Command - Status							
GET_STAT_DC_BUSENABLE_ACK_x_QUE	X	X	X	X	X	X	
GET_STAT_ERR_RESET_ACK_x_QFL	X	X	X	X	X	X	
GET_STAT_INVERTER_ON_ACK_x_QRF		X	X	X	X	X	
GET_STAT_SYSTEM_READY_x_SBM	X	X	X	X	X	X	
-Error							
GET_ERR_COMMUTATION		X	X	X		X	
GET_ERR_DC_BUS_OVERVOLT		X	X	X		X	
GET_ERR_DC_BUS_UNDERVOLT		X	X	X		X	
GET_ERR_ENCODER		X	X	X		X	
GET_ERR_EXCESS_FOLLOW		X	X	X		X	
GET_ERR_MOTOR_OVERTEMP		X	X	X		X	
GET_ERR_NOM_CUR_EXCESS		X	X	X		X	

Block name (folder name)	KE	KU/KW (R03)	KWZ	KWD	KWF	IDT4	
GET_ERR_SHORT_CIRCUIT		X	X	X		X	
GET_ERR_SUPPL_VOLT		X	X	X		X	
-Realtime							
GET_RT_ACTVAL_NORM_ACK		X	X	X		X	
GET_RT_DRIVE_ANGLE_SYNC		X	X	X		X	
GET_RT_DRIVE_SPEED_SYNC		X	X	X		X	
GET_RT_ON_NEG_SOFT_LIMIT		X	X	X		X	
GET_RT_ON_POS_SOFT_LIMIT		X	X	X		X	
GET_RT_OVERCUR_REACHED		X	X	X		X	
GET_RT_POS_WINDOW_REACHED		X	X	X		X	
GET_RT_POWER_LIMIT_REACHED		X	X	X		X	
GET_RT_RES_DIST_CLEARED		X	X	X		X	
GET_RT_SPEED_LIMIT		X	X	X		X	
GET_RT_SPEED_POS		X	X	X		X	
GET_RT_SPEED_THRESHOLD		X	X	X		X	
GET_RT_SPEED_WINDOW_REACHED		X	X	X		X	
GET_RT_SPEED_ZERO		X	X	X		X	
GET_RT_TORQUE_LIMIT		X	X	X		X	
GET_RT_TORQUE_THRESHOLD		X	X	X		X	
DeviceAccessSync -Controller - Actual values							
GET_ACTUAL_POSITION		X	X	X		X	
GET_ACTUAL_SPEED		X	X	X		X	
GET_ACTUAL_TORQUE		X	X	X		X	
-Controller - Set values - Preset values							
SET_PRE_SETPOINTS_SPEED		X	X	X		X	
SET_PRE_SETPOINTS_TORQUE		X	X	X		X	
-Controller - Set values							
SET_SETPOINT_POSITION		X	X	X		X	
SET_SETPOINT_SPEED		X	X	X		X	
SET_SETPOINT_TORQUE		X	X	X		X	
-Process IO							
GET_ENCODER1_LATCH		X	X	X		X	
GET_ENCODER1_STATUS							
GET_ENCODER1_VALUE		X	X	X		X	
GET_INPUT_ANALOG1							
GET_INPUT_ANALOG1_STATUS							
GET_INPUT_ANALOG2							
GET_INPUT_ANALOG2_STATUS							
GET_SETPOINT_SRC1		X	X	X		X	
GET_SETPOINT_SRC2		X	X	X		X	
GET_TS_INPUT							
GET_TS_INPUT1_LATCH_NEG							
GET_TS_INPUT1_LATCH_POS							
GET_TS_INPUT1_STATUS							
GET_TS_INPUT2_LATCH_NEG							
GET_TS_INPUT2_LATCH_POS							
GET_TS_INPUT2_STATUS							
SET_ENCODER1_CONTROL							

Block name (folder name)	KE	KU/KW (R03)	KWZ	KWD	KWF	IDT4	
SET_INPUT_ANALOG1_CONTROL							
SET_INPUT_ANALOG2_CONTROL							
SET_TS_OUTPUT							
SET_TS_OUTPUT_ACTIVATE							
SET_TS_OUTPUT_TIME							
-TimeStamp							
CAM_CONT_TS							
GET_TS_INPUTS							
SET_TS_OUTPUTS							
DeviceCmd							
DO_CMD_ONCE		X	X	X		X	
Blocks for specific devices or bus systems							
Special -DeviceAccessAsync							
GET_ERROR_ID11		X	X	X		X	
GET_STATUS_ID144		X	X	X		X	
-AmkCanCommunication_ACC							
GET_ERROR_OPT		X	X	X		X	
GET_ERROR_SYS		X	X	X		X	
-Local - iSA							
GET_DC_BUS_VOLTAGE							
GET_HEAT_SINK_TEMPERATURE							
GET_INTERIOR_TEMPERATURE							
-Sercos - Command - Control							
SET_CTRL_RT_BIT1							
SET_CTRL_RT_BIT2							
-Sercos - Command - Status							
GET_STAT_RT_BIT1							
GET_STAT_RT_BIT2							
-Sercos - Error							
GET_STAT_CLASS2							
-DeviceAccessSync - AmkCanCommunication_ACC							
GET_ACTVAL16_0		X	X	X		X	
GET_ACTVAL16_1		X	X	X		X	
GET_ACTVAL16_2		X	X	X		X	
GET_ACTVAL32_0		X	X	X		X	
GET_ACTVAL32_1		X	X	X		X	
GET_MESSAGE16	X	X[1)]	X[1)]	X[1)]		X[1)]	
GET_MESSAGE32	X	X[2)]	X[2)]	X[2)]		X[2)]	
SET_ADD_SETPOINT16		X	X	X		X	
SET_ADD_SETPOINT32		X	X	X		X	
SET_MAIN_SETPOINT		X[3)]	X[3)]	X[3)]	X		
SET_SETPOINT16_0		X[4)]	X[4)]	X[4)]		X[4)]	
SET_SETPOINT16_1		X	X	X		X	
SET_SETPOINT16_2		X	X	X		X	
SET_SETPOINT16_3		X	X	X		X	
SET_SETPOINT32_0		X[5)]	X[5)]	X[5)]		X[5)]	

Block name (folder name)	KE	KU/KW (R03)	KWZ	KWD	KWF	IDT4	
SET_SETPOINT32_1		X	X	X		X	
- Sercos							
GET_FOLLOW_ERR							
SET_LIM_SPEED_BIPOL							
SET_LIM_SPEED_POS							
SET_LIM_SPEED_NEG							
SET_LIM_TORQUE_BIPOL							
SET_LIM_TORQUE_POS							
SET_LIM_TORQUE_NEG							
SET_SETPOINT_MUL							
SET_SETPOINT_DIV							
SET_SETPOINT_SIWL							
- Sercos – Process IO							
GET_ACTPOS_LATCHED_NEG1							
GET_ACTPOS_LATCHED_NEG2							
GET_ACTPOS_LATCHED_POS1							
GET_ACTPOS_LATCHED_POS2							
GET_PROBE_STS							
Support -AmkCanCommunication_ACC							
DO_AFP		X	X	X		X	
DO_AFP_ONCE		X	X	X		X	
-Sercos							
CMD_BY_ID							
DO_CMD							
STATE_BY_ID							

Occupied by:

1) GET_ACTUAL_TORQUE
2) GET_ACTUAL_SPEED
3) SET_SETPOINT_POSITION,
   SET_SETPOINT_SPEED,
   SET_SETPOINT_TORQUE
4) SET_PRE_SETPOINT_TORQUE
5) SET_PRE_SETPOINT_SPEED

Devices on the EtherCAT bus

Block name (folder name)	I/O	KU/KW (R03)	KW (R05,R06) iX, iC, iDT5, ihX	KWZ	KWD (R05)		
Blocks that are not specific to devices or bus systems							
DeviceAccessAsync -Command - Control							
SET_CTRL_DC_BUSENABLE_x_UE		X	X	X	X		
SET_CTRL_ERR_RESET_x_FL		X	X	X	X		
SET_CTRL_INVERTER_ON_x_RF		X	X	X	X		
-Command - Status							
GET_STAT_DC_BUSENABLE_ACK_x_QUE		X	X	X	X		
GET_STAT_ERR_RESET_ACK_x_QFL		X	X	X	X		

Block name (folder name)	I/O	KU/KW (R03)	KW (R05,R06) iX, iC, iDT5, ihX	KWZ	KWD (R05)		
GET_STAT_INVERTER_ON_ACK_x_QRF		X	X	X	X		
GET_STAT_SYSTEM_READY_x_SBM		X	X	X	X		
-Error							
GET_ERR_COMMUTATION		X	X 1)	X	X 1)		
GET_ERR_DC_BUS_OVERVOLT		X	X 1)	X	X 1)		
GET_ERR_DC_BUS_UNDERVOLT		X	X 1)	X	X 1)		
GET_ERR_ENCODER		X	X 1)	X	X 1)		
GET_ERR_EXCESS_FOLLOW		X	X 1)	X	X 1)		
GET_ERR_MOTOR_OVERTEMP		X	X 1)	X	X 1)		
GET_ERR_NOM_CUR_EXCESS		X	X 1)	X	X 1)		
GET_ERR_SHORT_CIRCUIT		X	X 1)	X	X 1)		
GET_ERR_SUPPL_VOLT		X	X 1)	X	X 1)		
-Realtime							
GET_RT_ACTVAL_NORM_ACK		X	X 1)	X	X 1)		
GET_RT_DRIVE_ANGLE_SYNC		X	X 1)	X	X 1)		
GET_RT_DRIVE_SPEED_SYNC		X	X 1)	X	X 1)		
GET_RT_ON_NEG_SOFT_LIMIT		X	X 1)	X	X 1)		
GET_RT_ON_POS_SOFT_LIMIT		X	X 1)	X	X 1)		
GET_RT_OVERCUR_REACHED		X	X 1)	X	X 1)		
GET_RT_POS_WINDOW_REACHED		X	X 1)	X	X 1)		
GET_RT_POWER_LIMIT_REACHED		X	X 1)	X	X 1)		
GET_RT_RES_DIST_CLEARED		X	X 1)	X	X 1)		
GET_RT_SPEED_LIMIT		X	X 1)	X	X 1)		
GET_RT_SPEED_POS		X	X 1)	X	X 1)		
GET_RT_SPEED_THRESHOLD		X	X 1)	X	X 1)		
GET_RT_SPEED_WINDOW_REACHED		X	X 1)	X	X 1)		
GET_RT_SPEED_ZERO		X	X 1)	X	X 1)		
GET_RT_TORQUE_LIMIT		X	X 1)	X	X 1)		
GET_RT_TORQUE_THRESHOLD		X	X 1)	X	X 1)		
DeviceAccessSync -Controller - Actual values							
GET_ACTUAL_POSITION		X	X	X	X		
GET_ACTUAL_SPEED		X	X	X	X		
GET_ACTUAL_TORQUE		X	X	X	X		
-Controller - Set values - Preset values							
SET_PRE_SETPOINTS_SPEED			X		X		
SET_PRE_SETPOINTS_TORQUE			X		X		
-Controller - Set values							
SET_SETPOINT_POSITION		X	X	X	X		
SET_SETPOINT_SPEED		X	X	X	X		
SET_SETPOINT_TORQUE		X	X	X	X		
-Process IO							
GET_ENCODER1_LATCH		X	X	X	X		
GET_ENCODER1_STATUS							
GET_ENCODER1_VALUE		X	X	X	X		
GET_INPUT_ANALOG1		X	X	X	X		
GET_INPUT_ANALOG1_STATUS							

Block name (folder name)	I/O	KU/KW (R03)	KW (R05,R06) iX, iC, iDT5, ihX	KWZ	KWD (R05)		
GET_INPUT_ANALOG2		X	X		X		
GET_INPUT_ANALOG2_STATUS							
GET_SETPOINT_SRC1		X	X	X	X		
GET_SETPOINT_SRC2		X	X		X		
GET_TS_INPUT							
GET_TS_INPUT1_LATCH_NEG							
GET_TS_INPUT1_LATCH_POS							
GET_TS_INPUT1_STATUS							
GET_TS_INPUT2_LATCH_NEG							
GET_TS_INPUT2_LATCH_POS							
GET_TS_INPUT2_STATUS							
SET_ENCODER1_CONTROL							
SET_INPUT_ANALOG1_CONTROL							
SET_INPUT_ANALOG2_CONTROL							
SET_TS_OUTPUT							
SET_TS_OUTPUT_ACTIVATE							
SET_TS_OUTPUT_TIME							
-TimeStamp							
CAM_CONT_TS	X[3)]						
GET_TS_INPUTS	X[2)]						
SET_TS_OUTPUTS	X[3)]						
DeviceCmd							
DO_CMD_ONCE		X	X	X	X		
Blocks for specific devices or bus systems							
Special -DeviceAccessAsync							
GET_ERROR_ID11		X	X	X	X		
GET_STATUS_ID144		X	X	X	X		
-AmkCanCommunication_ACC							
GET_ERROR_OPT							
GET_ERROR_SYS							
-Local - iSA							
GET_DC_BUS_VOLTAGE							
GET_HEAT_SINK_TEMPERATURE							
GET_INTERIOR_TEMPERATURE							
-Sercos - Command - Control							
SET_CTRL_RT_BIT1		X	X	X	X		
SET_CTRL_RT_BIT2		X	X	X	X		
-Sercos - Command - Status							
GET_STAT_RT_BIT1		X	X	X	X		
GET_STAT_RT_BIT2		X	X	X	X		
-Sercos - Error							
GET_STAT_CLASS2		X	X	X	X		
-DeviceAccessSync - AmkCanCommunication_ACC							
GET_ACTVAL16_0							

Block name (folder name)	I/O	KU/KW (R03)	KW (R05,R06) iX, iC, iDT5, ihX	KWZ	KWD (R05)		
GET_ACTVAL16_1							
GET_ACTVAL16_2							
GET_ACTVAL32_0							
GET_ACTVAL32_1							
GET_MESSAGE16							
GET_MESSAGE32							
SET_ADD_SETPOINT16							
SET_ADD_SETPOINT32							
SET_MAIN_SETPOINT							
SET_SETPOINT16_0							
SET_SETPOINT16_1							
SET_SETPOINT16_2							
SET_SETPOINT16_3							
SET_SETPOINT32_0							
SET_SETPOINT32_1							
- Sercos							
GET_FOLLOW_ERR		X	X	X	X		
SET_LIM_SPEED_BIPOL		X	X	X	X		
SET_LIM_SPEED_POS		X	X	X	X		
SET_LIM_SPEED_NEG		X	X	X	X		
SET_LIM_TORQUE_BIPOL		X	X	X	X		
SET_LIM_TORQUE_POS		X	X	X	X		
SET_LIM_TORQUE_NEG		X	X	X	X		
SET_SETPOINT_MUL		X	X	X	X		
SET_SETPOINT_DIV		X	X	X	X		
SET_SETPOINT_SIWL		X	X	X	X		
- Sercos – Process IO							
GET_ACTPOS_LATCHED_NEG1		X	X	X	X		
GET_ACTPOS_LATCHED_NEG2			X		X		
GET_ACTPOS_LATCHED_POS1		X	X	X	X		
GET_ACTPOS_LATCHED_POS2			X		X		
GET_PROBE_STS		X	X	X	X		
Support -AmkCanCommunication_ACC							
DO_AFP							
DO_AFP_ONCE							
-Sercos							
CMD_BY_ID		X	X	X	X		
DO_CMD		X	X	X	X		
STATE_BY_ID		X	X	X	X		

1) Not yet supported by version "AER05 V1.02 2009/20"
2) EL1252 EtherCAT terminal
3) EL2252 EtherCAT terminal

Device with local bus connection

Block name (folder name)	**A4x-MxE 1)** **A5x-MxE 1)** **A6x-MxE 1)**	**iSA**					
Blocks that are not specific to devices or bus systems							
DeviceAccessAsync -Command - Control							
SET_CTRL_DC_BUSENABLE_x_UE							
SET_CTRL_ERR_RESET_x_FL	X	X					
SET_CTRL_INVERTER_ON_x_RF							
-Command - Status							
GET_STAT_DC_BUSENABLE_ACK_x_QUE		X 2)					
GET_STAT_ERR_RESET_ACK_x_QFL	X	X					
GET_STAT_INVERTER_ON_ACK_x_QRF							
GET_STAT_SYSTEM_READY_x_SBM	X	X					
-Error							
GET_ERR_COMMUTATION							
GET_ERR_DC_BUS_OVERVOLT							
GET_ERR_DC_BUS_UNDERVOLT							
GET_ERR_ENCODER							
GET_ERR_EXCESS_FOLLOW							
GET_ERR_MOTOR_OVERTEMP							
GET_ERR_NOM_CUR_EXCESS							
GET_ERR_SHORT_CIRCUIT							
GET_ERR_SUPPL_VOLT							
-Realtime							
GET_RT_ACTVAL_NORM_ACK							
GET_RT_DRIVE_ANGLE_SYNC							
GET_RT_DRIVE_SPEED_SYNC							
GET_RT_ON_NEG_SOFT_LIMIT							
GET_RT_ON_POS_SOFT_LIMIT							
GET_RT_OVERCUR_REACHED							
GET_RT_POS_WINDOW_REACHED							
GET_RT_POWER_LIMIT_REACHED							
GET_RT_RES_DIST_CLEARED							
GET_RT_SPEED_LIMIT							
GET_RT_SPEED_POS							
GET_RT_SPEED_THRESHOLD							
GET_RT_SPEED_WINDOW_REACHED							
GET_RT_SPEED_ZERO							
GET_RT_TORQUE_LIMIT							
GET_RT_TORQUE_THRESHOLD							
DeviceAccessSync -Controller - Actual values							
GET_ACTUAL_POSITION							
GET_ACTUAL_SPEED							
GET_ACTUAL_TORQUE							

Block name (folder name)	A4x-MxE[1)] A5x-MxE[1)] A6x-MxE[1)]	iSA					
-Controller - Set values - Preset values							
SET_PRE_SETPOINTS_SPEED							
SET_PRE_SETPOINTS_TORQUE							
-Controller - Set values							
SET_SETPOINT_POSITION							
SET_SETPOINT_SPEED							
SET_SETPOINT_TORQUE							
-Process IO							
GET_ENCODER1_LATCH	X						
GET_ENCODER1_STATUS	X						
GET_ENCODER1_VALUE	X						
GET_INPUT_ANALOG1	X						
GET_INPUT_ANALOG1_STATUS	X						
GET_INPUT_ANALOG2	X						
GET_INPUT_ANALOG2_STATUS	X						
GET_SETPOINT_SRC1	X						
GET_SETPOINT_SRC2							
GET_TS_INPUT	X						
GET_TS_INPUT1_LATCH_NEG	X						
GET_TS_INPUT1_LATCH_POS	X						
GET_TS_INPUT1_STATUS	X						
GET_TS_INPUT2_LATCH_NEG	X						
GET_TS_INPUT2_LATCH_POS	X						
GET_TS_INPUT2_STATUS	X						
SET_ENCODER1_CONTROL	X						
SET_INPUT_ANALOG1_CONTROL	X						
SET_INPUT_ANALOG2_CONTROL	X						
SET_TS_OUTPUT	X						
SET_TS_OUTPUT_ACTIVATE	X						
SET_TS_OUTPUT_TIME	X						
-TimeStamp							
CAM_CONT_TS	X						
GET_TS_INPUTS	X						
SET_TS_OUTPUTS	X						
DeviceCmd							
DO_CMD_ONCE							
Blocks for specific devices and bus systems							
Special -DeviceAccessAsync							
GET_ERROR_ID11							
GET_STATUS_ID144							
-AmkCanCommunication_ACC							
GET_ERROR_OPT							
GET_ERROR_SYS							
-Local - iSA							

Block name (folder name)	A4x-MxE1) A5x-MxE1) A6x-MxE1)	iSA					
GET_DC_BUS_VOLTAGE		X					
GET_HEAT_SINK_TEMPERATURE		X					
GET_INTERIOR_TEMPERATURE		X					
-Sercos - Command - Control							
SET_CTRL_RT_BIT1							
SET_CTRL_RT_BIT2							
-Sercos - Command - Status							
GET_STAT_RT_BIT1							
GET_STAT_RT_BIT2							
-Sercos - Error							
GET_STAT_CLASS2							
-DeviceAccessSync - AmkCanCommunication_ACC							
GET_ACTVAL16_0							
GET_ACTVAL16_1							
GET_ACTVAL16_2							
GET_ACTVAL32_0							
GET_ACTVAL32_1							
GET_MESSAGE16							
GET_MESSAGE32							
SET_ADD_SETPOINT16							
SET_ADD_SETPOINT32							
SET_MAIN_SETPOINT							
SET_SETPOINT16_0							
SET_SETPOINT16_1							
SET_SETPOINT16_2							
SET_SETPOINT16_3							
SET_SETPOINT32_0							
SET_SETPOINT32_1							
- Sercos							
GET_FOLLOW_ERR							
SET_LIM_SPEED_BIPOL							
SET_LIM_SPEED_POS							
SET_LIM_SPEED_NEG							
SET_LIM_TORQUE_BIPOL							
SET_LIM_TORQUE_POS							
SET_LIM_TORQUE_NEG							
SET_SETPOINT_MUL							
SET_SETPOINT_DIV							
SET_SETPOINT_SIWL							
- Sercos – ProcessIO							
GET_ACTPOS_LATCHED_NEG1							
GET_ACTPOS_LATCHED_NEG2							
GET_ACTPOS_LATCHED_POS1							
GET_ACTPOS_LATCHED_POS2							
GET_PROBE_STS							

Block name (folder name)	A4x-MxE[1)] A5x-MxE[1)] A6x-MxE[1)]	iSA					
Support -AmkCanCommunication_ACC							
DO_AFP							
DO_AFP_ONCE							
-Sercos							
CMD_BY_ID							
DO_CMD							
STATE_BY_ID							

1) PLC types: A5x-MxE; version AS V4.10 2013/06 and later
2) Always TRUE

13.1.2 Block dependency of device information configured automatically

The following tables list the assignments between bus access blocks and the associated necessary device information (ENUM values: EN_DEV_INFO type from the AmkBase library)

Abstraction to 'technological device information' means that the values can be mapped independently of devices and bus systems. This is done by AIPEX PRO during the automatic bus configuration process.

13.1.2.1 Blocks in the AmkDevAccess library

The following table lists the required device information for the blocks in the AmkDevAccess library which are displayed in the assignment window in the context of automatic bus configuration. For a list of the parameter settings for the IDs linked to the blocks: Siehe 'Parameterization' auf Seite 267.

Device information for the blocks in the AmkDevAccess library

Block name (folder name)	Device information (ENUM value)
Blocks that are not specific to devices or bus systems	
DeviceAccessAsync -Command - Control	
SET_CTRL_DC_BUSENABLE_x_UE	DEV_SET_CTRL_DC_BUSENABLE
SET_CTRL_ERR_RESET_x_FL	DEV_SET_CTRL_ERR_RESET
SET_CTRL_INVERTER_ON_x_RF	DEV_SET_CTRL_INVERTER_ON
-Command - Status	
GET_STAT_DC_BUSENABLE_ACK_x_QUE	DEV_GET_STAT_DC_BUSENABLE_ACK
GET_STAT_ERR_RESET_ACK_x_QFL	DEV_GET_STAT_ERR_RESET_ACK
GET_STAT_INVERTER_ON_ACK_x_QRF	DEV_GET_STAT_INVERTER_ON_ACK
GET_STAT_SYSTEM_READY_x_SBM	DEV_GET_STAT_SYSTEM_READY
-Error	
GET_ERR_COMMUTATION	DEV_GET_ERR_COMMUTATION
GET_ERR_DC_BUS_OVERVOLT	DEV_GET_ERR_DC_BUS_OVERVOLT
GET_ERR_DC_BUS_UNDERVOLT	DEV_GET_ERR_DC_BUS_UNDERVOLT
GET_ERR_ENCODER	DEV_GET_ERR_ENCODER
GET_ERR_EXCESS_FOLLOW	DEV_GET_ERR_EXCESS_FOLLOW
GET_ERR_MOTOR_OVERTEMP	DEV_GET_ERR_MOTOR_OVERTEMP
GET_ERR_NOM_CUR_EXCESS	DEV_GET_ERR_NOM_CUR_EXCESS
GET_ERR_SHORT_CIRCUIT	DEV_GET_ERR_SHORT_CIRCUIT

Block name (folder name)	Device information (ENUM value)
GET_ERR_SUPPL_VOLT	DEV_GET_ERR_SUPPL_VOLT
-Realtime	
GET_RT_ACTVAL_NORM_ACK	DEV_GET_RT_ACTVAL_NORM_ACK
GET_RT_DRIVE_ANGLE_SYNC	DEV_GET_RT_DRIVE_ANGLE_SYNC
GET_RT_DRIVE_SPEED_SYNC	DEV_GET_RT_DRIVE_SPEED_SYNC
GET_RT_ON_NEG_SOFT_LIMIT	DEV_GET_RT_ON_NEG_SOFT_LIMIT
GET_RT_ON_POS_SOFT_LIMIT	DEV_GET_RT_ON_POS_SOFT_LIMIT
GET_RT_OVERCUR_REACHED	DEV_GET_RT_OVERCUR_REACHED
GET_RT_POS_WINDOW_REACHED	DEV_GET_RT_POS_WINDOW_REACHED
GET_RT_POWER_LIMIT_REACHED	DEV_GET_RT_POWER_LIMIT_REACHED
GET_RT_RES_DIST_CLEARED	DEV_GET_RT_RES_DIST_CLEARED
GET_RT_SPEED_LIMIT	DEV_GET_RT_SPEED_LIMIT
GET_RT_SPEED_POS	DEV_GET_RT_SPEED_POS
GET_RT_SPEED_THRESHOLD	DEV_GET_RT_SPEED_THRESHOLD
GET_RT_SPEED_WINDOW_REACHED	DEV_GET_RT_SPEED_WINDOW_REACHED
GET_RT_SPEED_ZERO	DEV_GET_RT_SPEED_ZERO
GET_RT_TORQUE_LIMIT	DEV_GET_RT_TORQUE_LIMIT
GET_RT_TORQUE_THRESHOLD	DEV_GET_RT_TORQUE_THRESHOLD
DeviceAccessSync -Controller - Actual values	
GET_ACTUAL_POSITION	DEV_GET_ACTUAL_POSITION
GET_ACTUAL_SPEED	DEV_GET_ACTUAL_SPEED
GET_ACTUAL_TORQUE	DEV_GET_ACTUAL_TORQUE
-Controller - Set values - Preset values	
SET_PRE_SETPOINTS_SPEED	DEV_SET_PRE_SETPOINTS_SPEED
SET_PRE_SETPOINTS_TORQUE	DEV_SET_PRE_SETPOINTS_TORQUE
-Controller - Set values	
SET_SETPOINT_POSITION	DEV_SET_SETPOINT_POSITION
	DEV_SET_CTRL, DEV_GET_STAT
	DEV_GET_ACTUAL_POSITION 1)
	DEV_SET_ SETPOINT_POSITION_ABS 1)
	DEV_SET_SETPOINT_SPEED 1)
SET_SETPOINT_SPEED	DEV_SET_SETPOINT_SPEED
	DEV_SET_CTRL, DEV_GET_STAT
SET_SETPOINT_TORQUE	DEV_SET_SETPOINT_TORQUE
	DEV_SET_CTRL, DEV_GET_STAT
	DEV_SET_SETPOINT_SPEED 1)
-Process IO	
GET_ENCODER1_LATCH	DEV_GET_ENCODER1_LATCH
GET_ENCODER1_STATUS	DEV_GET_ENCODER1_STATUS
GET_ENCODER1_VALUE	DEV_GET_ENCODER1_VALUE
GET_INPUT_ANALOG1	DEV_GET_INPUT_ANALOG1
GET_INPUT_ANALOG1_STATUS	DEV_GET_INPUT_ANALOG1_STATUS
GET_INPUT_ANALOG2	DEV_GET_INPUT_ANALOG2
GET_INPUT_ANALOG2_STATUS	DEV_GET_INPUT_ANALOG2_STATUS
GET_SETPOINT_SRC1	DEV_GET_SETPOINT_SRC1
	DEV_GET_SETPOINT_SRC1_HIGH 1)
GET_SETPOINT_SRC2	DEV_GET_SETPOINT_SRC2
	DEV_GET_SETPOINT_SRC2_HIGH 1)

Block name (folder name)	Device information (ENUM value)
GET_TS_INPUT	DEV_GET_TS_INPUT
GET_TS_INPUT1_LATCH_NEG	DEV_GET_TS_INPUT1_LATCH_NEG
GET_TS_INPUT1_LATCH_POS	DEV_GET_TS_INPUT1_LATCH_POS
GET_TS_INPUT1_STATUS	DEV_GET_TS_INPUT1_STATUS
GET_TS_INPUT2_LATCH_NEG	DEV_GET_TS_INPUT2_LATCH_NEG
GET_TS_INPUT2_LATCH_POS	DEV_GET_TS_INPUT2_LATCH_POS
GET_TS_INPUT2_STATUS	DEV_GET_TS_INPUT2_STATUS
SET_ENCODER1_CONTROL	DEV_SET_ENCODER1_CONTROL
SET_INPUT_ANALOG1_CONTROL	DEV_SET_INPUT_ANALOG1_CONTROL
SET_INPUT_ANALOG2_CONTROL	DEV_SET_INPUT_ANALOG2_CONTROL
SET_TS_OUTPUT	DEV_SET_TS_OUTPUT
SET_TS_OUTPUT_ACTIVATE	DEV_SET_TS_OUTPUT_ACTIVATE
SET_TS_OUTPUT_TIME	DEV_SET_TS_OUTPUT_TIME
-TimeStamp	
CAM_CONT_TS	DEV_SET_TS_OUTPUT
	DEV_SET_TS_OUTPUT_ACTIVATE
	DEV_SET_TS_OUTPUT_TIME
GET_TS_INPUTS	DEV_GET_TS_INPUT
	DEV_GET_TS_INPUT1_LATCH_NEG
	DEV_GET_TS_INPUT1_LATCH_POS
	DEV_GET_TS_INPUT1_STATUS
	DEV_GET_TS_INPUT2_LATCH_NEG
	DEV_GET_TS_INPUT2_LATCH_POS
	DEV_GET_TS_INPUT2_STATUS
SET_TS_OUTPUTS	DEV_SET_TS_OUTPUT
	DEV_SET_TS_OUTPUT_ACTIVATE
	DEV_SET_TS_OUTPUT_TIME
DeviceCmd	
DO_CMD_ONCE	DEV_SET_CTRL
	DEV_GET_STAT
	DEV_SET_SETPOINT_SPEED 1)
Blocks for specific devices or bus systems	
Special -DeviceAccessAsync	
GET_ERROR_ID11	DEV_GET_ERROR_ID11
GET_STATUS_ID144	DEV_GET_STATUS_ID144
-AmkCanCommunication_ACC	
GET_ERROR_OPT	DEV_GET_ERROR_OPT
GET_ERROR_SYS	DEV_GET_ERROR_SYS
-Sercos - Command - Control	
SET_CTRL_RT_BIT1	DEV_SET_CTRL_RT_BIT1
SET_CTRL_RT_BIT2	DEV_SET_CTRL_RT_BIT2
-Sercos - Command - Status	
GET_STAT_RT_BIT1	DEV_GET_STAT_RT_BIT1
GET_STAT_RT_BIT2	DEV_GET_STAT_RT_BIT2
-Sercos - Error	
GET_STAT_CLASS2	DEV_GET_STAT_CLASS2
-DeviceAccessSync - AmkCanCommunication_ACC	

Block name (folder name)	Device information (ENUM value)
GET_ACTVAL16_0	DEV_GET_ACTVAL16_0
GET_ACTVAL16_1	DEV_GET_ACTVAL16_1
GET_ACTVAL16_2	DEV_GET_ACTVAL16_2
GET_ACTVAL32_0	DEV_GET_ACTVAL32_0
GET_ACTVAL32_1	DEV_GET_ACTVAL32_1
GET_MESSAGE16	DEV_GET_MESSAGE16
GET_MESSAGE32	DEV_GET_MESSAGE32
SET_ADD_SETPOINT16	DEV_SET_ADD_SETPOINT16
SET_ADD_SETPOINT32	DEV_SET_ADD_SETPOINT32
SET_MAIN_SETPOINT	DEV_SET_MAIN_SETPOINT
SET_SETPOINT16_0	DEV_SET_SETPOINT16_0
SET_SETPOINT16_1	DEV_SET_SETPOINT16_1
SET_SETPOINT16_2	DEV_SET_SETPOINT16_2
SET_SETPOINT16_3	DEV_SET_SETPOINT16_3
SET_SETPOINT32_0	DEV_SET_SETPOINT32_0
SET_SETPOINT32_1	DEV_SET_SETPOINT32_1
- Sercos	
GET_FOLLOW_ERR	DEV_GET_FOLLOW_ERR
SET_LIM_SPEED_BIPOL	DEV_SET_LIM_SPEED_BIPOL
SET_LIM_SPEED_POS	DEV_SET_LIM_SPEED_POS
SET_LIM_SPEED_NEG	DEV_SET_LIM_SPEED_NEG
SET_LIM_TORQUE_BIPOL	DEV_SET_LIM_TORQUE_BIPOL
SET_LIM_TORQUE_POS	DEV_SET_LIM_TORQUE_POS
SET_LIM_TORQUE_NEG	DEV_SET_LIM_TORQUE_NEG
SET_SETPOINT_MUL	DEV_SET_SETPOINT_MUL
SET_SETPOINT_DIV	DEV_SET_SETPOINT_DIV
SET_SETPOINT_SIWL	DEV_SET_SETPOINT_SIWL
- Sercos – Process IO	
GET_ACTPOS_LATCHED_NEG1	DEV_GET_ACTPOS_LATCHED_NEG1
GET_ACTPOS_LATCHED_NEG2	DEV_GET_ACTPOS_LATCHED_NEG2
GET_ACTPOS_LATCHED_POS1	DEV_GET_ACTPOS_LATCHED_POS1
GET_ACTPOS_LATCHED_POS2	DEV_GET_ACTPOS_LATCHED_POS2
GET_PROBE_STS	DEV_GET_PROBE_STS
Support -AmkCanCommunication_ACC	
DO_AFP	DEV_SET_CTRL
	DEV_GET_STAT
DO_AFP_ONCE	DEV_SET_CTRL
	DEV_GET_STAT
-Sercos	
CMD_BY_ID	-
DO_CMD	DEV_SET_CTRL
	DEV_GET_STAT
	DEV_SET_SETPOINT_SPEED
STATE_BY_ID	-

1) EtherCAT-specific

13.1.2.2 Parameterization

The following tables list the parameterization (parameter value) for various IDs based on the corresponding blocks in the AmkDevAccess library and the selected bus type. First, the 'ID description' provides an overview of the IDs that are currently relevant in the context of automatic bus configuration.

ID description

ID	Designation	Value	Meaning
32785	'Message 16'	84	Actual torque
32786	'Message 32'	40	Actual speed
32795	'Source UE'	5 9	DC bus enable via PLC DC bus enable via ID from master (KE)
32796	'Source RF'	5	Inverter on via PLC
32800	'AMK main operating mode'	410043 3C0043	Velocity control via PLC (new version) Velocity control via PLC (old version)
32801	'AMK secondary operating mode 1'	410404 3C0404	Position control via PLC (new version) Position control via PLC (old version)
32802	'AMK secondary operating mode 2'	410043 3C0043	Velocity control via PLC (new version) Velocity control via PLC (old version)
32803	'AMK secondary operating mode 3'	410002 3C0002	Torque control via PLC (new version) Torque control via PLC (old version)
32838-2	'Actual value list'	81	Torque feed-forward control
32838-12	'Actual value list'	37	Speed feed-forward control

ACC = Amk Can Communication

EC = EtherCAT

PLC = Programmable Logic Control

The following tables list specific parameter values for ACC, EtherCAT, and local bus based on the blocks used and the corresponding device (or the computer card used in the device) accessed with the block.

The specific parameterization for the local bus is used whenever the local axis is accessed via the KW-PLC2 option module (i.e. the drive of the DC bus in which the PLC module is located).

Parameterization specific to ACC

Block	Relevant ID	KE	KU	KW (R03)	KW (R05,R06), iX, iC, iDT5	KWZ	KWD	KWF	IDT4
DEFAULT_SET	32795	9							
GET_ACTUAL_TORQUE	32785		84	84		84	84		84
GET_ACTUAL_SPEED	32786		40	40		40	40		40
SET_CTRL_DC_BUSENABLE_x_UE	32795	5	5	5		5	5	5	
SET_CTRL_INVERTER_ON_x_RF	32796		5	5		5	5	5	5
SET_SETPOINT_POSITION	32800 32801		410404 410404	410404 410404		410404 410404	410404 410404		 410404
SET_SETPOINT_SPEED	32802		410043	410043		410043	410043		410043
SET_SETPOINT_TORQUE	32803		410002	410002		410002	410002		410002

Block	Relevant ID	KE	KU	KW (R03)	KW (R05,R06), iX, iC, iDT5	KWZ	KWD	KWF	IDT4
SET_MAIN_ SETPOINT	32800							F10003	
SET_PRE_ SETPOINT_ TORQUE	32838-2		81	81		81	81		81
SET_PRE_ SETPOINT_ SPEED	32838-12		37	37		37	37		37

Parameterization specific to EtherCAT

Block	Relevant ID	KE	KU	KW (R03)	KW (R05,R06), iX, iC, iDT5	KWZ	KWD	KWF	IDT4
DEFAULT_SET									
GET_ACTUAL_ TORQUE	32785								
GET_ACTUAL_ SPEED	32786								
SET_CTRL_DC_ BUSENABLE_x_UE	32795		5	5	5	5	5		
SET_CTRL_ INVERTER_ON_x_RF	32796		5	5	5	5	5		
SET_SETPOINT_ POSITION	32800 32801		410043 410404	410043 410404	410043 410404	410043 410404	410043 410404		
SET_SETPOINT_ SPEED	32802		410043	410043	410043	410043	410043		
SET_SETPOINT_ TORQUE	32803		410002	410002	410002	410002	410002		
SET_PRE_ SETPOINT_TORQUE	32838-2								
SET_PRE_ SETPOINT_SPEED	32838-12								

13.2 DeviceAccessAsync (asynchronous device access blocks)

Command

Control

SET_CTRL_DC_ BUSENABLE_x_UE	Set "DC bus enable" (UE)
SET_CTRL_ERR_RESET_x_ FL	Set "error reset" (FL)
SET_CTRL_INVERTER_ON_ x_RF	Set "inverter on" (RF)

Status

GET_STAT_DC_ BUSENABLE_ACK_x_QUE	Get "DC bus enable acknowledge" (QUE)
GET_STAT_ERR_RESET_ ACK_x_QFL	Get "error reset acknowledge" (QFL)
GET_STAT_INVERTER_ON_ ACK_x_QRF	Get "inverter on acknowledge" (QRF)

GET_STAT_SYSTEM_READY_x_SBM	Get "system ready" (SBM)

Error

GET_ERR_COMMUTATION	Get "commutation error"
GET_ERR_DC_BUS_OVERVOLT	Get "DC bus overvoltage error"
GET_ERR_DC_BUS_UNDERVOLT	Get "DC bus undervoltage error"
GET_ERR_ENCODER	Get "encoder error"
GET_ERR_EXCESS_FOLLOW	Get "excessive following error"
GET_ERR_MOTOR_OVERTEMP	Get "motor overtemperature"
GET_ERR_NOM_CUR_EXCESS	Get "nominal current excess" (I^2t monitoring)
GET_ERR_SHORT_CIRCUIT	Get "short-circuit or ground error"
GET_ERR_SUPPL_VOLT	Get "supply voltage error"

Real time

GET_RT_ACTVAL_NORM_ACK	Get "actual value normed acknowledge"
GET_RT_DRIVE_ANGLE_SYNC	Get "drive angle synchronous"
GET_RT_DRIVE_SPEED_SYNC	Get "drive speed synchronous"
GET_RT_ON_NEG_SOFT_LIMIT	Get "on negative software limit"
GET_RT_ON_POS_SOFT_LIMIT	Get "on positive software limit"
GET_RT_OVERCUR_REACHED	Get "overcurrent I^2t monitor reached > 50% load limit"
GET_RT_POS_WINDOW_REACHED	Get "position window reached"
GET_RT_POWER_LIMIT_REACHED	Get "power limit reached"
GET_RT_RES_DIST_CLEARED	Get "residual distance cleared"
GET_RT_SPEED_LIMIT	Get "speed limit"
GET_RT_SPEED_POS	Get "speed positive" (actual speed value >=0)
GET_RT_SPEED_THRESHOLD	Get "speed threshold"
GET_RT_SPEED_WINDOW_REACHED	Get "speed window reached"
GET_RT_SPEED_ZERO	Get "speed zero"
GET_RT_TORQUE_LIMIT	Get "torque limit"
GET_RT_TORQUE_THRESHOLD	Get "torque threshold"

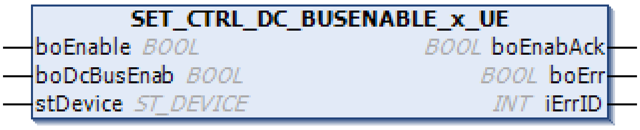
13.2.1 Command

13.2.1.1 Control

13.2.1.1.1 SET_CTRL_DC_BUSENABLE_x_UE (FB)

This block sets "DC bus enable" (UE) through the 'boDcBusEnab' variable.

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.
boDcBusEnab	BOOL	DC-Bus Enable (UE = converter on)

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table></td></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error:
<table>
<tr><td>Value</td><td>Meaning</td></tr>
<tr><td>1</td><td>Not configured device Information</td></tr>
<tr><td>2</td><td>Unassigned input / output variable</td></tr>
<tr><td>3</td><td>Invalid device instance (e.g. symbolic device identifier wrong assigned)</td></tr>
</table></td></tr>
</table>

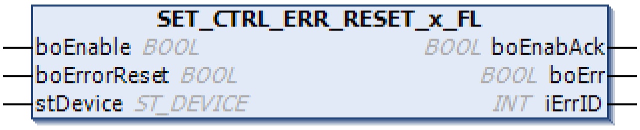
Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

13.2.1.1.2 SET_CTRL_ERR_RESET_x_FL (FB)

This block sets "error reset" (FL) through the 'boErrorReset' variable.

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.
boErrorReset	BOOL	Error Reset (FL = clear error)

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table></td></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error:
<table>
<tr><th>Value</th><th>Meaning</th></tr>
<tr><td>1</td><td>Not configured device Information</td></tr>
<tr><td>2</td><td>Unassigned input / output variable</td></tr>
<tr><td>3</td><td>Invalid device instance (e.g. symbolic device identifier wrong assigned)</td></tr>
</table></td></tr>
</table>

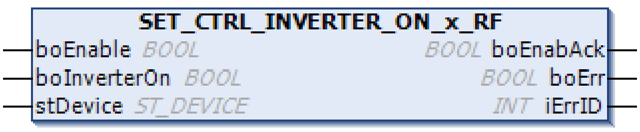
Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

13.2.1.1.3 SET_CTRL_INVERTER_ON_x_RF (FB)

This block sets "inverter on" (RF) through the 'boInverterOn' variable.

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.
boInverterOn	BOOL	Inverter On (RF = controller enable)

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table></td></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error:
<table>
<tr><td>Value</td><td>Meaning</td></tr>
<tr><td>1</td><td>Not configured device Information</td></tr>
<tr><td>2</td><td>Unassigned input / output variable</td></tr>
<tr><td>3</td><td>Invalid device instance (e.g. symbolic device identifier wrong assigned)</td></tr>
</table></td></tr>
</table>

Input and output variables

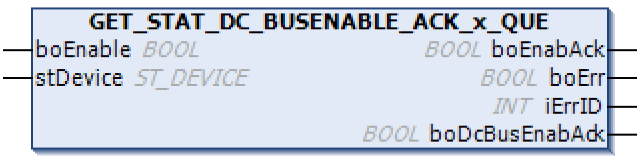
Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

13.2.1.2 Status

13.2.1.2.1 GET_STAT_DC_BUSENABLE_ACK_x_QUE (FB)

This block queries "DC bus enable acknowledge" (QUE) through the 'boDcBusEnabAck' variable.

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state<table><tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr><tr><td>TRUE</td><td>Error</td></tr></table></td></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output<table><tr><td colspan="2">iErrID = 0</td><td>No error</td></tr><tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr><tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr></table>Error:<table><tr><th>Value</th><th>Meaning</th></tr><tr><td>1</td><td>Not configured device Information</td></tr><tr><td>2</td><td>Unassigned input / output variable</td></tr><tr><td>3</td><td>Invalid device instance (e.g. symbolic device identifier wrong assigned)</td></tr></table></td></tr>
<tr><td>boDcBusEnabAck</td><td>BOOL</td><td>DC-Bus Enable Acknowledge (QUE = acknowledgement DC converter ON)</td></tr>
</table>

Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

13.2.1.2.2 GET_STAT_ERR_RESET_ACK_x_QFL (FB)

This block queries "error reset acknowledge" (QFL) through the 'boDcBusEnabAck' variable.

User interface

Input variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnable</td><td>BOOL</td><td>Enable signal: With a positive edge, the initialisation of the block starts.
As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC.
In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.</td></tr>
</table>

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state<table><tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr><tr><td>TRUE</td><td>Error</td></tr></table></td></tr>
</table>

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error:
<table>
<tr><td>Value</td><td>Meaning</td></tr>
<tr><td>1</td><td>Not configured device Information</td></tr>
<tr><td>2</td><td>Unassigned input / output variable</td></tr>
<tr><td>3</td><td>Invalid device instance (e.g. symbolic device identifier wrong assigned)</td></tr>
</table>
</td></tr>
<tr><td>boErrorResetAck</td><td>BOOL</td><td>Error Reset Acknowledge (QFL = acknowledgement clear error)</td></tr>
</table>

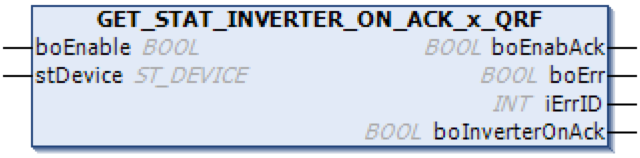
Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

13.2.1.2.3 GET_STAT_INVERTER_ON_ACK_x_QRF (FB)

This block queries "inverter on acknowledge" (QRF) through the 'boErrorResetAck' variable.

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error:
<table>
<tr><td>Value</td><td>Meaning</td></tr>
<tr><td>1</td><td>Not configured device Information</td></tr>
<tr><td>2</td><td>Unassigned input / output variable</td></tr>
<tr><td>3</td><td>Invalid device instance (e.g. symbolic device identifier wrong assigned)</td></tr>
</table>
</td></tr>
<tr><td>boInverterOnAck</td><td>BOOL</td><td>Inverter On Acknowledge (QRF = acknowledgement controller enable)</td></tr>
</table>

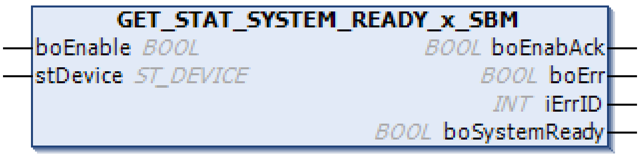
Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

13.2.1.2.4 GET_STAT_SYSTEM_READY_x_SBM (FB)

This block queries "system ready" (SBM) through the 'boSystemReady' variable.

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table>
</td></tr>
</table>

Name	Type	Description
iErrID	INT	Error identity number: Diagnostic number is output iErrID = 0 | No error iErrID ≠ 0 | boErr = TRUE | Error iErrID ≠ 0 | boErr = FALSE | Warning Error: Value | Meaning 1 | Not configured device Information 2 | Unassigned input / output variable 3 | Invalid device instance (e.g. symbolic device identifier wrong assigned)
boSystemReady	BOOL	System ready (SBM = system ready message)

Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

13.2.2 Error

13.2.2.1 GET_ERR_COMMUTATION (FB)

This block queries "commutation error" through the 'boCommutationError' variable.

User interface

Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.

Output variables

Name	Type	Description
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled
boErr	BOOL	The function block is in an error state FALSE | No error (permitted commanding or warning) TRUE | Error

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error:
<table>
<tr><td>Value</td><td>Meaning</td></tr>
<tr><td>1</td><td>Not configured device Information</td></tr>
<tr><td>2</td><td>Unassigned input / output variable</td></tr>
<tr><td>3</td><td>Invalid device instance (e.g. symbolic device identifier wrong assigned)</td></tr>
</table>
</td></tr>
<tr><td>boCommutationError</td><td>BOOL</td><td>Commutation Error</td></tr>
</table>

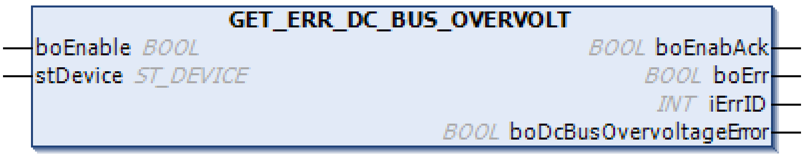
Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

13.2.2.2 GET_ERR_DC_BUS_OVERVOLT (FB)

This block queries "DC bus overvoltage error" through the 'boDcBusOvervoltageError' variable.

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error:
<table>
<tr><td>Value</td><td>Meaning</td></tr>
<tr><td>1</td><td>Not configured device Information</td></tr>
<tr><td>2</td><td>Unassigned input / output variable</td></tr>
<tr><td>3</td><td>Invalid device instance (e.g. symbolic device identifier wrong assigned)</td></tr>
</table>
</td></tr>
<tr><td>boDcBusOvervoltageError</td><td>BOOL</td><td>DC bus overvoltage error</td></tr>
</table>

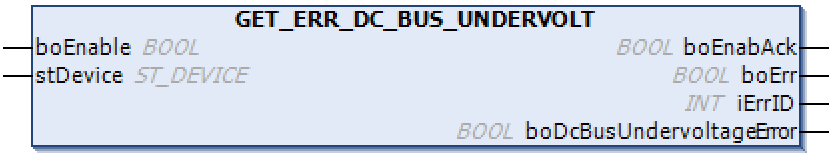
Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

13.2.2.3 GET_ERR_DC_BUS_UNDERVOLT (FB)

This block queries "DC bus undervoltage error" through the 'boDcBusUndervoltageError' variable.

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error:
<table>
<tr><td>Value</td><td>Meaning</td></tr>
<tr><td>1</td><td>Not configured device Information</td></tr>
<tr><td>2</td><td>Unassigned input / output variable</td></tr>
<tr><td>3</td><td>Invalid device instance (e.g. symbolic device identifier wrong assigned)</td></tr>
</table>
</td></tr>
<tr><td>boDcBusUndervoltageError</td><td>BOOL</td><td>DC bus undervoltage error</td></tr>
</table>

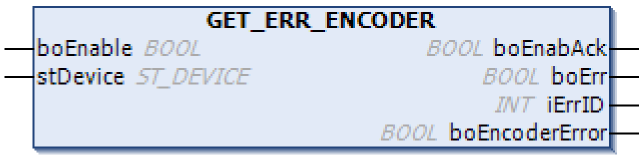
Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

13.2.2.4 GET_ERR_ENCODER (FB)

This block queries "encoder error" through the 'boEncoderError' variable.

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error:
<table>
<tr><td>Value</td><td>Meaning</td></tr>
<tr><td>1</td><td>Not configured device Information</td></tr>
<tr><td>2</td><td>Unassigned input / output variable</td></tr>
<tr><td>3</td><td>Invalid device instance (e.g. symbolic device identifier wrong assigned)</td></tr>
</table>
</td></tr>
<tr><td>boEncoderError</td><td>BOOL</td><td>Encoder error</td></tr>
</table>

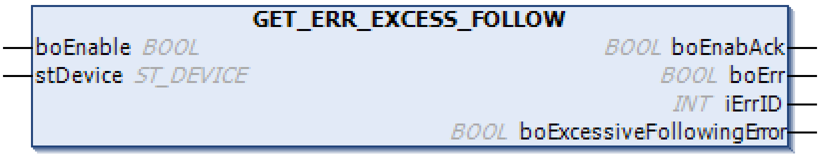
Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

13.2.2.5 GET_ERR_EXCESS_FOLLOW (FB)

This block queries "excessive following error" (ID159 'Excess error') through the 'boExessiveFollowingError' variable.

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table>
</td></tr>
</table>

Name	Type	Description
iErrID	INT	Error identity number: Diagnostic number is output iErrID = 0 — No error iErrID ≠ 0, boErr = TRUE — Error iErrID ≠ 0, boErr = FALSE — Warning Error: Value — Meaning 1 — Not configured device Information 2 — Unassigned input / output variable 3 — Invalid device instance (e.g. symbolic device identifier wrong assigned)
boExcessiveFollowingError	BOOL	ID159 'Excess error'

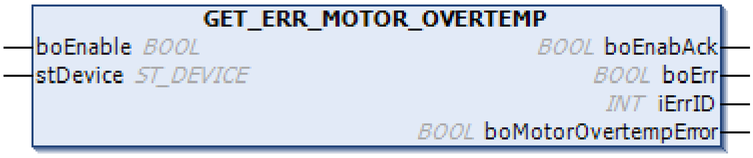
Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

13.2.2.6 GET_ERR_MOTOR_OVERTEMP (FB)

This block queries "motor overtemperature" through the 'boMotorOvertempError' variable.

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.

Output variables

Name	Type	Description
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled
boErr	BOOL	The function block is in an error state FALSE — No error (permitted commanding or warning) TRUE — Error

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error:
<table>
<tr><td>Value</td><td>Meaning</td></tr>
<tr><td>1</td><td>Not configured device Information</td></tr>
<tr><td>2</td><td>Unassigned input / output variable</td></tr>
<tr><td>3</td><td>Invalid device instance (e.g. symbolic device identifier wrong assigned)</td></tr>
</table>
</td></tr>
<tr><td>boMotorOvertempError</td><td>BOOL</td><td>Overtemperature motor</td></tr>
</table>

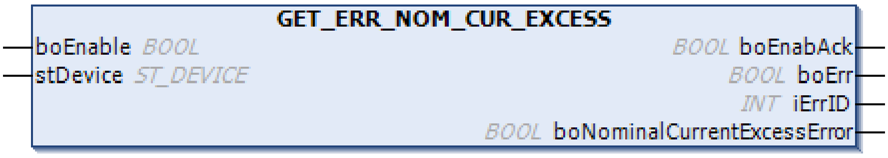
Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

13.2.2.7 GET_ERR_NOM_CUR_EXCESS (FB)

This block queries "nominal current excess" (I^2t monitoring) through the 'boNominalCurrentExcessError' variable.

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error:
<table>
<tr><td>Value</td><td>Meaning</td></tr>
<tr><td>1</td><td>Not configured device Information</td></tr>
<tr><td>2</td><td>Unassigned input / output variable</td></tr>
<tr><td>3</td><td>Invalid device instance (e.g. symbolic device identifier wrong assigned)</td></tr>
</table>
</td></tr>
<tr><td>boNominalCurrentExcessError</td><td>BOOL</td><td>Overcurrent monitoring (I²t) triggered</td></tr>
</table>

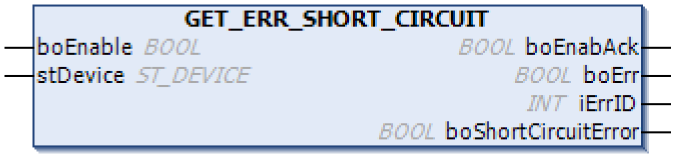
Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

13.2.2.8 GET_ERR_SHORT_CIRCUIT (FB)

This block queries "short-circuit or ground error" through the 'boShortCircuitError' variable.

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error:
<table>
<tr><td>Value</td><td>Meaning</td></tr>
<tr><td>1</td><td>Not configured device Information</td></tr>
<tr><td>2</td><td>Unassigned input / output variable</td></tr>
<tr><td>3</td><td>Invalid device instance (e.g. symbolic device identifier wrong assigned)</td></tr>
</table>
</td></tr>
<tr><td>boShortCircuitError</td><td>BOOL</td><td>Short circuit / ground fault</td></tr>
</table>

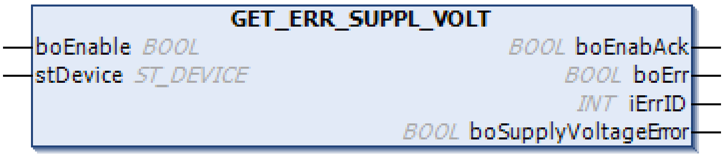
Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

13.2.2.9 GET_ERR_SUPPL_VOLT (FB)

This block queries "supply voltage error" through the 'boSupplyVoltageError' variable.

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error:
<table>
<tr><td>Value</td><td>Meaning</td></tr>
<tr><td>1</td><td>Not configured device Information</td></tr>
<tr><td>2</td><td>Unassigned input / output variable</td></tr>
<tr><td>3</td><td>Invalid device instance (e.g. symbolic device identifier wrong assigned)</td></tr>
</table>
</td></tr>
<tr><td>boSupplyVoltageError</td><td>BOOL</td><td>Supply voltage error</td></tr>
</table>

Input and output variables

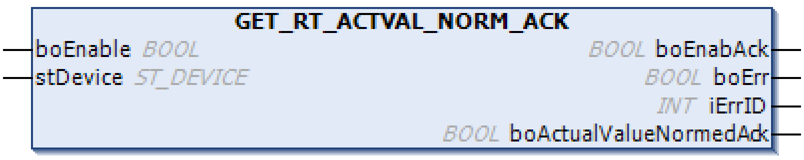
Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

13.2.3 Realtime

13.2.3.1 GET_RT_ACTVAL_NORM_ACK (FB)

This block queries 'boActualValueNormedAck' (actual value normed acknowledge)

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error:
<table>
<tr><td>Value</td><td>Meaning</td></tr>
<tr><td>1</td><td>Not configured device Information</td></tr>
<tr><td>2</td><td>Unassigned input / output variable</td></tr>
<tr><td>3</td><td>Invalid device instance (e.g. symbolic device identifier wrong assigned)</td></tr>
</table>
</td></tr>
<tr><td>boActualValueNormedAck</td><td>BOOL</td><td>Acknowledgement: actual value scaled</td></tr>
</table>

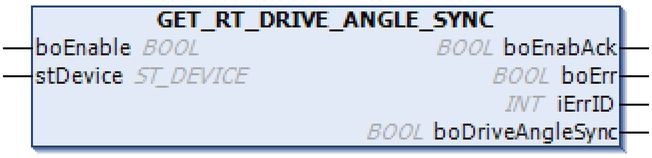
Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

13.2.3.2 GET_RT_DRIVE_ANGLE_SYNC (FB)

This block queries "drive angle synchronous" (drive according to ID228 'Synchron position window') through the 'boDriveAngleSync' variable.

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error:
<table>
<tr><td>Value</td><td>Meaning</td></tr>
<tr><td>1</td><td>Not configured device Information</td></tr>
<tr><td>2</td><td>Unassigned input / output variable</td></tr>
<tr><td>3</td><td>Invalid device instance (e.g. symbolic device identifier wrong assigned)</td></tr>
</table>
</td></tr>
<tr><td>boDriveAngleSync</td><td>BOOL</td><td>Drive according to ID228 'Synchron position window'</td></tr>
</table>

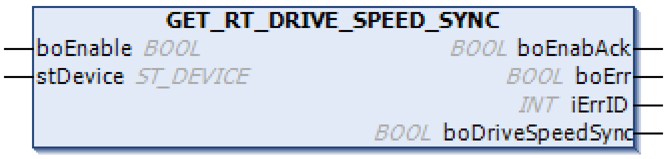
Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

13.2.3.3 GET_RT_DRIVE_SPEED_SYNC (FB)

This block queries 'boDriveSpeedSync' (drive according to ID32952 'At synchronous speed window').

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error:
<table>
<tr><td>Value</td><td>Meaning</td></tr>
<tr><td>1</td><td>Not configured device Information</td></tr>
<tr><td>2</td><td>Unassigned input / output variable</td></tr>
<tr><td>3</td><td>Invalid device instance (e.g. symbolic device identifier wrong assigned)</td></tr>
</table>
</td></tr>
<tr><td>boDriveSpeedSync</td><td>BOOL</td><td>Drive according to ID32952 'At synchronous speed window'</td></tr>
</table>

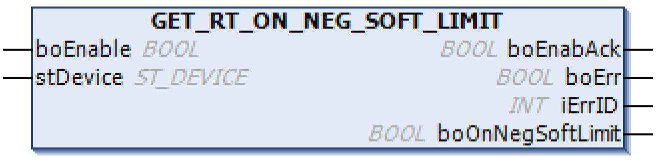
Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

13.2.3.4 GET_RT_ON_NEG_SOFT_LIMIT (FB)

This block queries "on negative software limit" (software end limit according to ID50 'Negative position limit') through the 'boOnNegSoftLimit' variable.

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error:
<table>
<tr><td>Value</td><td>Meaning</td></tr>
<tr><td>1</td><td>Not configured device Information</td></tr>
<tr><td>2</td><td>Unassigned input / output variable</td></tr>
<tr><td>3</td><td>Invalid device instance (e.g. symbolic device identifier wrong assigned)</td></tr>
</table>
</td></tr>
<tr><td>boOnNegSoftLimit</td><td>BOOL</td><td>Software limit according to ID50 'Negative position limit'</td></tr>
</table>

Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

13.2.3.5 GET_RT_ON_POS_SOFT_LIMIT (FB)

This block queries "on positive software limit" (software end limit according to ID49 'Positive position limit') through the 'boOnPosSoftLimit' variable.

User interface

Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table>
</td></tr>
</table>

Name	Type	Description
iErrID	INT	Error identity number: Diagnostic number is output iErrID = 0 — No error iErrID ≠ 0 — boErr = TRUE — Error iErrID ≠ 0 — boErr = FALSE — Warning Error: Value — Meaning 1 — Not configured device Information 2 — Unassigned input / output variable 3 — Invalid device instance (e.g. symbolic device identifier wrong assigned)
boOnPosSoftLimit	BOOL	Software limit according to ID49 'Positive position limit'

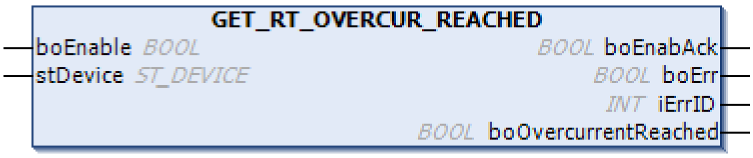
Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

13.2.3.6 GET_RT_OVERCUR_REACHED (FB)

This block queries "overcurrent I²t monitor reached > 50% load limit" through the 'boOvercurrentReached' variable.

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.

Output variables

Name	Type	Description
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled
boErr	BOOL	The function block is in an error state FALSE — No error (permitted commanding or warning) TRUE — Error

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error:
<table>
<tr><td>Value</td><td>Meaning</td></tr>
<tr><td>1</td><td>Not configured device Information</td></tr>
<tr><td>2</td><td>Unassigned input / output variable</td></tr>
<tr><td>3</td><td>Invalid device instance (e.g. symbolic device identifier wrong assigned)</td></tr>
</table>
</td></tr>
<tr><td>boOvercurrentReached</td><td>BOOL</td><td>Overcurrent message (I²t): load > 50% overload limit</td></tr>
</table>

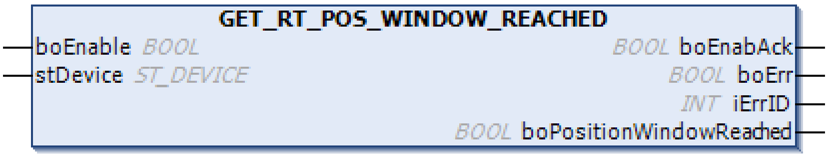
Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

13.2.3.7 GET_RT_POS_WINDOW_REACHED (FB)

This block queries "position window reached; |Xset-Xact|<InPositionWindow" (according to ID57 'In position window') through the 'boPositionWindowReached' variable.

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error:
<table>
<tr><td>Value</td><td>Meaning</td></tr>
<tr><td>1</td><td>Not configured device Information</td></tr>
<tr><td>2</td><td>Unassigned input / output variable</td></tr>
<tr><td>3</td><td>Invalid device instance (e.g. symbolic device identifier wrong assigned)</td></tr>
</table>
</td></tr>
<tr><td>boPositionWindowReached</td><td>BOOL</td><td>according to ID57 'In position window'</td></tr>
</table>

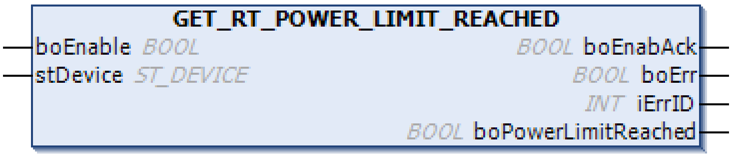
Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

13.2.3.8 GET_RT_POWER_LIMIT_REACHED (FB)

This block queries "power limit reached; |Pact|>PowerLimit" (according to ID158 'Power threshold') through the 'boPowerLimitReached' variable.

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error:
<table>
<tr><td>Value</td><td>Meaning</td></tr>
<tr><td>1</td><td>Not configured device Information</td></tr>
<tr><td>2</td><td>Unassigned input / output variable</td></tr>
<tr><td>3</td><td>Invalid device instance (e.g. symbolic device identifier wrong assigned)</td></tr>
</table>
</td></tr>
<tr><td>boPowerLimitReached</td><td>BOOL</td><td>according to ID158 'Power threshold'</td></tr>
</table>

Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

13.2.3.9 GET_RT_RES_DIST_CLEARED (FB)

This block queries "residual distance cleared" (according to ID32922 'Residual distance erase window') through the 'boResidualDistanceCleared' variable.

User interface

Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error:
<table>
<tr><td>Value</td><td>Meaning</td></tr>
<tr><td>1</td><td>Not configured device Information</td></tr>
<tr><td>2</td><td>Unassigned input / output variable</td></tr>
<tr><td>3</td><td>Invalid device instance (e.g. symbolic device identifier wrong assigned)</td></tr>
</table>
</td></tr>
<tr><td>boResidualDistanceCleared</td><td>BOOL</td><td>according to ID32922 'Residual distance erase window'</td></tr>
</table>

Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

13.2.3.10 GET_RT_SPEED_LIMIT (FB)

This block queries "speed limit; |Nset|>SpeedLimit" (speed limit according to ID38 'Positive velocity limit' / ID39 'Negative velocity limit') through the 'boSpeedLimit' variable.

User interface

Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error:
<table>
<tr><td>Value</td><td>Meaning</td></tr>
<tr><td>1</td><td>Not configured device Information</td></tr>
<tr><td>2</td><td>Unassigned input / output variable</td></tr>
<tr><td>3</td><td>Invalid device instance (e.g. symbolic device identifier wrong assigned)</td></tr>
</table>
</td></tr>
<tr><td>boSpeedLimit</td><td>BOOL</td><td>Speed limit according to ID38 'Positive velocity limit' / ID39 'Negative velocity limit'</td></tr>
</table>

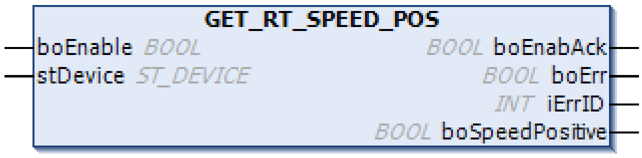
Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

13.2.3.11 GET_RT_SPEED_POS (FB)

This block queries "speed positive" (actual speed value >=0) through the 'boSpeedPositive' variable.

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error:
<table>
<tr><td>Value</td><td>Meaning</td></tr>
<tr><td>1</td><td>Not configured device Information</td></tr>
<tr><td>2</td><td>Unassigned input / output variable</td></tr>
<tr><td>3</td><td>Invalid device instance (e.g. symbolic device identifier wrong assigned)</td></tr>
</table>
</td></tr>
<tr><td>boSpeedPositive</td><td>BOOL</td><td>Actual speed value ≥0</td></tr>
</table>

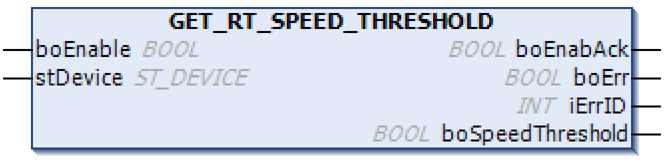
Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

13.2.3.12 GET_RT_SPEED_THRESHOLD (FB)

This block queries "speed threshold; |Nact|<SpeedThreshold" (according to ID125 'Velocity threshold') through the 'boSpeedThreshold' variable.

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error:
<table>
<tr><th>Value</th><th>Meaning</th></tr>
<tr><td>1</td><td>Not configured device Information</td></tr>
<tr><td>2</td><td>Unassigned input / output variable</td></tr>
<tr><td>3</td><td>Invalid device instance (e.g. symbolic device identifier wrong assigned)</td></tr>
</table>
</td></tr>
<tr><td>boSpeedThreshold</td><td>BOOL</td><td>according to ID125 'Velocity threshold'</td></tr>
</table>

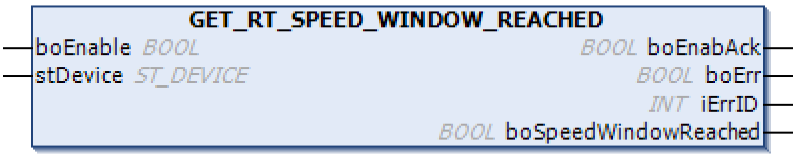
Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

13.2.3.13 GET_RT_SPEED_WINDOW_REACHED (FB)

This block queries "speed window reached; |Nset-Nact|<SpeedWindow" (according to ID157 'Velocity window') through the 'boSpeedWindowReached' variable.

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error:
<table>
<tr><td>Value</td><td>Meaning</td></tr>
<tr><td>1</td><td>Not configured device Information</td></tr>
<tr><td>2</td><td>Unassigned input / output variable</td></tr>
<tr><td>3</td><td>Invalid device instance (e.g. symbolic device identifier wrong assigned)</td></tr>
</table>
</td></tr>
<tr><td>boSpeedWindowReached</td><td>BOOL</td><td>according to ID157 'Velocity window' reached</td></tr>
</table>

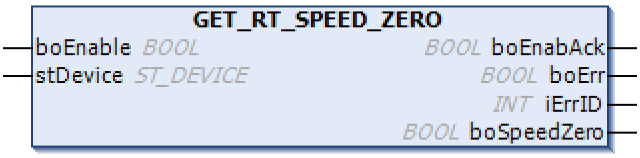
Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

13.2.3.14 GET_RT_SPEED_ZERO (FB)

This block queries "speed zero; |Nact|<ZeroWindow" (speed threshold according to ID124 'Zero velocity window') through the 'boSpeedZero' variable.

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error:
<table>
<tr><td>Value</td><td>Meaning</td></tr>
<tr><td>1</td><td>Not configured device Information</td></tr>
<tr><td>2</td><td>Unassigned input / output variable</td></tr>
<tr><td>3</td><td>Invalid device instance (e.g. symbolic device identifier wrong assigned)</td></tr>
</table></td></tr>
<tr><td>boSpeedZero</td><td>BOOL</td><td>Speed threshold according to ID124 'Zero velocity window'</td></tr>
</table>

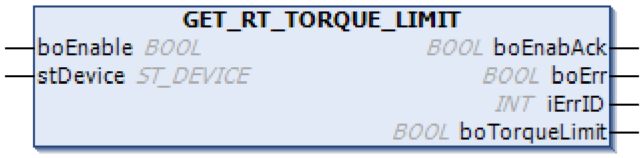
Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

13.2.3.15 GET_RT_TORQUE_LIMIT (FB)

This block queries "torque limit; |Mset|>TorqueLimit" (torque limit according to ID82 'Positive torque limit' / ID83 'Negative torque limit') through the 'boTorqueLimit' variable.

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table></td></tr>
</table>

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error:
<table>
<tr><td>Value</td><td>Meaning</td></tr>
<tr><td>1</td><td>Not configured device Information</td></tr>
<tr><td>2</td><td>Unassigned input / output variable</td></tr>
<tr><td>3</td><td>Invalid device instance (e.g. symbolic device identifier wrong assigned)</td></tr>
</table>
</td></tr>
<tr><td>boTorqueLimit</td><td>BOOL</td><td>Torque limit according to ID82 'Positive torque limit' / ID83 'Negative torque limit'</td></tr>
</table>

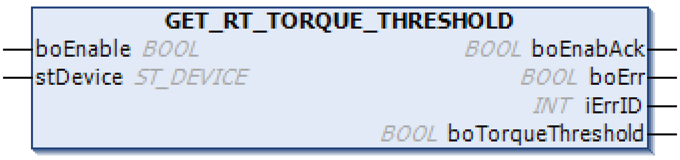
Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

13.2.3.16 GET_RT_TORQUE_THRESHOLD (FB)

This block queries "torque threshold; |Mact|>TorqueThreshold" (according to ID126 'Torque threshold') through the 'boTorqueThreshold' variable.

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error:
<table>
<tr><td>Value</td><td>Meaning</td></tr>
<tr><td>1</td><td>Not configured device Information</td></tr>
<tr><td>2</td><td>Unassigned input / output variable</td></tr>
<tr><td>3</td><td>Invalid device instance (e.g. symbolic device identifier wrong assigned)</td></tr>
</table>
</td></tr>
<tr><td>boTorqueThreshold</td><td>BOOL</td><td>according to ID126 'Torque threshold'</td></tr>
</table>

Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

13.3 DeviceAccessSync (synchronous device access blocks)

Controller

Actual values

GET_ACTUAL_POSITION	Get "actual position"
GET_ACTUAL_SPEED	Get "actual speed"
GET_ACTUAL_TORQUE	Get "actual torque"

Preset values

SET_PRE_SETPOINTS_SPEED	Set "precontrol speed setpoint"
SET_PRE_SETPOINTS_TORQUE	Set "precontrol torque setpoint"

Set values

SET_SETPOINT_POSITION	Set "position setpoint"
SET_SETPOINT_SPEED	Set "speed setpoint"
SET_SETPOINT_TORQUE	Set "torque setpoint"

ProcessIO

GET_ENCODER1_LATCH	Get latched encoder 1 value
GET_ENCODER1_STATUS	Get encoder 1 status information
GET_ENCODER1_VALUE	Get encoder 1 value
GET_INPUT_ANALOG1	Get analog input 1 (A1)
GET_INPUT_ANALOG1_STATUS	Get analog input 1 status
GET_INPUT_ANALOG2	Get analog input 2 (A2)
GET_INPUT_ANALOG2_STATUS	Get analog input 2 status
GET_SETPOINT_SRC1	Get ID32948, message 1 "configurable value"
GET_SETPOINT_SRC2	Get ID32948, message 2 "configurable value"
GET_TS_INPUT	Get "TimeStamp" inputs

GET_TS_INPUT1_LATCH_NEG	Get negative input 1 edge, latched "TimeStamp1" time information
GET_TS_INPUT1_LATCH_POS	Get positive input 1 edge, latched "TimeStamp1" time information
GET_TS_INPUT1_STATUS	Get "TimeStamp1" status information
GET_TS_INPUT2_LATCH_NEG	Get negative input 2 edge, latched "TimeStamp2" time information
GET_TS_INPUT2_LATCH_POS	Get positive input 2 edge, latched "TimeStamp2" time information
GET_TS_INPUT2_STATUS	Get "TimeStamp2" status information
SET_ENCODER1_CONTROL	Set encoder 1 control information
SET_INPUT_ANALOG1_CONTROL	Set analog 1 control information
SET_INPUT_ANALOG2_CONTROL	Set analog 2 control information
SET_TS_OUTPUT	Set "TimeStamp" outputs
SET_TS_OUTPUT_ACTIVATE	Set "TimeStamp" output activation information
SET_TS_OUTPUT_TIME	Set "TimeStamp" output time

TimeStamp

CAM_CONT_TS	Camshaft control for highly accurate control of the "TimeStamp" outputs
GET_TS_INPUTS	Get state of "TimeStamp" inputs
SET_TS_OUTPUTS	Set state of "TimeStamp" outputs

13.3.1 Controller

13.3.1.1 Actualvalues

13.3.1.1.1 GET_ACTUAL_POSITION (FB)

This block queries "actual position" through the 'diActualPosition' variable.

User interface

Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.

Output variables

Name	Type	Description
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled

Name	Type	Description
boErr	BOOL	The function block is in an error state FALSE: No error (permitted commanding or warning) TRUE: Error
iErrID	INT	Error identity number: Diagnostic number is output iErrID = 0: No error iErrID ≠ 0, boErr = TRUE: Error iErrID ≠ 0, boErr = FALSE: Warning Error: Value / Meaning 1: Not configured device Information 2: Unassigned input / output variable 3: Invalid device instance (e.g. symbolic device identifier wrong assigned)
diActualPosition	DINT	Actual position

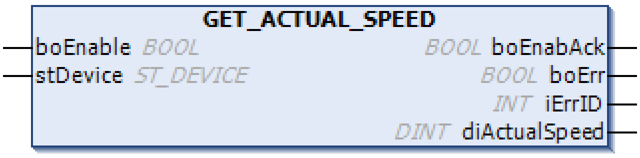
Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

13.3.1.1.2 GET_ACTUAL_SPEED (FB)

This block queries "actual speed" through the 'diActualSpeed' variable.

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.

Output variables

Name	Type	Description
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled
boErr	BOOL	The function block is in an error state FALSE: No error (permitted commanding or warning) TRUE: Error

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error:
<table>
<tr><td>Value</td><td>Meaning</td></tr>
<tr><td>1</td><td>Not configured device Information</td></tr>
<tr><td>2</td><td>Unassigned input / output variable</td></tr>
<tr><td>3</td><td>Invalid device instance (e.g. symbolic device identifier wrong assigned)</td></tr>
</table>
</td></tr>
<tr><td>diActualSpeed</td><td>DINT</td><td>Actual velocity</td></tr>
</table>

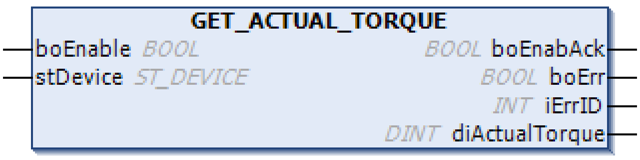
Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

13.3.1.1.3 GET_ACTUAL_TORQUE (FB)

This block queries "actual torque" through the 'diActualTorque' variable.
The query involves implicit type conversion if the device information for 'diActualTorque' is only transferred as an INT value.

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error:
<table>
<tr><td>Value</td><td>Meaning</td></tr>
<tr><td>1</td><td>Not configured device Information</td></tr>
<tr><td>2</td><td>Unassigned input / output variable</td></tr>
<tr><td>3</td><td>Invalid device instance (e.g. symbolic device identifier wrong assigned)</td></tr>
</table>
</td></tr>
<tr><td>diActualTorque</td><td>DINT</td><td>Current actual torque value [0.1% Mn]</td></tr>
</table>

Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

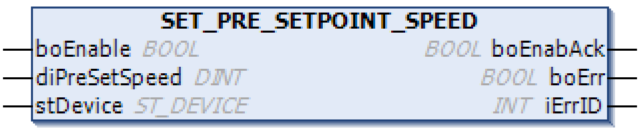
13.3.1.2 Setvalues

13.3.1.2.1 PreSetValues

13.3.1.2.1.1 SET_PRE_SETPOINTS_SPEED (FB)

This block sets the "precontrol speed setpoint" through the 'diPreSetSpeed' variable.

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.
diPreSetSpeed	DINT	Velocity feed-forward setpoint

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error:
<table>
<tr><td>Value</td><td>Meaning</td></tr>
<tr><td>1</td><td>Device information not configured</td></tr>
<tr><td>2</td><td>No input / output variable assigned (copy pointer = 0)</td></tr>
<tr><td>3</td><td>Illegal device instance (symbolic
device name might have been assigned incorrectly)</td></tr>
</table>
</td></tr>
</table>

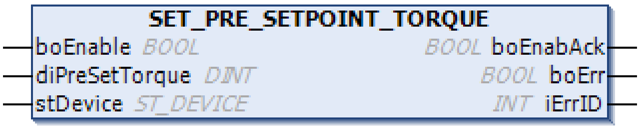
Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

13.3.1.2.1.2 SET_PRE_SETPOINTS_TORQUE (FB)

This block sets the "precontrol torque setpoint" through the 'diPreSetTorque' variable.

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.
diPreSetTorque	DINT	Torque feed-forward setpoint

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error:
<table>
<tr><th>Value</th><th>Meaning</th></tr>
<tr><td>1</td><td>Device information not configured</td></tr>
<tr><td>2</td><td>No input / output variable assigned (copy pointer = 0)</td></tr>
<tr><td>3</td><td>Illegal device instance (symbolic
device name might have been assigned incorrectly)</td></tr>
</table>
</td></tr>
</table>

Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

13.3.1.2.2 SET_SETPOINT_POSITION (FB)

This block sets "position setpoint" through the 'diSetPosition' variable.

When the block is set, from the point in time at which 'boEnable'=TRUE:

- One-off change to NBA1 (secondary operating mode 1: position control) and 'diSetPositon'(k-1) = 'diSetPositon'(k)' is set. With EtherCAT, "position setpoint = actual position" is also set.
- The differences resulting from 'diSetPositon'(k) - 'diSetPositon'(k-1)' are continuously added to the position setpoint.
- The current position setpoint is output at the 'diActSetPos' output variable.

User interface

Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.
diSetPosition	DINT	Specification of the position setpoint (position setpoint system) [increments]

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error:
<table>
<tr><td>Value</td><td>Meaning</td></tr>
<tr><td>1</td><td>Device information not configured</td></tr>
<tr><td>2</td><td>No input / output variable assigned (copy pointer = 0)</td></tr>
<tr><td>3</td><td>Illegal device instance (symbolic
device name might have been assigned incorrectly)</td></tr>
</table>
</td></tr>
<tr><td>diActSetPos</td><td>DINT</td><td>Current position setpoint</td></tr>
</table>

Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

Actions

Name	Description
SetAutoMode()	Sets the default mode. This corresponds to the behavior of the block described above.
SetIncMode()	This action corresponds to the incremental behavior described above; when 'boEnable'=TRUE, "position setpoint = actual position" is always set (regardless of the bus).
SetAbsMode()	Corresponds to an absolute position value default. In other words, 'diSetPosition' is specified directly as the position setpoint (no adaptation to the current actual position).

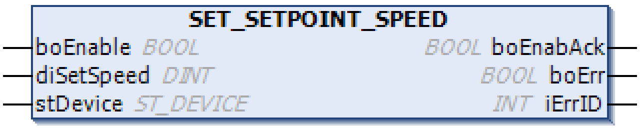
13.3.1.2.3 SET_SETPOINT_SPEED (FB)

This block sets the "speed setpoint" through the 'diSetSpeed' variable.

When the block is set, from the point in time at which 'boEnable'=TRUE:

- One-off change to NBA2 (secondary operating mode 2: speed control).
- The 'diSetSpeed' variable is output as the setpoint speed.

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.
diSetSpeed	DINT	Set the velocity setpoint

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table></td></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error:
<table>
<tr><td>Value</td><td>Meaning</td></tr>
<tr><td>1</td><td>Device information not configured</td></tr>
<tr><td>2</td><td>No input / output variable assigned (copy pointer = 0)</td></tr>
<tr><td>3</td><td>Illegal device instance (symbolic
device name might have been assigned incorrectly)</td></tr>
</table></td></tr>
</table>

Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

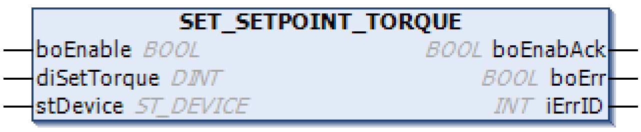
13.3.1.2.4 SET_SETPOINT_TORQUE (FB)

This block sets "torque setpoint" through the 'diSetTorque' variable.

When the block is set, from the point in time at which 'boEnable'=TRUE:

- One-off change to NBA3 (secondary operating mode 3: torque control).
- The 'diSetTorque' variable is output as the set torque.

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.
diSetTorque	DINT	Specification of the torque setpoint [0.1% Mn]

Output variables

Name	Type	Description
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled

Name	Type	Description
boErr	BOOL	The function block is in an error state FALSE: No error (permitted commanding or warning) TRUE: Error
iErrID	INT	Error identity number: Diagnostic number is output iErrID = 0: No error iErrID ≠ 0, boErr = TRUE: Error iErrID ≠ 0, boErr = FALSE: Warning Error: Value / Meaning 1: Device information not configured 2: No input / output variable assigned (copy pointer = 0) 3: Illegal device instance (symbolic device name might have been assigned incorrectly)

Input and output variables

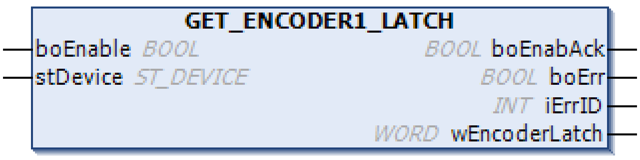
Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

13.3.2 ProcessIO

13.3.2.1 GET_ENCODER1_LATCH (FB)

This block synchronously queries the latched encoder value through the 'wEncoderLatch' variable.

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.

Output variables

Name	Type	Description
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled
boErr	BOOL	The function block is in an error state FALSE: No error (permitted commanding or warning) TRUE: Error

Name	Type	Description
iErrID	INT	Error identity number: Diagnostic number is output iErrID = 0 | No error iErrID ≠ 0 | boErr = TRUE | Error iErrID ≠ 0 | boErr = FALSE | Warning Error: Value | Meaning 1 | Device information not configured 2 | No input / output variable assigned (copy pointer = 0) 3 | Illegal device instance (symbolic device name might have been assigned incorrectly)
wEncoderLatch	WORD	Get latched encoder value

Input and output variables

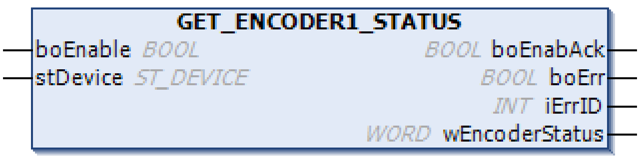
Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

13.3.2.2 GET_ENCODER1_STATUS (FB)

This block synchronously queries the latched encoder value through the 'wEncoderLatch' variable.

Only in conjunction with A5x MxE controllers.

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.

Output variables

Name	Type	Description
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled
boErr	BOOL	The function block is in an error state FALSE | No error (permitted commanding or warning) TRUE | Error

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error:
<table>
<tr><td>Value</td><td>Meaning</td></tr>
<tr><td>1</td><td>Device information not configured</td></tr>
<tr><td>2</td><td>No input / output variable assigned (copy pointer = 0)</td></tr>
<tr><td>3</td><td>Illegal device instance (symbolic
device name might have been assigned incorrectly)</td></tr>
</table>
</td></tr>
<tr><td>wEncoderStatus</td><td>WORD</td><td>Encoder status information</td></tr>
</table>

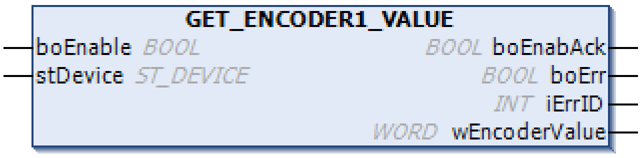
Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

13.3.2.3 GET_ENCODER1_VALUE (FB)

This block synchronously queries the current encoder value through the 'wEncoderValue' variable.

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table>
</td></tr>
</table>

Name	Type	Description
iErrID	INT	Error identity number: Diagnostic number is output iErrID = 0 – No error iErrID ≠ 0 – boErr = TRUE – Error iErrID ≠ 0 – boErr = FALSE – Warning Error: Value – Meaning 1 – Device information not configured 2 – No input / output variable assigned (copy pointer = 0) 3 – Illegal device instance (symbolic device name might have been assigned incorrectly)
wEncoderValue	WORD	Current encoder value

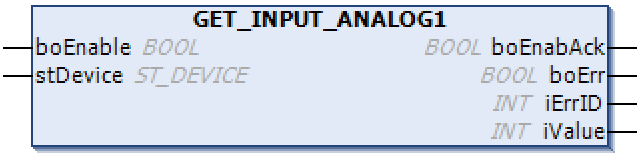
Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

13.3.2.4 GET_INPUT_ANALOG1 (FB)

This block synchronously queries analog input 1 (A1) through the 'iValue' variable.

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.

Output variables

Name	Type	Description
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled
boErr	BOOL	The function block is in an error state FALSE – No error (permitted commanding or warning) TRUE – Error

Name	Type	Description
iErrID	INT	Error identity number: Diagnostic number is output iErrID = 0: No error iErrID ≠ 0, boErr = TRUE: Error iErrID ≠ 0, boErr = FALSE: Warning Error: Value / Meaning 1: Device information not configured 2: No input / output variable assigned (copy pointer = 0) 3: Illegal device instance (symbolic device name might have been assigned incorrectly)
iValue	INT	Analog input voltage according to ID32897 'Analog Input A1'

Input and output variables

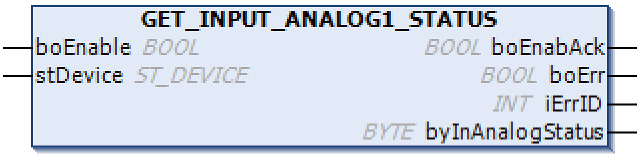
Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

13.3.2.5 GET_INPUT_ANALOG1_STATUS (FB)

This block synchronously queries the status of analog input 1 through the 'byInAnalogStatus' variable.

Only in conjunction with A5x MxE controllers.

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.

Output variables

Name	Type	Description
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled
boErr	BOOL	The function block is in an error state FALSE: No error (permitted commanding or warning) TRUE: Error

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error:
<table>
<tr><td>Value</td><td>Meaning</td></tr>
<tr><td>1</td><td>Device information not configured</td></tr>
<tr><td>2</td><td>No input / output variable assigned (copy pointer = 0)</td></tr>
<tr><td>3</td><td>Illegal device instance (symbolic
device name might have been assigned incorrectly)</td></tr>
</table>
</td></tr>
<tr><td>byInAnalogStatus</td><td>BYTE</td><td>Status analog input</td></tr>
</table>

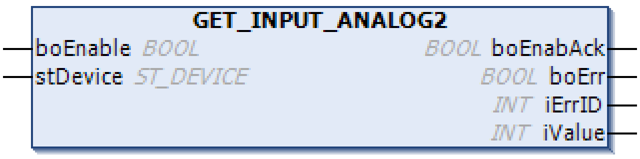
Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

13.3.2.6 GET_INPUT_ANALOG2 (FB)

This block synchronously queries analog input 2 (A2) through the 'iValue' variable.

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error:
<table>
<tr><td>Value</td><td>Meaning</td></tr>
<tr><td>1</td><td>Device information not configured</td></tr>
<tr><td>2</td><td>No input / output variable assigned (copy pointer = 0)</td></tr>
<tr><td>3</td><td>Illegal device instance (symbolic
device name might have been assigned incorrectly)</td></tr>
</table>
</td></tr>
<tr><td>iValue</td><td>INT</td><td>Analog input voltage according to ID32898 'Analog Input A2'</td></tr>
</table>

Input and output variables

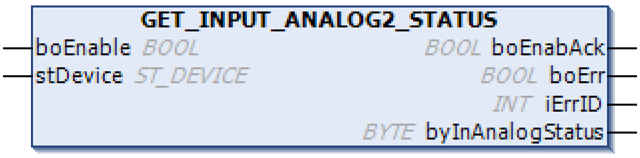
Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

13.3.2.7 GET_INPUT_ANALOG2_STATUS (FB)

This block synchronously queries the status of analog input 2 through the 'byInAnalogStatus' variable.

Only in conjunction with A5x MxE controllers.

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error:
<table>
<tr><td>Value</td><td>Meaning</td></tr>
<tr><td>1</td><td>Device information not configured</td></tr>
<tr><td>2</td><td>No input / output variable assigned (copy pointer = 0)</td></tr>
<tr><td>3</td><td>Illegal device instance (symbolic
device name might have been assigned incorrectly)</td></tr>
</table>
</td></tr>
<tr><td>byInAnalogStatus</td><td>BYTE</td><td>Status analog input</td></tr>
</table>

Input and output variables

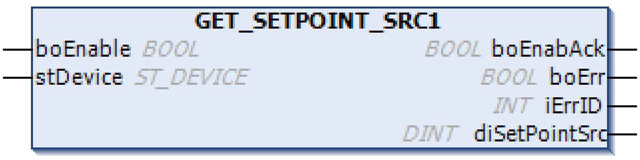
Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

13.3.2.8 GET_SETPOINT_SRC1 (FB)

This block queries the configurable values "ID32948 'Message 4x32', 1st message" through the 'diSetPointSrc' variable.

Only in conjunction with A5x MxE controllers.

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error:
<table>
<tr><td>Value</td><td>Meaning</td></tr>
<tr><td>1</td><td>Device information not configured</td></tr>
<tr><td>2</td><td>No input / output variable assigned (copy pointer = 0)</td></tr>
<tr><td>3</td><td>Illegal device instance (symbolic
device name might have been assigned incorrectly)</td></tr>
</table>
</td></tr>
<tr><td>diSetPointSrc</td><td>DINT</td><td>Gets the setpoint SRC according to ID32948 'Message 4x32'
Info1: ID34074 'Homing Counter 1' / ID34075 'Actual Counter 1'</td></tr>
</table>

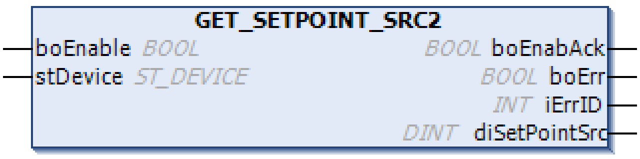
Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

13.3.2.9 GET_SETPOINT_SRC2 (FB)

This block queries the configurable values "ID32948 'Message 4x32', 2st message" through the 'diSetPointSrc' variable.

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error:
<table>
<tr><td>Value</td><td>Meaning</td></tr>
<tr><td>1</td><td>Device information not configured</td></tr>
<tr><td>2</td><td>No input / output variable assigned (copy pointer = 0)</td></tr>
<tr><td>3</td><td>Illegal device instance (symbolic
device name might have been assigned incorrectly)</td></tr>
</table>
</td></tr>
<tr><td>diSetPointSrc</td><td>DINT</td><td>Gets the setpoint SRC according to ID32948 'Message 4x32'
Info2: ID34076 'Homing Counter 2' / ID34077 'Actual Counter 2'</td></tr>
</table>

Input and output variables

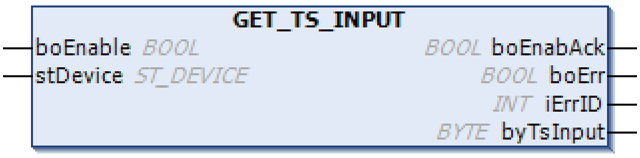
Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

13.3.2.10 GET_TS_INPUT (FB)

This block synchronously queries the binary "TimeStamp" inputs through the 'byTsInput' variable.

Only in conjunction with A5x MxE controllers or the EL1252 EtherCAT terminal.

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error:
<table>
<tr><td>Value</td><td>Meaning</td></tr>
<tr><td>1</td><td>Device information not configured</td></tr>
<tr><td>2</td><td>No input / output variable assigned (copy pointer = 0)</td></tr>
<tr><td>3</td><td>Illegal device instance (symbolic
device name might have been assigned incorrectly)</td></tr>
</table>
</td></tr>
<tr><td>byTsInput</td><td>BYTE</td><td>Get binary TimeStamp inputs</td></tr>
</table>

Input and output variables

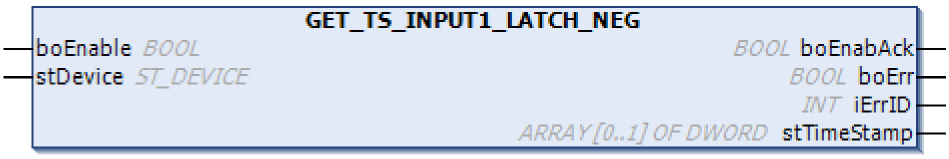
Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

13.3.2.11 GET_TS_INPUT1_LATCH_NEG (FB)

This block synchronously queries the latched "TimeStamp1" time information through the negative edge at input1. The information is queried with the 'stTimeStamp' ARRAY.

Only in conjunction with A5x MxE controllers or the EL1252 EtherCAT terminal.

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error:
<table>
<tr><td>Value</td><td>Meaning</td></tr>
<tr><td>1</td><td>Device information not configured</td></tr>
<tr><td>2</td><td>No input / output variable assigned (copy pointer = 0)</td></tr>
<tr><td>3</td><td>Illegal device instance (symbolic
device name might have been assigned incorrectly)</td></tr>
</table>
</td></tr>
<tr><td>stTimeStamp</td><td>ARRAY</td><td>Queries the latched TimeStamp time information through the negative edge at input1</td></tr>
</table>

Input and output variables

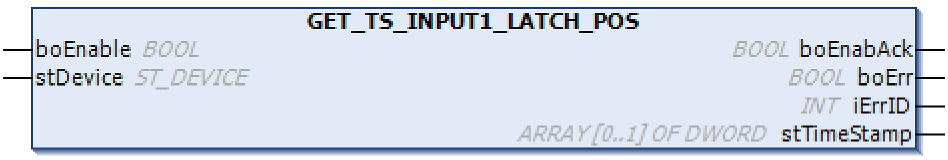
Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

13.3.2.12 GET_TS_INPUT1_LATCH_POS (FB)

This block synchronously queries the latched "TimeStamp1" time information through the positive edge at input1. The information is queried with the 'stTimeStamp' ARRAY.

Only in conjunction with A5x MxE controllers or the EL1252 EtherCAT terminal.

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table>
</td></tr>
</table>

Name	Type	Description
iErrID	INT	Error identity number: Diagnostic number is output iErrID = 0 | No error iErrID ≠ 0 | boErr = TRUE | Error iErrID ≠ 0 | boErr = FALSE | Warning Error: Value | Meaning 1 | Device information not configured 2 | No input / output variable assigned (copy pointer = 0) 3 | Illegal device instance (symbolic device name might have been assigned incorrectly)
stTimeStamp	ARRAY	Queries the latched TimeStamp time information through the positive edge at input1

Input and output variables

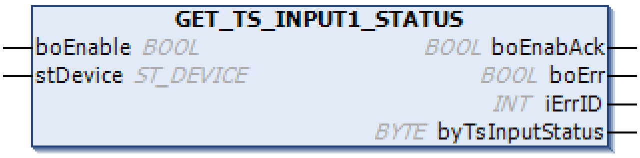
Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

13.3.2.13 GET_TS_INPUT1_STATUS (FB)

This block synchronously queries the binary "TimeStamp1" status information through the 'byTsInput'Status' variable.

Only in conjunction with A5x MxE controllers or the EL1252 EtherCAT terminal.

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.

Output variables

Name	Type	Description
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled
boErr	BOOL	The function block is in an error state FALSE | No error (permitted commanding or warning) TRUE | Error

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error:
<table>
<tr><td>Value</td><td>Meaning</td></tr>
<tr><td>1</td><td>Device information not configured</td></tr>
<tr><td>2</td><td>No input / output variable assigned (copy pointer = 0)</td></tr>
<tr><td>3</td><td>Illegal device instance (symbolic
device name might have been assigned incorrectly)</td></tr>
</table>
</td></tr>
<tr><td>byTsInputStatus</td><td>BYTE</td><td>TimeStamp status information</td></tr>
</table>

Input and output variables

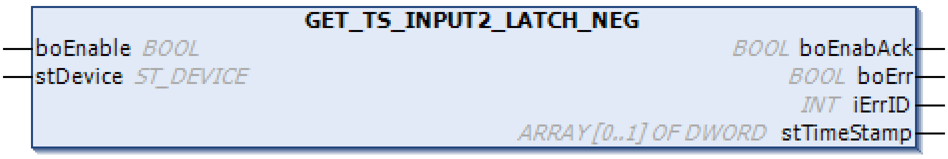
Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

13.3.2.14 GET_TS_INPUT2_LATCH_NEG (FB)

This block synchronously queries the latched "TimeStamp2" time information through the negative edge at input2. The information is queried with the 'stTimeStamp' ARRAY.

Only in conjunction with A5x MxE controllers or the EL1252 EtherCAT terminal.

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error:
<table>
<tr><td>Value</td><td>Meaning</td></tr>
<tr><td>1</td><td>Device information not configured</td></tr>
<tr><td>2</td><td>No input / output variable assigned (copy pointer = 0)</td></tr>
<tr><td>3</td><td>Illegal device instance (symbolic
device name might have been assigned incorrectly)</td></tr>
</table>
</td></tr>
<tr><td>stTimeStamp</td><td>ARRAY</td><td>Queries the latched TimeStamp time information through the negative edge at input2</td></tr>
</table>

Input and output variables

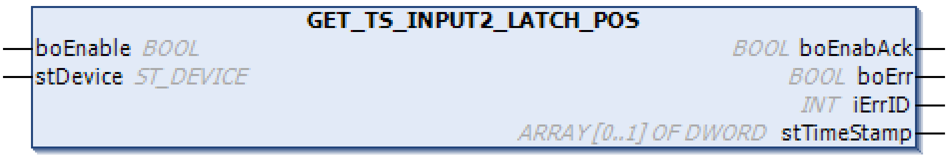
Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

13.3.2.15 GET_TS_INPUT2_LATCH_POS (FB)

This block synchronously queries the latched "TimeStamp2" time information through the positive edge at input2. The information is queried with the 'stTimeStamp' ARRAY.

Only in conjunction with A5x MxE controllers or the EL1252 EtherCAT terminal.

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error:
<table>
<tr><td>Value</td><td>Meaning</td></tr>
<tr><td>1</td><td>Device information not configured</td></tr>
<tr><td>2</td><td>No input / output variable assigned (copy pointer = 0)</td></tr>
<tr><td>3</td><td>Illegal device instance (symbolic
device name might have been assigned incorrectly)</td></tr>
</table>
</td></tr>
<tr><td>stTimeStamp</td><td>ARRAY</td><td>Queries the latched TimeStamp time information through the positive edge at input2</td></tr>
</table>

Input and output variables

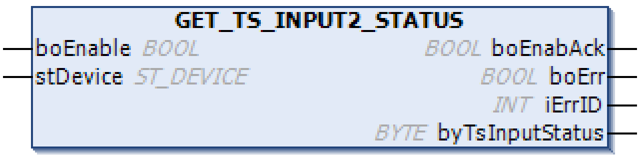
Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

13.3.2.16 GET_TS_INPUT2_STATUS (FB)

This block synchronously queries the binary "TimeStamp2" status information through the 'byTsInput'Status' variable.

Only in conjunction with A5x MxE controllers or the EL1252 EtherCAT terminal.

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table>
</td></tr>
</table>

Name	Type	Description
iErrID	INT	Error identity number: Diagnostic number is output iErrID = 0 — No error iErrID ≠ 0 — boErr = TRUE — Error iErrID ≠ 0 — boErr = FALSE — Warning Error: Value — Meaning 1 — Device information not configured 2 — No input / output variable assigned (copy pointer = 0) 3 — Illegal device instance (symbolic device name might have been assigned incorrectly)
byTsInputStatus	BYTE	TimeStamp status information

Input and output variables

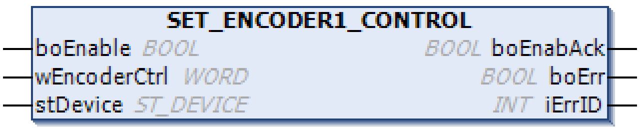
Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

13.3.2.17 SET_ENCODER1_CONTROL (FB)

This block synchronously sets the encoder control information through the 'wEncoderCtrl' variable.

Only in conjunction with A5x MxE controllers.

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.
wEncoderCtrl	WORD	Encoder control information

Output variables

Name	Type	Description
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled
boErr	BOOL	The function block is in an error state FALSE — No error (permitted commanding or warning) TRUE — Error

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error:
<table>
<tr><td>Value</td><td>Meaning</td></tr>
<tr><td>1</td><td>Device information not configured</td></tr>
<tr><td>2</td><td>No input / output variable assigned (copy pointer = 0)</td></tr>
<tr><td>3</td><td>Illegal device instance (symbolic
device name might have been assigned incorrectly)</td></tr>
</table>
</td></tr>
</table>

Input and output variables

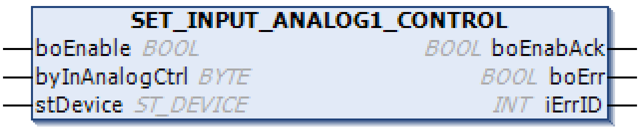
Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

13.3.2.18 SET_INPUT_ANALOG1_CONTROL (FB)

This block synchronously sets the analog1 control information through the 'byInAnalogCtrl' variable.

Only in conjunction with A5x MxE controllers.

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.
byInAnalogCtrl	BYTE	Control information at the analog input

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error:
<table>
<tr><td>Value</td><td>Meaning</td></tr>
<tr><td>1</td><td>Device information not configured</td></tr>
<tr><td>2</td><td>No input / output variable assigned (copy pointer = 0)</td></tr>
<tr><td>3</td><td>Illegal device instance (symbolic
device name might have been assigned incorrectly)</td></tr>
</table>
</td></tr>
</table>

Input and output variables

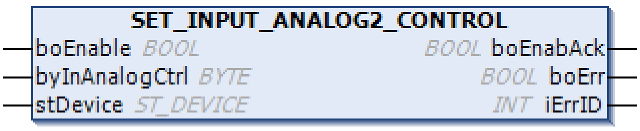
Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

13.3.2.19 SET_INPUT_ANALOG2_CONTROL (FB)

This block synchronously sets the analog2 control information through the 'byInAnalogCtrl' variable.

Only in conjunction with A5x MxE controllers.

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.
byInAnalogCtrl	BYTE	Control information at the analog input

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error:
<table>
<tr><td>Value</td><td>Meaning</td></tr>
<tr><td>1</td><td>Device information not configured</td></tr>
<tr><td>2</td><td>No input / output variable assigned (copy pointer = 0)</td></tr>
<tr><td>3</td><td>Illegal device instance (symbolic
device name might have been assigned incorrectly)</td></tr>
</table>
</td></tr>
</table>

Input and output variables

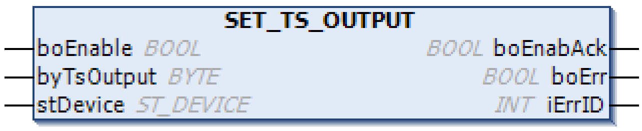
Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

13.3.2.20 SET_TS_OUTPUT (FB)

This block synchronously sets the binary "TimeStamp" outputs through the 'byTSOutput' variable.

Only in conjunction with A5x MxE controllers or the EL2252 EtherCAT terminal.

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.
byTsOutput	BYTE	Set binary TimeStamp outputs

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error:
<table>
<tr><td>Value</td><td>Meaning</td></tr>
<tr><td>1</td><td>Device information not configured</td></tr>
<tr><td>2</td><td>No input / output variable assigned (copy pointer = 0)</td></tr>
<tr><td>3</td><td>Illegal device instance (symbolic
device name might have been assigned incorrectly)</td></tr>
</table>
</td></tr>
</table>

Input and output variables

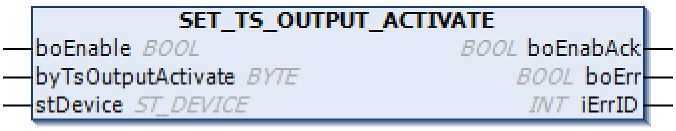
Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

13.3.2.21 SET_TS_OUTPUT_ACTIVATE (FB)

This block synchronously sets the "TimeStamp" output activation information through the 'byTsOutputActivate' variable.

Only in conjunction with A5x MxE controllers or the EL2252 EtherCAT terminal.

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.
byTsOutputActivate	BYTE	Set activation information for the TimeStamp output

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error:
<table>
<tr><td>Value</td><td>Meaning</td></tr>
<tr><td>1</td><td>Device information not configured</td></tr>
<tr><td>2</td><td>No input / output variable assigned (copy pointer = 0)</td></tr>
<tr><td>3</td><td>Illegal device instance (symbolic
device name might have been assigned incorrectly)</td></tr>
</table>
</td></tr>
</table>

Input and output variables

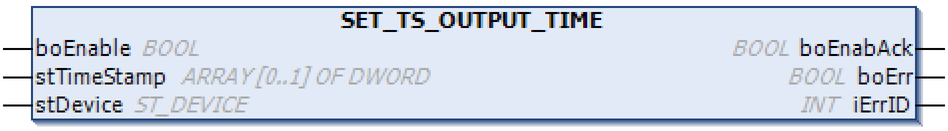
Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

13.3.2.22 SET_TS_OUTPUT_TIME (FB)

This block synchronously sets the "TimeStamp" output time through the 'stTimeStamp' variable.

Only in conjunction with A5x MxE controllers or the EL2252 EtherCAT terminal.

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.
stTimeStamp	ARRAY	Set latched TimeStamp time information

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error:
<table>
<tr><td>Value</td><td>Meaning</td></tr>
<tr><td>1</td><td>Device information not configured</td></tr>
<tr><td>2</td><td>No input / output variable assigned (copy pointer = 0)</td></tr>
<tr><td>3</td><td>Illegal device instance (symbolic
device name might have been assigned incorrectly)</td></tr>
</table>
</td></tr>
</table>

Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

13.3.3 TimeStamp

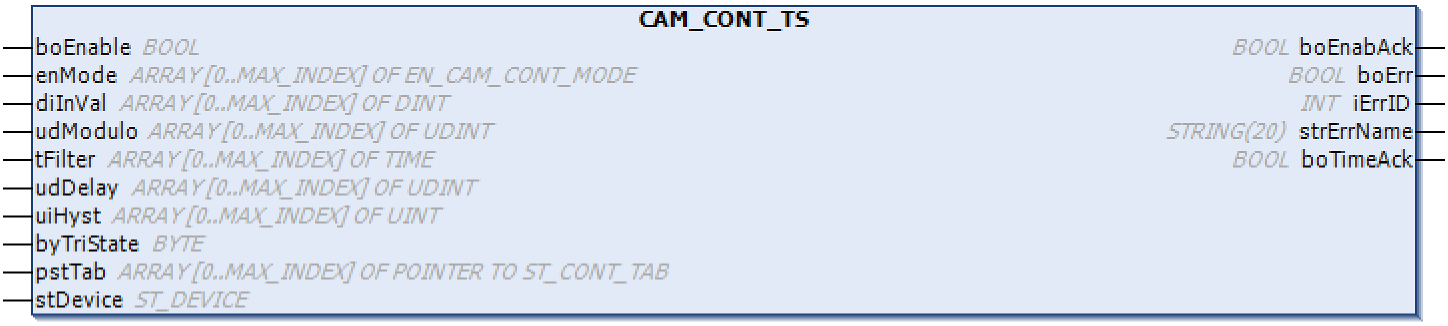
13.3.3.1 CAM_CONT_TS (FB)

This block is based on a combination of the function blocks 'CAM_CONT_1' and 'SET_TS_OUTPUTS'

Through this combination, two positionally accurate cam controllers are made possible in connection with timestamp outputs. The corresponding input variables are array based on a two-channel design.

'byTriState' is not supported by the local IO terminal of the A5x MxE type controller.

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>enMode</td><td>ARRAY</td><td>ARRAY [0..MAX_INDEX] OF EN_CAM_CONT_MODE
Selection mode between incremental and absolute input evaluation
<table><tr><td>Default</td><td>CAM_CONT_INC</td></tr></table>
<table><tr><th>Range</th><th>Meaning</th></tr><tr><td>CAM_CONT_INC</td><td>Incremental input value evaluation</td></tr><tr><td>CAM_CONT_ABS</td><td>Absolute input value evaluation</td></tr></table></td></tr>
<tr><td>diInVal</td><td>ARRAY</td><td>ARRAY [0..MAX_INDEX] OF DINT
Input value of the camshaft control (position)</td></tr>
<tr><td>udModulo</td><td>ARRAY</td><td>ARRAY [0..MAX_INDEX] OF UDINT
Modulo value
In mode 'enMode' = CAM_CONT_INC, this is the value at which cam table evaluation restarts at "0"
<table><tr><td>Range</td><td>0 … $+2^{31}-1$</td></tr><tr><td>Default</td><td>20000</td></tr></table></td></tr>
<tr><td>tFilter</td><td>ARRAY</td><td>ARRAY [0..MAX_INDEX] OF TIME
Filter time constant
Attenuates the impact of changes in velocity in the context of dead-time compensation
<table><tr><td>Default</td><td>t#1 ms</td></tr></table></td></tr>
<tr><td>udDelay</td><td>ARRAY</td><td>ARRAY [0..MAX_INDEX] OF UDINT
Dead-time constant
To calculate the offset of the binary information depending on the current velocity in the context of dead-time compensation
<table><tr><td>Resolution</td><td>t#0.001 ms</td></tr><tr><td>Default</td><td>0 (dead-time compensation not active)</td></tr></table></td></tr>
<tr><td>uiHyst</td><td>ARRAY</td><td>ARRAY [0..MAX_INDEX] OF UINT
Hysteresis value
(H), applied to the on and off edges (X_{on}, X_{off}) of a cam signal
<table><tr><td>Default</td><td>0 (hysteresis not active)</td></tr></table>
In conjunction with dead-time compensation, the hysteresis must be set higher than the dead-time compensation path X_{dead}

Thus:
$X_{dead} = T_{dead} * n * G / 60000$

where
<table><tr><td>X_{dead}</td><td>Dead-time compensation path [incr]</td></tr><tr><td>T_{dead}</td><td>Dead time [ms]</td></tr><tr><td>n</td><td>Speed [rpm]</td></tr><tr><td>G</td><td>Encoder resolution [incr/rev]</td></tr></table>
In incremental input value evaluation mode ('enMode' = CAM_CONT_INC), the following must be true:
<table><tr><td>$H < udModulo - (X_{off} - X_{on})$</td><td>for $X_{off} > X_{on}$</td></tr><tr><td>$H < X_{on} - X_{off}$</td><td>for $X_{off} < X_{on}$</td></tr></table></td></tr>
</table>

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>byTriState</td><td>BYTE</td><td>Output check of the tri-state
<table>
<tr><td>0</td><td>Channel 1</td></tr>
<tr><td>1</td><td>Channel 2</td></tr>
</table></td></tr>
<tr><td>pstTab</td><td>ARRAY</td><td>POINTER TO ST_CONT_TAB
Pointer to the cam table</td></tr>
</table>

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table></td></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
strErrName = 'CAM_CONT_1_CH1' or 'CAM_CONT_1_CH2'
Warning
<table>
<tr><th>Value</th><th>Meaning</th></tr>
<tr><td>1</td><td>Modulo value limited to maximum</td></tr>
<tr><td>2</td><td>Filter time constant set to 1</td></tr>
<tr><td>3</td><td>Filter time constant limited to maximum</td></tr>
<tr><td>4</td><td>Dead-time constant set to 0</td></tr>
<tr><td>5</td><td>Dead-time constant set to 1</td></tr>
<tr><td>6</td><td>Dead-time constant limited to maximum</td></tr>
</table>
strErrName = 'SET_TS_OUTPUTS'
Warning
<table>
<tr><th>Value</th><th>Meaning</th></tr>
<tr><td>20</td><td>Illegal request for a new output value (the set time has not yet expired).</td></tr>
</table>
Error
<table>
<tr><th>Value</th><th>Meaning</th></tr>
<tr><td>1-9</td><td>Error codes for base function blocks</td></tr>
<tr><td>20</td><td>The value of 'diStartTime' is illegal:
'diStartTime' <0
'diStartTime' <3x ID2 [ns]
'diStartTime' +3x ID2 [ns] > 16#7FFFFFFF</td></tr>
</table></td></tr>
<tr><td>strErrName</td><td>STRING (20)</td><td>Block name of the block causing the error
(CAM_CONT_1_CH1, CAM_CONT_1_CH2, SET_TS_OUTPUTS)</td></tr>
<tr><td>boTimeAck</td><td>BOOL</td><td>Time for change of the state is expired
It can be done a new output</td></tr>
</table>

Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

Description

The camshaft control has the following properties:

- Incremental or absolute mode
- Filter in the context of dead-time compensation
- Dead-time compensation
- Hysteresis

Mode

- **Set incremental input value** ('enMode' = CAM_CONT_INC):
  The 'diInVal' input variable is processed as a 32-bit signed fixed-point number (32-bit integer value).
  In response to every call, the block generates the input value differences from two consecutive items of input information and adds these up to a positive 32-bit value. The internal counter works modulo; in other words, it counts up to a configurable final value 'udModulo' and then starts again at zero.
- **Set absolute input value** ('enMode' = CAM_CONT_ABS):
  The 'diInVal' input variable is processed as a 32-bit signed fixed-point number (32-bit integer value). Overshoot at the end of the travel range is limited.

Filter

To attenuate the impact of changes in velocity for dead-time compensation, multiple speed values are averaged. The 'tFilter' filter time constant determines the number of velocity values for which averaging is performed (number = 'tFilter' [ms]/stDevice.uiCycleTime [ms]).

Dead-time compensation

For dead-time compensation the binary information is offset leading based on the current velocity. The 'tDelay' dead-time constant accounts for the time taken to calculate the offset.

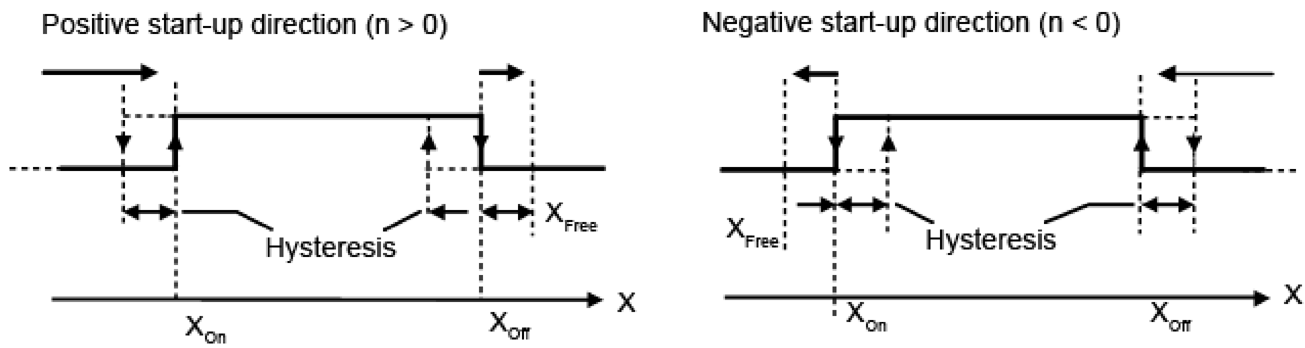
Hysteresis

The hysteresis ensures that the binary output always adopts a stable state, even if the input value of the block is moving around a rising or falling cam edge at the time.

The generation of the hysteresis (X_{on}, X_{off}) is illustrated in the figure below:

- Positive approach direction (n > 0; X increasing)
- Negative approach direction (n < 0; X decreasing)

Abbildung 57: CAM_CONT: Hysteresis generation



A "positive approach direction" results in the following behavior at the binary output (cam):

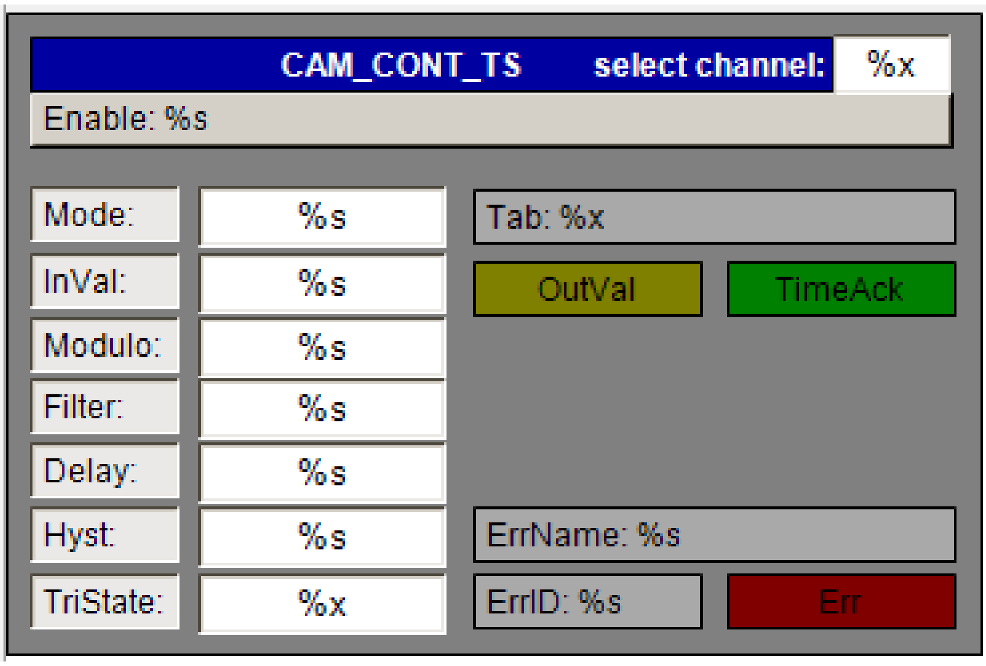
- Cam information "0" is output starting from a position $X < X_{on}$.
- Cam information "1" is output as of position $X \geq X_{on}$.
- Cam information "1" is retained during reverse rotation to position $X \geq X_{on}$–H.
  Cam information "0" is output during further reverse rotation to position $X < X_{on}$–H.
- Cam information "0" is output during forward rotation starting from position $X \geq X_{off}$.
  - Cam information "0" is retained in the event of reverse rotation to position $X \geq X_{off}$–H before position $X = X_{free} = X_{off}$+H is reached.
    Cam information "1" is output during further reverse rotation.
  - In the event of reverse rotation after position $X \geq X_{free}$ has been reached, the cam signal is generated according to the "negative approach direction".

A "negative approach direction" results in the following behavior:

- Cam information "0" is output starting from a position $X \geq X_{off}$.
- Cam information "1" is output as of position $X < X_{off}$.
- Cam information "1" is retained during forward rotation to position $X < X_{off}$+H.
  Cam information "0" is output during further forward rotation.
- Cam information "0" is output during reverse rotation starting from position $X < X_{on}$.
  - Cam information "0" is retained in the event of reverse rotation to position $X < X_{on}$+H prior to overshooting position $X = X_{free} = X_{on}$–H.
    Cam information "1" is output during further forward rotation.
  - In the event of reverse rotation after position $X < X_{free}$ has been reached, the cam signal is generated according to the "positive approach direction".

Switchover between hysteresis generation of positive (negative) approach direction takes place once a cam has completed its rotation and position X_{free} has been reached or overshot.

13.3.3.1.1 Visualization



13.3.3.2 GET_TS_INPUTS (FB)

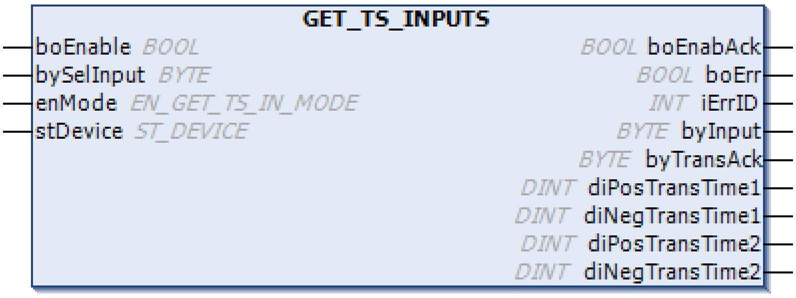
The 'GET_TS_INPUTS' block provides binary information for the local IO terminals of the A4x-, A5x-, A6X-MXE-control variants, or the terminal EL 1252. It is designed with 2 channels.

As well as the 'byInput' binary input levels of the channels, changes in the 'byTransAck' channel levels and the exact time of the changes 'diPosTransTime1', 'diNegTransTime1', 'diPosTransTime2', 'diNegTransTime2' are output.

The channels (channel1, channel2, or both channels) are selected with 'bySelInput'.

To reduce runtime, the function is only executed for the selected channel(s).

User interface



Input variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnable</td><td>BOOL</td><td>Enable signal: With a positive edge, the initialisation of the block starts.
As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC.
In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.</td></tr>
<tr><td>bySelInput</td><td>BYTE</td><td>Selection of the input channel<table><tr><td>0</td><td>Channel 1</td></tr><tr><td>1</td><td>Channel 2</td></tr></table></td></tr>
<tr><td>enMode</td><td>ENUM</td><td>EN_GET_TS_IN_MODE
Selection mode
(Not used in the context of the specific function for AMK)</td></tr>
</table>

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state<table><tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr><tr><td>TRUE</td><td>Error</td></tr></table></td></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output<table><tr><td colspan="2">iErrID = 0</td><td>No error</td></tr><tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr><tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr></table>Error<table><tr><td>Value</td><td>Meaning</td></tr><tr><td>1-9</td><td>Error codes for base function blocks</td></tr><tr><td>10</td><td>Value of positive timestamp of input 1 too high</td></tr><tr><td>11</td><td>Value of negative timestamp of input 1 too high</td></tr><tr><td>12</td><td>Value of positive timestamp of input 2 too high</td></tr><tr><td>13</td><td>Value of negative timestamp of input 2 too high</td></tr></table></td></tr>
</table>

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>byInput</td><td>BYTE</td><td>Current state of the input<table><tr><td>0</td><td>Channel 1</td></tr><tr><td>1</td><td>Channel 2</td></tr></table></td></tr>
<tr><td>byTransAck</td><td>BYTE</td><td>Display for a valid edge<table><tr><td>0</td><td>Positive edge channel 1</td></tr><tr><td>1</td><td>Negative edge channel 1</td></tr><tr><td>2</td><td>Positive edge channel 2</td></tr><tr><td>3</td><td>Negative edge channel 2</td></tr></table></td></tr>
<tr><td>diPosTransTime1</td><td>DINT</td><td>Channel 1:
Time offset for a positive edge<table><tr><td>Unit</td><td>ns</td></tr></table></td></tr>
<tr><td>diNegTransTime1</td><td>DINT</td><td>Channel 1:
Time offset for a negative edge<table><tr><td>Unit</td><td>ns</td></tr></table></td></tr>
<tr><td>diPosTransTime2</td><td>DINT</td><td>Channel 2:
Time offset for a positive edge<table><tr><td>Unit</td><td>ns</td></tr></table></td></tr>
<tr><td>diNegTransTime2</td><td>DINT</td><td>Channel 2:
Time offset for a negative edge<table><tr><td>Unit</td><td>ns</td></tr></table></td></tr>
</table>

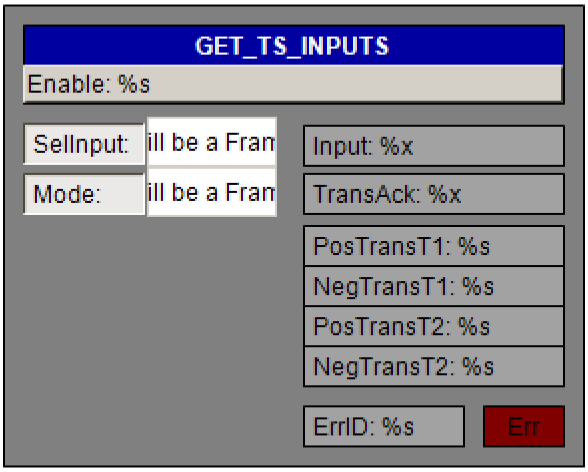
Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

13.3.3.2.1 Prerequisite for the block

1. AmkBase:
   - FdiGetDiffToBusSysTime();
2. AmkDevAccess:
   - GET_TS_INPUT;
   - GET_TS_INPUT1_STATUS;
   - GET_TS_INPUT1_LATCH_POS;
   - GET_TS_INPUT1_LATCH_NEG;
   - GET_TS_INPUT2_STATUS;
   - GET_TS_INPUT2_LATCH_POS;
   - GET_TS_INPUT2_LATCH_NEG;

13.3.3.2.2 Visualization



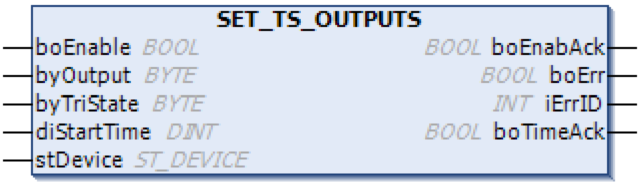
13.3.3.3 SET_TS_OUTPUTS (FB)

The 'SET_TS_OUTPUTS' block supplies the binary output information for the local IO terminal of the A4x-, A5x-, A6X-MXE-control variants, or the terminal EL 1252. It is designed for 2 binary channels.

The exact output time 'diStartTime' can be set for a binary output level 'byOutput' and the required output structure 'byTriState'. The occurrence of the output time can be detected with 'boTimeAck'.

'byTriState' is not supported by the local IO terminal of the A5x MxE type controller.

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.
byOutput	BYTE	Current state of the output 0: Channel 1 1: Channel 2
byTriState	BYTE	Output check of the tri-state 0: Channel 1 1: Channel 2
diStartTime	DINT	Start time of current cycle Unit: ns

Output variables

Name	Type	Description
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled
boErr	BOOL	The function block is in an error state FALSE: No error (permitted commanding or warning) TRUE: Error
iErrID	INT	Error identity number: Diagnostic number is output iErrID = 0: No error iErrID ≠ 0, boErr = TRUE: Error iErrID ≠ 0, boErr = FALSE: Warning Warning Value: Meaning 20: Illegal request for a new output value (the set time has not yet expired). Error Value: Meaning 1-9: Error codes for base function blocks 20: The value of 'diStartTime' is illegal: 'diStartTime' <0 'diStartTime' <3x ID2 [ns] 'diStartTime' +3x ID2 [ns] > 16#7FFFFFFF
boTimeAck	BOOL	Time for change of the state is expired It can be done a new output

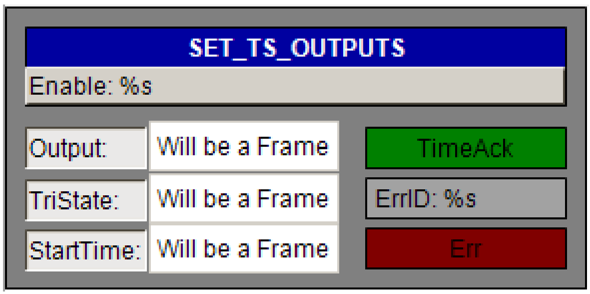
Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

13.3.3.3.1 Prerequisite for the block

1. AmkBase:
   - FboAddToBusSysTime();
2. AmkDevAccess:
   - SET_TS_OUTPUT;
   - SET_TS_OUTPUT_ACTIVATE;
   - SET_TS_OUTPUT_TIME;

13.3.3.3.2 Visualization



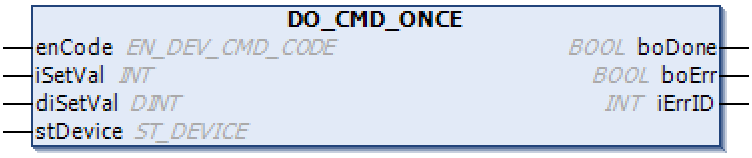
13.4 DeviceCmd (device commanding)

DO_CMD_ONCE	One-off commanding cycle

13.4.1 DO_CMD_ONCE (FB)

Device commanding for the 'DO_CMD_ONCE' block triggers a one-off command cycle that is not specific to a bus system.

User interface



Input variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>enCode</td><td>ENUM</td><td>EN_DEV_CMD_CODE
Select: Command code
<table>
<tr><td>Default</td><td>TAB_CALC_OP</td></tr>
</table>
<table>
<tr><th>Range</th><th>Meaning</th></tr>
<tr><td>DEV_CMD_MODE0</td><td>MainMode0</td></tr>
<tr><td>DEV_CMD_POS</td><td>Position control</td></tr>
<tr><td>DEV_CMD_SPEED</td><td>Speed control</td></tr>
<tr><td>DEV_CMD_TORQUE</td><td>Torque control</td></tr>
<tr><td>DEV_CMD_HOME</td><td>Homing cycle</td></tr>
<tr><td>DEV_CMD_STOP</td><td>Stop (speed control, n=0)</td></tr>
<tr><td>DEV_CMD_MSTART</td><td>Start touch probe</td></tr>
<tr><td>DEV_CMD_MSTOP</td><td>Stop touch probe</td></tr>
<tr><td>DEV_CMD_HOME_TMP_PAR</td><td>Homing cycle (block setting)</td></tr>
</table></td></tr>
<tr><td>iSetVal</td><td>INT</td><td>16-bit Setpoint (depends on command code)
<table>
<tr><th>Range</th><th>Meaning</th></tr>
<tr><td>DEV_CMD_HOME_TMP_PAR</td><td>Approach direction according to ID147 'Homing parameter'
Cam according to ID32926 'AMK homing cycle parameter'
(See document Parameter description, Part no. 26249)</td></tr>
</table></td></tr>
<tr><td>diSetVal</td><td>DINT</td><td>32-bit Setpoint (depends on command code)
<table>
<tr><th>Range</th><th>Meaning</th></tr>
<tr><td>DEV_CMD_HOME_TMP_PAR</td><td>Offset according to ID153 'Spindle angle position'
(See document Parameter description, Part no. 26249)</td></tr>
</table></td></tr>
</table>

Output variables

Name	Type	Description
boDone	BOOL	Response that the function block has been completely executed.
boErr	BOOL	The function block is in an error state FALSE: No error (permitted commanding or warning) TRUE: Error
iErrID	INT	Error identity number: Diagnostic number is output iErrID = 0: No error iErrID ≠ 0, boErr = TRUE: Error iErrID ≠ 0, boErr = FALSE: Warning

Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

Actions

Name	Description
Start	The process is started with the start action and acknowledged with 'boDone' = TRUE • The acknowledgement not revoked until the next start action is underway • The input parameters must be specified before the start action is triggered
Stop	Abort all movements in process (transition to speed control with setpoint speed = 0).

13.5 PlcVarAccess (PLC-PLC communication)

The blocks for PLC-PLC communication provide the basis for the automatic bus configuration of asynchronous and synchronous communication links between AMK controllers.

Asynchronous

GET_PLCVAR_ASYNC_BYTE08	Receive an asynchronous 'mapped' BYTE-ARRAY 8 bytes in length
GET_PLCVAR_ASYNC_BYTE16	Receive an asynchronous 'mapped' BYTE-ARRAY 16 bytes in length
GET_PLCVAR_ASYNC_BYTE32	Receive an asynchronous 'mapped' BYTE-ARRAY 32 bytes in length
GET_PLCVAR_ASYNC_BYTE64	Receive an asynchronous 'mapped' BYTE-ARRAY 64 bytes in length
GET_PLCVAR_ASYNC_DINT	Receive an asynchronous 'mapped' DINT type variable
GET_PLCVAR_ASYNC_INT	Receive an asynchronous 'mapped' INT type variable
SET_PLCVAR_ASYNC_BYTE08	Send an asynchronous 'mapped' BYTE-ARRAY 8 bytes in length
SET_PLCVAR_ASYNC_BYTE16	Send an asynchronous 'mapped' BYTE-ARRAY 16 bytes in length
SET_PLCVAR_ASYNC_BYTE32	Send an asynchronous 'mapped' BYTE-ARRAY 32 bytes in length
SET_PLCVAR_ASYNC_BYTE64	Send an asynchronous 'mapped' BYTE-ARRAY 64 bytes in length
SET_PLCVAR_ASYNC_DINT	Send an asynchronous 'mapped' DINT type variable
SET_PLCVAR_ASYNC_INT	Send an asynchronous 'mapped' INT type variable

Synchronous

GET_PLCVAR_SYNC_BYTE08	Receive a synchronous 'mapped' BYTE-ARRAY 8 bytes in length
GET_PLCVAR_SYNC_BYTE16	Receive a synchronous 'mapped' BYTE-ARRAY 16 bytes in length
GET_PLCVAR_SYNC_BYTE32	Receive a synchronous 'mapped' BYTE-ARRAY 32 bytes in length

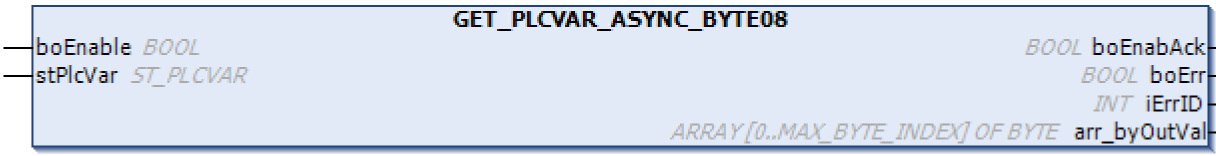
GET_PLCVAR_SYNC_BYTE64	Receive a synchronous 'mapped' BYTE-ARRAY 64 bytes in length
GET_PLCVAR_SYNC_DINT	Receive a synchronous 'mapped' DINT type variable
GET_PLCVAR_SYNC_INT	Receive a synchronous 'mapped' INT type variable
SET_PLCVAR_SYNC_BYTE08	Send a synchronous 'mapped' BYTE-ARRAY 8 bytes in length
SET_PLCVAR_SYNC_BYTE16	Send a synchronous 'mapped' BYTE-ARRAY 16 bytes in length
SET_PLCVAR_SYNC_BYTE32	Send a synchronous 'mapped' BYTE-ARRAY 32 bytes in length
SET_PLCVAR_SYNC_BYTE64	Send a synchronous 'mapped' BYTE-ARRAY 64 bytes in length
SET_PLCVAR_SYNC_DINT	Send a synchronous 'mapped' DINT type variable
SET_PLCVAR_SYNC_INT	Send a synchronous 'mapped' INT type variable

13.5.1 Asynchronous

13.5.1.1 GET_PLCVAR_ASYNC_BYTE08 (FB)

This block receives an asynchronous "mapped" byte array 8 bytes in length through 'arr_byOutVal' (MAX_BYTE_INDEX:=8).

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table></td></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error:
<table>
<tr><th>Value</th><th>Meaning</th></tr>
<tr><td>1</td><td>Device information not configured</td></tr>
<tr><td>2</td><td>No input / output variable assigned (copy pointer = 0)</td></tr>
<tr><td>3</td><td>Illegal device instance (symbolic
device name might have been assigned incorrectly)</td></tr>
</table></td></tr>
<tr><td>arr_byOutVal</td><td>ARRAY</td><td>BYTE08 Output value</td></tr>
</table>

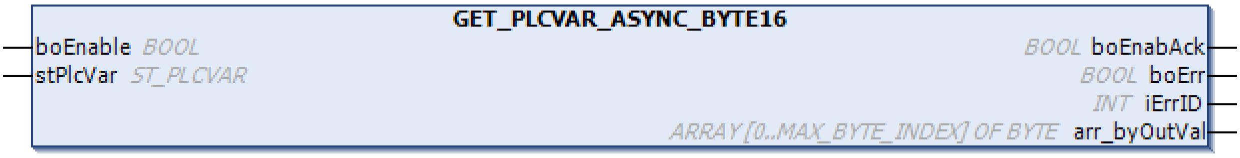
Input and output variables

Name	Type	Description
stPlcVar	STRUCT	ST_PLCVAR Information about the PLC variables The 'stPlcVar' input variable must always be assigned a handle (ST_PLCVAR type global variable). This variable must be added to the device tree of the CODESYS project.

13.5.1.2 GET_PLCVAR_ASYNC_BYTE16 (FB)

This block receives an asynchronous "mapped" byte array 16 bytes in length through 'arr_byOutVal' (MAX_BYTE_INDEX:=16).

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table></td></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error:
<table>
<tr><th>Value</th><th>Meaning</th></tr>
<tr><td>1</td><td>Device information not configured</td></tr>
<tr><td>2</td><td>No input / output variable assigned (copy pointer = 0)</td></tr>
<tr><td>3</td><td>Illegal device instance (symbolic
device name might have been assigned incorrectly)</td></tr>
</table></td></tr>
<tr><td>arr_byOutVal</td><td>ARRAY</td><td>BYTE16 Output value</td></tr>
</table>

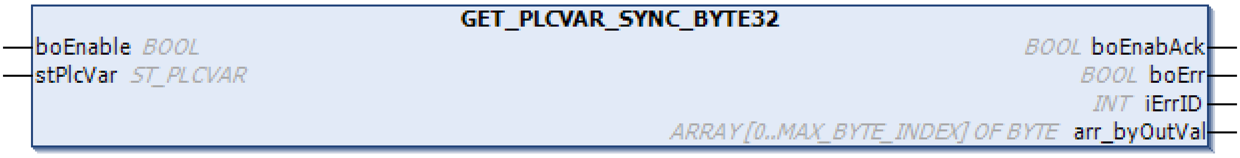
Input and output variables

Name	Type	Description
stPlcVar	STRUCT	ST_PLCVAR Information about the PLC variables The 'stPlcVar' input variable must always be assigned a handle (ST_PLCVAR type global variable). This variable must be added to the device tree of the CODESYS project.

13.5.1.3 GET_PLCVAR_ASYNC_BYTE32 (FB)

This block receives an asynchronous "mapped" byte array 32 bytes in length through 'arr_byOutVal' (MAX_BYTE_INDEX:=32).

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table></td></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error:
<table>
<tr><td>Value</td><td>Meaning</td></tr>
<tr><td>1</td><td>Device information not configured</td></tr>
<tr><td>2</td><td>No input / output variable assigned (copy pointer = 0)</td></tr>
<tr><td>3</td><td>Illegal device instance (symbolic
device name might have been assigned incorrectly)</td></tr>
</table></td></tr>
<tr><td>arr_byOutVal</td><td>ARRAY</td><td>BYTE32 Output value</td></tr>
</table>

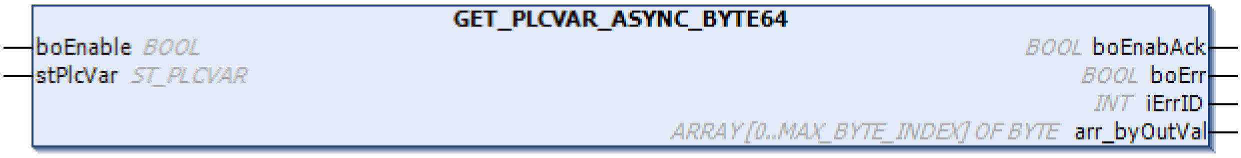
Input and output variables

Name	Type	Description
stPlcVar	STRUCT	ST_PLCVAR Information about the PLC variables The 'stPlcVar' input variable must always be assigned a handle (ST_PLCVAR type global variable). This variable must be added to the device tree of the CODESYS project.

13.5.1.4 GET_PLCVAR_ASYNC_BYTE64 (FB)

This block receives an asynchronous "mapped" byte array 64 bytes in length through 'arr_byOutVal' (MAX_BYTE_INDEX:=64).

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table></td></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error:
<table>
<tr><th>Value</th><th>Meaning</th></tr>
<tr><td>1</td><td>Device information not configured</td></tr>
<tr><td>2</td><td>No input / output variable assigned (copy pointer = 0)</td></tr>
<tr><td>3</td><td>Illegal device instance (symbolic
device name might have been assigned incorrectly)</td></tr>
</table></td></tr>
<tr><td>arr_byOutVal</td><td>ARRAY</td><td>BYTE64 Output value</td></tr>
</table>

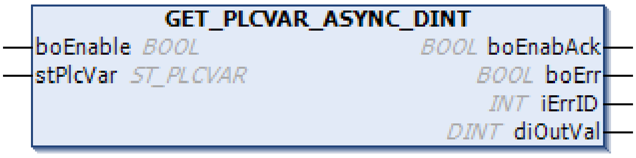
Input and output variables

Name	Type	Description
stPlcVar	STRUCT	ST_PLCVAR Information about the PLC variables The 'stPlcVar' input variable must always be assigned a handle (ST_PLCVAR type global variable). This variable must be added to the device tree of the CODESYS project.

13.5.1.5 GET_PLCVAR_ASYNC_DINT (FB)

This block receives an asynchronous "mapped" DINT type variable through 'diOutVal'.

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table></td></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error:
<table>
<tr><th>Value</th><th>Meaning</th></tr>
<tr><td>1</td><td>Device information not configured</td></tr>
<tr><td>2</td><td>No input / output variable assigned (copy pointer = 0)</td></tr>
<tr><td>3</td><td>Illegal device instance (symbolic
device name might have been assigned incorrectly)</td></tr>
</table></td></tr>
<tr><td>diOutVal</td><td>DINT</td><td>Output value</td></tr>
</table>

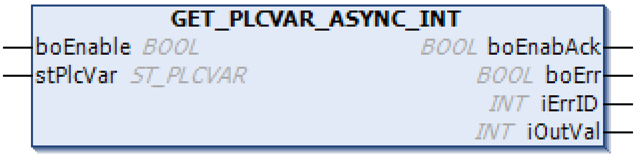
Input and output variables

Name	Type	Description
stPlcVar	STRUCT	ST_PLCVAR Information about the PLC variables The 'stPlcVar' input variable must always be assigned a handle (ST_PLCVAR type global variable). This variable must be added to the device tree of the CODESYS project.

13.5.1.6 GET_PLCVAR_ASYNC_INT (FB)

This block receives an asynchronous "mapped" INT type variable through 'iOutVal'.

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table></td></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error:
<table>
<tr><td>Value</td><td>Meaning</td></tr>
<tr><td>1</td><td>Device information not configured</td></tr>
<tr><td>2</td><td>No input / output variable assigned (copy pointer = 0)</td></tr>
<tr><td>3</td><td>Illegal device instance (symbolic
device name might have been assigned incorrectly)</td></tr>
</table></td></tr>
<tr><td>iOutVal</td><td>INT</td><td>Output value</td></tr>
</table>

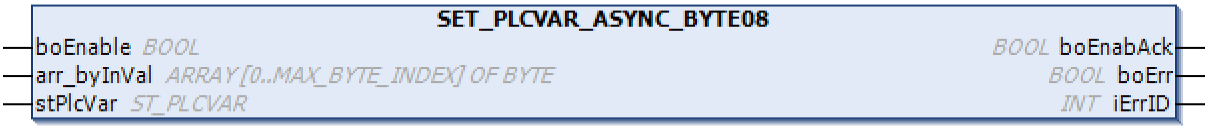
Input and output variables

Name	Type	Description
stPlcVar	STRUCT	ST_PLCVAR Information about the PLC variables The 'stPlcVar' input variable must always be assigned a handle (ST_PLCVAR type global variable). This variable must be added to the device tree of the CODESYS project.

13.5.1.7 SET_PLCVAR_ASYNC_BYTE08 (FB)

This block sends an asynchronous "mapped" byte array 8 bytes in length through 'arr_byInVal' (MAX_BYTE_INDEX:=8).

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.
arr_byInVal	ARRAY	BYTE08 Input value

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table></td></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error:
<table>
<tr><th>Value</th><th>Meaning</th></tr>
<tr><td>1</td><td>Device information not configured</td></tr>
<tr><td>2</td><td>No input / output variable assigned (copy pointer = 0)</td></tr>
<tr><td>3</td><td>Illegal device instance (symbolic
device name might have been assigned incorrectly)</td></tr>
</table></td></tr>
</table>

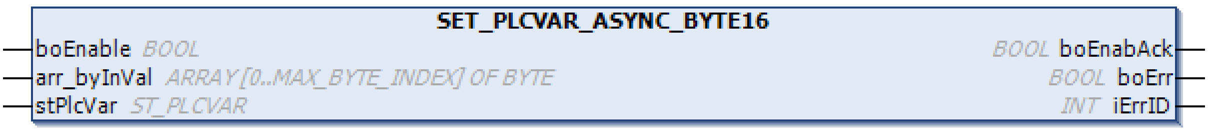
Input and output variables

Name	Type	Description
stPlcVar	STRUCT	ST_PLCVAR Information about the PLC variables The 'stPlcVar' input variable must always be assigned a handle (ST_PLCVAR type global variable). This variable must be added to the device tree of the CODESYS project.

13.5.1.8 SET_PLCVAR_ASYNC_BYTE16 (FB)

This block sends an asynchronous "mapped" byte array 16 bytes in length through 'arr_byInVal' (MAX_BYTE_INDEX:=16).

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.
arr_byInVal	ARRAY	BYTE16 Input value

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table></td></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error:
<table>
<tr><td>Value</td><td>Meaning</td></tr>
<tr><td>1</td><td>Device information not configured</td></tr>
<tr><td>2</td><td>No input / output variable assigned (copy pointer = 0)</td></tr>
<tr><td>3</td><td>Illegal device instance (symbolic
device name might have been assigned incorrectly)</td></tr>
</table></td></tr>
</table>

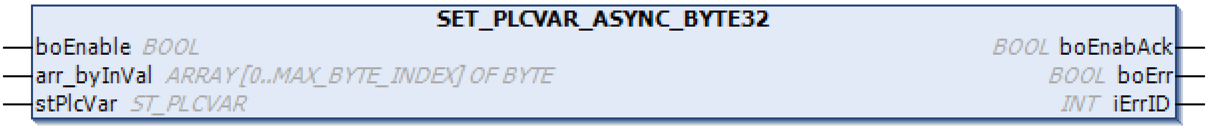
Input and output variables

Name	Type	Description
stPlcVar	STRUCT	ST_PLCVAR Information about the PLC variables The 'stPlcVar' input variable must always be assigned a handle (ST_PLCVAR type global variable). This variable must be added to the device tree of the CODESYS project.

13.5.1.9 SET_PLCVAR_ASYNC_BYTE32 (FB)

This block sends an asynchronous "mapped" byte array 32 bytes in length through 'arr_byInVal' (MAX_BYTE_INDEX:=32).

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.
arr_byInVal	ARRAY	BYTE32 Input value

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table></td></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error:
<table>
<tr><th>Value</th><th>Meaning</th></tr>
<tr><td>1</td><td>Device information not configured</td></tr>
<tr><td>2</td><td>No input / output variable assigned (copy pointer = 0)</td></tr>
<tr><td>3</td><td>Illegal device instance (symbolic
device name might have been assigned incorrectly)</td></tr>
</table></td></tr>
</table>

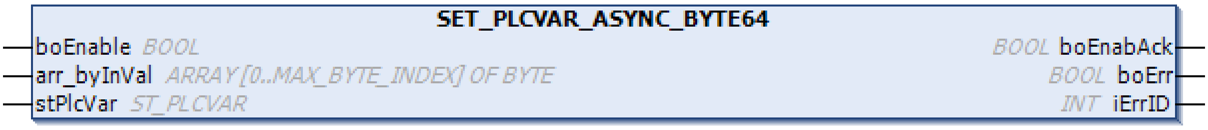
Input and output variables

Name	Type	Description
stPlcVar	STRUCT	ST_PLCVAR Information about the PLC variables The 'stPlcVar' input variable must always be assigned a handle (ST_PLCVAR type global variable). This variable must be added to the device tree of the CODESYS project.

13.5.1.10 SET_PLCVAR_ASYNC_BYTE64 (FB)

This block sends an asynchronous "mapped" byte array 64 bytes in length through 'arr_byInVal' (MAX_BYTE_INDEX:=64).

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.
arr_byInVal	ARRAY	BYTE64 Input value

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table></td></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error:
<table>
<tr><th>Value</th><th>Meaning</th></tr>
<tr><td>1</td><td>Device information not configured</td></tr>
<tr><td>2</td><td>No input / output variable assigned (copy pointer = 0)</td></tr>
<tr><td>3</td><td>Illegal device instance (symbolic
device name might have been assigned incorrectly)</td></tr>
</table></td></tr>
</table>

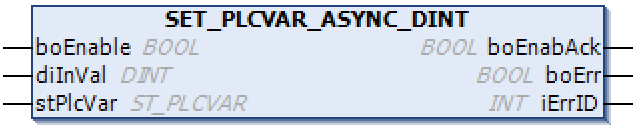
Input and output variables

Name	Type	Description
stPlcVar	STRUCT	ST_PLCVAR Information about the PLC variables The 'stPlcVar' input variable must always be assigned a handle (ST_PLCVAR type global variable). This variable must be added to the device tree of the CODESYS project.

13.5.1.11 SET_PLCVAR_ASYNC_DINT (FB)

This block sends an asynchronous "mapped" DINT type variable through 'diInVal'.

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.
diInVal	DINT	Input value

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table></td></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error:
<table>
<tr><td>Value</td><td>Meaning</td></tr>
<tr><td>1</td><td>Device information not configured</td></tr>
<tr><td>2</td><td>No input / output variable assigned (copy pointer = 0)</td></tr>
<tr><td>3</td><td>Illegal device instance (symbolic
device name might have been assigned incorrectly)</td></tr>
</table></td></tr>
</table>

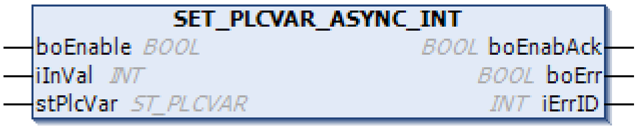
Input and output variables

Name	Type	Description
stPlcVar	STRUCT	ST_PLCVAR Information about the PLC variables The 'stPlcVar' input variable must always be assigned a handle (ST_PLCVAR type global variable). This variable must be added to the device tree of the CODESYS project.

13.5.1.12 SET_PLCVAR_ASYNC_INT (FB)

This block sends an asynchronous "mapped" INT type variable through 'iInVal'.

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.
iInVal	INT	Input value

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table></td></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error:
<table>
<tr><th>Value</th><th>Meaning</th></tr>
<tr><td>1</td><td>Device information not configured</td></tr>
<tr><td>2</td><td>No input / output variable assigned (copy pointer = 0)</td></tr>
<tr><td>3</td><td>Illegal device instance (symbolic
device name might have been assigned incorrectly)</td></tr>
</table></td></tr>
</table>

Input and output variables

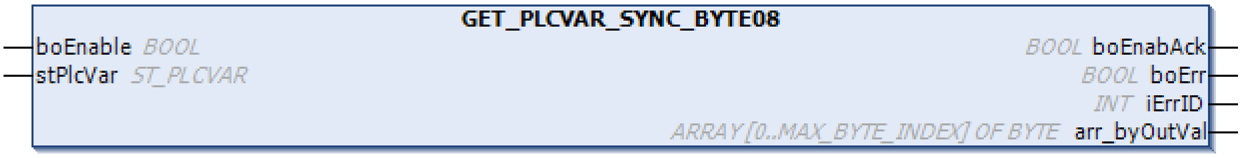
Name	Type	Description
stPlcVar	STRUCT	ST_PLCVAR Information about the PLC variables The 'stPlcVar' input variable must always be assigned a handle (ST_PLCVAR type global variable). This variable must be added to the device tree of the CODESYS project.

13.5.2 Synchronous

13.5.2.1 GET_PLCVAR_SYNC_BYTE08 (FB)

This block receives a synchronous "mapped" byte array 8 bytes in length through 'arr_byOutVal' (MAX_BYTE_INDEX:=8).

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table></td></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error:
<table>
<tr><th>Value</th><th>Meaning</th></tr>
<tr><td>1</td><td>Device information not configured</td></tr>
<tr><td>2</td><td>No input / output variable assigned (copy pointer = 0)</td></tr>
<tr><td>3</td><td>Illegal device instance (symbolic
device name might have been assigned incorrectly)</td></tr>
</table></td></tr>
<tr><td>arr_byOutVal</td><td>ARRAY</td><td>BYTE08 Output value</td></tr>
</table>

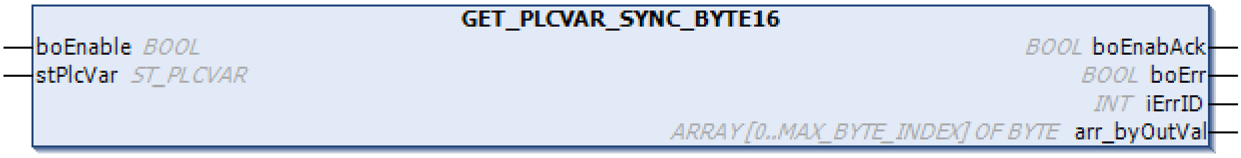
Input and output variables

Name	Type	Description
stPlcVar	STRUCT	ST_PLCVAR Information about the PLC variables The 'stPlcVar' input variable must always be assigned a handle (ST_PLCVAR type global variable). This variable must be added to the device tree of the CODESYS project.

13.5.2.2 GET_PLCVAR_SYNC_BYTE16 (FB)

This block receives a synchronous "mapped" byte array 16 bytes in length through 'arr_byOutVal' (MAX_BYTE_INDEX:=16).

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table></td></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error:
<table>
<tr><td>Value</td><td>Meaning</td></tr>
<tr><td>1</td><td>Device information not configured</td></tr>
<tr><td>2</td><td>No input / output variable assigned (copy pointer = 0)</td></tr>
<tr><td>3</td><td>Illegal device instance (symbolic
device name might have been assigned incorrectly)</td></tr>
</table></td></tr>
<tr><td>arr_byOutVal</td><td>ARRAY</td><td>BYTE16 Output value</td></tr>
</table>

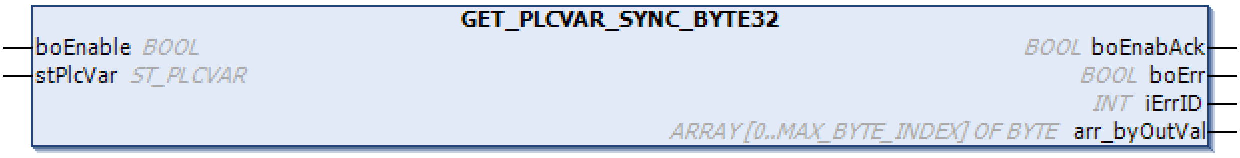
Input and output variables

Name	Type	Description
stPlcVar	STRUCT	ST_PLCVAR Information about the PLC variables The 'stPlcVar' input variable must always be assigned a handle (ST_PLCVAR type global variable). This variable must be added to the device tree of the CODESYS project.

13.5.2.3 GET_PLCVAR_SYNC_BYTE32 (FB)

This block receives a synchronous "mapped" byte array 32 bytes in length through 'arr_byOutVal' (MAX_BYTE_INDEX:=32).

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table></td></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error:
<table>
<tr><td>Value</td><td>Meaning</td></tr>
<tr><td>1</td><td>Device information not configured</td></tr>
<tr><td>2</td><td>No input / output variable assigned (copy pointer = 0)</td></tr>
<tr><td>3</td><td>Illegal device instance (symbolic
device name might have been assigned incorrectly)</td></tr>
</table></td></tr>
<tr><td>arr_byOutVal</td><td>ARRAY</td><td>BYTE32 Output value</td></tr>
</table>

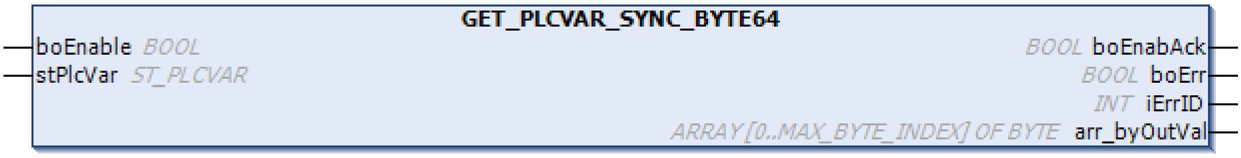
Input and output variables

Name	Type	Description
stPlcVar	STRUCT	ST_PLCVAR Information about the PLC variables The 'stPlcVar' input variable must always be assigned a handle (ST_PLCVAR type global variable). This variable must be added to the device tree of the CODESYS project.

13.5.2.4 GET_PLCVAR_SYNC_BYTE64 (FB)

This block receives a synchronous "mapped" byte array 64 bytes in length through 'arr_byOutVal' (MAX_BYTE_INDEX:=64).

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table></td></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error:
<table>
<tr><td>Value</td><td>Meaning</td></tr>
<tr><td>1</td><td>Device information not configured</td></tr>
<tr><td>2</td><td>No input / output variable assigned (copy pointer = 0)</td></tr>
<tr><td>3</td><td>Illegal device instance (symbolic
device name might have been assigned incorrectly)</td></tr>
</table></td></tr>
<tr><td>arr_byOutVal</td><td>ARRAY</td><td>BYTE64 Output value</td></tr>
</table>

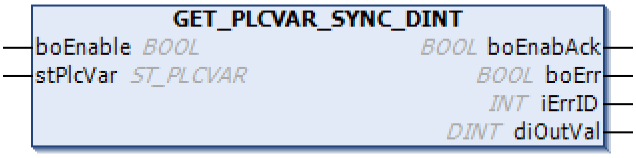
Input and output variables

Name	Type	Description
stPlcVar	STRUCT	ST_PLCVAR Information about the PLC variables The 'stPlcVar' input variable must always be assigned a handle (ST_PLCVAR type global variable). This variable must be added to the device tree of the CODESYS project.

13.5.2.5 GET_PLCVAR_SYNC_DINT (FB)

This block receives a synchronous "mapped" DINT type variable through 'diOutVal'.

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table></td></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error:
<table>
<tr><td>Value</td><td>Meaning</td></tr>
<tr><td>1</td><td>Device information not configured</td></tr>
<tr><td>2</td><td>No input / output variable assigned (copy pointer = 0)</td></tr>
<tr><td>3</td><td>Illegal device instance (symbolic
device name might have been assigned incorrectly)</td></tr>
</table></td></tr>
<tr><td>diOutVal</td><td>DINT</td><td>Output value</td></tr>
</table>

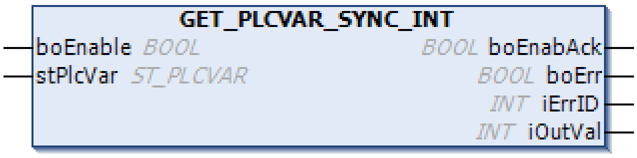
Input and output variables

Name	Type	Description
stPlcVar	STRUCT	ST_PLCVAR Information about the PLC variables The 'stPlcVar' input variable must always be assigned a handle (ST_PLCVAR type global variable). This variable must be added to the device tree of the CODESYS project.

13.5.2.6 GET_PLCVAR_SYNC_INT (FB)

This block receives a synchronous "mapped" INT type variable through 'iOutVal'.

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table></td></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error:
<table>
<tr><td>Value</td><td>Meaning</td></tr>
<tr><td>1</td><td>Device information not configured</td></tr>
<tr><td>2</td><td>No input / output variable assigned (copy pointer = 0)</td></tr>
<tr><td>3</td><td>Illegal device instance (symbolic
device name might have been assigned incorrectly)</td></tr>
</table></td></tr>
<tr><td>iOutVal</td><td>INT</td><td>Output value</td></tr>
</table>

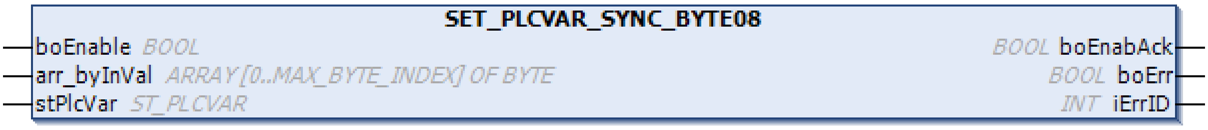
Input and output variables

Name	Type	Description
stPlcVar	STRUCT	ST_PLCVAR Information about the PLC variables The 'stPlcVar' input variable must always be assigned a handle (ST_PLCVAR type global variable). This variable must be added to the device tree of the CODESYS project.

13.5.2.7 SET_PLCVAR_SYNC_BYTE08 (FB)

This block sends a synchronous "mapped" byte array 8 bytes in length through 'arr_byInVal' (MAX_BYTE_INDEX:=8).

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.
arr_byInVal	ARRAY	BYTE08 Input value

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table></td></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error:
<table>
<tr><td>Value</td><td>Meaning</td></tr>
<tr><td>1</td><td>Device information not configured</td></tr>
<tr><td>2</td><td>No input / output variable assigned (copy pointer = 0)</td></tr>
<tr><td>3</td><td>Illegal device instance (symbolic
device name might have been assigned incorrectly)</td></tr>
</table></td></tr>
</table>

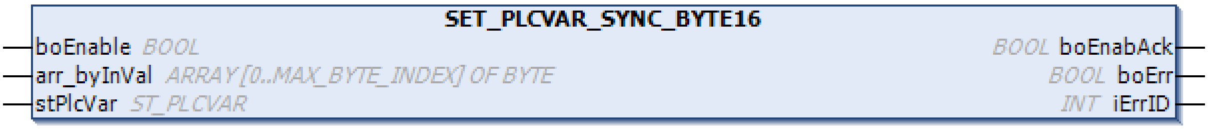
Input and output variables

Name	Type	Description
stPlcVar	STRUCT	ST_PLCVAR Information about the PLC variables The 'stPlcVar' input variable must always be assigned a handle (ST_PLCVAR type global variable). This variable must be added to the device tree of the CODESYS project.

13.5.2.8 SET_PLCVAR_SYNC_BYTE16 (FB)

This block sends a synchronous "mapped" byte array 16 bytes in length through 'arr_byInVal' (MAX_BYTE_INDEX:=16).

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.
arr_byInVal	ARRAY	BYTE16 Input value

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table></td></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error:
<table>
<tr><td>Value</td><td>Meaning</td></tr>
<tr><td>1</td><td>Device information not configured</td></tr>
<tr><td>2</td><td>No input / output variable assigned (copy pointer = 0)</td></tr>
<tr><td>3</td><td>Illegal device instance (symbolic
device name might have been assigned incorrectly)</td></tr>
</table></td></tr>
</table>

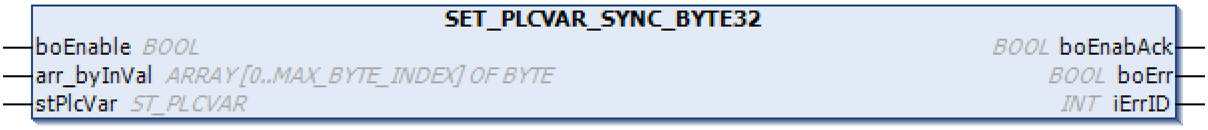
Input and output variables

Name	Type	Description
stPlcVar	STRUCT	ST_PLCVAR Information about the PLC variables The 'stPlcVar' input variable must always be assigned a handle (ST_PLCVAR type global variable). This variable must be added to the device tree of the CODESYS project.

13.5.2.9 SET_PLCVAR_SYNC_BYTE32 (FB)

This block sends a synchronous "mapped" byte array 32 bytes in length through 'arr_byInVal' (MAX_BYTE_INDEX:=32).

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.
arr_byInVal	ARRAY	BYTE32 Input value

Output variables

Name	Type	Description
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled
boErr	BOOL	The function block is in an error state FALSE: No error (permitted commanding or warning) TRUE: Error
iErrID	INT	Error identity number: Diagnostic number is output iErrID = 0: No error iErrID ≠ 0, boErr = TRUE: Error iErrID ≠ 0, boErr = FALSE: Warning Error: Value / Meaning 1: Device information not configured 2: No input / output variable assigned (copy pointer = 0) 3: Illegal device instance (symbolic device name might have been assigned incorrectly)

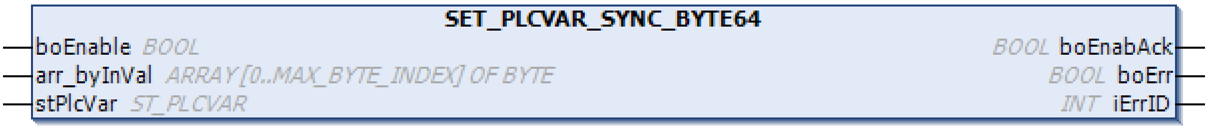
Input and output variables

Name	Type	Description
stPlcVar	STRUCT	ST_PLCVAR Information about the PLC variables The 'stPlcVar' input variable must always be assigned a handle (ST_PLCVAR type global variable). This variable must be added to the device tree of the CODESYS project.

13.5.2.10 SET_PLCVAR_SYNC_BYTE64 (FB)

This block sends a synchronous "mapped" byte array 64 bytes in length through 'arr_byInVal' (MAX_BYTE_INDEX:=64).

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.
arr_byInVal	ARRAY	BYTE64 Input value

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table></td></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error:
<table>
<tr><td>Value</td><td>Meaning</td></tr>
<tr><td>1</td><td>Device information not configured</td></tr>
<tr><td>2</td><td>No input / output variable assigned (copy pointer = 0)</td></tr>
<tr><td>3</td><td>Illegal device instance (symbolic
device name might have been assigned incorrectly)</td></tr>
</table></td></tr>
</table>

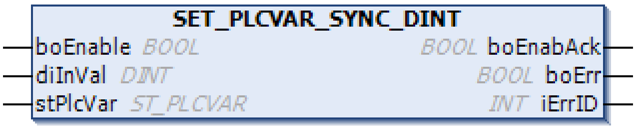
Input and output variables

Name	Type	Description
stPlcVar	STRUCT	ST_PLCVAR Information about the PLC variables The 'stPlcVar' input variable must always be assigned a handle (ST_PLCVAR type global variable). This variable must be added to the device tree of the CODESYS project.

13.5.2.11 SET_PLCVAR_SYNC_DINT (FB)

This block sends a synchronous "mapped" DINT type variable through 'diInVal'.

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.
diInVal	DINT	Input value

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table></td></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error:
<table>
<tr><td>Value</td><td>Meaning</td></tr>
<tr><td>1</td><td>Device information not configured</td></tr>
<tr><td>2</td><td>No input / output variable assigned (copy pointer = 0)</td></tr>
<tr><td>3</td><td>Illegal device instance (symbolic
device name might have been assigned incorrectly)</td></tr>
</table></td></tr>
</table>

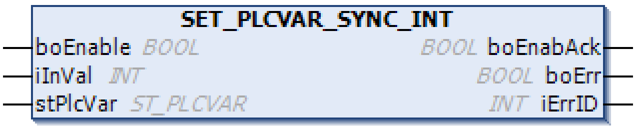
Input and output variables

Name	Type	Description
stPlcVar	STRUCT	ST_PLCVAR Information about the PLC variables The 'stPlcVar' input variable must always be assigned a handle (ST_PLCVAR type global variable). This variable must be added to the device tree of the CODESYS project.

13.5.2.12 SET_PLCVAR_SYNC_INT (FB)

This block sends a synchronous "mapped" INT type variable through 'iInVal'.

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.
iInVal	INT	Input value

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table></td></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error:
<table>
<tr><th>Value</th><th>Meaning</th></tr>
<tr><td>1</td><td>Device information not configured</td></tr>
<tr><td>2</td><td>No input / output variable assigned (copy pointer = 0)</td></tr>
<tr><td>3</td><td>Illegal device instance (symbolic
device name might have been assigned incorrectly)</td></tr>
</table></td></tr>
</table>

Input and output variables

Name	Type	Description
stPlcVar	STRUCT	ST_PLCVAR Information about the PLC variables The 'stPlcVar' input variable must always be assigned a handle (ST_PLCVAR type global variable). This variable must be added to the device tree of the CODESYS project.

13.6 Special (blocks for specific buses and devices)

For access to special quantities which are specific to bus systems and / or manufacturers and cannot be managed in the same way for all systems.

The blocks in the special folder are specific to bus systems and / or manufacturers. Changing the default parameter settings required for the automatic bus configuration can lead to problems affecting predefined functionality. Siehe 'Parameterization' auf Seite 267.

DeviceAccessAsync

GET_ERROR_ID11	Get ID11 'Status class 1-errors'
GET_STATUS_ID144	Get ID144 'Status word'

AmkCanCommunication_ACC

GET_ERROR_OPT	Get 'option error info'
GET_ERROR_SYS	Get 'system error info'

Local

- iSA

GET_DC_BUS_VOLTAGE	Get "DC bus voltage"
GET_HEAT_SINK_TEMPERATURE	Get "heat sink temperature"
GET_INTERIOR_TEMPERATURE	Get "interior temperature"

Sercos

- Command
  - Control / Status

SET_CTRL_RT_BIT1	Set real-time control bit 1
SET_CTRL_RT_BIT2	Set real-time control bit2
GET_STAT_RT_BIT1	Set real-time status bit1
GET_STAT_RT_BIT2	Set real-time status bit2

- Error

GET_STAT_CLASS2	Get ID12 'Status class 2-warnings'

DeviceAccessSync

AmkCanCommunikation_ACC

GET_ACTVAL16_0	Get ID32839/2 'Actual value list'/'actvalue16_0'
GET_ACTVAL16_1	Get ID32839/3 'Actual value list'/'actvalue16_1'
GET_ACTVAL16_2	Get ID32839/4 'Actual value list'/'actvalue16_2'
GET_ACTVAL32_0	Get ID32839/12 'Actual value list'/'actvalue32_0'

GET_ACTVAL32_1	Get ID32839/13 'Actual value list'/'actvalue32_1'
GET_MESSAGE16	Get ID32785 'Message 16'
GET_MESSAGE32	Get ID32786 'Message 32'
SET_ADD_SETPOINT16	Set 'additional setpoint16'
SET_ADD_SETPOINT32	Set 'additional setpoint32'
SET_MAIN_SETPOINT	Set 'main setpoint'
SET_SETPOINT16_0	Set ID32838/2 'Actual value list'/'setpoint16_0'
SET_SETPOINT16_1	Set ID32838/3 'Actual value list'/'setpoint16_1'
SET_SETPOINT16_2	Set ID32838/4 'Actual value list'/'setpoint16_2'
SET_SETPOINT16_3	Set ID32838/5 'Actual value list'/'setpoint16_3'
SET_SETPOINT32_0	Set ID32838/12 'Actual value list'/'setpoint32_0'
SET_SETPOINT32_1	Set ID32838/13 'Actual value list'/'setpoint32_1'

Sercos

GET_FOLLOW_ERR	Get ID189 error 'Following distance'
SET_LIM_SPEED_BIPOL	Set ID91 'Bipolar velocity limit'
SET_LIM_SPEED_NEG	Set ID39 'Negative velocity limit'
SET_LIM_SPEED_POS	Set ID38 'Positive velocity limit'
SET_LIM_TORQUE_BIPOL	Set ID92 'Bipolar torque limit'
SET_LIM_TORQUE_NEG	Set ID83 'Negative torque limit'
SET_LIM_TORQUE_POS	Set ID82 'Positive torque limit'
SET_SETPOINT_DIV	Set ID32892 'Synchronous setpoint pulses divider'
SET_SETPOINT_MUL	Set ID32893 'Synchronous setpoint pulses multiplier'
SET_SETPOINT_SIWL	Set ID33911 'SIWL setpoint'

- ProcessIO

GET_ACTPOS_LATCHED_NEG1	Get ID131 'Probe value 1 negative edge'
GET_ACTPOS_LATCHED_NEG2	Get ID133 'Probe value 2 negative edge'
GET_ACTPOS_LATCHED_POS1	Get ID130 'Probe value 1 positive edge'
GET_ACTPOS_LATCHED_POS2	Get ID132 'Probe value 2 positive edge'
GET_PROBE_STS	Get ID179 'Probe status'

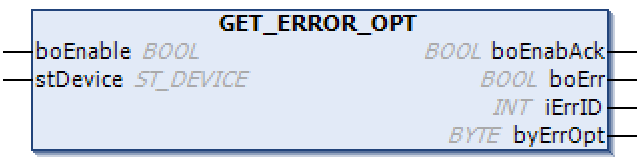
13.6.1 DeviceAccessAsync

13.6.1.1 AmkCanCommunication_ACC

13.6.1.1.1 GET_ERROR_OPT (FB)

This block queries "option error" through the 'byErrOpt' variable.

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table></td></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error:
<table>
<tr><td>Value</td><td>Meaning</td></tr>
<tr><td>1</td><td>Device information not configured</td></tr>
<tr><td>2</td><td>No input / output variable assigned (copy pointer = 0)</td></tr>
<tr><td>3</td><td>Illegal device instance (symbolic
device name might have been assigned incorrectly)</td></tr>
</table></td></tr>
<tr><td>byErrOpt</td><td>BYTE</td><td>Option error information</td></tr>
</table>

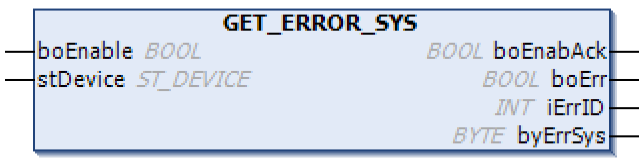
Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

13.6.1.1.2 GET_ERROR_SYS (FB)

This block queries "system error info" through the 'byErrSys' variable.

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table></td></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error:
<table>
<tr><td>Value</td><td>Meaning</td></tr>
<tr><td>1</td><td>Device information not configured</td></tr>
<tr><td>2</td><td>No input / output variable assigned (copy pointer = 0)</td></tr>
<tr><td>3</td><td>Illegal device instance (symbolic
device name might have been assigned incorrectly)</td></tr>
</table></td></tr>
<tr><td>byErrSys</td><td>BYTE</td><td>System error information</td></tr>
</table>

Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

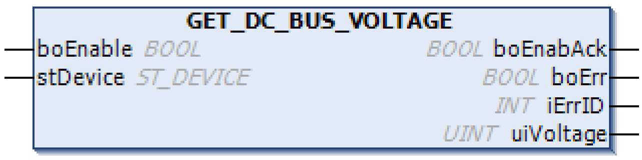
13.6.1.2 Local

13.6.1.2.1 iSA

13.6.1.2.1.1 GET_DC_BUS_VOLTAGE (FB)

This block queries "DC bus voltage" (ID32836 'DC bus voltage') through the 'uiVoltage' variable.

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.

Output variables

Name	Type	Description
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled

Name	Type	Description
boErr	BOOL	The function block is in an error state FALSE: No error (permitted commanding or warning) TRUE: Error
iErrID	INT	Error identity number: Diagnostic number is output iErrID = 0: No error iErrID ≠ 0, boErr = TRUE: Error iErrID ≠ 0, boErr = FALSE: Warning Error: Value / Meaning 1: Device information not configured 2: No input / output variable assigned (copy pointer = 0) 3: Illegal device instance (symbolic device name might have been assigned incorrectly)
uiVoltage	UINT	Actual voltage DC bus [Volt] Unit: V

Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

13.6.1.2.1.2 GET_HEAT_SINK_TEMPERATURE (FB)

This block queries "heat sink temperature" (ID33116) through the 'uiTemperature' variable.

User interface

Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.

Output variables

Name	Type	Description
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled
boErr	BOOL	The function block is in an error state FALSE: No error (permitted commanding or warning) TRUE: Error

Name	Type	Description
iErrID	INT	Error identity number: Diagnostic number is output iErrID = 0 — No error iErrID ≠ 0 — boErr = TRUE — Error iErrID ≠ 0 — boErr = FALSE — Warning Error: Value — Meaning 1 — Device information not configured 2 — No input / output variable assigned (copy pointer = 0) 3 — Illegal device instance (symbolic device name might have been assigned incorrectly)
uiTemperature	UINT	Actual heat sink temperature [°C] Unit — 0.1 °C

Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

13.6.1.2.1.3 GET_INTERIOR_TEMPERATURE (FB)

This block queries "interior temperature" (ID32810 'Inner room temperature') through the 'uiTemperature' variable.

User interface

Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.

Output variables

Name	Type	Description
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled
boErr	BOOL	The function block is in an error state FALSE — No error (permitted commanding or warning) TRUE — Error

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error:
<table>
<tr><td>Value</td><td>Meaning</td></tr>
<tr><td>1</td><td>Device information not configured</td></tr>
<tr><td>2</td><td>No input / output variable assigned (copy pointer = 0)</td></tr>
<tr><td>3</td><td>Illegal device instance (symbolic
device name might have been assigned incorrectly)</td></tr>
</table>
</td></tr>
<tr><td>uiTemperature</td><td>UINT</td><td>Actual interior temperature power supply [°C]
<table>
<tr><td>Unit</td><td>0.1 °C</td></tr>
</table>
</td></tr>
</table>

Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

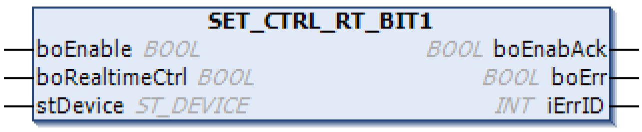
13.6.1.3 Sercos

13.6.1.3.1 Command

13.6.1.3.1.1 SET_CTRL_RT_BIT1 (FB)

This block sets real-time control bit1 (ID134 bit 6 'Master control word') through the 'boRealtimeCtrl' variable.

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.
boRealtimeCtrl	BOOL	Real-time control bit1 (ID134 bit 6 'Master control word') (See document Parameter description ID134 'Master control word', Part no. 203704)

Output variables

Name	Type	Description
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled

Name	Type	Description
boErr	BOOL	The function block is in an error state FALSE — No error (permitted commanding or warning) TRUE — Error
iErrID	INT	Error identity number: Diagnostic number is output iErrID = 0 — No error iErrID ≠ 0, boErr = TRUE — Error iErrID ≠ 0, boErr = FALSE — Warning Error: Value — Meaning 1 — Device information not configured 2 — No input / output variable assigned (copy pointer = 0) 3 — Illegal device instance (symbolic device name might have been assigned incorrectly)

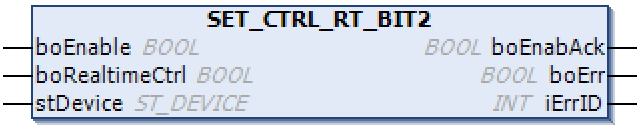
Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

13.6.1.3.1.2 SET_CTRL_RT_BIT2 (FB)

This block sets real-time control bit2 (ID134 bit 7 'Master control word') through the 'boRealtimeCtrl' variable.

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.
boRealtimeCtrl	BOOL	Real-time control bit2 (ID134 bit 7 'Master control word') (See document Parameter description ID134 'Master control word', Part no. 203704)

Output variables

Name	Type	Description
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled
boErr	BOOL	The function block is in an error state FALSE — No error (permitted commanding or warning) TRUE — Error

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error:
<table>
<tr><td>Value</td><td>Meaning</td></tr>
<tr><td>1</td><td>Device information not configured</td></tr>
<tr><td>2</td><td>No input / output variable assigned (copy pointer = 0)</td></tr>
<tr><td>3</td><td>Illegal device instance (symbolic
device name might have been assigned incorrectly)</td></tr>
</table>
</td></tr>
</table>

Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

13.6.1.3.1.3 GET_STAT_RT_BIT1 (FB)

This block queries real-time status bit1 (ID135 bit 6 'Drive status word') through the 'boRealtimeStat' variable.

User interface

Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table>
</td></tr>
</table>

Name	Type	Description
iErrID	INT	Error identity number: Diagnostic number is output iErrID = 0 — No error iErrID ≠ 0 — boErr = TRUE — Error iErrID ≠ 0 — boErr = FALSE — Warning Error: Value — Meaning 1 — Device information not configured 2 — No input / output variable assigned (copy pointer = 0) 3 — Illegal device instance (symbolic device name might have been assigned incorrectly)
boRealtimeStat	BOOL	Real-time status bit1 (ID135 bit 6 'Drive status word') (See document Parameter description ID135 'Drive status word', Part no. 203704)

Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

13.6.1.3.1.4 GET_STAT_RT_BIT2 (FB)

This block queries real-time status bit2 (ID135 bit 7 'Drive status word') through the 'boRealtimeStat' variable.

User interface

Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.

Output variables

Name	Type	Description
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled
boErr	BOOL	The function block is in an error state FALSE — No error (permitted commanding or warning) TRUE — Error

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error:
<table>
<tr><th>Value</th><th>Meaning</th></tr>
<tr><td>1</td><td>Device information not configured</td></tr>
<tr><td>2</td><td>No input / output variable assigned (copy pointer = 0)</td></tr>
<tr><td>3</td><td>Illegal device instance (symbolic
device name might have been assigned incorrectly)</td></tr>
</table>
</td></tr>
<tr><td>boRealtimeStat</td><td>BOOL</td><td>Real-time status bit2 (ID135 bit 6 'Drive status word')
(See document Parameter description ID135 'Drive status word', Part no. 203704)</td></tr>
</table>

Input and output variables

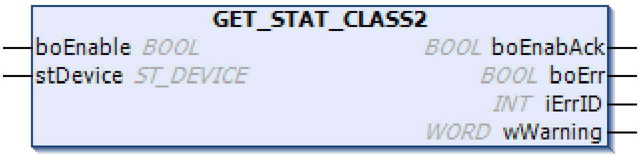
Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

13.6.1.3.2 Error

13.6.1.3.2.1 GET_STAT_CLASS2 (FB)

This block queries ID12 'Status class 2-warnings' through the 'wWarning' variable.

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error:
<table>
<tr><td>Value</td><td>Meaning</td></tr>
<tr><td>1</td><td>Device information not configured</td></tr>
<tr><td>2</td><td>No input / output variable assigned (copy pointer = 0)</td></tr>
<tr><td>3</td><td>Illegal device instance (symbolic
device name might have been assigned incorrectly)</td></tr>
</table>
</td></tr>
<tr><td>wWarning</td><td>WORD</td><td>Warning ID12 'Status class 2-warnings'
(See document Parameter description ID12 'Status class 2-warnings' , Part no. 203704)</td></tr>
</table>

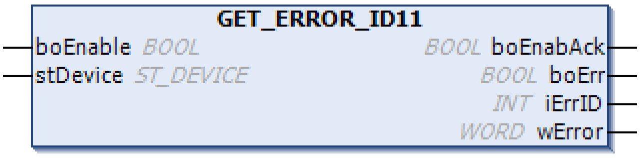
Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

13.6.1.4 GET_ERROR_ID11 (FB)

This block queries ID11 'Status class 1-errors' through the 'wError' variable.

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error:
<table>
<tr><td>Value</td><td>Meaning</td></tr>
<tr><td>1</td><td>Device information not configured</td></tr>
<tr><td>2</td><td>No input / output variable assigned (copy pointer = 0)</td></tr>
<tr><td>3</td><td>Illegal device instance (symbolic
device name might have been assigned incorrectly)</td></tr>
</table>
</td></tr>
<tr><td>wError</td><td>WORD</td><td>Get error information ID11 'Status class 1-errors'
(See document Parameter description ID11'Status class 1-errors', Part no. 203704)</td></tr>
</table>

Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

13.6.1.5 GET_STATUS_ID144 (FB)

This block queries ID144 'Status word' through the 'wStatus' variable.

User interface

Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error:
<table>
<tr><td>Value</td><td>Meaning</td></tr>
<tr><td>1</td><td>Device information not configured</td></tr>
<tr><td>2</td><td>No input / output variable assigned (copy pointer = 0)</td></tr>
<tr><td>3</td><td>Illegal device instance (symbolic
device name might have been assigned incorrectly)</td></tr>
</table>
</td></tr>
<tr><td>wStatus</td><td>WORD</td><td>Status word information ID144 'Status word'
(See document Parameter description ID144 'Status word', Part no. 26249)</td></tr>
</table>

Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

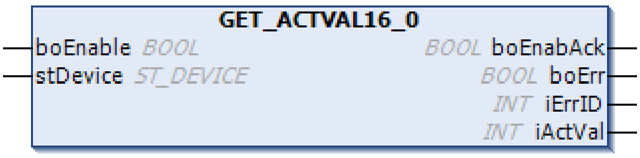
13.6.2 DeviceAccessSync

13.6.2.1 AmkCanCommunication_ACC

13.6.2.1.1 GET_ACTVAL16_0 (FB)

This block queries "actvalue16_0" (ID32839 'Actual value list', list element2) through the 'iActVal' variable.

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error:
<table>
<tr><td>Value</td><td>Meaning</td></tr>
<tr><td>1</td><td>Device information not configured</td></tr>
<tr><td>2</td><td>No input / output variable assigned (copy pointer = 0)</td></tr>
<tr><td>3</td><td>Illegal device instance (symbolic
device name might have been assigned incorrectly)</td></tr>
</table>
</td></tr>
<tr><td>iActVal</td><td>INT</td><td>Actual value (ID32839 'Actual value list', list element2)
(See document Parameter description ID32839 'Actual value list' , Part no. 26249)</td></tr>
</table>

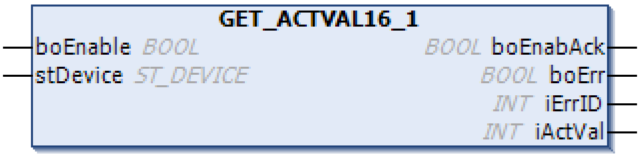
Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

13.6.2.1.2 GET_ACTVAL16_1 (FB)

This block queries "actvalue16_1" (ID32839 'Actual value list', list element3) through the 'iActVal' variable.

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error:
<table>
<tr><td>Value</td><td>Meaning</td></tr>
<tr><td>1</td><td>Device information not configured</td></tr>
<tr><td>2</td><td>No input / output variable assigned (copy pointer = 0)</td></tr>
<tr><td>3</td><td>Illegal device instance (symbolic
device name might have been assigned incorrectly)</td></tr>
</table></td></tr>
<tr><td>iActVal</td><td>INT</td><td>Actual value (ID32839 'Actual value list', list element3)
(See document Parameter description ID32839 'Actual value list' , Part no. 26249)</td></tr>
</table>

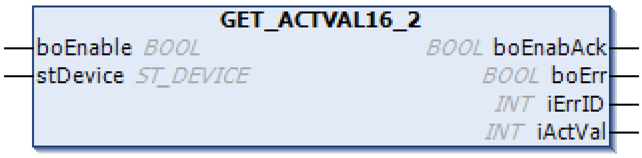
Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

13.6.2.1.3 GET_ACTVAL16_2 (FB)

This block queries "actvalue16_2" (ID32839 'Actual value list', list element4) through the 'iActVal' variable.

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table></td></tr>
</table>

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error:
<table>
<tr><td>Value</td><td>Meaning</td></tr>
<tr><td>1</td><td>Device information not configured</td></tr>
<tr><td>2</td><td>No input / output variable assigned (copy pointer = 0)</td></tr>
<tr><td>3</td><td>Illegal device instance (symbolic
device name might have been assigned incorrectly)</td></tr>
</table>
</td></tr>
<tr><td>iActVal</td><td>INT</td><td>Actual value (ID32839 'Actual value list', list element4)
(See document Parameter description ID32839 'Actual value list' , Part no. 26249)</td></tr>
</table>

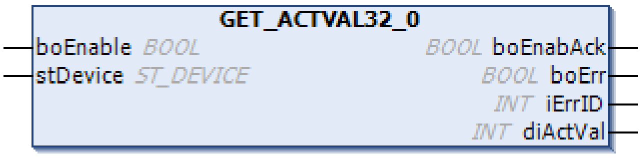
Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

13.6.2.1.4 GET_ACTVAL32_0 (FB)

This block queries "actvalue32_0" (ID32839 'Actual value list', list element12) through the 'diActVal' variable.

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error:
<table>
<tr><td>Value</td><td>Meaning</td></tr>
<tr><td>1</td><td>Device information not configured</td></tr>
<tr><td>2</td><td>No input / output variable assigned (copy pointer = 0)</td></tr>
<tr><td>3</td><td>Illegal device instance (symbolic
device name might have been assigned incorrectly)</td></tr>
</table>
</td></tr>
<tr><td>diActVal</td><td>DINT</td><td>Actual value (ID32839 'Actual value list', list element12)
(See document Parameter description ID32839 'Actual value list' , Part no. 26249)</td></tr>
</table>

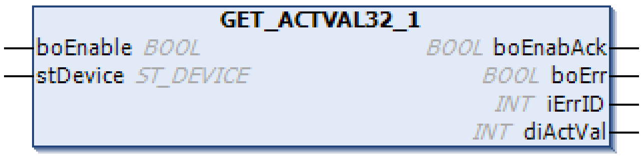
Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

13.6.2.1.5 GET_ACTVAL32_1 (FB)

This block queries "actvalue32_1" (ID32839 'Actual value list', list element13) through the 'diActVal' variable.

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error:
<table>
<tr><td>Value</td><td>Meaning</td></tr>
<tr><td>1</td><td>Device information not configured</td></tr>
<tr><td>2</td><td>No input / output variable assigned (copy pointer = 0)</td></tr>
<tr><td>3</td><td>Illegal device instance (symbolic
device name might have been assigned incorrectly)</td></tr>
</table>
</td></tr>
<tr><td>diActVal</td><td>DINT</td><td>Actual value (ID32839 'Actual value list', list element13)
(See document Parameter description ID32839 'Actual value list' , Part no. 26249)</td></tr>
</table>

Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

13.6.2.1.6 GET_MESSAGE16 (FB)

This block queries "config message16" (ID32785 'Message 16') through the 'iMessage' variable.

User interface

Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error:
<table>
<tr><td>Value</td><td>Meaning</td></tr>
<tr><td>1</td><td>Device information not configured</td></tr>
<tr><td>2</td><td>No input / output variable assigned (copy pointer = 0)</td></tr>
<tr><td>3</td><td>Illegal device instance (symbolic
device name might have been assigned incorrectly)</td></tr>
</table>
</td></tr>
<tr><td>iMessage</td><td>INT</td><td>Configuration message (ID32785 'Message 16')
(See document Parameter description ID32785 'Message 16' , Part no. 26249)</td></tr>
</table>

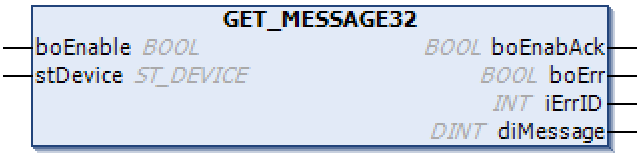
Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

13.6.2.1.7 GET_MESSAGE32 (FB)

This block queries "config message32" (ID32786 'Message 32') through the 'diMessage' variable.

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error:
<table>
<tr><td>Value</td><td>Meaning</td></tr>
<tr><td>1</td><td>Device information not configured</td></tr>
<tr><td>2</td><td>No input / output variable assigned (copy pointer = 0)</td></tr>
<tr><td>3</td><td>Illegal device instance (symbolic
device name might have been assigned incorrectly)</td></tr>
</table>
</td></tr>
<tr><td>diMessage</td><td>DINT</td><td>Configuration message (ID32786 'Message 32')
(See document Parameter description ID32786 'Message 32' , Part no. 26249)</td></tr>
</table>

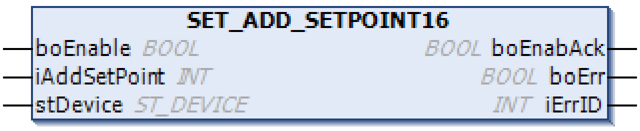
Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

13.6.2.1.8 SET_ADD_SETPOINT16 (FB)

This block sets "additional setpoint16" through the 'iAddSetPoint' variable.

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.
iAddSetPoint	INT	Additional setpoint16

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table>
</td></tr>
</table>

Name	Type	Description
iErrID	INT	Error identity number: Diagnostic number is output iErrID = 0 — No error iErrID ≠ 0, boErr = TRUE — Error iErrID ≠ 0, boErr = FALSE — Warning Error: Value — Meaning 1 — Device information not configured 2 — No input / output variable assigned (copy pointer = 0) 3 — Illegal device instance (symbolic device name might have been assigned incorrectly)

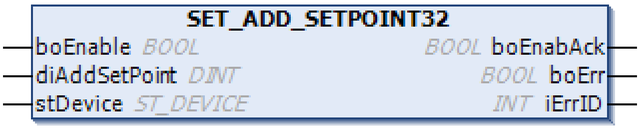
Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

13.6.2.1.9 SET_ADD_SETPOINT32 (FB)

This block sets "additional setpoint32" through the 'diAddSetPoint' variable.

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.
diAddSetPoint	DINT	Additional setpoint32

Output variables

Name	Type	Description
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled
boErr	BOOL	The function block is in an error state FALSE — No error (permitted commanding or warning) TRUE — Error

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error:
<table>
<tr><td>Value</td><td>Meaning</td></tr>
<tr><td>1</td><td>Device information not configured</td></tr>
<tr><td>2</td><td>No input / output variable assigned (copy pointer = 0)</td></tr>
<tr><td>3</td><td>Illegal device instance (symbolic
device name might have been assigned incorrectly)</td></tr>
</table>
</td></tr>
</table>

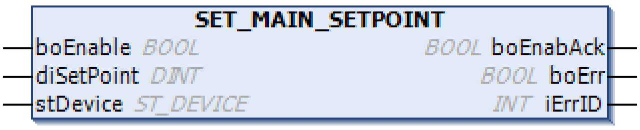
Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

13.6.2.1.10 SET_MAIN_SETPOINT (FB)

This block sets "main setpoint" through the 'diSetPoint' variable.

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.
diSetPoint	INT	Main Setpoint

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error:
<table>
<tr><td>Value</td><td>Meaning</td></tr>
<tr><td>1</td><td>Device information not configured</td></tr>
<tr><td>2</td><td>No input / output variable assigned (copy pointer = 0)</td></tr>
<tr><td>3</td><td>Illegal device instance (symbolic
device name might have been assigned incorrectly)</td></tr>
</table>
</td></tr>
</table>

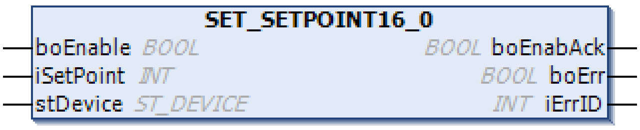
Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

13.6.2.1.11 SET_SETPOINT16_0 (FB)

This block queries ""setpoint16_0" (ID32838 'Actual value list', list element2) through the 'iSetPoint' variable.

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.
iSetPoint	INT	Setpoint16_0 (ID32838 'Actual value list', list element2) (See document Parameter description ID32838 'Actual value list' , Part no. 26249)

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error:
<table>
<tr><td>Value</td><td>Meaning</td></tr>
<tr><td>1</td><td>Device information not configured</td></tr>
<tr><td>2</td><td>No input / output variable assigned (copy pointer = 0)</td></tr>
<tr><td>3</td><td>Illegal device instance (symbolic
device name might have been assigned incorrectly)</td></tr>
</table>
</td></tr>
</table>

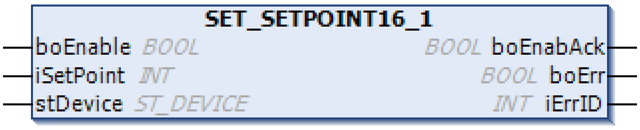
Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

13.6.2.1.12 SET_SETPOINT16_1 (FB)

This block queries ""setpoint16_1" (ID32838 'Actual value list', list element3) through the 'iSetPoint' variable.

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.
iSetPoint	INT	Setpoint16_1 (ID32838 'Actual value list', list element3) (See document Parameter description ID32838 'Actual value list' , Part no. 26249)

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error:
<table>
<tr><td>Value</td><td>Meaning</td></tr>
<tr><td>1</td><td>Device information not configured</td></tr>
<tr><td>2</td><td>No input / output variable assigned (copy pointer = 0)</td></tr>
<tr><td>3</td><td>Illegal device instance (symbolic
device name might have been assigned incorrectly)</td></tr>
</table>
</td></tr>
</table>

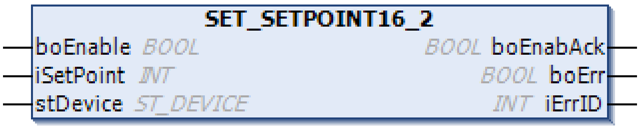
Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

13.6.2.1.13 SET_SETPOINT16_2 (FB)

This block queries ""setpoint16_2" (ID32838 'Actual value list', list element4) through the 'iSetPoint' variable.

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.
iSetPoint	INT	Setpoint16_2 (ID32838 'Actual value list', list element4) (See document Parameter description ID32838 'Actual value list' , Part no. 26249)

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error:
<table>
<tr><td>Value</td><td>Meaning</td></tr>
<tr><td>1</td><td>Device information not configured</td></tr>
<tr><td>2</td><td>No input / output variable assigned (copy pointer = 0)</td></tr>
<tr><td>3</td><td>Illegal device instance (symbolic
device name might have been assigned incorrectly)</td></tr>
</table>
</td></tr>
</table>

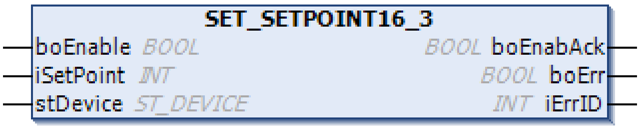
Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

13.6.2.1.14 SET_SETPOINT16_3 (FB)

This block queries ""setpoint16_3" (ID32838 'Actual value list', list element5) through the 'iSetPoint' variable.

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.
iSetPoint	INT	Setpoint16_3 (ID32838 'Actual value list', list element5) (See document Parameter description ID32838 'Actual value list' , Part no. 26249)

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error:
<table>
<tr><td>Value</td><td>Meaning</td></tr>
<tr><td>1</td><td>Device information not configured</td></tr>
<tr><td>2</td><td>No input / output variable assigned (copy pointer = 0)</td></tr>
<tr><td>3</td><td>Illegal device instance (symbolic
device name might have been assigned incorrectly)</td></tr>
</table>
</td></tr>
</table>

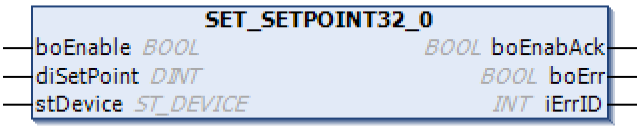
Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

13.6.2.1.15 SET_SETPOINT32_0 (FB)

This block queries ""setpoint32_0" (ID32838 'Actual value list', list element12) through the 'diSetPoint' variable.

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.
diSetPoint	DINT	Setpoint32_0 (ID32838 'Actual value list', list element12) (See document Parameter description ID32838 'Actual value list' , Part no. 26249)

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table>
</td></tr>
</table>

Name	Type	Description
iErrID	INT	Error identity number: Diagnostic number is output iErrID = 0 – No error iErrID ≠ 0, boErr = TRUE – Error iErrID ≠ 0, boErr = FALSE – Warning Error: Value – Meaning 1 – Device information not configured 2 – No input / output variable assigned (copy pointer = 0) 3 – Illegal device instance (symbolic device name might have been assigned incorrectly)

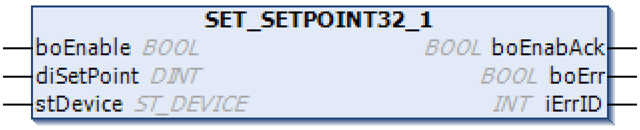
Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

13.6.2.1.16 SET_SETPOINT32_1 (FB)

This block queries ""setpoint32_1" (ID32838 'Actual value list', list element13) through the 'diSetPoint' variable.

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.
diSetPoint	DINT	Setpoint32_1 (ID32838 'Actual value list', list element13) (See document Parameter description ID32838 'Actual value list' , Part no. 26249)

Output variables

Name	Type	Description
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled
boErr	BOOL	The function block is in an error state FALSE – No error (permitted commanding or warning) TRUE – Error

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error:
<table>
<tr><td>Value</td><td>Meaning</td></tr>
<tr><td>1</td><td>Device information not configured</td></tr>
<tr><td>2</td><td>No input / output variable assigned (copy pointer = 0)</td></tr>
<tr><td>3</td><td>Illegal device instance (symbolic
device name might have been assigned incorrectly)</td></tr>
</table>
</td></tr>
</table>

Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

13.6.2.2 Sercos

13.6.2.2.1 ProcessIO

13.6.2.2.1.1 GET_ACTPOS_LATCHED_NEG1 (FB)

This block queries the currently latched position through the negative edge of sensor1 (ID131 'Probe value 1 negative edge') through the 'diLatchedVal' variable.

User interface

Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error:
<table>
<tr><td>Value</td><td>Meaning</td></tr>
<tr><td>1</td><td>Device information not configured</td></tr>
<tr><td>2</td><td>No input / output variable assigned (copy pointer = 0)</td></tr>
<tr><td>3</td><td>Illegal device instance (symbolic
device name might have been assigned incorrectly)</td></tr>
</table></td></tr>
<tr><td>diLatchedVal</td><td>DINT</td><td>Get currently latched position through negative edge of touch probe1 (ID131 'Probe value 1 negative edge')
(See document Parameter description ID131 'Probe value 1 negative edge' , Part no. 203704)</td></tr>
</table>

Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

13.6.2.2.1.2 GET_ACTPOS_LATCHED_NEG2 (FB)

This block queries the currently latched position through the negative edge of sensor2 (ID133 'Probe value 2 negative edge') through the 'diLatchedVal' variable.

User interface

Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table></td></tr>
</table>

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error:
<table>
<tr><td>Value</td><td>Meaning</td></tr>
<tr><td>1</td><td>Device information not configured</td></tr>
<tr><td>2</td><td>No input / output variable assigned (copy pointer = 0)</td></tr>
<tr><td>3</td><td>Illegal device instance (symbolic
device name might have been assigned incorrectly)</td></tr>
</table>
</td></tr>
<tr><td>diLatchedVal</td><td>DINT</td><td>Get currently latched position through negative edge of touch probe2 (ID133 'Probe value 2 negative edge')
(See document Parameter description ID133 'Probe value 2 negative edge' , Part no. 203704)</td></tr>
</table>

Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

13.6.2.2.1.3 GET_ACTPOS_LATCHED_POS1 (FB)

This block queries the currently latched position through the positive edge of sensor1 (ID130 'Probe value 1 positive edge') through the 'diLatchedVal' variable.

User interface

Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error:
<table>
<tr><td>Value</td><td>Meaning</td></tr>
<tr><td>1</td><td>Device information not configured</td></tr>
<tr><td>2</td><td>No input / output variable assigned (copy pointer = 0)</td></tr>
<tr><td>3</td><td>Illegal device instance (symbolic
device name might have been assigned incorrectly)</td></tr>
</table>
</td></tr>
<tr><td>diLatchedVal</td><td>DINT</td><td>Get currently latched position through positive edge of touch probe1 (ID130 'Probe value 1 positive edge')
(See document Parameter description ID130 'Probe value 1 positive edge' , Part no. 203704)</td></tr>
</table>

Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

13.6.2.2.1.4 GET_ACTPOS_LATCHED_POS2 (FB)

This block queries the currently latched position through the positive edge of sensor2 (ID132 'Probe value 2 positive edge') through the 'diLatchedVal' variable.

User interface

Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error:
<table>
<tr><td>Value</td><td>Meaning</td></tr>
<tr><td>1</td><td>Device information not configured</td></tr>
<tr><td>2</td><td>No input / output variable assigned (copy pointer = 0)</td></tr>
<tr><td>3</td><td>Illegal device instance (symbolic
device name might have been assigned incorrectly)</td></tr>
</table>
</td></tr>
<tr><td>diLatchedVal</td><td>DINT</td><td>Get currently latched position through positive edge of touch probe2 (ID132 'Probe value 2 positive edge')
(See document Parameter description ID132 'Probe value 2 positive edge' , Part no. 203704)</td></tr>
</table>

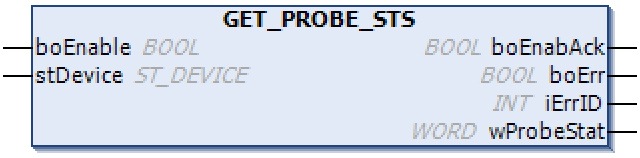
Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

13.6.2.2.1.5 GET_PROBE_STS (FB)

This block queries the measured value status (ID179 'Probe status') through the 'wProbeStat' variable.

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error:
<table>
<tr><td>Value</td><td>Meaning</td></tr>
<tr><td>1</td><td>Device information not configured</td></tr>
<tr><td>2</td><td>No input / output variable assigned (copy pointer = 0)</td></tr>
<tr><td>3</td><td>Illegal device instance (symbolic
device name might have been assigned incorrectly)</td></tr>
</table>
</td></tr>
<tr><td>wProbeStat</td><td>WORD</td><td>Get measured value status (ID179 'Probe status')
(See document Parameter description ID179 'Probe status' , Part no. 203704)</td></tr>
</table>

Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

13.6.2.2.2 GET_FOLLOW_ERR (FB)

This block queries the following error (ID189 'Following distance') through the 'diFollowErr' variable.

User interface

Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error:
<table>
<tr><td>Value</td><td>Meaning</td></tr>
<tr><td>1</td><td>Device information not configured</td></tr>
<tr><td>2</td><td>No input / output variable assigned (copy pointer = 0)</td></tr>
<tr><td>3</td><td>Illegal device instance (symbolic
device name might have been assigned incorrectly)</td></tr>
</table>
</td></tr>
<tr><td>diFollowErr</td><td>DINT</td><td>Error ID189 'Following distance'
(See document Parameter description ID189 'Following distance' , Part no. 203704)</td></tr>
</table>

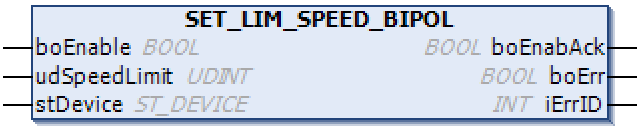
Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

13.6.2.2.3 SET_LIM_SPEED_BIPOL (FB)

This block sets the bipolar speed limit (ID91 'Bipolar velocity limit') through the 'udSpeedLimit' variable.

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.
udSpeedLimit	UDINT	Set ID91 'Bipolar velocity limit' (See document Parameter description ID91 'Bipolar velocity limit' , Part no. 203704)

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error:
<table>
<tr><td>Value</td><td>Meaning</td></tr>
<tr><td>1</td><td>Device information not configured</td></tr>
<tr><td>2</td><td>No input / output variable assigned (copy pointer = 0)</td></tr>
<tr><td>3</td><td>Illegal device instance (symbolic
device name might have been assigned incorrectly)</td></tr>
</table>
</td></tr>
</table>

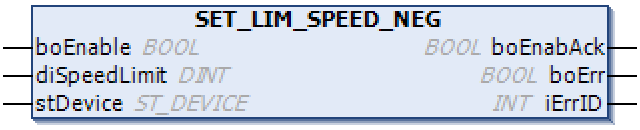
Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

13.6.2.2.4 SET_LIM_SPEED_NEG (FB)

This block sets the negative speed limit (ID39 'Negative velocity limit') through the 'diSpeedLimit' variable.

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.
diSpeedLimit	DINT	Set limit speed negative (ID39 'Negative velocity limit') (See document Parameter description ID39 'Negative velocity limit' , Part no. 203704)

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error:
<table>
<tr><td>Value</td><td>Meaning</td></tr>
<tr><td>1</td><td>Device information not configured</td></tr>
<tr><td>2</td><td>No input / output variable assigned (copy pointer = 0)</td></tr>
<tr><td>3</td><td>Illegal device instance (symbolic
device name might have been assigned incorrectly)</td></tr>
</table>
</td></tr>
</table>

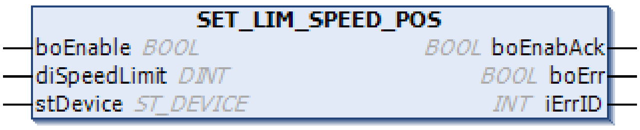
Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

13.6.2.2.5 SET_LIM_SPEED_POS (FB)

This block sets the positive speed limit (ID38 'Positive velocity limit') through the 'diSpeedLimit' variable.

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.
diSpeedLimit	DINT	Set limit speed positive (ID38 'Positive velocity limit') (See document Parameter description ID38 'Positive velocity limit' , Part no. 203704)

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error:
<table>
<tr><td>Value</td><td>Meaning</td></tr>
<tr><td>1</td><td>Device information not configured</td></tr>
<tr><td>2</td><td>No input / output variable assigned (copy pointer = 0)</td></tr>
<tr><td>3</td><td>Illegal device instance (symbolic
device name might have been assigned incorrectly)</td></tr>
</table>
</td></tr>
</table>

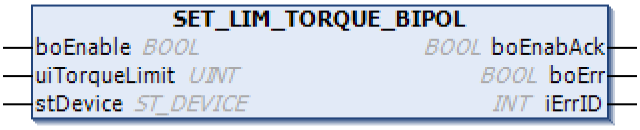
Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

13.6.2.2.6 SET_LIM_TORQUE_BIPOL (FB)

This block sets ID92 'Bipolar torque limit' through the 'uiTorqueLimit' variable.

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.
uiTorqueLimit	UINT	Set ID92 'Bipolar torque limit' (See document Parameter description ID92 'Bipolar torque limit' , Part no. 203704)

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error:
<table>
<tr><td>Value</td><td>Meaning</td></tr>
<tr><td>1</td><td>Device information not configured</td></tr>
<tr><td>2</td><td>No input / output variable assigned (copy pointer = 0)</td></tr>
<tr><td>3</td><td>Illegal device instance (symbolic
device name might have been assigned incorrectly)</td></tr>
</table>
</td></tr>
</table>

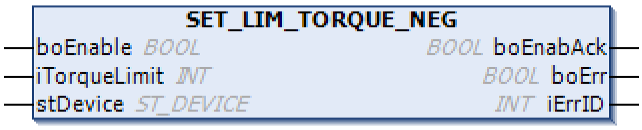
Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

13.6.2.2.7 SET_LIM_TORQUE_NEG (FB)

This block sets ID83 'Negative torque limit' through the 'iTorqueLimit' variable.

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.
iTorqueLimit	INT	Set torque limit negative (ID83 'Negative torque limit') (See document Parameter description ID83 'Negative torque limit' , Part no. 203704)

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error:
<table>
<tr><td>Value</td><td>Meaning</td></tr>
<tr><td>1</td><td>Device information not configured</td></tr>
<tr><td>2</td><td>No input / output variable assigned (copy pointer = 0)</td></tr>
<tr><td>3</td><td>Illegal device instance (symbolic
device name might have been assigned incorrectly)</td></tr>
</table>
</td></tr>
</table>

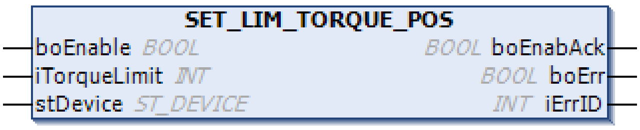
Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

13.6.2.2.8 SET_LIM_TORQUE_POS (FB)

This block sets ID82 'Positive torque limit' through the 'iTorqueLimit' variable.

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.
iTorqueLimit	INT	Set torque limit positive (ID82 'Positive torque limit') (See documentParameter description ID82 'Positive torque limit' , Part no. 203704)

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error:
<table>
<tr><td>Value</td><td>Meaning</td></tr>
<tr><td>1</td><td>Device information not configured</td></tr>
<tr><td>2</td><td>No input / output variable assigned (copy pointer = 0)</td></tr>
<tr><td>3</td><td>Illegal device instance (symbolic
device name might have been assigned incorrectly)</td></tr>
</table>
</td></tr>
</table>

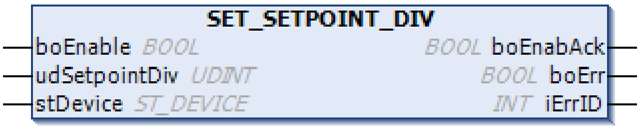
Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

13.6.2.2.9 SET_SETPOINT_DIV (FB)

This block sets ID32892 'Synchronous setpoint pulses divider' through the 'udSetpointDiv' variable.

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.
udSetpointDiv	UDINT	Set ID32892 'Synchronous setpoint pulses divider' (See document Parameter description ID32892 'Synchronous setpoint pulses divider' , Part no. 203704)

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error:
<table>
<tr><td>Value</td><td>Meaning</td></tr>
<tr><td>1</td><td>Device information not configured</td></tr>
<tr><td>2</td><td>No input / output variable assigned (copy pointer = 0)</td></tr>
<tr><td>3</td><td>Illegal device instance (symbolic
device name might have been assigned incorrectly)</td></tr>
</table>
</td></tr>
</table>

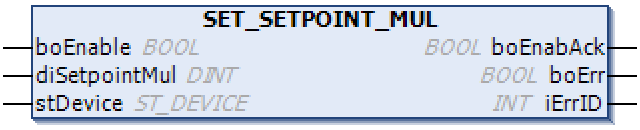
Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

13.6.2.2.10 SET_SETPOINT_MUL (FB)

This block sets ID32893 'Synchronous setpoint pulses multiplier' through the 'udSetpointMul' variable.

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.
udSetpointMul	UDINT	Set ID32893 'Synchronous setpoint pulses multiplier' (See document Parameter description ID32893 'Synchronous setpoint pulses multiplier' , Part no. 203704)

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error:
<table>
<tr><td>Value</td><td>Meaning</td></tr>
<tr><td>1</td><td>Device information not configured</td></tr>
<tr><td>2</td><td>No input / output variable assigned (copy pointer = 0)</td></tr>
<tr><td>3</td><td>Illegal device instance (symbolic
device name might have been assigned incorrectly)</td></tr>
</table>
</td></tr>
</table>

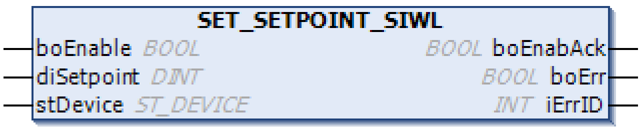
Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

13.6.2.2.11 SET_SETPOINT_SIWL (FB)

This block sets the soft pulse forwarding setpoint (ID33911 'SIWL setpoint') through the 'diSetPoint' variable.

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.
diSetpoint	DINT	Sets Setpoint soft pulse forwarding (ID33911 'SIWL setpoint') (See document Parameter description ID33911 'SIWL setpoint' , Part no. 203704)

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error:
<table>
<tr><td>Value</td><td>Meaning</td></tr>
<tr><td>1</td><td>Device information not configured</td></tr>
<tr><td>2</td><td>No input / output variable assigned (copy pointer = 0)</td></tr>
<tr><td>3</td><td>Illegal device instance (symbolic
device name might have been assigned incorrectly)</td></tr>
</table>
</td></tr>
</table>

Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

13.7 Support (support functions)

The blocks are essentially intended for internal system development. They are used, for example, in superior quality blocks and made available to users in a format customized to meet the requirements of their applications.

AmkCanCommunikation_ACC

DO_AFP	Executes the AFP (AMK fieldbus protocol)
DO_AFP_ONCE	Executes a one-off complete cycle of the AFP.

CamVarAccess

Asynchronous

GET_COMVAR_ASYNC_DINT	Reads an asynchronous 4-byte communication input variable
GET_COMVAR_ASYNC_INT	Reads an asynchronous 2-byte communication input variable
SET_COMVAR_ASYNC_DINT	Writes an asynchronous 4-byte communication output variable
SET_COMVAR_ASYNC_INT	Writes an asynchronous 2-byte communication output variable

Synchronous

GET_COMVAR_SYNC_DINT	Reads a synchronous 4-byte communication input variable
GET_COMVAR_SYNC_INT	Reads a synchronous 2-byte communication input variable
SET_COMVAR_SYNC_DINT	Writes a synchronous 4-byte communication output variable
SET_COMVAR_SYNC_INT	Writes a synchronous 2-byte communication output variable

Sercos

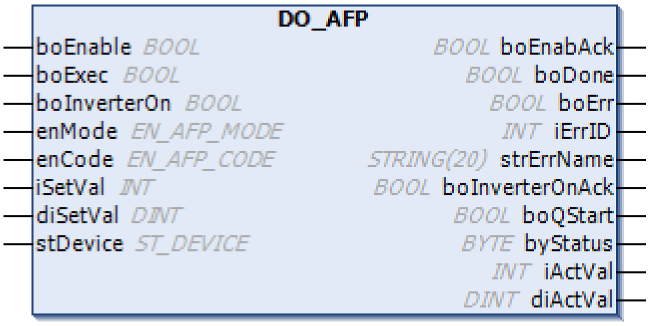
CMD_BY_ID	Executes ID-based commanding
CMD_START_STOP_BY_ID	Executes ID-based start / stop commanding
DO_CMD	EC-specific commanding
STATE_BY_ID	Checks the status of ID-based commanding

13.7.1 AmkCanCommunication_ACC

13.7.1.1 DO_AFP (FB)

The 'DO_AFP' block executes the AFP (AMK fieldbus protocol) via the ACC bus (ACC = AMK CAN communication).

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.
boExec	BOOL	Function execution: With a positive edge, the execution of the block starts. As long as 'boExec' = TRUE, the block is processed by the PLC. In the state 'boExec' = FALSE execution of the block is ended.
boInverterOn	BOOL	Inverter On (RF = controller enable)
enMode	ENUM	EN_AFP_MODE Selection mode AFP
enCode	ENUM	EN_AFP_CODE Select: Command code AFP
iSetVal	INT	16-bit Setpoint
diSetVal	DINT	32-bit Setpoint

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boDone</td><td>BOOL</td><td>Response that the function block has been completely executed.</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table></td></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table></td></tr>
<tr><td>strErrName</td><td>STRING</td><td>Block name of the block causing the error</td></tr>
<tr><td>boInverterOnAck</td><td>BOOL</td><td>Inverter On Acknowledge (QRF = acknowledgement controller enable)</td></tr>
<tr><td>boQStart</td><td>BOOL</td><td>With a positive edge, the execution of the block starts.</td></tr>
<tr><td>byStatus</td><td>BYTE</td><td>Drive parameters (ID34029 'AFP status bits')
(See document Parameter description ID34029 'AFP status bits' , Part no. 26249)</td></tr>
</table>

Name	Type	Description
iActVal	INT	16-bit Actual value
diActVal	DINT	32-bit Actual value

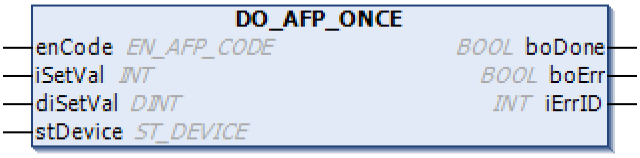
Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

13.7.1.2 DO_AFP_ONCE (FB)

The 'DO_AFP_ONCE' block executes a one-off complete cycle of the AFP.

User interface



Input variables

Name	Type	Description
enCode	ENUM	EN_AFP_CODE Select: Command code AFP
iSetVal	INT	16-bit Setpoint
diSetVal	DINT	32-bit Setpoint

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boDone</td><td>BOOL</td><td>Response that the function block has been completely executed.</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state<table><tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr><tr><td>TRUE</td><td>Error</td></tr></table></td></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output<table><tr><td colspan="2">iErrID = 0</td><td>No error</td></tr><tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr><tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr></table></td></tr>
</table>

Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

13.7.2 ComVarAccess

The instance of the bus system must be identified. 'stDevice' is to be connected to the device assigned to the communication variables by means of the 'symbolic device identifier'.

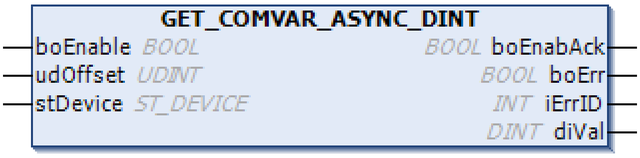
From the installation of CODESYS V3.5 SP10 Patch 4 (AIPEX PRO V3.04), the local instance (interface) of the controller is also allowed.

13.7.2.1 Asynchronous

13.7.2.1.1 GET_COMVAR_ASYNC_DINT (FB)

Reads an asynchronous 4-byte communication input variable and transfers as DINT type through 'diVal'.

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.
udOffset	UDINT	Reads an asynchronous 4-byte Communication variables

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table></td></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error:
<table>
<tr><td>Value</td><td>Meaning</td></tr>
<tr><td>1</td><td>Device information not configured</td></tr>
<tr><td>2</td><td>No input / output variable assigned (copy pointer = 0)</td></tr>
<tr><td>3</td><td>Illegal device instance (symbolic
device name might have been assigned incorrectly)</td></tr>
</table></td></tr>
<tr><td>diVal</td><td>DINT</td><td>Output value</td></tr>
</table>

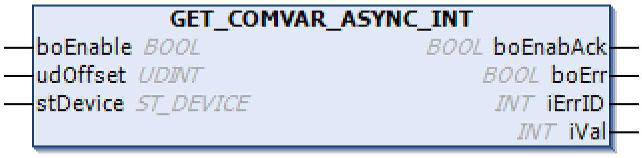
Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

13.7.2.1.2 GET_COMVAR_ASYNC_INT (FB)

Reads an asynchronous 2-byte communication input variable and transfers as INT type through 'iVal'.

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.
udOffset	UDINT	Reads an asynchronous 2-byte Communication variables

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table></td></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error:
<table>
<tr><th>Value</th><th>Meaning</th></tr>
<tr><td>1</td><td>Device information not configured</td></tr>
<tr><td>2</td><td>No input / output variable assigned (copy pointer = 0)</td></tr>
<tr><td>3</td><td>Illegal device instance (symbolic
device name might have been assigned incorrectly)</td></tr>
</table></td></tr>
<tr><td>iVal</td><td>INT</td><td>Output value</td></tr>
</table>

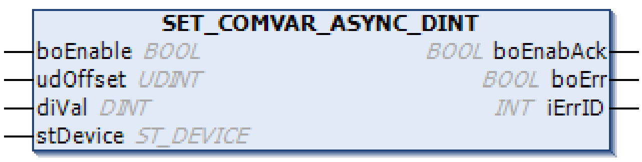
Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

13.7.2.1.3 SET_COMVAR_ASYNC_DINT (FB)

Writes an asynchronous 4-byte communication output variable as DINT through 'diVal'.

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.
udOffset	UDINT	Writes an asynchronous 4-byte Communication variables
diVal	DINT	Input value

Output variables

Name	Type	Description
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled
boErr	BOOL	The function block is in an error state FALSE: No error (permitted commanding or warning) TRUE: Error
iErrID	INT	Error identity number: Diagnostic number is output iErrID = 0: No error iErrID ≠ 0, boErr = TRUE: Error iErrID ≠ 0, boErr = FALSE: Warning Error: Value / Meaning 1: Device information not configured 2: No input / output variable assigned (copy pointer = 0) 3: Illegal device instance (symbolic device name might have been assigned incorrectly)

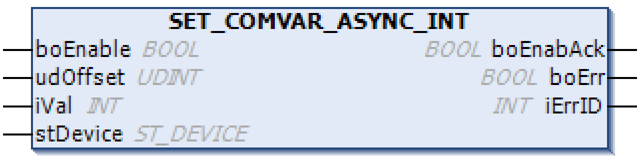
Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

13.7.2.1.4 SET_COMVAR_ASYNC_INT (FB)

Writes an asynchronous 2-byte communication output variable as INT through 'iVal'.

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.
udOffset	UDINT	Writes an asynchronous 2-byte Communication variables
iVal	INT	Input value

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table></td></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error:
<table>
<tr><td>Value</td><td>Meaning</td></tr>
<tr><td>1</td><td>Device information not configured</td></tr>
<tr><td>2</td><td>No input / output variable assigned (copy pointer = 0)</td></tr>
<tr><td>3</td><td>Illegal device instance (symbolic
device name might have been assigned incorrectly)</td></tr>
</table></td></tr>
</table>

Input and output variables

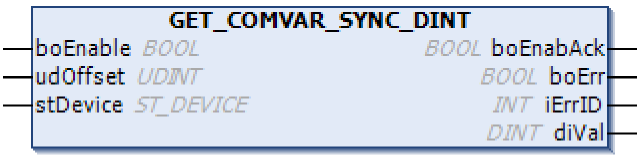
Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

13.7.2.2 Synchronous

13.7.2.2.1 GET_COMVAR_SYNC_DINT (FB)

Reads a synchronous 4-byte communication input variable and transfers as DINT type through 'diVal'.

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.
udOffset	UDINT	Reads a synchronous 4-byte Communication variables

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table></td></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error:
<table>
<tr><td>Value</td><td>Meaning</td></tr>
<tr><td>1</td><td>Device information not configured</td></tr>
<tr><td>2</td><td>No input / output variable assigned (copy pointer = 0)</td></tr>
<tr><td>3</td><td>Illegal device instance (symbolic
device name might have been assigned incorrectly)</td></tr>
</table></td></tr>
<tr><td>diVal</td><td>DINT</td><td>Output value</td></tr>
</table>

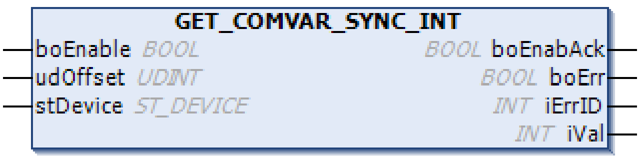
Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

13.7.2.2.2 GET_COMVAR_SYNC_INT (FB)

Reads a synchronous 2-byte communication input variable and transfers as INT type through 'iVal'.

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.

Name	Type	Description
udOffset	UDINT	Reads a synchronous 2-byte Communication variables

Output variables

Name	Type	Description
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled
boErr	BOOL	The function block is in an error state FALSE: No error (permitted commanding or warning) TRUE: Error
iErrID	INT	Error identity number: Diagnostic number is output iErrID = 0: No error iErrID ≠ 0, boErr = TRUE: Error iErrID ≠ 0, boErr = FALSE: Warning Error: Value – Meaning 1 – Device information not configured 2 – No input / output variable assigned (copy pointer = 0) 3 – Illegal device instance (symbolic device name might have been assigned incorrectly)
iVal	INT	Output value

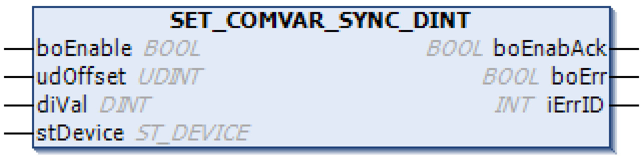
Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

13.7.2.2.3 SET_COMVAR_SYNC_DINT (FB)

Writes a synchronous 4-byte communication output variable as DINT through 'diVal'.

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.
udOffset	UDINT	Writes a synchronous 4-byte Communication variables
diVal	DINT	Input value

Output variables

Name	Type	Description
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled
boErr	BOOL	The function block is in an error state FALSE: No error (permitted commanding or warning) TRUE: Error
iErrID	INT	Error identity number: Diagnostic number is output iErrID = 0: No error iErrID ≠ 0, boErr = TRUE: Error iErrID ≠ 0, boErr = FALSE: Warning Error: Value – Meaning 1 – Device information not configured 2 – No input / output variable assigned (copy pointer = 0) 3 – Illegal device instance (symbolic device name might have been assigned incorrectly)

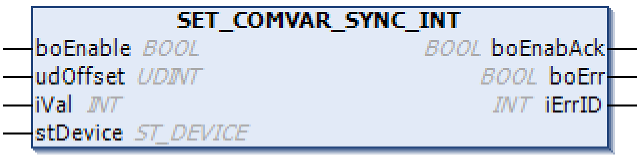
Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

13.7.2.2.4 SET_COMVAR_SYNC_INT (FB)

Writes a synchronous 2-byte communication output variable as INT through 'iVal'.

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.
udOffset	UDINT	Writes a synchronous 2-byte Communication variables
iVal	INT	Input value

Output variables

Name	Type	Description
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table></td></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error:
<table>
<tr><td>Value</td><td>Meaning</td></tr>
<tr><td>1</td><td>Device information not configured</td></tr>
<tr><td>2</td><td>No input / output variable assigned (copy pointer = 0)</td></tr>
<tr><td>3</td><td>Illegal device instance (symbolic
device name might have been assigned incorrectly)</td></tr>
</table></td></tr>
</table>

Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

13.7.3 Sercos

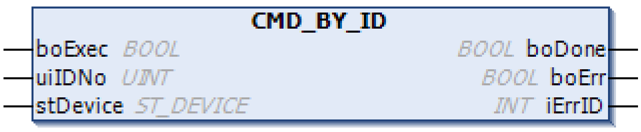
13.7.3.1 CMD_BY_ID (FB)

This block executes ID-based commanding via the EtherCAT bus. In accordance with to the SERCOS standard, a complete commanding cycle is executed referencing the ID number specified in the 'uiIDNo' input variable.

For a commandable ID (e.g. ID148 'Drive homing cycle command'):

- a value of 3 is written,
- a check is made to ascertain if 3 can be read back (no error, otherwise error code 15),
- a value of 0 is written,
- a check is made to ascertain if 0 can be read back (end)

User interface



Input variables

Name	Type	Description
boExec	BOOL	Function execution: With a positive edge, the execution of the block starts. As long as 'boExec' = TRUE, the block is processed by the PLC. In the state 'boExec' = FALSE execution of the block is ended.
uiIDNo	UINT	Parameter number (ID)

Output variables

Name	Type	Description
boDone	BOOL	Response that the function block has been completely executed.

Name	Type	Description
boErr	BOOL	The function block is in an error state FALSE | No error (permitted commanding or warning) TRUE | Error
iErrID	INT	Error identity number: Diagnostic number is output iErrID = 0 | No error iErrID ≠ 0 | boErr = TRUE | Error iErrID ≠ 0 | boErr = FALSE | Warning Error Value | Meaning 15 | Commanding error

Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

13.7.3.2 CMD_START_STOP_BY_ID (FB)

This block executes ID-based start / stop commanding via the EtherCAT bus.

User interface

Input variables

Name	Type	Description
boExec	BOOL	Function execution: With a positive edge, the execution of the block starts. As long as 'boExec' = TRUE, the block is processed by the PLC. In the state 'boExec' = FALSE execution of the block is ended.
boStartNotStop	BOOL	Commanding ID number FALSE | Stop (code=0) TRUE | Start (code=3)
uiIDNo	UINT	Parameter number (ID)

Output variables

Name	Type	Description
boDone	BOOL	Acknowledgement: Function block is initialised and enabled
boErr	BOOL	The function block is in an error state FALSE | No error (permitted commanding or warning) TRUE | Error

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td>iErrID = 0</td><td></td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error
<table>
<tr><td>Value</td><td>Meaning</td></tr>
<tr><td>15</td><td>Commanding error</td></tr>
</table>
</td></tr>
</table>

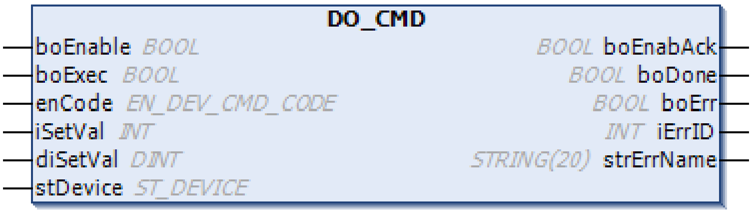
Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

13.7.3.3 DO_CMD (FB)

This block is used for commanding that is specific to EtherCAT (based on control / status or ID).

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.
boExec	BOOL	Function execution: With a positive edge, the execution of the block starts. As long as 'boExec' = TRUE, the block is processed by the PLC. In the state 'boExec' = FALSE execution of the block is ended.

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>enCode</td><td>ENUM</td><td>EN_DEV_CMD_CODE
Select: Command code
<table><tr><td>Default</td><td>TAB_CALC_OP</td></tr></table>
<table>
<tr><th>Range</th><th>Meaning</th></tr>
<tr><td>DEV_CMD_MODE0</td><td>MainMode0</td></tr>
<tr><td>DEV_CMD_POS</td><td>Position control</td></tr>
<tr><td>DEV_CMD_SPEED</td><td>Speed control</td></tr>
<tr><td>DEV_CMD_TORQUE</td><td>Torque control</td></tr>
<tr><td>DEV_CMD_HOME</td><td>Homing cycle</td></tr>
<tr><td>DEV_CMD_STOP</td><td>Stop (speed control, n=0)</td></tr>
<tr><td>DEV_CMD_MSTART</td><td>Start touch probe</td></tr>
<tr><td>DEV_CMD_MSTOP</td><td>Stop touch probe</td></tr>
<tr><td>DEV_CMD_HOME_TMP_PAR</td><td>Homing cycle (block setting)</td></tr>
</table></td></tr>
<tr><td>iSetVal</td><td>INT</td><td>16-bit Setpoint (depends on command code)
<table>
<tr><th>Range</th><th>Meaning</th></tr>
<tr><td>DEV_CMD_HOME_TMP_PAR</td><td>Approach direction according to ID147 'Homing parameter'
Cam according to ID32926 'AMK homing cycle parameter'
(See document Parameter description , Part no. 26249)</td></tr>
</table></td></tr>
<tr><td>diSetVal</td><td>DINT</td><td>32-bit Setpoint (depends on command code)
<table>
<tr><th>Range</th><th>Meaning</th></tr>
<tr><td>DEV_CMD_HOME_TMP_PAR</td><td>Offset according to ID153 'Spindle angle position'
(See document Parameter description , Part no. 26249)</td></tr>
</table></td></tr>
</table>

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boDone</td><td>BOOL</td><td>Response that the function block has been completely executed.</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table></td></tr>
</table>

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Warning
if 'strErrName' = 'DO_CMD'
<table>
<tr><th>Value</th><th>Meaning</th></tr>
<tr><td>2</td><td>Commanding without inverter on acknowledge (QRF)</td></tr>
</table>
Error
if 'strErrName' = 'DO_CMD'
<table>
<tr><th>Value</th><th>Meaning</th></tr>
<tr><td>1</td><td>Illegal command code</td></tr>
</table>
if 'strErrName' = 'DEV_SET_CTRL' or 'DEV_GET_STAT'
<table>
<tr><th>Value</th><th>Meaning</th></tr>
<tr><td>1</td><td>Device information not configured</td></tr>
<tr><td>2</td><td>No input / output variable assigned (copy pointer = 0)</td></tr>
<tr><td>3</td><td>Illegal device instance (symbolic
device name might have been assigned incorrectly)</td></tr>
</table>
</td></tr>
<tr><td>strErrName</td><td>STRING</td><td>Block name of the block causing the error
<table>
<tr><th>Range</th><th>Meaning</th></tr>
<tr><td>DEV_SET_CTRL</td><td>Write 'control word'</td></tr>
<tr><td>DEV_GET_STAT</td><td>Read 'status word'</td></tr>
<tr><td>DO_CMD</td><td></td></tr>
</table>
</td></tr>
</table>

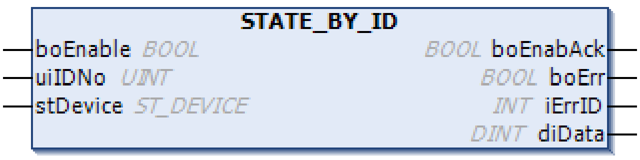
Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

13.7.3.4 STATE_BY_ID (FB)

This block checks the status of ID-based commanding via the EtherCAT bus.

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed. If 'boEnable' = TRUE, the ID is read continuously

Name	Type	Description
uiIDNo	UINT	Parameter number (ID)

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Response that the function block has been completely executed.</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table></td></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table></td></tr>
<tr><td>diData</td><td>DINT</td><td>Parameter value
<table>
<tr><th>Range</th><th>Meaning</th></tr>
<tr><td>0</td><td>Inactive</td></tr>
<tr><td>3</td><td>Active</td></tr>
<tr><td>7</td><td>Idle</td></tr>
<tr><td>15</td><td>Error</td></tr>
</table></td></tr>
</table>

Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

14 AmkEasyDev - Simplified device interface

AmkEasyDev is an internal library which provides a simple functional interface for access to general device information along with information about the drive controller. These blocks are essentially based on blocks from the AmkDevAccess library and, therefore, support automatic bus configuration specific to AMK.

The library is divided into:

DeviceAccessAsync	Asynchronous device access blocks
DeviceAccessSync	Synchronous device access blocks
Support	Support blocks

14.1 Block dependency of device information configured automatically

The following tables list the assignments between bus access blocks and the associated necessary device information (ENUM values: EN_DEV_INFO type from the AmkBase library)

Abstraction to 'technological device information' means that the values can be mapped independently of devices and bus systems. This is done by AIPEX PRO during the automatic bus configuration process.

14.1.1 Blocks in the AmkEasyDev library

The following table lists the blocks in the AmkDevAccess library on which the blocks in the AmkEasyDev library are based. The 'Device information for blocks in the AmkDevAccess library thus establishes the connection between the linked device information from the blocks (ENUM values: EN_DEV_INFO type from the AmkBase library).

Base blocks in the AmkEasyDev library

Block name (folder name)	AmkDevAccess block
Blocks that are not specific to devices or bus systems	
DeviceAccessAsync	
EASY_DEVICE	GET_STAT_SYSTEM_READY_X_SBM
	GET_STAT_DC_BUSENABLE_ACK_x_QUE
	GET_STAT_ERR_RESET_ACK_x_QFL
	SET_CTRL_DC_BUSENABLE_x_UE
	SET_CTRL_ERR_RESET_x_FL
EASY_HOMING	SET_SETPOINT_SPEED
	DO_CMD_ONCE
-Command	
GET_STATUS_BITS	GET_STAT_SYSTEM_READY_x_SBM
	GET_STAT_DC_BUSENABLE_ACK_x_QUE
	GET_STAT_INVERTER_ON_ACK_x_QRF
	GET_STAT_ERR_RESET_ACK_x_QFL
SET_CONTROL_BITS	SET_CTRL_DC_BUSENABLE_x_UE
	SET_CTRL_ERR_RESET_x_FL
	SET_CTRL_INVERTER_ON_x_RF
HANDLE_FL_QFL	SET_CTRL_ERR_RESET_x_FL
	GET_STAT_ERR_RESET_ACK_x_QFL
HANDLE_RF_QRF	SET_CTRL_INVERTER_ON_x_RF
	GET_STAT_INVERTER_ON_ACK_x_QRF
HANDLE_UE_QUE	SET_CTRL_DC_BUSENABLE_x_UE
	GET_STAT_DC_BUSENABLE_ACK_x_QUE
DeviceAccessSync	

Block name (folder name)	AmkDevAccess block
EASY_CONTROL	GET_STAT_INVERTER_ON_ACK_x_QRF
	SET_CTRL_INVERTER_ON_x_RF
	GET_ACTUAL_POSITION
	GET_ACTUAL_SPEED
	GET_ACTUAL_TORQUE
	SET_SETPOINT_POSITION
	SET_SETPOINT_SPEED
	SET_SETPOINT_TORQUE
	DO_CMD_ONCE
EASY_POSITIONING	SET_SETPOINT_POSITION
Support -AmkCanCom_ACC	
EASY_DRIVE	GET_STATUS_BITS
	SET_CONTROL_BITS
	GET_ACTUAL_POSITION
	DO_AFP

14.2 DeviceAccessAsync

The blocks in the DeviceAccessAsync folder comprise the following blocks with asynchronous device access:

EASY_DEVICE
EASY_HOMING
EASY_PROBE

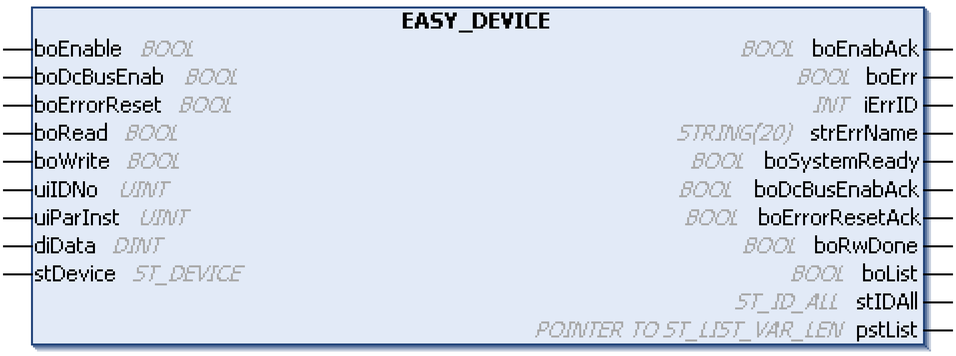
14.2.1 EASY_DEVICE (FB)

The 'EASY_DEVICE' block facilitates access to a device.

The following options are supported:

- Set "DC bus enable" (UE).
- Set "error reset" (FL).
- Get "system ready" (SBM).
- Get "DC bus enable acknowledge" (QUE).
- Get "error reset acknowledge" (QFL).
- Read IDs
- Write IDs

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.
boDcBusEnab	BOOL	DC-Bus Enable (UE = converter on)
boErrorReset	BOOL	Error Reset (FL = clear error)
boRead	BOOL	Read parameter / ID (with all elements)
boWrite	BOOL	Write parameter / ID 'boList' = TRUE: → WRITE_ID_DINT 'boList' = FALSE → WRITE_ID_LIST
uiIDNo	UINT	Parameter number (ID)
uiParInst	UINT	Parameter set number or instance number
diData	DINT	Parameter value Data for write ID (if 'boList' =FALSE)

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table></td></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table></td></tr>
<tr><td>strErrName</td><td>STRING (20)</td><td>Block name of the module generating the error
<table>
<tr><th>Range</th><th>Meaning</th></tr>
<tr><td>'DEV_SYR__SBM'</td><td>GET_STAT_SYSTEM_READY_x_SBM</td></tr>
<tr><td>'DEV_BEA__QUE'</td><td>GET_STAT_DC_BUSENABLE_ACK_x_QUE</td></tr>
<tr><td>'DEV_ERA__QFL'</td><td>GET_STAT_ERR_RESET_ACK_x_QFL</td></tr>
<tr><td>'DEV_BE__UE'</td><td>SET_CTRL_DC_BUSENABLE_x_UE</td></tr>
<tr><td>'DEV_ER__FL'</td><td>SET_CTRL_ERR_RESET_x_FL</td></tr>
<tr><td>'HANDLE_IDS'</td><td>HANDLE_IDS</td></tr>
</table></td></tr>
<tr><td>boSystemReady</td><td>BOOL</td><td>System ready (SBM = system ready message)</td></tr>
<tr><td>boDcBusEnabAck</td><td>BOOL</td><td>DC-Bus Enable Acknowledge (QUE = acknowledgement DC converter ON)</td></tr>
<tr><td>boErrorResetAck</td><td>BOOL</td><td>Error Reset Acknowledge (QFL = acknowledgement clear error)</td></tr>
<tr><td>boRwDone</td><td>BOOL</td><td>Handshake ID read/write completed</td></tr>
</table>

Name	Type	Description
boList	BOOL	Identifier for a list parameter FALSE: The data to be read is in 'stIDAll.diData' TRUE: List parameter: The list to be read is transferred to the list structure referenced by 'pbyData'
stIDAll	STRUCT	ST_ID_ALL Parameter information Accommodates the element information ID information structure with: data, min. value, max. value, attribute, unit, name
pstList	POINTER	POINTER TO ST_LIST_VAR_LEN Pointer to the internal ID list

Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

The 'EASY_DEVICE' block combines the following basic functions.
Based on the AmkDevAccess library:

- SET_CTRL_DC_BUSENABLE_x_UE
- SET_CTRL_ERR_RESET_x_FL
- GET_STAT_SYSTEM_READY_x_SBM
- GET_STAT_DC_BUSENABLE_ACK_x_QUE
- GET_STAT_ERR_RESET_ACK_x_QFL

(See document Software description AmkDevAccess Bibliothek , Part no. 109903)

Based on the AmkSystem library:

- READ_ID_LIST_ALL
- WRITE_ID_DINT
- READ_ID_DINT
- WRITE_ID_LIST
- WRITE_ID_DINT_TMP
- READ_ID_DINT_TMP

(See document Software description AmkSystem library , Part no. 205004)

For logical reasons, this block should not be called in the event-driven PGT task (PGT = Peripherie Grund Takt (peripheral basic cycle)) FPLC_TASK but in a cyclic or free-running task (PLC_TASK, for example).

Integrating the 'SHOW_LIST' support block enables list IDs to be displayed and edited with the 'ViEasyDevice' visualization, for example.

IDs are read with 'boRead'=TRUE, using the 'READ_ID_LIST_ALL' block. IDs are selected based on 'uiIDNo' and 'uiParInst'. The corresponding device is identified by the 'stDevice' variable, which is initialized automatically. For a standard ID (not a list ID: 'boList'=FALSE), the complete ID information (data, min. value, max. value, attribute, unit, name) is made available in the 'stIDAll' structure. For a list ID: ('boList'= TRUE), the list value (data) is saved in a local 'ST_LIST_VAR_LEN' type structure. This structure can be read or edited with the 'ViEasyDevice' visualization. In programming terms, it can be accessed with the 'pstList' pointer. A distinction can be made between standard IDs and list IDs with 'boList' (see above).

Based on this variable, when writing an ID 'boWrite'=TRUE, either the value is taken from 'diData' (not a list ID: 'boList'=FALSE') or the information is written back to the 'ST_LIST_4096' type structure ('boList'=TRUE).

In the context of ID access based on the 'EASY_DEVICE' block, read access must always be carried out before commencing a write operation.

During a read operation, the ID type for the subsequent write operation is defined by reading the 'boList' variable:

'boList'=FALSE: → simple data type

'boList'=TRUE → list type

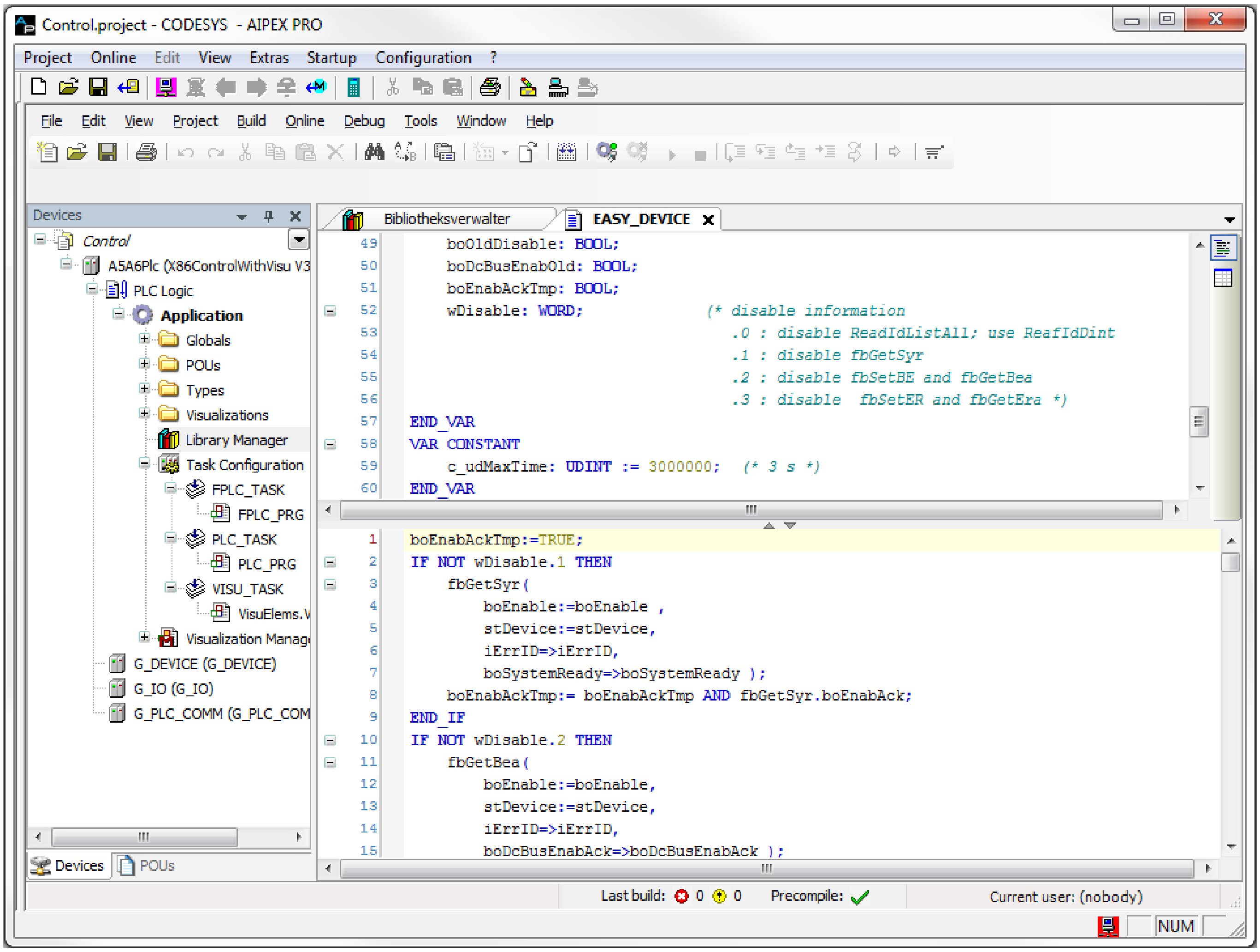
The 'SelectWriteAuto', 'SelectWriteSimple', and 'SelectWriteList' operations can be executed to influence the automatic definition of the write behavior described above.

Actions

Name	Description
SelectWriteAuto()	'boList' is determined automatically when reading the ID.
SelectWriteSimple()	'boList'=FALSE; a simple data type is always written, based on the 'WRITE_ID_DINT' block
SelectWriteList()	'boList'=TRUE; a list type is always written, based on the 'WRITE_ID_LIST' block. The list header, which consists of the current and the maximum list lengths, must be specified correctly.

The local variable 'wDisable' is used to disable the base function contained in the block. If the base function is disabled, it is not processed when the entire block is enabled. As a result, the necessary bus information does not have to be "mapped". (See figure)

Figure: wDisable information from EASY_DEVICE



The individual bits of the 'wDisable' variable have the following meanings:

wDisable.0:	disable ReadIdListAll; use ReafIdDint	(action: Disable ReadIdListAll) 1)
wDisable.1:	disable fbGetSyr	(action: DisableSysRdy)
wDisable.2:	disable fbSetBE and fbGetBea	(action: DisableBeBea)
wDisable.3:	disablefbSetER and fbGetEra	(action: DisableErEra)

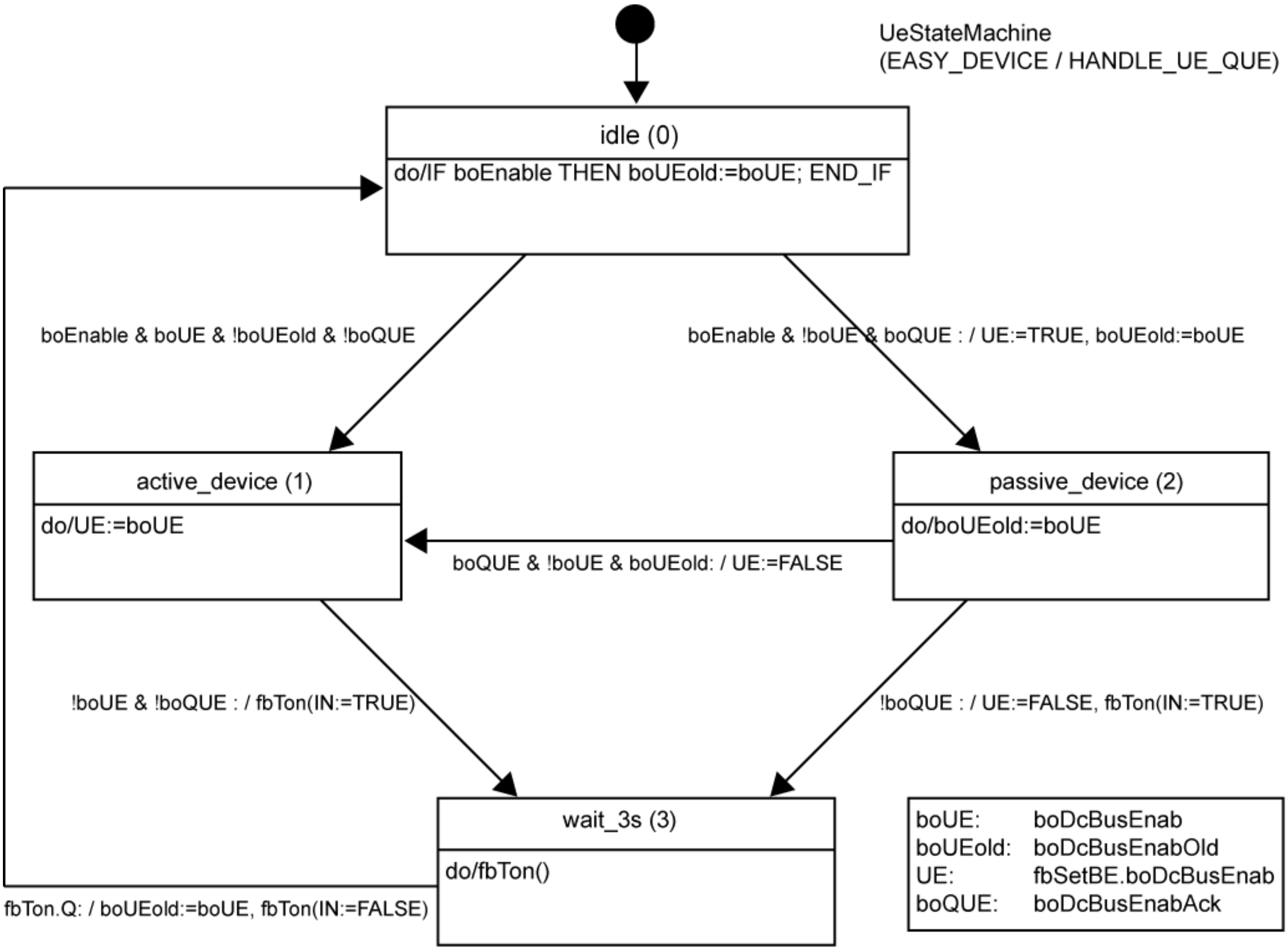
The setting of the corresponding bit(s) can either be organized as an initial value when the block instance is created or it can be set during the course of the assigned actions (see figure). The 'EnableAll' action clears all 'disable bits' ('wDisable':=0).

1) Setting 'wDisable.0'=TRUE deselects the rather more complex mechanism which involves using the 'READ_ID_LIST_ALL' block and the type distinction for write operations based on 'boList' (see above). Instead, only the ID value (the data) is read or written (with the 'READ_ID_DINT' or 'WRITE_ID_DINT' block).

The 'SelectAccessTmp' action writes or reads the temporary value of the ID.

Prerequisite: 'wDisable.0'=TRUE; or 'DisableReadIdListAll' action. (Instead of the 'READ_ID_DINT' or 'WRITE_ID_DINT' blocks – see above – only the 'READ_ID_DINT_TMP' or 'WRITE_ID_DINT_TMP' blocks are used here.)

The following behavior applies for the UE graphs in the context of the 'EASY_DEVICE' block (and 'HANDLE_UE_QUE' block):



The "active_device" state is relevant if

a) The KE (ID32795 = 5) is linked to the block instance directly as a device

b) The drive inverter linked to the block instance is also the bus master for the KE (ID32795 = 9)

The "passive_device" state is relevant if

a) The drive inverter linked to the block instance is not the bus master for the KE (in this case, UE is derived from QUE)

14.2.2 EASY_HOMING (FB)

The 'EASY_HOMING' block supports the homing of a drive and stopping of this movement (through transition to speed control with "speed=0") independent of the bus system.

The homing cycle mode is specified with 'enStdMode' and 'enAmkMode'.

The 'diSetPosition' input defines the setpoint position at the end of the homing cycle (see below: Input variables).

User interface

Input variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnable</td><td>BOOL</td><td>Enable signal: With a positive edge, the initialisation of the block starts.
As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC.
In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.</td></tr>
<tr><td>boExec</td><td>BOOL</td><td>Function execution: With a positive edge, the execution of the block starts.
As long as 'boExec' = TRUE, the block is processed by the PLC.
In the state 'boExec' = FALSE execution of the block is ended.</td></tr>
<tr><td>boStop</td><td>BOOL</td><td>With a positive edge, the execution of the block is aborted or completed.
Stop homing cycle, based on 'enStdMode' and 'enAmkMode'</td></tr>
<tr><td>enStdMode</td><td>ENUM</td><td>EN_HOME_MODE_STD
Standard homing cycle mode (operates according to ID147 'Homing parameter')
<table>
<tr><td>Default</td><td>POSTRANS_POSDIR</td></tr>
</table>
<table>
<tr><th>Range</th><th>Meaning</th></tr>
<tr><td>REM_PARA_ USED</td><td>Homing cycle according to remanent parameters (ID147 'Homing parameter' / ID32926 'AMK homing cycle parameter')</td></tr>
<tr><td>POSTRANS_ POSDIR</td><td>Positive homing cycle direction / positive cam edge</td></tr>
<tr><td>POSTRANS_ NEGDIR</td><td>Positive homing cycle direction / negative cam edge</td></tr>
<tr><td>NEGTRANS_ POSDIR</td><td>Positive homing cycle direction / positive cam edge</td></tr>
<tr><td>NEGTRANS_ NEGDIR</td><td>Positive homing cycle direction / negative cam edge</td></tr>
</table></td></tr>
<tr><td>enAmkMode</td><td>ENUM</td><td>EN_HOME_MODE
Homing cycle mode specific to AMK (operates according to ID32926 'AMK homing cycle parameter')
<table>
<tr><td>Default</td><td>CAM_OFF</td></tr>
</table>
<table>
<tr><th>Range</th><th>Meaning</th></tr>
<tr><td>CAM_OFF</td><td>No cam evaluation</td></tr>
<tr><td>LIN_PULS_ZON</td><td>Linear axis – pulse cam with zero pulse evaluation</td></tr>
<tr><td>LIN_PULS_ZOFF</td><td>Linear axis – pulse cam without zero pulse evaluation</td></tr>
<tr><td>LIN_RANGE_ ZON</td><td>Linear axis – range cam with zero pulse evaluation</td></tr>
<tr><td>LIN_RANGE_ ZOFF</td><td>Linear axis – range cam without zero pulse evaluation</td></tr>
<tr><td>ROT_PULS_ZON</td><td>Rotary axis – pulse cam with zero pulse evaluation</td></tr>
<tr><td>ROT_PULS_ ZOFF</td><td>Rotary axis – pulse cam without zero pulse evaluation</td></tr>
<tr><td>ROT_RANGE_ ZON</td><td>Rotary axis – range cam with zero pulse evaluation</td></tr>
<tr><td>ROT_RANGE_ ZOFF</td><td>Rotary axis – range cam without zero pulse evaluation</td></tr>
</table></td></tr>
<tr><td>diSetPosition</td><td>DINT</td><td>Specification of the position setpoint (position setpoint system) [increments] (principle of operation according to ID153 'Spindle angle position')
<table>
<tr><td>Unit</td><td>inc</td></tr>
</table></td></tr>
</table>

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table></td></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table></td></tr>
<tr><td>strErrName</td><td>STRING (20)</td><td>Block name of the module generating the error
<table>
<tr><td>Range</td><td>Meaning</td></tr>
<tr><td>'SET_SETVEL'</td><td>SET_SETPOINT_SPEED</td></tr>
</table></td></tr>
<tr><td>boDone</td><td>BOOL</td><td>Response that the function block has been completely executed.</td></tr>
</table>

Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

The 'EASY_HOMING' block combines the following basic functions

Based on the AmkDevAccess library:

- SET_SETPOINT_SPEED
- DO_CMD_ONCE

(See document Software description AmkDevAccess Bibliothek , Part no. 109903)

For logical reasons, this block should not be called in the event-driven PGT task (PGT = Peripherie Grund Takt (peripheral basic cycle)) FPLC_TASK but in a cyclic or "free-running" task (PLC_TASK, for example).

14.2.3 EASY_PROBE (FB)

The 'EASY_PROBE' blocks facilitate easy use of the touch probe functions supported by the drives (or drive controller assemblies KW-R03, KW-R05, etc.).

The touch probe function is activated with the 'boEnable' enable signal. The enable is acknowledged with 'boEnabAck'. Since the touch probe function and homing are mutually exclusive, homing (e.g. with 'EASY_CONTROL' or 'EASY_HOMING') can only take place if 'boEnable'=FALSE for 'EASY_PROBE'.

When the touch probe function is activated ('boEnabAck'=TRUE), a positive edge at 'boExec' will trigger the start of a measuring cycle (in other words, the measuring signal input (touch probe) selected with 'iNumber' is enabled). The current position value is then detected on the first active edge (set in ID169 'Probe control parameter'; see below) at the measuring signal input; this is signaled with 'boDone'=TRUE and the value detected is written to 'diData'.

The touch probe can only be specified upstream of each edge of 'boExec' (with 'iNumber'). However, the measuring cycle cannot be restarted until the current cycle is at an end ('boDone'=TRUE).

If two touch probes are to be evaluated in parallel, this can be achieved with two instances of 'EASY_PROBE'. However, as the touch probe function can only be activated for all touch probes together, both 'boEnable' signals of the instances must be coupled (e.g. fbMT2.boEnable:= fbMT1.boEnabAck).

The prerequisites for the touch probe function are:

BE3: ID32980 'Port 3 Bit 2' = 401 (touch probe 1)

BE2: ID32979 'Port 3 Bit 1' = 402 (touch probe 2; only available for KW-R05)

Moreover, the active edge on which the sample is to be taken must be set in ID169:

ID169, Bit0 = 1: positive edge at BE3 (touch probe 1)

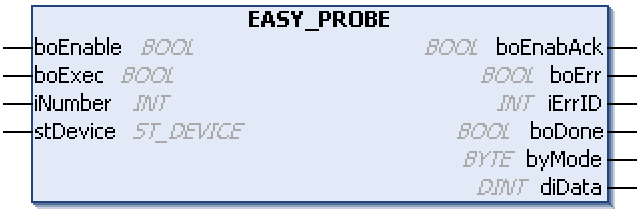
ID169, Bit1 = 1: negative edge at BE3 (touch probe 1)

ID169, Bit2 = 1: positive edge at BE2 (touch probe 2; only available for KW-R05)

ID169, Bit3 = 1: negative edge at BE2 (touch probe 2; only available for KW-R05)

Only one edge (positive or negative) may be selected per touch probe. If no edge is selected, an error message with 'iErrID=1' is output at the start of a measuring cycle (see below).

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed. The touch probe function and the homing cycle are mutually exclusive. The enable signal must be inactive during homing (e.g. set with 'EASY_HOMING' or 'EASY_CONTROL')
boExec	BOOL	Function execution: With a positive edge, the execution of the block starts. As long as 'boExec' = TRUE, the block is processed by the PLC. In the state 'boExec' = FALSE execution of the block is ended.
iNumber	INT	Number of the measuring signal input (touch probe). Note: One or a number of touch probes are supported based on the drive controller assemblies. Range: KW-R03: 1 Touch probe 1 (binary input BE3); KW-R05: 1..2 Touch probe 1/2 (binary input BE3/BE2) Default: 1

Output variables

Name	Type	Description
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled
boErr	BOOL	The function block is in an error state FALSE: No error (permitted commanding or warning) TRUE: Error

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error
<table>
<tr><th>Range</th><th>Meaning</th></tr>
<tr><td>0</td><td>No error</td></tr>
<tr><td>1</td><td>No compatible edge change in ID169 'Probe control parameter'</td></tr>
<tr><td>2</td><td>Error reading ID169 'Probe control parameter'</td></tr>
<tr><td>3</td><td>Illegal iNumber</td></tr>
<tr><td>4</td><td>Start of touch probe function failed</td></tr>
<tr><td>5</td><td>Error reading
ID409 'Probe 1 positive latch'...ID412 'Probe 2 negative latch'</td></tr>
<tr><td>6</td><td>Error reading ID130 'Probe value 1 positive edge'...ID133 'Probe value 2 negative edge'</td></tr>
<tr><td>7</td><td>Error writing ID405 'Probe 1 enable' / ID406 'Probe 2 enable'</td></tr>
</table>
</td></tr>
<tr><td>boDone</td><td>BOOL</td><td>Response that the function block has been completely executed.</td></tr>
<tr><td>byMode</td><td>BYTE</td><td>Mode according to ID169 'Probe control parameter'
Bit0: touch probe 1, positive edge
Bit1: touch probe 1, negative edge
Bit2: touch probe 2, positive edge
Bit3: touch probe 2, negative edge</td></tr>
<tr><td>diData</td><td>DINT</td><td>Parameter value
at the time the selected switching edge occurs (according to ID169 'Probe control parameter')</td></tr>
</table>

Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

The 'EASY_PROBE' block requires the following basic blocks:

Based on the AmkDevAccess library:

- DO_CMD_ONCE

(See document Software description AmkDevAccess Bibliothek , Part no. 109903)

For logical reasons, this block should not be called in the event-driven PGT task (PGT = Peripherie Grund Takt (peripheral basic cycle)) FPLC_TASK but in a cyclic or "free-running" task (PLC_TASK, for example).

14.2.4 Command

The following blocks are combined to organize access to device status and control information:

GET_STATUS_BITS

HANDLE_FL_QFL

HANDLE_RF_QRF

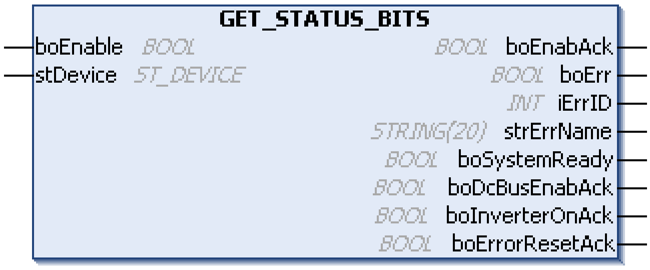
HANDLE_UE_QUE

SET_CONTROL_BITS

14.2.4.1 GET_STATUS_BITS (FB)

The 'GET_STATUS_BITS' block queries the following information:
"System ready" (SBM),
"DC bus enable acknowledge" (QUE),
"Inverter on acknowledge" (QRF),
"Error reset acknowledge" (QFL).

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table></td></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table></td></tr>
<tr><td>strErrName</td><td>STRING (20)</td><td>Block name of the module generating the error
<table>
<tr><th>Range</th><th>Meaning</th></tr>
<tr><td>'DEV_SYR__ SBM'</td><td>GET_STAT_SYSTEM_READY_x_SBM</td></tr>
<tr><td>'DEV_BEA__ QUE'</td><td>GET_STAT_DC_BUSENABLE_ACK_x_QUE</td></tr>
<tr><td>'DEV_IOA__QRF'</td><td>GET_STAT_INVERTER_ON_ACK_x_QRF</td></tr>
<tr><td>'DEV_ERA__ QFL'</td><td>GET_STAT_ERR_RESET_ACK_x_QFL</td></tr>
</table></td></tr>
<tr><td>boSystemReady</td><td>BOOL</td><td>System ready (SBM = system ready message)</td></tr>
<tr><td>boDcBusEnabAck</td><td>BOOL</td><td>DC-Bus Enable Acknowledge (QUE = acknowledgement DC converter ON)</td></tr>
<tr><td>boInverterOnAck</td><td>BOOL</td><td>Inverter On Acknowledge (QRF = acknowledgement controller enable)</td></tr>
<tr><td>boErrorResetAck</td><td>BOOL</td><td>Error Reset Acknowledge (QFL = acknowledgement clear error)</td></tr>
</table>

Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

14.2.4.2 HANDLE_FL_QFL (FB)

The 'Handle_FL_QFL' block is used to organize error resets (FL) with generation of acknowledgement information (QFL). In EtherCAT-based devices, ID99 'Diagnosis reset status class 1' is only read; as long as 'boErrorReset' is activated.

User interface

Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.
boErrorReset	BOOL	Error Reset (FL = clear error)

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table></td></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table></td></tr>
<tr><td>strErrName</td><td>STRING (20)</td><td>Block name of the module generating the error
<table>
<tr><th>Range</th><th>Meaning</th></tr>
<tr><td>'DEV_ER__FL'</td><td>SET_CTRL_ERR_RESET_x_FL</td></tr>
<tr><td>'DEV_ERA__QFL'</td><td>GET_STAT_ERR_RESET_ACK_x_QFL</td></tr>
</table></td></tr>
<tr><td>boErrorResetAck</td><td>BOOL</td><td>Error Reset Acknowledge (QFL = acknowledgement clear error)</td></tr>
</table>

Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

14.2.4.3 HANDLE_RF_QRF (FB)

The 'Handle_RF_QRF' block is used to organize the controller enable (RF) with generation of acknowledgement information (QRF).

User interface

Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.
boInverterOn	BOOL	Inverter On (RF = controller enable)

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table></td></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table></td></tr>
<tr><td>strErrName</td><td>BOOL</td><td>Block name of the module generating the error
<table>
<tr><th>Range</th><th>Meaning</th></tr>
<tr><td>'DEV_IO__RF'</td><td>SET_CTRL_INVERTER_ON_x_RF</td></tr>
<tr><td>'DEV_IOA__QRF'</td><td>GET_STAT_INVERTER_ON_ACK_x_QRF</td></tr>
</table></td></tr>
<tr><td>boInverterOnAck</td><td>BOOL</td><td>Inverter On Acknowledge (QRF = acknowledgement controller enable)</td></tr>
</table>

Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

14.2.4.4 HANDLE_UE_QUE (FB)

The 'Handle_UE_QUE' block is used to organize DC bus enable (UE) with generation of acknowledgement information (QUE).

User interface

Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.
boDcBusEnab	BOOL	DC-Bus Enable (UE = converter on)

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table></td></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table></td></tr>
<tr><td>strErrName</td><td>STRING (20)</td><td>Block name of the module generating the error
<table>
<tr><th>Range</th><th>Meaning</th></tr>
<tr><td>'DEV_BE__UE'</td><td>SET_CTRL_DC_BUSENABLE_x_UE</td></tr>
</table></td></tr>
<tr><td>boDcBusEnabAck</td><td>BOOL</td><td>DC-Bus Enable Acknowledge (QUE = acknowledgement DC converter ON)</td></tr>
</table>

Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

14.2.4.5 SET_CONTROL_BITS (FB)

The 'SET_CONTROL_BITS' block sets the following information:
"DC bus enable" (UE),
"Inverter on" (RF),
"Error reset" (FL).

User interface

Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.
boDcBusEnab	BOOL	DC-Bus Enable (UE = converter on)
boInverterOn	BOOL	Inverter On (RF = controller enable)
boErrorReset	BOOL	Error Reset (FL = clear error)

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table></td></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table></td></tr>
<tr><td>strErrName</td><td>STRING (20)</td><td>Block name of the module generating the error
<table>
<tr><th>Range</th><th>Meaning</th></tr>
<tr><td>'DEV_BE__UE'</td><td>SET_CTRL_DC_BUSENABLE_x_UE</td></tr>
<tr><td>'DEV_IO__RF'</td><td>SET_CTRL_INVERTER_ON_x_RF</td></tr>
<tr><td>'DEV_ER__FL'</td><td>SET_CTRL_ERR_RESET_x_FL</td></tr>
</table></td></tr>
</table>

Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

14.3 DeviceAccessSync (synchronous device access blocks)

The blocks in the DeviceAccessSync folder comprise the following blocks with synchronous device access:

EASY_CONTROL

EASY_POSITIONING

14.3.1 EASY_CONTROL (FB)

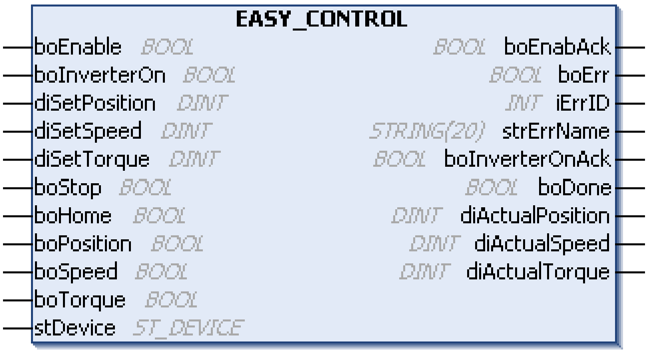
The 'EASY_CONTROL' block facilitates access to the drive controller independent of the bus system.

The following options are supported:

- Set "inverter on" (RF).
- Set "position setpoint".
- Set "speed setpoint".
- Set "torque setpoint".
- Get "inverter on acknowledge" (QRF).
- Get "actual position".
- Get "actual speed".
- Get "actual torque.

It also supports the execution of the "homing cycle" (according to the settings made in the relevant IDs "147 'Homing parameter', 32926 'AMK homing cycle parameter', 150 'Homing offset 1', ...") and stopping of this movement (through transition to speed control with "speed=0").

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.
boInverterOn	BOOL	Inverter On (RF = controller enable)
diSetPosition	DINT	Specification of the position setpoint (position setpoint system) [increments] Unit: inc
diSetSpeed	DINT	Set the velocity setpoint Unit: 1/10000 rpm
diSetTorque	DINT	Specification of the torque setpoint [0.1% Mn] Unit: 1/10% rated torque
boStop	BOOL	With a positive edge, the execution of the block is aborted or completed. (transition to speed control with speed = 0")
boHome	BOOL	Homing drive Enable signal: With a positive edge, the homing cycle function starts. As long as 'boHome' = TRUE, the homing drive is carried out. Use a negative edge 'boHome' = FALSE to cancel the current referencing or terminate the completed referencing.
boPosition	BOOL	Change to secondary operating mode 1 (position control), based on 'SET_ SETPOINT_POSITION' block and setting of 'diSetPosition' as position setpoint (See documentSoftware descriptionAmkDevAccess Bibliothek , Part no. 109903).
boSpeed	BOOL	Change to secondary operating mode 2 (speed control), based on 'SET_ SETPOINT_SPEED' block and setting of 'diSetSpeed' as speed setpoint (See documentSoftware descriptionAmkDevAccess Bibliothek , Part no. 109903)
boTorque	BOOL	Change to secondary operating mode 3 (torque control), based on 'SET_ SETPOINT_TORQUE' block and setting of 'diSetTorque' as torque setpoint (See documentSoftware descriptionAmkDevAccess Bibliothek , Part no. 109903)

The binary inputs 'boStop', 'boHome', 'boPosition', 'boSpeed', and 'boTorque' are prioritized in this order; logically, only one binary input (with the exception of 'boStop') should be active at any one time.

Output variables

Name	Type	Description
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled
boErr	BOOL	The function block is in an error state FALSE | No error (permitted commanding or warning) TRUE | Error
iErrID	INT	Error identity number: Diagnostic number is output iErrID = 0 | No error iErrID ≠ 0 | boErr = TRUE | Error iErrID ≠ 0 | boErr = FALSE | Warning
strErrName	STRING (20)	Block name of the module generating the error Range: 'GET_INVERTER_ON_ACK' | GET_STAT_INVERTER_ON_ACK_x_QRF 'GET_ACTPOS' | GET_ACTUAL_POSITION 'GET_ACTVEL' | GET_ACTUAL_SPEED 'GET_ACTTOR' | GET_ACTUAL_TORQUE 'SET_INVERTER_ON' | SET_CTRL_INVERTER_ON_x_RF 'SET_SETPOS' | SET_SETPOINT_POSITION 'SET_SETVEL' | SET_SETPOINT_SPEED 'SET_SETTOR' | SET_SETPOINT_TORQUE
boInverterOnAck	BOOL	Inverter On Acknowledge (QRF = acknowledgement controller enable)
boDone	BOOL	Response that the function block has been completely executed. Handshake, for the execution of 'boStop', 'boHome', 'boPosition', 'boSpeed', or 'boTorque'
diActualPosition	DINT	Actual position Unit | inc
diActualSpeed	DINT	Actual velocity Unit | 1/10000 rpm
diActualTorque	DINT	Actual torque Unit | 1/10% rated torque

Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

The 'EASY_CONTROL' block combines the following basic functions from the AmkDevAccess library.

- SET_CTRL_INVERTER_ON_x_RF
- SET_SETPOINT_POSITION
- SET_SETPOINT_SPEED
- SET_SETPOINT_TORQUE
- GET_STAT_INVERTER_ON_ACK_x_QRF
- GET_ACTUAL_POSITION
- GET_ACTUAL_SPEED
- GET_ACTUAL_TORQUE
- DO_CMD_ONCE (for the execution of the homing cycle)

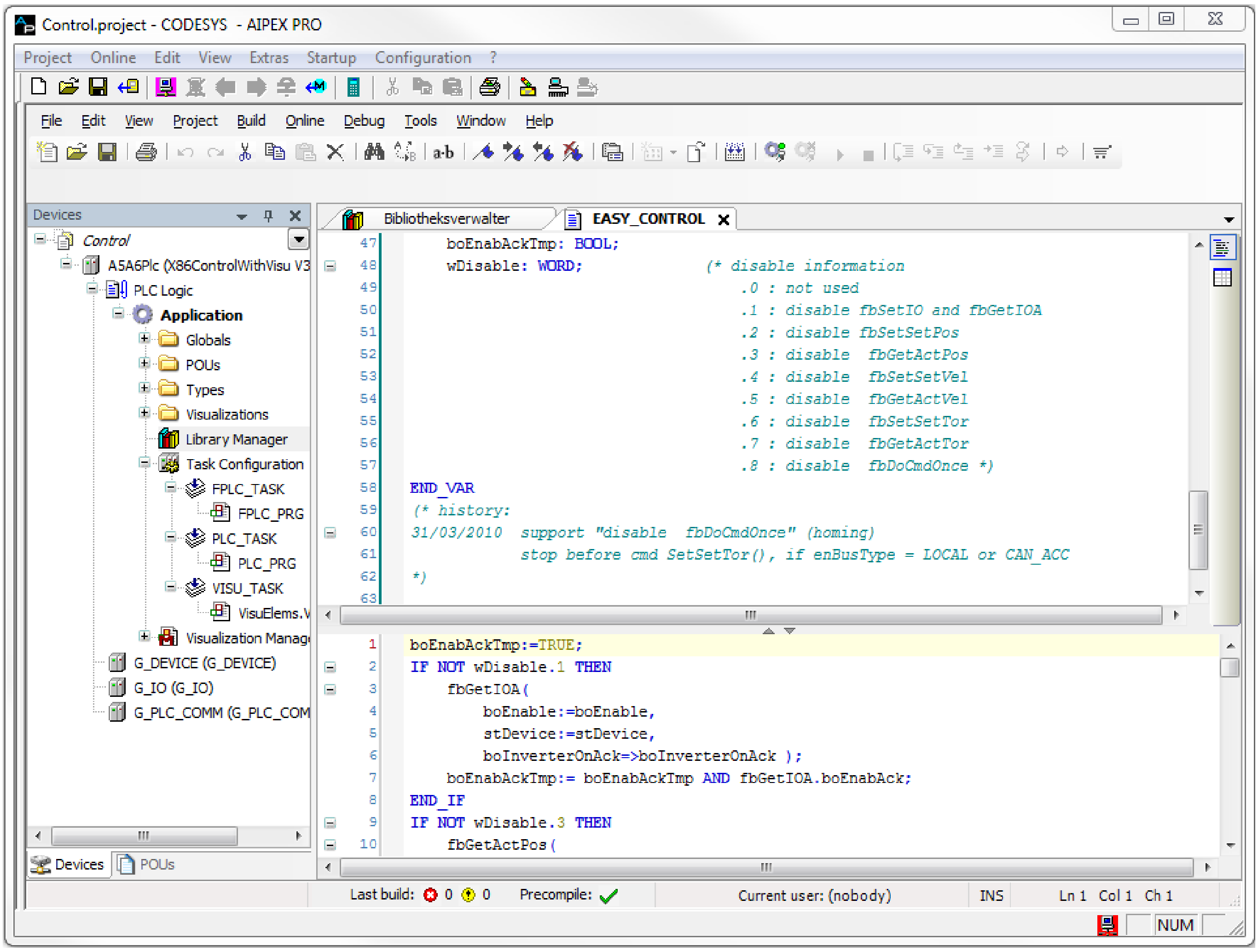
(See documentSoftware descriptionAmkDevAccess Bibliothek , Part no. 109903)

For logical reasons, this block is called in the event-driven PGT task (PGT = Peripherie Grund Takt (peripheral basic cycle)) FPLC_TASK, because synchronous command variables (in the PGT grid according to ID2 'SERCOS cycle time') are set for the drive controller or actual values are received via it.

Once the corresponding operating mode has been selected, the 'EASY_CONTROL' block copies essentially only synchronous values to / from the drive controller. A defined movement in position control requires, therefore, a synchronous command variable generator (such as the 'VGEN' or 'POS' block, for example; to supply 'diSetPosition'

(See documentSoftware description IEC 61131-3 function block libraries , Part no. 201977

The local variable 'wDisable' is used to disable the base function contained in the block (see figure). If the base function is disabled, it is not processed when the entire block is enabled. As a result, the necessary bus information does not have to be "mapped".



The individual bits of the wDisable variable have the following meanings:

wDisable.0:	not currently used	
wDisable.1:	disable fbSetIO and fbGetIOA	(action: DisableIoIoA)
wDisable.2:	disable fbSetSetPos	(action: DisableSetPos)
wDisable.3:	disable fbGetActPos	(action: DisableActPos)
wDisable.4:	disable fbSetSetVel	(action: DisableSetSpeed)
wDisable.5:	disable fbGetActVel	(action: DisableActSpeed)
wDisable.6:	disable fbSetSetTor	(action: DisableSetTorque)
wDisable.7:	disable fbGetActTor	(action: DisableActTorque)
wDisable.8:	disable fbDoCmdOnce	(action: DisableDoCmdOnce)

The setting of the corresponding bit(s) can either be organized as an initial value when the block instance is created or it can be set during the course of the assigned actions (see figure). The "EnableAll" action clears all "disable bits" ('wDisable':=0).

14.3.2 EASY_POSITIONING (FB)

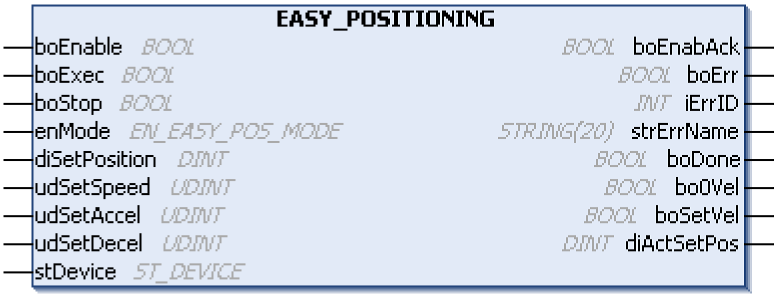
The 'EASY_POSITIONING' supports relative and absolute positioning independent of the bus system; with setting of 'diSetPosition', 'udSetSpeed', 'udSetAccel', and 'udSetDecel' according to the behavior of the 'POS'-positioning block.

(See documentSoftware descriptionAmkBase Bibliothek, Part no. 204986).

It also supports stopping of the movement (by decelerating the movement until standstill "speed=0").

- If 'boExec'=TRUE (start positioning):
  Set 'boStop'=TRUE/FALSE to stop/resume positioning (until position end: 'boDone'=TRUE).
- If 'boExec'=FALSE (resume positioning):
  Set 'boStop'=TRUE (before end of positioning) to abort positioning (abort before position end: 'boDone'=TRUE).

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.
boExec	BOOL	Function execution: With a positive edge, the execution of the block starts. As long as 'boExec' = TRUE, the block is processed by the PLC. In the state 'boExec' = FALSE execution of the block is ended.
boStop	BOOL	With a positive edge, the execution of the block is aborted or completed.
enMode	ENUM	EN_EASY_POS_MODE Selection mode positioning Range: EASY_POS_REL: position difference EASY_POS_ABS: absolute end position Default: EASY_POS_REL
diSetPosition	DINT	Specification of the position setpoint (position setpoint system) [increments] EASY_POS_REL: position difference EASY_POS_ABS: absolute end position Unit: inc Default: 0
udSetSpeed	UDINT	Setpoint velocity to define the final velocity (increment difference of the output value over time). Range: 0...300000000 Unit: inc/s Default: 200000

Name	Type	Description
udSetAccel	UDINT	Acceleration (increment difference increase of the output value over time). Range: 0 ... 400000000 Unit: inc/s^2 Default: 100000
udSetDecel	UDINT	Deceleration (increment difference decrease of the output value over time). Range: 0 ... 4000000000 Unit: inc/s^2 Default: 100000

Output variables

Name	Type	Description
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled
boErr	BOOL	The function block is in an error state FALSE: No error (permitted commanding or warning) TRUE: Error
iErrID	INT	Error identity number: Diagnostic number is output iErrID = 0: No error iErrID ≠ 0, boErr = TRUE: Error iErrID ≠ 0, boErr = FALSE: Warning
strErrName	STRING (20)	Block name of the module generating the error Range: Meaning SET_SP_ POSITION: Error according to the 'SET_SETPOINT_POSITION' block POS: Error according to the 'POS' block
boDone	BOOL	Response that the function block has been completely executed.
bo0Vel	BOOL	When 'bo0Vel' is active, no setpoint is output.
boSetVel	BOOL	When 'boSetVel' is active, the target velocity has been reached.
diActSetPos	DINT	Current position setpoint of the integrated block 'SET_SETPOINT_POSITION'

Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

The 'EASY_POSITIONING' block uses the following basic functions from the AmkDevAccess library in combination.

- SET_SETPOINT_POSITION

(See document Software description AmkDevAccess Bibliothek , Part no. 109903)

The 'POS' function block (in 'POS_REL' mode) from the AmkBase library is used to organize the positioning operation.

For logical reasons, this block is called in the event-driven PGT task (PGT = Peripherie Grund Takt (peripheral basic cycle)) FPLC_TASK, because synchronous command variables (in the PGT grid according to ID2 'SERCOS cycle time') are set for the drive controller via it.

14.4 Support (support blocks)

The support blocks in the support folder comprise:

AmkCanCom_ACC

EASY_DRIVE

General

HANDLE_IDS
SHOW_CHAR_LIST
SHOW_LIST

14.4.1 AmkCanCommunication_ACC

14.4.1.1 EASY_DRIVE (FB)

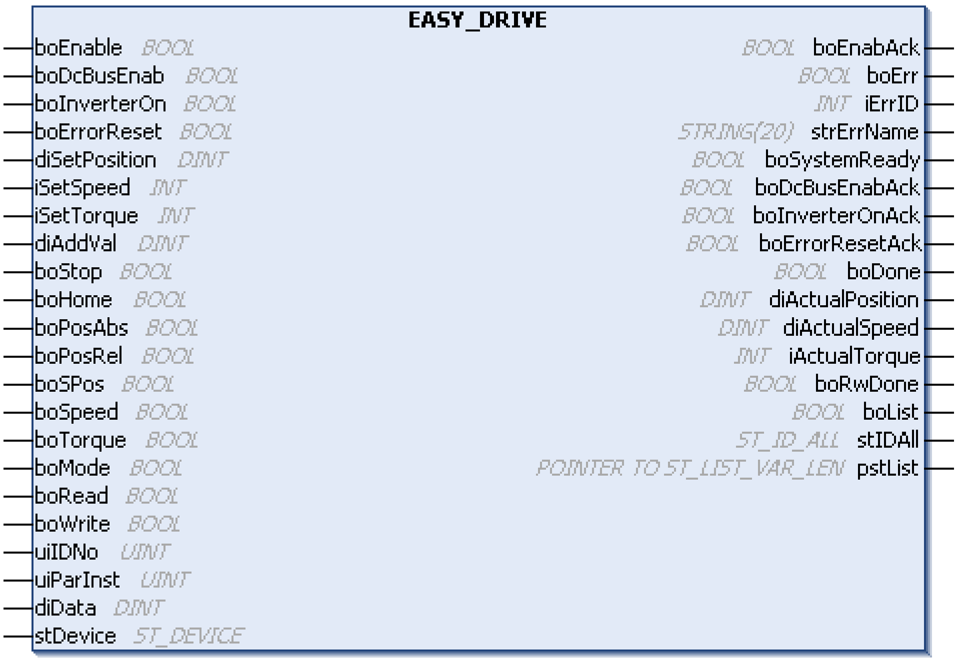
In the context of an ACC link involving AMK drives, the 'EASY_DRIVE' block is a simple drive interface with:

- Mapping of binary information (UE, RF, FL, SBM, QUE, QRF, QFL).
- Setting of movement setpoints and mapping of drive actual values.
- Commanding of drive basic functions (homing cycle, positioning, etc.).
- ID read / write access.

The 'EASY_DRIVE' block is only compatible with the ACC bus (ACC = AMK CAN communication). Based on the "AmkDriveAfp" library, it provides the function specific to the AMP (AMP = AMK fieldbus protocol) via the ACC bus.

(See document Software description AFP - AMK fieldbus protocol , Part no. 27872)

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.
boDcBusEnab	BOOL	DC-Bus Enable (UE = converter on)
boInverterOn	BOOL	Inverter On (RF = controller enable)
boErrorReset	BOOL	Error Reset (FL = clear error)

Name	Type	Description
diSetPosition	DINT	Specification of the position setpoint (position setpoint system) [increments] Unit: inc
iSetSpeed	INT	Set the velocity setpoint Unit: 1 rpm
iSetTorque	INT	Specification of the torque setpoint [0.1% Mn] Unit: 1/10% rated torque
diAddVal	DINT	Additive value (e.g. for operating mode selection with 'boMode') Value: 0..5: corresponding to operating mode 0..5
boStop	BOOL	With a positive edge, the execution of the block is aborted or completed.
boHome	BOOL	Homing drive Enable signal: With a positive edge, the homing cycle function starts. As long as 'boHome' = TRUE, the homing drive is carried out. Use a negative edge 'boHome' = FALSE to cancel the current referencing or terminate the completed referencing. (based on the homing cycle parameters ID147 'Homing parameter' or ID32926 'AMK homing cycle parameter').
boPosAbs	BOOL	Absolute positioning Enable signal: With a positive edge, the absolute positioning function starts. As long as 'boPosAbs' = TRUE, positioning is carried out. Use a negative edge 'boPosAbs' = FALSE to cancel the current positioning or terminate the completed positioning. Requirement: The homing point must be known, boID33036_RPF_known = TRUE Corresponding to 'iSetSpeed'.
boPosRel	BOOL	Relative positioning Enable signal: With a positive edge, the relative positioning function starts. As long as 'boPosRel' = TRUE, positioning is carried out. Use a negative edge 'boPosRel' = FALSE to cancel the current positioning or terminate the completed positioning. Corresponding to 'diSetPosition' and 'iSetSpeed'.
boSPos	BOOL	Start spindle positioning corresponding to 'diSetPosition' and 'iSetSpeed'.

Name	Type	Description
boSpeed	BOOL	Speed control Enable signal: With a positive edge, the speed control function starts. As long as 'boSpeed' = TRUE, speed control (speed setpoint) is carried out. Use a negative edge 'boSpeed' = FALSE to cancel the current speed control (with setpoint value = 0). Acceleration and deceleration ramp Variation 1 ID32780 'Acceleration ramp' and ID32781'Deceleration ramp'. By setting bit 6 = 1 in ID32802 'AMK secondary operating mode 2', a ramp generator (acceleration / deceleration) acts on the speed controller input. The entered times apply for acceleration and deceleration between the speed 0 U/min and ±ID113 'Maximum speed'. Requirement STANDARD_AXIS.fbVelocity.diVelocityRamp = 1 ms. Variation 2 The variable diVelocityRamp is the Ramp time in which the drive is accelerate or decelerate from the current velocity to the new set point velocity. Requirement: Setting bit 6 = 0 in ID32800 'AMK secondary operating mode 2'. (Hint: By activated automatically parametrisations the bit 6 will be automatically overwritten with 1). Corresponding to 'iSetSpeed'.
boTorque	BOOL	Start torque control, corresponding to 'iSetTorque'.
boMode	BOOL	Start of Operating mode switching with operating mode 0..5 in 'diAddVal'.
boRead	BOOL	Read parameter / ID
boWrite	BOOL	Write parameter / ID 'boList'=TRUE: WRITE_ID_LIST 'boList'=FALSE: WRITE_ID_DINT
uiIDNo	UINT	Parameter number (ID)
uiParInst	UINT	Parameter set number or instance number
diData	DINT	Parameter value

The binary inputs 'boStop', 'boHome', 'boPosAbs', 'boPosRel', 'boSPos', 'boSpeed', 'boTorque', and 'boMode' are prioritized in this order; logically, only one binary input (with the exception of 'boStop') should be active at any one time.

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table></td></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table></td></tr>
</table>

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>strErrName</td><td>STRING(20)</td><td>Block name of the module generating the error
<table>
<tr><td>Range</td><td>Meaning</td></tr>
<tr><td>'DEV_SYR__ SBM'</td><td>GET_STAT_SYSTEM_READY_x_SBM</td></tr>
<tr><td>'DEV_BEA__ QUE'</td><td>GET_STAT_DC_BUSENABLE_ACK_x_QUE</td></tr>
<tr><td>'DEV_IOA__ QRF'</td><td>GET_STAT_INVERTER_ON_ACK_x_QRF</td></tr>
<tr><td>'DEV_ERA__ QFL'</td><td>GET_STAT_ERR_RESET_ACK_x_QFL</td></tr>
<tr><td>'GET_ACTPOS'</td><td>GET_ACTUAL_POSITION</td></tr>
<tr><td>'DEV_GET_ STAT‘</td><td>DEV_GET_STAT</td></tr>
<tr><td>'DEV_SET_ CTRL‘</td><td>DEV_SET_CTRL</td></tr>
<tr><td>'AFP_BASIC‘</td><td>DO_AFP</td></tr>
<tr><td>'HANDLE_IDS'</td><td>HANDLE_IDS</td></tr>
</table></td></tr>
<tr><td>boSystemReady</td><td>BOOL</td><td>System ready (SBM = system ready message)</td></tr>
<tr><td>boDcBusEnabAck</td><td>BOOL</td><td>DC-Bus Enable Acknowledge (QUE = acknowledgement DC converter ON)</td></tr>
<tr><td>boInverterOnAck</td><td>BOOL</td><td>Inverter On Acknowledge (QRF = acknowledgement controller enable)</td></tr>
<tr><td>boErrorResetAck</td><td>BOOL</td><td>Error Reset Acknowledge (QFL = acknowledgement clear error)</td></tr>
<tr><td>boDone</td><td>BOOL</td><td>Response that the function block has been completely executed.</td></tr>
<tr><td>diActualPosition</td><td>DINT</td><td>Actual position
<table><tr><td>Unit</td><td>inc</td></tr></table></td></tr>
<tr><td>diActualSpeed</td><td>DINT</td><td>Actual velocity
<table><tr><td>Unit</td><td>1/10000 rpm</td></tr></table></td></tr>
<tr><td>iActualTorque</td><td>INT</td><td>Actual torque
<table><tr><td>Unit</td><td>1/10% rated torque</td></tr></table></td></tr>
<tr><td>boRwDone</td><td>BOOL</td><td>Handshake ID read/write completed</td></tr>
<tr><td>boList</td><td>BOOL</td><td>Identifier for a list parameter
<table>
<tr><td>FALSE</td><td>The data to be read is in 'stIDAll.diData'</td></tr>
<tr><td>TRUE</td><td>List parameter:
The list to be read is transferred to the list structure referenced by 'pbyData'</td></tr>
</table></td></tr>
<tr><td>stIDAll</td><td>STRUCT</td><td>ST_ID_ALL
Parameter information
Accommodates the element information
Structure with: data, min. value, max. value, attribute, unit, name</td></tr>
<tr><td>pstList</td><td>POINTER</td><td>POINTER TO ST_LIST_VAR_LEN
Pointer to the internal ID list</td></tr>
</table>

Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

The 'EASY_DRIVE' block combines the following basic functions.

Based on the AmkDevAccess library:

- SET_CTRL_DC_BUSENABLE_x_UE
- SET_CTRL_INVERTER_ON_x_RF
- SET_CTRL_ERR_RESET_x_FL
- GET_STAT_SYSTEM_READY_x_SBM
- GET_STAT_DC_BUSENABLE_ACK_x_QUE
- GET_STAT_INVERTER_ON_ACK_x_QRF
- GET_STAT_ERR_RESET_ACK_x_QFL
- GET_ACTUAL_POSITION
- DO_AFP,
  for the execution of the AFP (AMK fieldbus protocol) with the AFP codes:
- AFP_ZERO, AFP_STOP, AFP_HOME, AFP_POSA, AFP_POSR, AFP_SPOS, AFP_SPEED, AFP_TORQUE, AFP_MODE0, .., AFP_MODE5

(See document Software description AmkDevAccess Bibliothek , Part no. 109903)

Based on the AmkSystem library:

- READ_ID_LIST_ALL
- WRITE_ID_DINT
- WRITE_ID_LIST

(See document Software description AmkSystem library , Part no. 205004)

For logical reasons, this block should not be called in the event-driven PGT task (PGT = Peripherie Grund Takt (peripheral basic cycle)) FPLC_TASK but in a cyclic or "free-running" task (PLC_TASK, for example).

Integrating the 'SHOW_LIST' support block enables list IDs to be displayed and edited with the 'ViEasyDevice' visualization, for example.

IDs are read with 'boRead'=TRUE, using the 'READ_ID_LIST_ALL' block. IDs are selected based on 'uiIDNo' and 'uiParInst' (see the AmkSystem documentation). The corresponding device is identified by the 'stDevice' variable, which is initialized automatically. For a standard ID (not a list ID: 'boList'=FALSE), the complete ID information (data, min. value, max. value, attribute, unit, name) is made available in the 'stIDAll' structure. For a list ID: ('boList'= TRUE), the list value (data) is saved in a local 'ST_LIST_VAR_LEN' type structure. This structure can be read or edited with the 'ViEasyDrive' visualization. In programming terms, it can be accessed with the 'pstList' pointer. A distinction can be made between standard IDs and list IDs with 'boList' (see above).

Based on this variable, when writing an ID 'boWrite'=TRUE, either the value is taken from 'diData' (not a list ID: 'boList'=FALSE') or the information is written back to the 'ST_LIST_VAR_LEN' type structure ('boList'=TRUE).

In the context of ID access based on the 'EASY_DEVICE' block, read access must always be carried out before commencing a write operation.

During a read operation, the ID type for the subsequent write operation is defined by reading the 'boList' variable:

'boList'=FALSE: → simple data type

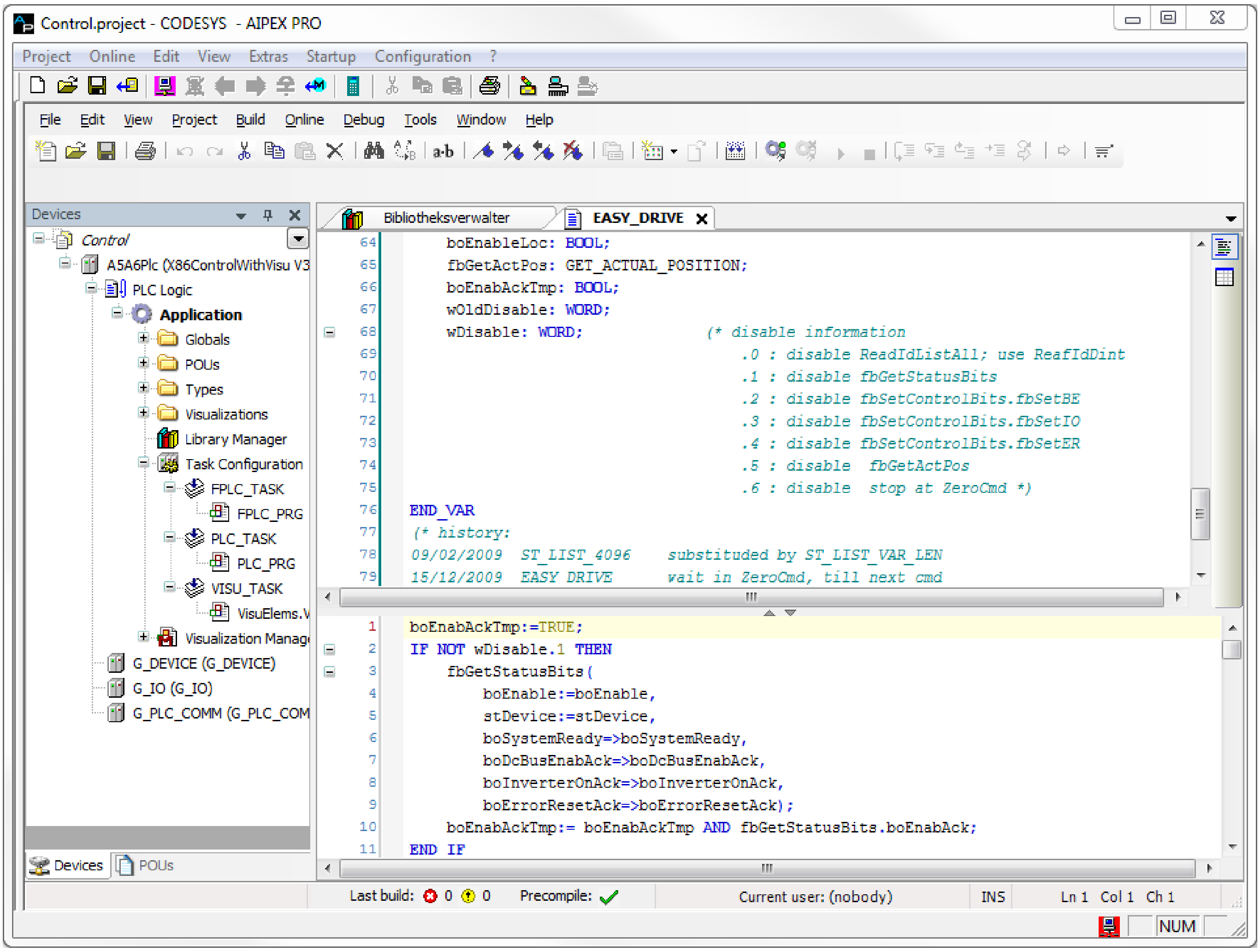
'boList'=TRUE: → list type.

The 'SelectWriteAuto', 'SelectWriteSimple', and 'SelectWriteList' operations can be executed to influence the automatic definition of the write behavior described above.

Actions

Name	Description
SelectWriteAuto()	'boList' is determined automatically when reading the ID.
SelectWriteSimple()	'boList'=FALSE; a simple data type is always written, based on the 'WRITE_ID_DINT' block
SelectWriteList()	'boList'=TRUE; a list type is always written, based on the 'WRITE_ID_LIST' block. The list header, which consists of the current and the maximum list lengths, must be specified correctly.

The local variable 'wDisable' is used to disable the base function contained in the block (see figure). If the base function is disabled, it is not processed when the entire block is enabled. As a result, the necessary bus information does not have to be "mapped".



The individual bits of the 'wDisable' variable have the following meanings:

wDisable.0:	disable ReadIdListAll; use ReadIdDint	(action: DisableReadIdListAll) 1)
wDisable.1:	disable fbGetStatusBits	(action: DisableGetStatus)
wDisable.2:	disable fbSetControlBits.fbSetBE	(action: DisableBe)
wDisable.3:	disable fbSetControlBits.fbSetIO	(action: DisableIo)
wDisable.4:	disable fbSetControlBits.fbSetEr	(action: DisableEr)
wDisable.5:	disable fbGetActPos	(action: DisableActPos)
wDisable.6:	disable stop at zero	(action: DisableStopAtZero)

The setting of the corresponding bit(s) can either be organized as an initial value when the block instance is created or it can be set during the course of the assigned actions. The 'EnableAll' action clears all 'disable bits' (wDisable:=0).

1) Setting 'wDisable.0':=TRUE deselects the rather more complex mechanism which involves using the 'READ_ID_LIST_ALL' block and the type distinction for write operations based on 'boList' (see above). Instead, only the ID value (the data) is read or written (with the 'READ_ID_DINT' or 'WRITE_ID_DINT' block).

14.4.2 General

14.4.2.1 HANDLE_IDS (FB)

The 'HANDLE_IDS' block is a support block for reading / writing remanent and temporary IDs.

The block is used in the context of the 'EASY_DEVICE' and 'EASY_DRIVE' blocks, for example. It is based on the blocks in the AmkSystem library and combines their function in a more compact form.

The following blocks are integrated:

- READ_ID_LIST_ALL
- READ_ID_DINT
- WRITE_ID_DINT
- WRITE_ID_LIST

- READ_ID_DINT_TMP
- WRITE_ID_DINT_TMP

User interface

Input variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boRead</td><td>BOOL</td><td>Read parameter / ID</td></tr>
<tr><td>boWrite</td><td>BOOL</td><td>Write parameter / ID</td></tr>
<tr><td>enMode</td><td>ENUM</td><td>EN_WRITE_ID_MODE
Selection mode
<table>
<tr><td>Default</td><td>WRITE_ID_AUTO</td></tr>
</table>
<table>
<tr><th>Range</th><th>Meaning</th></tr>
<tr><td>WRITE_ID_
AUTO</td><td>To write an ID, 'boList' is taken from an upstream 'ReadIdListAll'.
FALSE: The ID read is not a list ID. The 'WRITE_ID_DINT' block is used for the subsequent write operation.
TRUE: The ID read is a list ID. The 'WRITE_ID_LIST' block is used for the subsequent write operation.
In this mode, an ID read function must be called prior to every write operation.</td></tr>
<tr><td>WRITE_
SIMPLE_ID</td><td>Regardless of 'ReadIdListAll', 'boList' is set to FALSE. The 'WRITE_ID_DINT' block is used for the write operation</td></tr>
<tr><td>WRITE_LIST_ID</td><td>Regardless of 'ReadIdListAll', 'boList' is set to TRUE. The 'WRITE_ID_LIST block is used for the write operation</td></tr>
<tr><td>ACCESS_ID_
TMP</td><td>Temporary ID access follows. Only effective in conjunction with the 'DisableReadIdListAll' action (see actions)</td></tr>
</table></td></tr>
<tr><td>uiIDNo</td><td>UINT</td><td>Parameter number (ID)(ID) whose element is read / written</td></tr>
<tr><td>uiParInst</td><td>UINT</td><td>Parameter set number or instance number</td></tr>
<tr><td>diData</td><td>DINT</td><td>Parameter value that is written to the database or temporary value (if 'boList'=FALSE).
The ID value read from a standard ID (not a list ID) is always returned in 'stIDAll.diData' (see output variables)</td></tr>
<tr><td>uiSize</td><td>UINT</td><td>Maximum data length available to accommodate the information to be read.
uiSize ≤ SIZEOF(variable) referenced by 'pbyData'!
<table>
<tr><td>Unit</td><td>BYTE</td></tr>
</table></td></tr>
<tr><td>pbyData</td><td>POINTER</td><td>POINTER TO READ DATA
Pointer referencing the structure / variable which is receiving the information read.</td></tr>
</table>

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boDone</td><td>BOOL</td><td>Response that the function block has been completely executed.</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table></td></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error
<table>
<tr><td>Range</td><td>Meaning</td></tr>
<tr><td>0</td><td>No error</td></tr>
<tr><td>otherwise</td><td>Error information according to SERCOS definition
(See documentSoftware description AmkBase Bibliothek , Part no. 204986 (error information)</td></tr>
</table></td></tr>
<tr><td>boList</td><td>BOOL</td><td>Identifier for a list parameter
<table>
<tr><td>FALSE</td><td>The data to be read is in 'stIDAll.diData'</td></tr>
<tr><td>TRUE</td><td>List parameter:
The list to be read is transferred to the list structure referenced by 'pbyData'</td></tr>
</table></td></tr>
<tr><td>stIDAll</td><td>STRUCT</td><td>ST_ID_ALL
Parameter information
Accommodates the element information
The ID values for all read functions of standard IDs (not list IDs) is provided in 'stIDAll.diData'</td></tr>
</table>

Input and output variables

Name	Type	Description
stDevice	STRUCT	The device description structure assigns the block a device.

Actions

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>DisableReadIdListAll</td><td>The 'ReadIdListAll' (READ_ID_LIST_ALL) function is deactivated.

If 'enMode'='ACCESS_ID_TMP', access (read / write) takes place with the 'WRITE_ID_DINT_TMP' / 'READ_ID_DINT_TMP' blocks.
Otherwise, access (read / write) is via the 'WRITE_ID_DINT' / 'READ_ID_DINT' blocks.
</td></tr>
<tr><td>EnableAll</td><td>The 'ReadIdListAll' function is activated (default).</td></tr>
</table>

14.4.2.2 SHOW_CHAR_LIST (FB)

The 'SHOW_CHAR_LIST' support block displays "character" lists. The block is used, for example, in the context of the 'ViShowList', 'ViEasyDevice', and 'ViEasyDrive' visualizations.

User interface

Input variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>uiIndex</td><td>UINT</td><td>Search index starting from which the list is visualized with the 'ViShowList' visualization</td></tr>
<tr><td>boNoInput</td><td>BOOL</td><td>Controls visualization visibility<table><tr><th>Range</th><th>Meaning</th></tr><tr><td>FALSE</td><td>List element visible</td></tr><tr><td>TRUE</td><td>List element not visible</td></tr></table></td></tr>
</table>

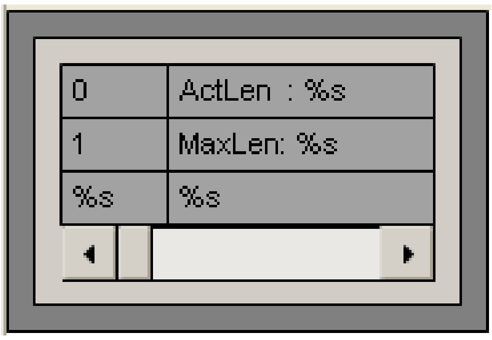
Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>uiActLen</td><td>UINT</td><td>Current list length<table><tr><td>Unit</td><td>Byte</td></tr></table></td></tr>
<tr><td>uiMaxLen</td><td>UINT</td><td>Maximum list length<table><tr><td>Unit</td><td>Byte</td></tr></table></td></tr>
</table>

Input and output variables

Name	Type	Description
stList	STRUCT	ST_LIST_VAR_LEN List to be visualized with 'ViShowList', for example.

The 'SHOW_CHAR_LIST' block supports the display of a subset of a 'ST_LIST_VAR_LEN' type list, as well as the list header information (current, maximum list length); e.g. with the 'ViShowCharList' visualization (see figure). The list values are displayed in a STRING(14) type ASCII character string.



14.4.2.3 SHOW_LIST (FB)

The 'SHOW_LIST' support block supports the display and editing of lists. The block is used, for example, in the context of the 'ViShowList', 'ViEasyDevice', and 'ViEasyDrive' visualizations.

User interface

Input variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>uiIndex</td><td>UINT</td><td>Search index starting from which the list is visualized with the 'ViShowList' visualization.</td></tr>
<tr><td>boNoInput</td><td>BOOL</td><td>Controls visualization visibility<table><tr><th>Range</th><th>Meaning</th></tr><tr><td>FALSE</td><td>List element visible</td></tr><tr><td>TRUE</td><td>List element not visible</td></tr></table></td></tr>
</table>

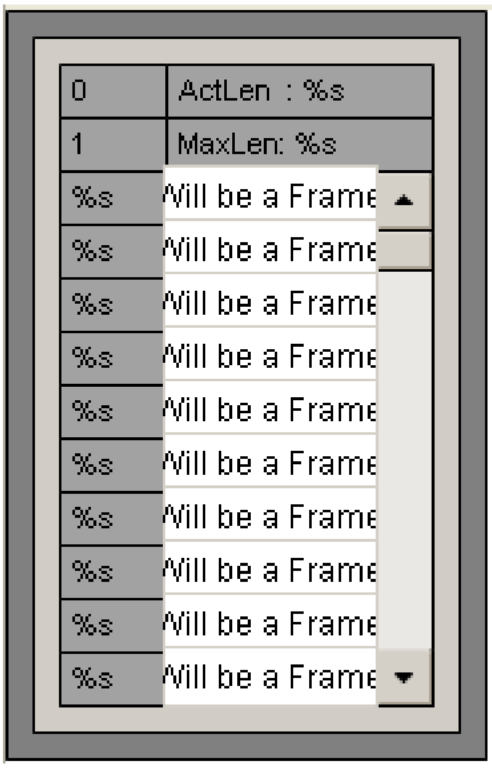
Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>uiActLen</td><td>UINT</td><td>Current list length<table><tr><td>Unit</td><td>Byte</td></tr></table></td></tr>
<tr><td>uiMaxLen</td><td>UINT</td><td>Maximum list length<table><tr><td>Unit</td><td>Byte</td></tr></table></td></tr>
</table>

Input and output variables

Name	Type	Description
stList	STRUCT	ST_LIST_VAR_LEN List to be visualized with 'ViShowList', for example.

The 'SHOW_LIST' block supports the display of a subset of a 'ST_LIST_VAR_LEN' type list, as well as the list header information (current, maximum list length); e.g. with the 'ViShowList' visualization. The list values are displayed in a DINT type array of 10 values.



15 AmkFile - File function specific to AMK

AmkFile is an external file library which provides the file function specific to AMK. It is divided into:

DirectoryAccess	Directory access functions
FileAccess	File access functions
SupportFunctions	Support functions

With regard to file name identifiers for the following blocks:
- File name extension is permitted
- The file name may contain up to 64 characters (plus extension)

Furthermore:
- The file size is limited only by the available physical file system memory
- The file name is case-sensitive
- The access blocks must be processed in a lower-priority task with sufficient cycle time
- The blocks are processed in full on each call

The following characters must not be used when creating a directory or a file:
- \ : * ?\ : * ? " |" < > |
- In CODESYS, $ is interpreted as a reference to the ASCII table
- ' indicates the end of the input string
- & characters are not displayed correctly in the visualization.

15.1 DirectoryAccess

CREATE_DIR_1	Create a directory / folder
REMOVE_DIR_1	Delete a directory / folder

15.1.1 CREATE_DIR_1 (FB)

The 'CREATE_DIR_1' function block creates a directory or a directory structure.

User interface

Input variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boExec</td><td>BOOL</td><td>Function execution: With a positive edge, the execution of the block starts.
As long as 'boExec' = TRUE, the block is processed by the PLC.
In the state 'boExec' = FALSE execution of the block is ended.</td></tr>
<tr><td>enCreateMode</td><td>ENUM</td><td>EN_CREATE_MODE
Selection of creation mode
<table>
<tr><td>Default</td><td>CREATE_IF_PARENTS</td></tr>
</table>
<table>
<tr><th>Range</th><th>Meaning</th></tr>
<tr><td>CREATE_IF_
PARENTS</td><td>Only creates the directory if the higher-level directory exists.</td></tr>
<tr><td>CREATE_
ALWAYS</td><td>Creates a directory and the specific subdirectories (a complete directory structure)</td></tr>
</table></td></tr>
</table>

Name	Type	Description
strDirName	STRING (64)	Name of the directory or the complete directory structure. A maximum of 64 characters are permitted.

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boDone</td><td>BOOL</td><td>Response that the function block has been completely executed.</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table></td></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error
<table>
<tr><th>Range</th><th>Meaning</th></tr>
<tr><td>0</td><td>No error</td></tr>
<tr><td>1</td><td>Illegal file name</td></tr>
</table>
In addition, the global error codes listed may also be displayed Siehe 'Table 1: Global AmkFile function block error codes' auf Seite 534.</td></tr>
</table>

15.1.2 REMOVE_DIR_1 (FB)

The 'REMOVE_DIR_1' function block deletes a directory or a directory structure.

User interface

Input variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boExec</td><td>BOOL</td><td>Function execution: With a positive edge, the execution of the block starts.
As long as 'boExec' = TRUE, the block is processed by the PLC.
In the state 'boExec' = FALSE execution of the block is ended.</td></tr>
<tr><td>enRemoveMode</td><td>ENUM</td><td>EN_REMOVE_MODE
Selects delete mode
<table>
<tr><td>Default</td><td>REMOVE_IF_EMPTY</td></tr>
</table>
<table>
<tr><th>Range</th><th>Meaning</th></tr>
<tr><td>REMOVE_IF_EMPTY</td><td>Deletes a directory only if it is empty.</td></tr>
<tr><td>REMOVE_ALWAYS</td><td>Deletes a directory along with all subordinate files and directories.</td></tr>
</table></td></tr>
<tr><td>strDirName</td><td>STRING (64)</td><td>Name of the directory or the complete directory structure.
A maximum of 64 characters are permitted.</td></tr>
</table>

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boDone</td><td>BOOL</td><td>Response that the function block has been completely executed.</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table></td></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error
<table>
<tr><th>Range</th><th>Meaning</th></tr>
<tr><td>0</td><td>No error</td></tr>
<tr><td>1</td><td>Illegal file name</td></tr>
</table>
In addition, the global error codes listed may also be displayed Siehe 'Table 1: Global AmkFile function block error codes' auf Seite 534.</td></tr>
</table>

15.2 FileAccess

FIND_FILE_1	Find a file
READ_FILE_1	Read a file
REMOVE_FILE_1	Delete a file
RENAME_FILE_1	Rename and / or move a file
SIZE_FILE_1	Determine the length of a file
WRITE_FILE_1	Write to a file

15.2.1 FIND_FILE_1 (FB)

The 'FIND_FILE_1' function block finds a file in the file system and returns the file name and file size.

User interface

Input variables

Name	Type	Description
boExec	BOOL	Function execution: With a positive edge, the execution of the block starts. As long as 'boExec' = TRUE, the block is processed by the PLC. In the state 'boExec' = FALSE execution of the block is ended.
strFileSpec	STRING (64)	File specification. All standard rules for file name generation are permitted (even the wildcards '*' and '?' are permitted). A maximum of 64 characters are permitted.
uiIndex	UINT	Search index(0, 1, ...), to select the file found and displayed in the 'strFileName', 'udSize' output variables. If a file with this file specification cannot be found, error code 16 is displayed ('iErrID'=16) Siehe 'Table 1: Global AmkFile function block error codes' auf Seite 534.

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boDone</td><td>BOOL</td><td>Response that the function block has been completely executed.</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state<table><tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr><tr><td>TRUE</td><td>Error</td></tr></table></td></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output<table><tr><td colspan="2">iErrID = 0</td><td>No error</td></tr><tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr><tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr></table>Error<table><tr><td>Range</td><td>Meaning</td></tr><tr><td>0</td><td>No error</td></tr><tr><td>1</td><td>Illegal file specification</td></tr><tr><td>2</td><td>Illegal search index</td></tr></table>In addition, the global error codes may also be displayed. Siehe 'Table 1: Global AmkFile function block error codes' auf Seite 534.</td></tr>
<tr><td>strFileName</td><td>STRING (64)</td><td>File name, including extension
A maximum of 64 characters are permitted.</td></tr>
<tr><td>udSize</td><td>UDINT</td><td>Specifies the size of the data memory [byte]
If this values is greater than the actual data range, then an undefined data range is accessed.</td></tr>
</table>

15.2.2 READ_FILE_1 (FB)

The 'READ_FILE_1' function block reads a file and saves 'udSize' of the file content to a buffer structure which references the 'bpyBuffer' pointer variable.

User interface

Input variables

Name	Type	Description
boExec	BOOL	Function execution: With a positive edge, the execution of the block starts. As long as 'boExec' = TRUE, the block is processed by the PLC. In the state 'boExec' = FALSE execution of the block is ended.

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>enReadMode</td><td>ENUM</td><td>EN_READ_MODE
Selects read mode
<table>
<tr><td>Default</td><td>READ_BEGIN</td></tr>
</table>
<table>
<tr><th>Range</th><th>Meaning</th></tr>
<tr><td>READ_BEGIN</td><td>The buffer structure is populated with the content of the file. The process commences at the start of the file</td></tr>
<tr><td>READ_ CURRENT</td><td>The buffer structure is populated with the content of the file. The process resumes at the current position of the file. (The current position is the position at which the last read operation ended with the instance.)</td></tr>
</table></td></tr>
<tr><td>strFileName</td><td>STRING (64)</td><td>File name, including extension
A maximum of 64 characters are permitted.</td></tr>
<tr><td>udSize</td><td>UDINT</td><td>Specifies the size of the data memory [byte]
If this values is greater than the actual data range, then an undefined data range is accessed.
The number of the last byte written to the buffer is displayed in the 'udNoByte' variable (see output variables).</td></tr>
<tr><td>pbyFileBuff</td><td>POINTER TO BYTE</td><td>Pointer variable referencing the buffer structure to which the file information is copied.
There must be an assurance that the buffer structure is greater than or equal to the 'udSize' variable. In the absence of such assurance, adjacent data in the buffer structure may be damaged beyond repair!</td></tr>
</table>

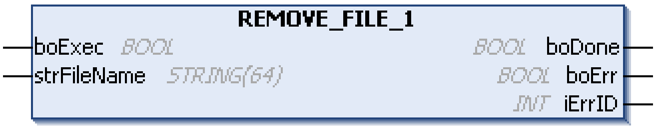
Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boDone</td><td>BOOL</td><td>Response that the function block has been completely executed.</td></tr>
<tr><td>strFileNameTmp</td><td>STRING</td><td>File name, including extension</td></tr>
<tr><td>udNoByte</td><td>UDINT</td><td>Byte number of the last file information transferred to the buffer. This corresponds to the last byte index of the buffer structure in which file information was transferred (subject to commencing with 'udNoByte = 0' in the first byte of the buffer).</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table></td></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error
<table>
<tr><th>Range</th><th>Meaning</th></tr>
<tr><td>0</td><td>No error</td></tr>
<tr><td>1</td><td>Illegal file name</td></tr>
<tr><td>2</td><td>Illegal size 'udSize'</td></tr>
<tr><td>3</td><td>Illegal buffer pointer</td></tr>
</table>
In addition, the global error codes may also be displayed. Siehe 'Table 1: Global AmkFile function block error codes' auf Seite 534.</td></tr>
</table>

15.2.3 REMOVE_FILE_1 (FB)

The 'REMOVE_FILE_1' function block removes a file from the file system.

User interface



Input variables

Name	Type	Description
boExec	BOOL	Function execution: With a positive edge, the execution of the block starts. As long as 'boExec' = TRUE, the block is processed by the PLC. In the state 'boExec' = FALSE execution of the block is ended.
strFileName	STRING (64)	File name, including extension A maximum of 64 characters are permitted.

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boDone</td><td>BOOL</td><td>Response that the function block has been completely executed.</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table></td></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error
<table>
<tr><th>Range</th><th>Meaning</th></tr>
<tr><td>0</td><td>No error</td></tr>
<tr><td>1</td><td>Illegal file name</td></tr>
</table>
In addition, the global error codes may also be displayed. Siehe 'Table 1: Global AmkFile function block error codes' auf Seite 534.</td></tr>
</table>

15.2.4 RENAME_FILE_1 (FB)

The 'RENAME_FILE_1' function block is used to rename or move a file.

User interface

Input variables

Name	Type	Description
boExec	BOOL	Function execution: With a positive edge, the execution of the block starts. As long as 'boExec' = TRUE, the block is processed by the PLC. In the state 'boExec' = FALSE execution of the block is ended.

Name	Type	Description
strCurrFileName	STRING (64)	Specification of the full path to the file which is to be renamed or moved. A maximum of 64 characters are permitted.
strNewFileName	STRING (64)	Specification of the full path to the location in which the file with the new name is to be created. A maximum of 64 characters are permitted.

Output variables

Name	Type	Description
boDone	BOOL	Response that the function block has been completely executed.
boErr	BOOL	The function block is in an error state FALSE: No error (permitted commanding or warning) TRUE: Error
iErrID	INT	Error identity number: Diagnostic number is output iErrID = 0: No error iErrID ≠ 0, boErr = TRUE: Error iErrID ≠ 0, boErr = FALSE: Warning Error Range: Meaning 0: No error 1: Illegal file name In addition, the global error codes may also be displayed. Siehe 'Table 1: Global AmkFile function block error codes' auf Seite 534.

15.2.5 SIZE_FILE_1 (FB)

The 'SIZE_FILE_1' function block returns the file length of a file.

User interface

Input variables

Name	Type	Description
boExec	BOOL	Function execution: With a positive edge, the execution of the block starts. As long as 'boExec' = TRUE, the block is processed by the PLC. In the state 'boExec' = FALSE execution of the block is ended.
strFileName	STRING (64)	File name, including extension A maximum of 64 characters are permitted.

Output variables

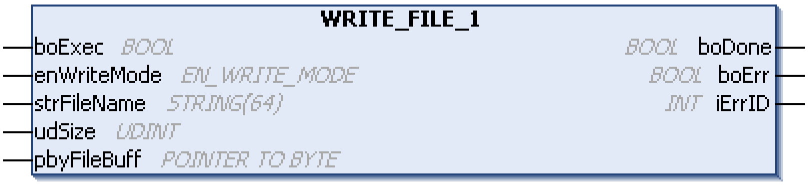
Name	Type	Description
boDone	BOOL	Response that the function block has been completely executed.
boErr	BOOL	The function block is in an error state FALSE: No error (permitted commanding or warning) TRUE: Error

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error
<table>
<tr><th>Range</th><th>Meaning</th></tr>
<tr><td>0</td><td>No error</td></tr>
<tr><td>1</td><td>Illegal file name</td></tr>
</table>
In addition, the global error codes may also be displayed. Siehe 'Table 1: Global AmkFile function block error codes' auf Seite 534.</td></tr>
<tr><td>udSize</td><td>UDINT</td><td>Specifies the size of the data memory [byte]
If this values is greater than the actual data range, then an undefined data range is accessed.</td></tr>
</table>

15.2.6 WRITE_FILE_1 (FB)

The 'WRITE_FILE_1' function block is used to write the 'udSize' of a buffer structure to a file. The buffer structure is assigned with the 'pbyFileBuff' pointer variable.

User interface



Input variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boExec</td><td>BOOL</td><td>Function execution: With a positive edge, the execution of the block starts.
As long as 'boExec' = TRUE, the block is processed by the PLC.
In the state 'boExec' = FALSE execution of the block is ended.</td></tr>
<tr><td>enWriteMode</td><td>ENUM</td><td>EN_WRITE_MODE
Selection of write mode
<table>
<tr><td>Default</td><td>WRITE_NEW</td></tr>
</table>
<table>
<tr><th>Range</th><th>Meaning</th></tr>
<tr><td>WRITE_NEW</td><td>The content of the buffer structure overwrites the current content of the file</td></tr>
<tr><td>WRITE_APPEND</td><td>The content of the buffer structure is appended to the current position in the file.
(The current position is the position at which the last write operation ended with this instance.)</td></tr>
</table></td></tr>
<tr><td>strFileName</td><td>STRING (64)</td><td>File name, including extension
A maximum of 64 characters are permitted.</td></tr>
<tr><td>udSize</td><td>UDINT</td><td>Specifies the size of the data memory [byte]
If this values is greater than the actual data range, then an undefined data range is accessed.</td></tr>
</table>

Name	Type	Description
pbyFileBuff	POINTER	POINTER_TO_BYTE Pointer variable referencing the buffer structure containing the file information to be written. There must be an assurance that the buffer structure is greater than or equal to the 'udSize' variable.

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boDone</td><td>BOOL</td><td>Response that the function block has been completely executed.</td></tr>
<tr><td>udiByteWrite</td><td>UDINT</td><td>Number of written bytes</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table></td></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error
<table>
<tr><td>Range</td><td>Meaning</td></tr>
<tr><td>0</td><td>No error</td></tr>
<tr><td>1</td><td>Illegal file name</td></tr>
<tr><td>2</td><td>Illegal file size (udSize)</td></tr>
<tr><td>3</td><td>Illegal buffer pointer</td></tr>
<tr><td>4</td><td>Illegal write mode</td></tr>
</table>
In addition, the global error codes may also be displayed. Siehe 'Table 1: Global AmkFile function block error codes' auf Seite 534.</td></tr>
</table>

15.3 SupportFunctions

FdiGetFreeSpace	Displays the free memory capacity
FiFileConnect	Establish connection to logical device
FiGetFileAttr	Get file attributes

15.3.1 FdiGetFreeSpace (F)

The 'FdiGetFreeSpace' function shows the current memory capacity on the selected logical device.

User interface

Input variables

Name	Type	Description
strPath	STRING (64)	Device path (optional: logical device + additional path)

Output variables

Name	Type	Description
FdiGetFreeSpace	DINT	Range: -1 = Error Range: >0 = Free memory capacity in Kbytes

Range	Meaning
-1	Error
>0	Free memory capacity in Kbytes

15.3.2 FiFileConnect (F)

The 'FiFileConnect' function is used to assign a logical device identifier (drive identifier) to a physical memory space (external file system). There are 3 different variants.

Logical device identifier / drive identifier	Physical memory space
CST_1 .. CST_99	"Customer" directory on the controller's internal hard disk
 USB11_1 .. USB_11_99 USB12_1 .. USB_12_99 USB21_1 .. USB_21_99 USB22_1 .. USB_22_99	USB stick 1st USB device, 1st partition 1st USB device, 2nd partition 2nd USB device, 1st partition 3rd USB device, 2nd partition
EXT_1 .. EXT_99	External file system, e.g. on MS Windows PC
Index_1 ... _99 at the end of the identifier can be used to define multiple logical identifiers for a single device.	

The 'FiFileConnect' function requires the 'Server Message Block Version 1' (SMBv1) network protocol. This network protocol is not installed on Windows 10, Windows Server 2016 and other versions of Windows.

Diagnostic message when a function block is called without active 'SMBv1':
iExt1DriveStatus: = FiFileConnect ('EXT_1', '192.168.0.153/A5protocols','user=yyyyy,password=xxxx,sec=ntlm');
The FiFileConnect function block returns status 4.
iExt1DriveStatus = 4

An installation example for 'SMBv1' can be found after the function block description.
See: Install / activate the network protocol 'SMBv1' in Windows 10 Professional

During data access, the 'strPhysicalAddress' option can be used to extend the path identifier by up to 64 characters.

In case when the external file system is interrupted, all file access functions and the function FiFileConnect are interrupted for 20 seconds.
Therefore, call the file functions in a 'free-running task' with a real-time priority higher than 15.

User interface

Input variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>strLogicalDevice</td><td>STRING (10)</td><td>Logical device / drive identifier with which a connection is to be established:
<table>
<tr><td>CST_<x></td><td>"Customer" directory on the controller's internal hard disk; where <x>={1..99}</td></tr>
<tr><td>USB<ab>_<x></td><td>USB device <a>, partition <b>; where <a>, <b> = {1..2}, <x> ={1..99}</td></tr>
<tr><td>EXT_<x></td><td>External file system (e.g. MS Windows PC); where <x>= {1..99}</td></tr>
</table></td></tr>
<tr><td>strPhysicalAddress</td><td>STRING (64)</td><td>Physical address which can be entered dependent on a logical device / drive identifier:
<table>
<tr><td>CST_<x></td><td>Optional path</td></tr>
<tr><td>USB<ab>_<x></td><td>Optional path</td></tr>
<tr><td>EXT_<x></td><td>IP address or computer name</td></tr>
</table></td></tr>
<tr><td>strOptions</td><td>STRING (128)</td><td>Option for access to an external file system; where EXT_<x> in the format: "user=xxxx,password=yyyy,domain=zzzz"</td></tr>
</table>

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>FiFileConnect</td><td>INT</td><td>
<table>
<tr><td>0</td><td>Connection successfully established</td></tr>
<tr><td>1</td><td>Incorrect logical drive</td></tr>
<tr><td>2</td><td>Drive missing (e.g. USB stick is not plugged in)</td></tr>
<tr><td>3</td><td>Incorrect physical address (e.g. for EXT drive without physical address)</td></tr>
<tr><td>4</td><td>Unable to establish connection</td></tr>
<tr><td>5</td><td>Connection already exists; when attempted again, unable to re-establish the connection</td></tr>
</table></td></tr>
</table>

15.3.2.1 CST_<x>

CST_<x> establishes a connection to the controller's non-volatile internal user data space ("cst directory"; where: cst = customer).

<x> connection number: Up to 99 connections can be opened to the CST directory.

Example:

strLogicalDevice:=CST_1 (1st connection to the CST directory)

When connecting to the CST directory, a directory (or an entire directory hierarchy) can be specified at the strPhysicalAddress input. The strFileName variable of an AmkFile function blocks can be prefixed with this string. Thus the strFileName input of the AmkFile function blocks can be extended by approx. 64 characters.

The directories specified with strPhysicalAddress must be created. The FiFileConnect function does not create directories or files!

The strOptions variable is not relevant when connecting to a CST directory.

Call syntax:

FiFileConnect (strLogicalDevice:= 'CST_1', strPhysicalAddress:= 'Test/myFolder', strOptions:='')

→ strFileName:= 'CST_1:Datei.txt' (e.g. in the context of the READ_FILE_1 block) then addresses the "/Test/myFolder/Datei.txt" file in the customer directory on the controller. Siehe 'READ_FILE_1 (FB)' auf Seite 460.

15.3.2.2 USB<ab>_<x>

USB<ab>_<x> establishes a connection to a USB device that is connected to the USB interfaces. The file system on the USB device must be a FAT file system.

<a> is the USB device (a maximum of two devices can be connected)

<b> is the partition (a maximum of two partitions per USB device are possible)

<x> connection number (up to 99 connections can be opened to a USB device)

Example:

strLogicalDevice:=USB12_1 (1st USB device, 2nd partition, 1st connection)

Connecting more than two USB devices may alter the controller startup sequence!

When connecting to a USB device, a directory (or an entire directory hierarchy) can be specified at the strPhysicalAddress input. The strFileName variable of an AmkFile function blocks can be prefixed with this string.

The directories specified with strPhysicalAddress must be created. The FiFileConnect function does not create directories or files!

The strOptions variable is not relevant when connecting to a USB device.

Call syntax:

FiFileConnect (strLogicalDevice:= 'USB12_1', strPhysicalAddress:= 'Test/myFolder', strOptions:='')

→ strFileName:= 'USB12_1':Datei.txt' (e.g. in the context of the READ_FILE_1 block) then addresses the "/Test/myFolder/Datei.txt" file on the 2nd partition of the 1st USB device. Siehe 'READ_FILE_1 (FB)' auf Seite 460.

15.3.2.3 EXT_<x>

EXT_<x> establishes a connection to an external file system (e.g. a Microsoft Windows PC) via the network.

<x> connection number (up to 99 network connections to various computers are possible)

The directories and files to be accessed via the network must have corresponding authorizations / access rights.

For a connection to an external file system (EXT_<x>), the strPhysicalAddress can be assigned the following two options:

1. Enter IP address.
   This results in a direct attempt to establish a connection to the IP address. The IP address must be assigned in the network.
2. Enter computer name.
   If a computer name is entered, there must be a DNS (domain name system) server present on the network. The DNS server converts the computer name into the corresponding IP address. So that the controller can find the DNS server on the network, the IP address of the DNS server must be entered in ID34216 'DNS server address' (in hexadecimal notation).

A directory must be specified after the entry of the IP address or the computer name. If subdirectories are to be accessed, they must be enabled separately. These subdirectories can be specified directly after the entry of the IP address or the computer name.

Example:

strLogicalDevice:=EXT_1 (1st connection to the external directory)

strPhysicalAddress:='192.168.0.1/MyFolder'

For a connection to an external file system (EXT_<x>), the strOptions must be assigned the following values:

- user
- password
- domain (optional)

The strOptions input is a STRING type variable with 128 characters including the predefined values (user=,password=,domain,=).

Example for strOptions:

Domain not specified	strOptions:='user=xxxxxx,password=yyyyyy'
Domain specified	strOptions:='user=xxxxxx,password=yyyyyy,domain=zzzzzz'

Call syntax:

FiFileConnect(strLogicalDevice:= 'EXT_1', strPhysicalAddress:= '192.168.0.1/MyFolder', strOptions:='user=MyName,password=MyPass')

→ strFileName:='EXT_1:Datei.txt' (e.g. in the context of the READ_FILE_1 block) then addresses the "/Test/myFolder/Datei.txt" file on the external device (e.g. Windows PC) with the IP address "192.168.0.1".

The MyFolder folder on the external device must be enabled with the corresponding rights for the MyName user, with the MyPass password!

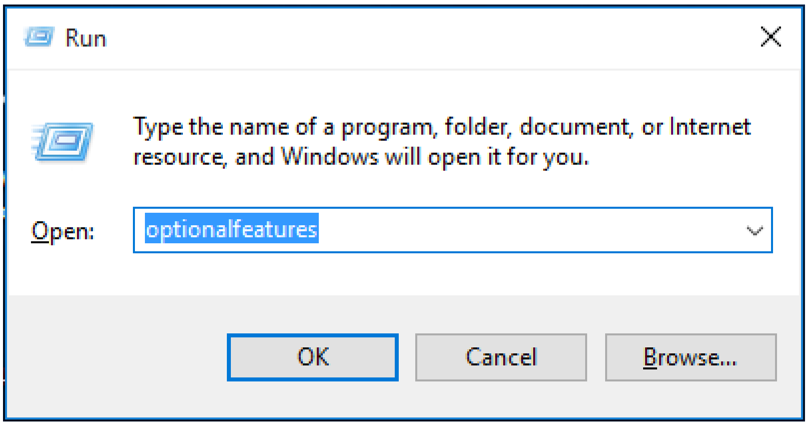
Install / activate the network protocol 'SMBv1' in Windows 10 Professional

Local administrator rights are required.

1. Open the 'Run' window.

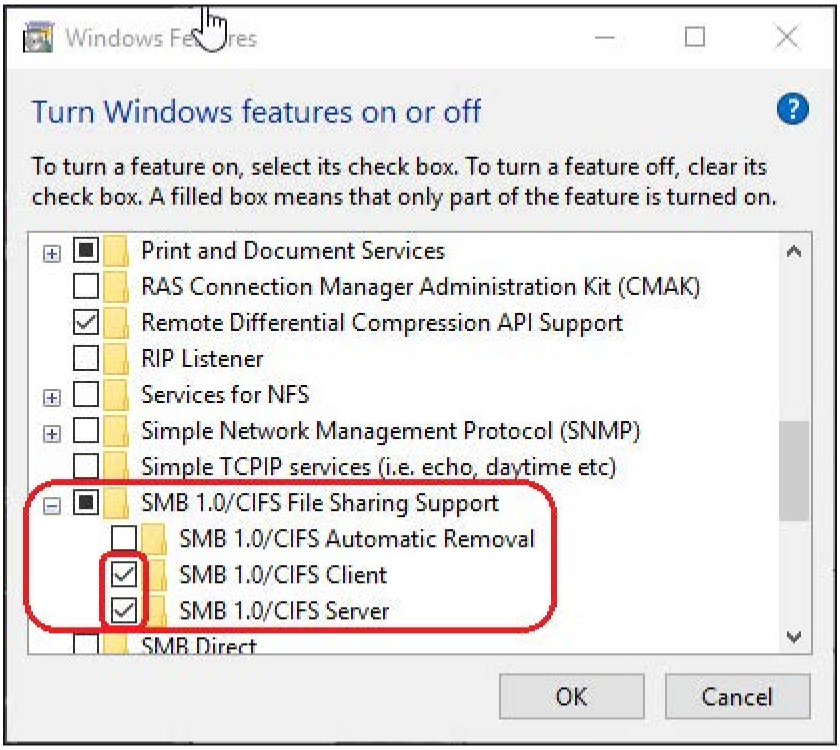
Press at the same time the keys <WINDOWS> and <R>.

2. Enter the command 'optionalfeatures' and confirm with the <OK> button.



3. In the folder 'SMB 1.0/CIFS File Sharing Support' you will find the required 'SMBv1' network protocols.

Activate SMB 1.0/CIFS client and SMB 1.0/CIFS server. Then confirm with the <OK> button.



The PC must be restarted after installation.

15.3.3 FiGetFileAttr (F)

The 'FiGetFileAttr' function enters the file attributes of a file in an 'ST_FILE_ATTR' type structure variable.

User interface

Input variables

Name	Type	Description
strFileName	STRING(64)	File name, including extension File name (optional: logical device + additional path)
pstFileAttr	POINTER	POINTER TO ST_FILE_ATTR Pointer to a structure variable in which the file attributes are entered.

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>FiGetFileAttr</td><td>INT</td><td>
<table>
<tr><td>Range</td><td>Meaning</td></tr>
<tr><td>0</td><td>No error (the structure variable referenced contains the file attributes).</td></tr>
<tr><td>-1</td><td>The enable is missing for one of the directories in the search path prefix</td></tr>
<tr><td>-2</td><td>There are too many symlinks</td></tr>
<tr><td>-4</td><td>The path string is incomplete or empty.</td></tr>
<tr><td>-5</td><td>Illegal memory access</td></tr>
<tr><td>-6</td><td>Part of the path string is not a directory</td></tr>
<tr><td>20</td><td>General system error</td></tr>
</table>
</td></tr>
</table>

15.3.4 ST_FILE_ATTR (ST)

The file attributes are saved in an 'ST_FILE_ATTR' type structure.

Structure elements

Name	Type	Description
strFileName	STRING (64)	File name, including extension The attributes are saved in the structure.

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>enFileType</td><td>ENUM</td><td>EN_FILE_TYP
File type
<table>
<tr><th>Range</th><th>Meaning</th></tr>
<tr><td>FILE_TYPE_REG</td><td>Regular file</td></tr>
<tr><td>FILE_TYPE_DIR</td><td>Directory</td></tr>
<tr><td>FILE_TYPE_CHR</td><td>Character device</td></tr>
<tr><td>FILE_TYPE_BLK</td><td>Block device</td></tr>
<tr><td>FILE_TYPE_FIFO</td><td>Named pipe</td></tr>
<tr><td>FILE_TYPE_LINK</td><td>Symbolic link</td></tr>
<tr><td>FILE_TYPE_SOCK</td><td>Socket</td></tr>
</table></td></tr>
<tr><td>wPermission</td><td>WORD</td><td>Access rights to file(byte-coded).
Where, e.g. 8#644 → '-,rw-,r--,r--'
where "-", rwx-Owner, rwx-Gruppe, rwx-Sonstige", and
where: r=read right, w=write right, x=execute right</td></tr>
<tr><td>udFileSize</td><td>UDINT</td><td>File length [bytes](corresponding to the return value of the 'SIZE_FILE_1' FB)</td></tr>
<tr><td>dtLastAccessTime</td><td>DATE_AND_TIME</td><td>Date and time of last access. (CODESYS "DT" type)</td></tr>
<tr><td>dtLastModificationTime</td><td>DATE_AND_TIME</td><td>Date and time of last change (CODESYS "DT" type)</td></tr>
<tr><td>dtLastStatChangeTime</td><td>DATE_AND_TIME</td><td>Date and time of last change of state (CODESYS "DT" type)</td></tr>
</table>

Structure definition

```
TYPE ST_FILE_ATTR:
    STRUCT
        strFileName: STRING(64);
        enFileType: EN_FILE_TYPE;
        wPermission: WORD;
        udFileSize: UDINT;
        dtLastAccessTime: DT;
        dtLastModificationTime: DT;
        dtLastStatChangeTime: DT;
    END_STRUCT
END_TYPE
```

16 AmkSockets - Ethernet socket functions specific to AMK

AmKSockets is an external communication library for application-programmable TCP/IP communication. It is divided into:

BasicFunctions	Basic functions
ConversionFunctions	Conversion functions
SupportFunctions	Support functions
TCPSpecific	Functions specific to TCP

16.1 General information about the term 'sockets'.

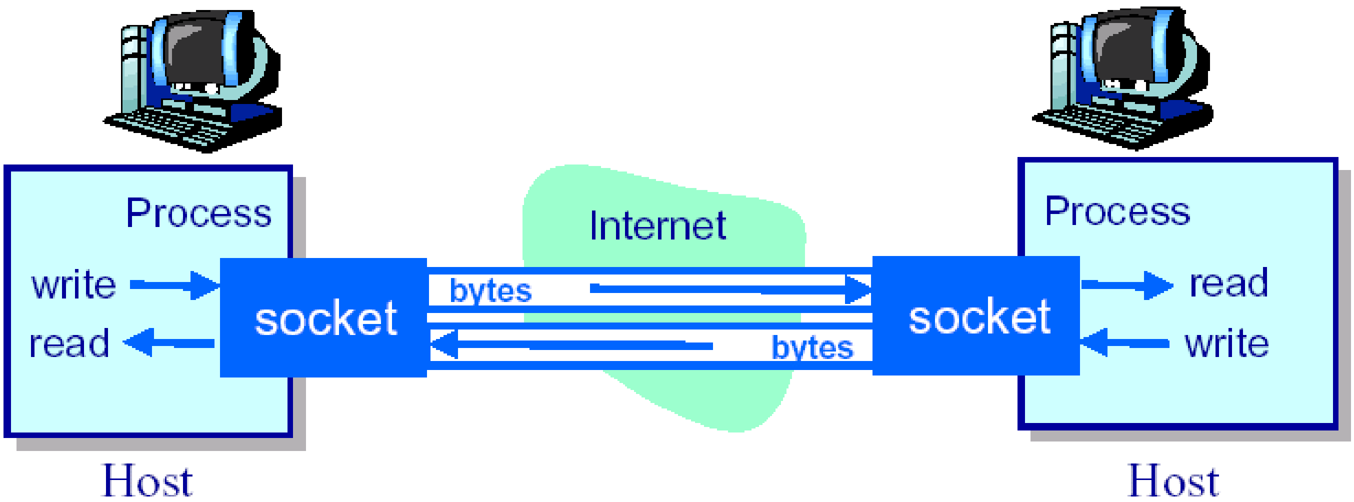
16.1.1 Introduction to 'sockets'

A socket is an established interface application. The interface is controlled by an operating system which can be used to send and receive messages.

A 'TCP' socket provides a reliable bidirectional communication route from one process to another. A 'TCP' socket should be used whenever a secure means of transferring multiple data items is required.

A 'UDP' socket provides a unreliable bidirectional communication route from one process to another. The 'UDP' socket is useful for sending broadcast messages (to multiple devices).

Figure illustrating how a socket works:



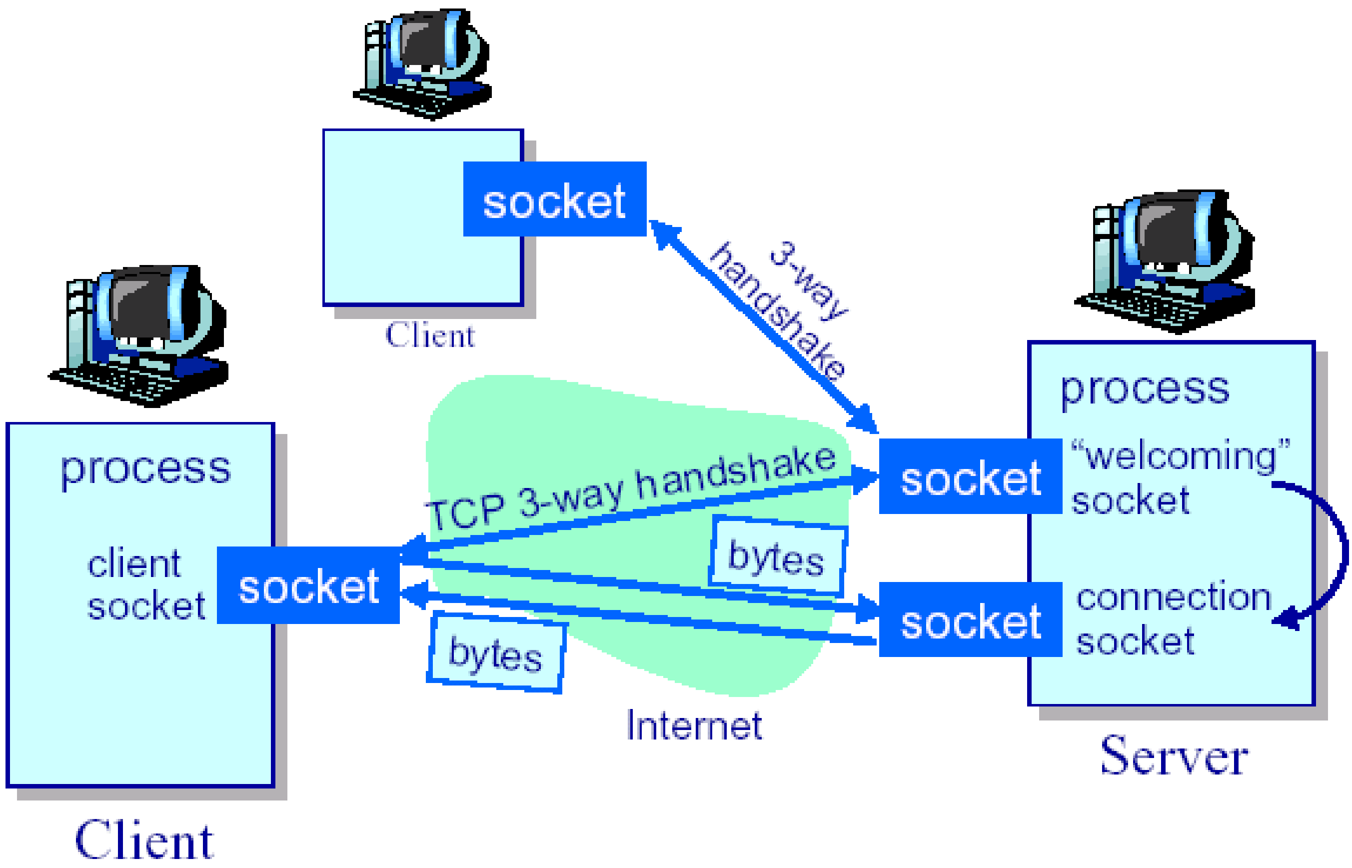
This figure shows that a socket-to-socket connection is similar to a cable into which data packets from the two connected applications are placed. The data packets are transferred automatically and received by the other application.

The data packets are received in the order in which they are placed into the cable. Each data packet is acknowledged by the receiver.

The sockets are addressed with an IP address and a port. On a network, each host has a unique IP address. The IP address consists of 4 bytes. It is usually written in the following format: e.g. 192.168.0.1

16.1.2 Client/server principle

Ordinarily, a client requests a connection to the server and is connected to another client. The approach differs in terms of what happens during the connection process. The 2 approaches are outlined in the text below.



The description below assumes that only non-blocking sockets are used. A blocking socket (default mode of a new socket when created) triggers the blocking of the executed function (e.g. no return) until execution is completed or an error occurs. Particular caution must be exercised when using this feature, otherwise the function will be blocked for a prolonged period. This may cause the watchdog time to be exceeded.

The blocking of a socket can be changed by using the 'FdiSockIoCtl' function. The function is described in this document.

The 'FboSockSelect' function can be used in conjunction with 'FboSockIsRead', 'FboSockIsWrite', and 'FboSockIsConnect' to determine the status of the non-blocking socket.

Server:

1. Create a special socket which receives the requirements of the incoming connection.
2. (Optional) Set the socket to blocking mode and also set the other initializations.
3. Connect the socket to the IP address and the specific port (well-known port). Well-known ports are defined and used by each port number for a number of well-known services (23 is the default for TELNET). It is advisable to use a number higher than 1500 for server applications, since it is less likely that the selected number will already have been assigned. To avoid conflicts in this case, the port must be changed. If more than one application is used to receive data in a single machine, the application for which the port is being used must be determined. The port splits an IP address into many logical subaddresses.
4. Call the function for the socket with the source of the incoming OS queue.
   The initialization can be selected for a specific point at any time. However, the connection to the client must be checked first. Otherwise, an attempt is made to re-establish the connection.
5. When the socket is prompted to receive a connection, the application can then accept the called function and connection. This can be refused or the newly connected socket can be returned. If there is more than one connection request, the function is called again after this sequence.
6. The connection is established. Data can be sent and received.
7. When the connection is no longer required, the sockets must be closed by the returned function in order to release the resources.
8. If the server application will not permit more connections, the listening socket must be closed.

Client:

1. Create the socket to be connected.
2. (Optional) Set the socket to blocking mode and also set the other initializations.

3. Call the connected function which is taking over the target parameters of the IP address and the corresponding port. A 3-way handshake with the server is initiated. Clients do not have to be added first, because the local end address (TCP port and IP address) is managed by the operating system.
4. The connection is established. Data can be sent and received. If this is not possible, the socket must be closed and the process must be repeated at a later point in time starting from step 1.
5. When the connection is no longer required, the socket must be closed in order to release the resources.

16.2 BasicFunctions

FboSockBind	Connects a specific port number and IP address to a socket
FboSockClose	Closes a connection
FboSockConnect	Sets up a connection to a server device
FboSockGetOption	Returns the current socket options
FboSockSelect	Performs a timeout function
FboSockSetOption	This function is used to select options for sockets
FboSockShutDown	Blocks send and/or receive based on parameter
FdiSockCreate	Creates a socket interface for the remaining socket functions
FdiSockIoCtl	This function is used to control the socket mode
FdiSockRecv	Function receives a defined data block from the connected device
FdiSockRecvFrom	Function receives a defined data block from the specified device
FdiSockSend	Sends data from the send buffer to the connected device
FdiSockSendTo	Sends data from the send buffer to the specified address

16.2.1 FboSockBind (F)

The 'FboSockBind' function connects a specific input port and an IP address to a socket that is not connected. The port is used for secure processing of the socket descriptor so that incoming packets can be received. If the 'FboSockConnect' function is used, this is not necessary. The server application calls 'FboSockBind' to take up a known port for clients. The 'FboSockBind' function is not actually able to establish a connection.

User interface

Input variables

Name	Type	Description
diSocket	DINT	Return value from 'FdiSockCreate'
pstSockAddr	POINTER	POINTER TO ST_SOCK_ADDR Pointer to the structure which contains the IP address and port number to be added. This structure must be created in the network byte sequence.

Output variables

Name	Type	Description
FboSockBind	BOOL	Return successful =TRUE. Return not successful =FALSE. In the event of an error, the 'FuiSockGetLastError' function must be called to obtain the error code.

If 'IPADDR_ANY' is used as an IP address, the packet are accepted by many interfaces. If port 0 is entered when the 'FboSockBind' is called, the system sets a specific port number.

To receive broadcast and multicast packets for 'UDP' and 'RAW', the 'IPADDR_ANY' socket must be linked to a broadcast address or a multicast address.

The 'FboSockBind' function does not have to be called if the 'FboSockConnect' or 'FdiSockSend' function has already been called. The connection is closed with any port and the local IP address connects when used with the same interface 'FboSockConnect' or 'FboSockSendTo'. 'FdiSockCreate' assigns a port at random. To receive a broadcast address or a multicast address, a connection to the local address must be established. So, to send all packet to one port, a connection to 'IPADDR_ANY' can be established.

16.2.2 FboSockClose (F)

For 'TCP': This function starts the process to close a connection. All sources involved in the connection are released when the connection is acknowledged and if all input data in the socket queue has been read.

For 'UDP' and 'RAW': The connection is lost immediately.

With 'TCP', the connection remains established if the handshake to close it has been initiated. Once the connection is no longer in place, the resources are released immediately.

If the connection is still in place, the sources are released as soon as the handshake closes and the input window is empty. If 'FboSockClose' is called to receive data, the socket for sending data is no longer valid.

The connection can be closed in 3 ways.

- Graceful close without timeout: At the start of the the closing process, the 'FboSockClose' function immediately returns all acknowledged data in the output window.
- Graceful close with timeout: The 'FboSockClose' applies a block until the output window has been emptied or a timeout occurs. In the event of a timeout, the data in the output window is rejected and a reset signal is subsequently sent back to the host. Once the output window has been emptied, the handshake to close the connection is initiated.
- Hard close: Some data which has not been detected in the output window is rejected. A reset signal is sent to the host and 'FboSockClose' is returned.

At the end of the graceful close without timeout, the received side takes over all data before the socket returns the resources. In the event of a hard close or a graceful close with timeout, there is no guarantee that all data sent will actually have been received.

User interface

Input variables

Name	Type	Description
diSocket	DINT	Return value from 'FdiSockCreate'

Output variables

Name	Type	Description
FboSockClose	BOOL	Return successful =TRUE ('TCP' closes but not a completed handshake. 'UDP' or 'RAW' is closed). Return not successful =FALSE. In the event of an error, the 'FuiSockGetLastError' function must be called to obtain the error code.

The 'FboSockClose' is used to close the socket. It is dependent on the 'SO_LINGER' option and lingers at the connected value set by the called 'FboSockSetOption' function.

If 'SO_LINGER' is set and the connected value is '0', 'FboSockClose' is hard-closed. If 'SO_LINGER' is set and the connected value is not 0, the write signal for the data is blocked by the 'FboSockClose' function until the output window is confirmed or a timeout occurs.

'FboSockClose' is hard-closed in the event of a timeout. If a timeout does not occur, all data is detected and 'FboSockClose' can close gracefully. If 'SO_LINGER' is not set, 'FboSockClose' is closed gracefully. The default setting is a graceful close without the value lingering

The source is then used up if the call to 'FboSockClose' fails.

16.2.3 FboSockConnect (F)

The 'FboSockConnect' function connects a socket to a server device. The IP address and the port must be known.

If the socket is not connected, a connection is established with any port and the local IP address.

User interface

Input variables

Name	Type	Description
diSocket	DINT	Return value from 'FdiSockCreate' is linked to 'FboSockBind'; it is defined when 'boSockListen' is called.
pstSockAddr	POINTER	POINTER TO ST_SOCK_ADDR Pointer to the structure which contains the IP address and port number to be added. This structure must be created in the network byte sequence.

Output variables

Name	Type	Description
FboSockConnect	BOOL	Return UDP =TRUE. TCP =TRUE is returned if the connection has been established successfully. In the event of an error, the 'FuiSockGetLastError' function must be called to obtain the error code.

For TCP: The 'FboSockConnect' function initiates the handshake. If the socket is blocked (Siehe 'FdiSockIoCtl (F)' auf Seite 481.), the established connection or an error message is blocked while the write signal is pending. Timeout detection is closed with the identical method and uses the same timeout values as 'FdiSockSend'.

If the socket is not blocked (Siehe 'FdiSockIoCtl (F)' auf Seite 481.), the 'FboSockConnect' returns an error after the handshake has been initialized. 'FboSockSelect' can be called after 'FboSockConnect' for the purpose of blocking (with timeout) as soon as the connection is established.

For 'UDP': The IP address and the port are linked to the socket and the function is returned immediately.

For 'RAW': The IP address is linked to the socket and the function is returned immediately. The port is not taken into account.

When in the blocked state, the function is blocked permanently. Use 'FboSockSelect' to apply a timeout.

The function sends to the broadcast or multicast addresses in sequence. Use 'FdiSockSendto' or 'FboSockConnect' with remote address to set the broadcast ('IPADDR_BROADCAST') or multicast address.

16.2.4 FboSockGetOption (F)

The 'FboSockGetOption' function returns the current values of the connected socket options. The option is specific with 'enSockOpt' and the value derived is returned in the 'stOptData'. The meaning of the value is determined by the option polled.

To support the options and the format of the option value. Siehe 'FboSockSetOption (F)' auf Seite 478.

User interface

Input variables

Name	Type	Description
diSocket	DINT	Return value from 'FdiSockCreate'
enSockOpt	ENUM	EN_SOCK_OPT Selection of options Siehe 'FboSockSetOption (F)' auf Seite 478.
pstSockOpt	POINTER	POINTER TO ST_OPT_DATA Pointer to the structure containing the data for the selected option Siehe 'Socket options' auf Seite 496.

Output variables

Name	Type	Description
FboSockGetOption	BOOL	Return successful =TRUE. Return not successful =FALSE. In the event of an error, the 'FuiSockGetLastError' function must be called to obtain the error code.

The 'FboSockGetOption' is not able to output values for the 'IP_ADD_MEMBERSHIP' and 'IP_DROP_MEMBERSHIP' options.

16.2.5 FboSockSelect (F)

The 'FboSockSelect' function returns a timeout function for 'FboSockConnect', 'FdiSockAccept', 'FdiSockRecv', 'FdiSockRecvFrom', 'FdiSockSend', and 'FdiSockSendTo'. If the connected socket is in block mode or, in the event of immediate feedback, is not in block mode, each of these function blocks is permanent. Siehe 'FdiSockIoCtl (F)' auf Seite 481.

The connected socket must not be set to blocking before 'FboSockConnect', 'FdiSockAccept', 'FdiSockRecv', or 'FdiSockSend' is called. This is to ensure that the function is not blocked.

User interface

Input variables

Name	Type	Description
pstReadList	POINTER	POINTER TO ST_SOCK_LIST Ready to read. The socket list is checked by the optional pointer.
pstWriteList	POINTER	POINTER TO ST_SOCK_LIST Ready to write. The socket list is checked by the optional pointer.
pstExceptList	POINTER	POINTER TO ST_SOCK_LIST In the event of error messages, the socket list is checked by the optional pointer. (RTIP 3.0 is not supported.) If there are differences then =0. Return error from 'FboSockSelect'
tTimeOut	TIME	Timeout

Output variables

Name	Type	Description
FboSockSelect	BOOL	'FboSockSelect' =TRUE if the sockets are not blocked in the input list following reading or writing or if no sockets are ready (timeout) 'FboSockSelect' =FALSE: In the event of an error, the 'FuiSockGetLastError' function must be called to obtain the error code.

'FboSockSelect' blocks the selected signal until either timeout occurs or the selected criteria from the defined list are met by at least one socket.

Under CoDeSys, the 'tTimeOut' variable is initialized with time values not equal to 0 when 'FboSockSelect' is called. The software watchdog triggers in an exceptional case and terminates the program if the program sequence times out.

Therefore, it is better to set the time value to 0.

If 'pstReadList' is not 0, 'FboSockSelect' is returned. The status is illegal if it contains data from the 'UDP' or 'TCP' socket or reading the socket (TCP) fails.

For 'TCP': Sockets are the receivers, 'FboSockSelect' is returned when a connection is established and 'FdiSockAccept' is successful.

If a 'UDP' socket is specified in 'pstWriteList' and 'pstWriteList' is not 0, 'FboSockSelect' is returned immediately. It is returned if there is a blocking 'TCP' socket in the output window of 'pstWriteList'.

The function is returned if there is a blocking TCP socket with an empty output window in 'pstWriteList' or if a socket is in an illegal state in write mode.

For non-blocking TCP sockets, 'FboSockSelect' is returned immediately if the connection has been established or an error occurs.

A non-blocking TCP socket is considered by 'FboSockConnect' in the context of a completed connection process if the write operation (e.g. 'FdiSockSend' or 'FdiSockSendTo') has been completed. If the write operation has not been completed by 'FboSockSelect' , the socket is not returned.

The 3 modified directories only contain the sockets which meet the criteria.

To be determined if the returned sockets are ready to read/write or the sockets for calling 'FboSockIsRead', 'FboSockIsWrite', and 'FboSockIsConnect' are connected. FboSockSelect is only determined if the socket is blocking. It is returned if an error is pending at the socket.

The setting of the 'tTimeOut' variable determines how long the 'FboSockSelect' function is blocked for. If the timeout value = 0, 'FboSockSelect' is not locked. This function supports send calls to multiple sockets.

'FboSockSelect' changes the directory of the sockets in line with the variable. It deletes the sockets from the directory which does not meet the criteria. The following functions are available to generate the adapted directories of 'FboSockSelect', 'FboSockListSet', 'FboSockListClr', 'FboSockListIsSet', and 'FboSockListZero'.

The maximum number of sockets entered in the socket directory SOCK_LIST_SIZE_MAX + 1 (64)

16.2.6 FboSockSetOption (F)

The 'FboSockSetOption' is used to set the connected options for the socket. The 'enSockOpt' parameter contains the setting and 'stOptData' contains the information for setting the option

User interface

Input variables

Name	Type	Description
diSocket	DINT	Return value from 'FdiSockCreate'

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>enSockOpt</td><td>ENUM</td><td>EN_SOCK_OPT
Selection of options
<table>
<tr><th>Range</th><th>Meaning</th></tr>
<tr><td>SO_NAGLE</td><td>NAGLE algorithm activated/deactivated</td></tr>
<tr><td>SO_REUSEADDR</td><td>Enables the local address to be reused</td></tr>
<tr><td>SO_KEEPALIVE</td><td>Retain connection</td></tr>
<tr><td>SO_MAX_UDP_QUE</td><td>Maximum number of UDP input packets in the buffer</td></tr>
<tr><td>SO_UDPCKSUM_IN</td><td>Check UDP input packets (checksum)</td></tr>
<tr><td>SO_UDPCKSUM_OUT</td><td>Generate UDP output packets (checksum)</td></tr>
<tr><td>SO_LINGER</td><td>Linger in write mode if data is available</td></tr>
<tr><td>SO_TCP_NO_COPY</td><td>TCP is in packet mode rather than window mode</td></tr>
<tr><td>SO_REUSESOCK</td><td>Enables the socket to be reused in TWAIT status. Without this option, the reusable socket is closed for 120 seconds following initialization and has responded to ACK and FIN. (Closing of the socket has been initialized and accepted.) If the system proceeds without the socket and the 'SO_REUSEOCK' socket option is set, the socket can be released and reassigned during the period (120 s).</td></tr>
<tr><td>SO_DELAYED_ACK</td><td>Acknowledge send delay</td></tr>
<tr><td>SO_IP_TTL</td><td>IP TTL (time to live)</td></tr>
<tr><td>SO_SELECT_SIZE</td><td>TCP write selection: wake up when</td></tr>
<tr><td>SO_TCP_TIMESTAMP</td><td>TCP timestamp option</td></tr>
<tr><td>SO_802_2</td><td>802.2 socket</td></tr>
<tr><td>SO_TOS</td><td>TOS data length: value in IP overview</td></tr>
<tr><td>IP_MULTICAST_IF</td><td>Set/retain IP multicast interface</td></tr>
<tr><td>IP_MULTICAST_TTL</td><td>Set/retain IP multicast TTL</td></tr>
<tr><td>IP_MULTICAST_LOOP</td><td>Set/retain IP multicast feedback</td></tr>
<tr><td>IP_ADD_MEMBERSHIP</td><td>Connection to multicast group</td></tr>
<tr><td>IP_DROP_MEMBERSHIP</td><td>Leave multicast group</td></tr>
</table></td></tr>
<tr><td>pstSockOpt</td><td>POINTER</td><td>POINTER TO ST_OPT_DATA
Pointer to the structure containing the data for the selected option
Siehe 'Socket options' auf Seite 496.</td></tr>
</table>

Output variables

Name	Type	Description
FboSockSetOption	BOOL	Return successful =TRUE. Return not successful =FALSE. In the event of an error, the 'FuiSockGetLastError' function must be called to obtain the error code.

16.2.7 FboSockShutDown (F)

The 'FboSockShutDown' function depends on the value of the 'enSthdMode' variable. This function activates or deactivates sending and/or receiving at a socket.

User interface

Input variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>diSocket</td><td>DINT</td><td>Return value from 'FdiSockCreate'</td></tr>
<tr><td>enShtdMode</td><td>ENUM</td><td>EN_SHTD_MODE
Definition of block in operation
<table>
<tr><th>Range</th><th>Meaning</th></tr>
<tr><td>DISABLE_ RECEIVES</td><td>Packets cannot be received</td></tr>
<tr><td>DISABLE_ SENDS</td><td>Packets cannot be sent</td></tr>
<tr><td>DISABLE_BOTH</td><td>Packets cannot be sent or received</td></tr>
</table></td></tr>
</table>

Output variables

Name	Type	Description
FboSockShutDown	BOOL	Return successful =TRUE. Return not successful =FALSE. In the event of an error, the 'FuiSockGetLastError' function must be called to obtain the error code.

For 'TCP': Feedback is deactivated, incoming data is accepted until the input window is full but the data is not acknowledged. A FIN is set when sending is deactivated.

For 'UDP' and 'RAW': Receiving is deactivated, incoming packets are queued for 'UDP' exchange. The resources are not released until after 'FboSockClose' is complete.

16.2.8 FdiSockCreate (F)

The 'FdiSockCreate' function creates a socket interface for the remaining socket functions and API functions. The number of available sockets is defined in the operating system.

User interface

Input variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>enFamily</td><td>ENUM</td><td>EN_ADDR_FAMILY
Address format (family)
<table>
<tr><td>Range</td><td>enFamily is a member of the 'ST_SOCK_ADDR' structure</td></tr>
</table></td></tr>
</table>

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>enSockType</td><td>ENUM</td><td>EN_SOCK_TYPE
Socket type
<table>
<tr><th>Range</th><th>Meaning</th></tr>
<tr><td>SOCK_STREAM</td><td>Stream socket</td></tr>
<tr><td>SOCK_DGRAM</td><td>Diagram socket</td></tr>
<tr><td>SOCK_RAW</td><td>RAW protocol interface</td></tr>
<tr><td>SOCK_RDM</td><td>Reliable message transfer</td></tr>
<tr><td>SOCK_ SEQPACKET</td><td>Consecutive packets (stream)</td></tr>
</table></td></tr>
<tr><td>enProtType</td><td>ENUM</td><td>EN_PROT_TYPE
Protocol type
<table>
<tr><th>Range</th><th>Meaning</th></tr>
<tr><td>IPPROTO_TCP</td><td>TCP</td></tr>
<tr><td>IPPROTO_IP</td><td>Placeholder for IP</td></tr>
<tr><td>IPPROTO_UDP</td><td>UDP</td></tr>
</table></td></tr>
</table>

Output variables

Name	Type	Description
FdiSockCreate	DINT	SOCK_INVALID (-1) is returned if more sockets are available or a socket is greater than or equal to 0. In the event of an error, the 'FuiSockGetLastError' function must be called to obtain the error code.

If the socket is in use in its entirety, 'FboSockClose' can be called in order to make multiple sockets available with other function calls 'FdiSockCreate'. The 'TCP' sockets cannot be used until 'FboSockConnect', 'FboSockListen', 'FboSockBind', or 'FdiSockAccept' have been called.

16.2.9 FdiSockIoCtl (F)

This function controls the socket mode. The command is executed with the 'enCmd' variable.

User interface

Input variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>diSocket</td><td>DINT</td><td>Return value from 'FdiSockCreate'</td></tr>
<tr><td>enCmd</td><td>ENUM</td><td>EN_SOCK_CMD
Command execution
<table>
<tr><th>Range</th><th>Meaning</th></tr>
<tr><td>FIONBIO</td><td>Set without block</td></tr>
<tr><td>FIONREAD</td><td>Number of characters to be read and available</td></tr>
<tr><td>FIONWRITE</td><td>Number of characters to be written and available (AS-PL12 and AS-PL14 target systems only).</td></tr>
</table></td></tr>
<tr><td>boCmdVal</td><td>BOOL</td><td>Commanding value: Definition of the switching options ON or OFF for commanding of 'FIONBIO'. It is of no significance for the other commandings.</td></tr>
</table>

Output variables

Name	Type	Description
FdiSockIoCtl	DINT	The available characters to be read or written are returned. The return value is depending upon the 'FIONREAD' or 'FIONWRITE' commanding. If the 'FIONBIO' commanding is set, the return value =0. If the function could not be executed successfully, 'SOCK_INVALID (-1)' is returned. In the event of an error, the 'FuiSockGetLastError' function must be called to obtain the error code.

'FIONBIO' commanding activates or deactivates non-blocking mode. If the value of variable 'boCmdVal =TRUE, the socket is set to non-blocking. Otherwise, it is set to the default mode (blocking).

'FIONREAD' commanding returns the number of available read bytes.

For 'TCP': The maximum number of bytes returned in the input window.

For 'UDP': The maximum number of bytes queued in the first 'UDP' socket ready to be returned.

16.2.10 FdiSockRecv (F)

The 'FdiSockRecv' function receives a defined data block from the connected device and copies the data to a user buffer. It is no longer returned as the 'uiSize' bytes. Feedback is sent as soon as the bytes are available in the buffer.

For 'UDP' and 'RAW': The 'FboSockConnect' must be called before 'FdiSockRecv'.

For 'TCP': The block must be connected before 'FdiSockRecv' is called.

If no data is available and the socket is in blocking mode, the read signal of the 'FdiSockRecv' function is blocked until data is available. If no data is available and the socket is not in blocking mode, the 'FdiSockRecv' function is returned immediately with an error.

If data is available, it can be determined that 'FboSockSelect' is called before 'FdiSockRecv'.

For 'TCP': The function returns a 0 if an end of the data packet has been received (e.g. the remote host outputs 'FboSockClose') and the input window is empty.

If the 'RAW' data packet matches the protocol range of the socket, the packet is added to the queue. The block protocol is a parameter of 'FdiSockCreate'. The data written in the buffer contains the IP header.

User interface

Input variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>diSocket</td><td>DINT</td><td>Return value from 'FdiSockCreate'</td></tr>
<tr><td>enFlags</td><td>ENUM</td><td>EN_CTRL_FLAGS
Control flags
<table>
<tr><th>Range</th><th>Meaning</th></tr>
<tr><td>MSG_OOB</td><td>Process out-of-band data</td></tr>
<tr><td>MSG_PEEK</td><td>View incoming message</td></tr>
<tr><td>MSG_ DONTROUTE</td><td>Send without using routing table</td></tr>
<tr><td>MSG_EOR</td><td>All data accepted</td></tr>
<tr><td>MSG_TRUNC</td><td>Data deleted prior to transfer</td></tr>
<tr><td>MSG_CTRUNC</td><td>Control data lost prior to transfer</td></tr>
<tr><td>MSG_WAITALL</td><td>Wait for complete query or error</td></tr>
<tr><td>MSG_QUEUE</td><td>Buffer cannot send, in queue (target systems AS-PL12 and AS-PL14 only)</td></tr>
</table></td></tr>
</table>

Name	Type	Description
uiSize	UINT	Maximum data length available to accommodate the information to be read. uiSize ≤ SIZEOF(variable) referenced by 'pbyData'!
pbyBuffer	POINTER	POINTER TO BYTE Pointer variable referencing the receive buffer in which the data received is written

Output variables

Name	Type	Description
FdiSockRecv	DINT	The number of bytes located in the buffer is returned. At the end of the process, 'FdiSockRecv' =0 (TCP only) or 'SOCK_INVALID (-1) if an error has occurred. In the event of an error, the 'FuiSockGetLastError' function must be called to obtain the error code.

In the blocked state, the function is permanently blocked. Use 'FboSockSelect' to apply a timeout.

In the receive sequence of broadcast or multicast addresses the local addresses are connected.

'IPADDR_ANY' can also be integrated to receive all sent packets.

For UDP: A packet is removed from the input queue every time 'FdiSockRecv' is called. If the packet contains several data items, the call is retrieved and the remaining data is lost.

The function comes to an end once the other side is closed (e.g. sends a FIN) but 'FboSockClose' has not been called and the input window is empty. (TCP only)

16.2.11 FdiSockRecvFrom (F)

The 'FdiSockRecvFrom' function receives a defined data block from the specified device and copies the data to a user buffer. It is no longer returned as the 'uiSize' bytes. Feedback is sent as soon as the bytes are available in the buffer.

The IP address and the port of the end point are returned to the caller in the 'stSockAddr' structure.

For 'UDP': The 'FboSockConnect' function can (but does not have to) be called before 'FdiSockRecvFrom'.

For 'TCP': The block must be connected before 'FdiSockRecvFrom' is called.

If no data is available and the block is in blocking mode, the read signal of the 'FdiSockRecvFrom' function is blocked until data is available.

If no data is available and the block is not in blocking mode, the 'FdiSockRecv' function is returned immediately with an error.

If data is available, it can be determined that 'FboSockSelect' is called before 'FdiSockRecvFrom'.

For 'TCP': The function returns a 0 if an end of the data packet has been received (e.g. the remote host outputs 'FboSockClose') and the input window is empty. The data written in the buffer contains the IP header.

If the 'RAW' data packet matches the protocol range of the block, the packet is added to the queue. The block protocol is a parameter of 'FdiSockCreate'. The data written in the buffer contains the IP header.

User interface

Input variables

Name	Type	Description
diSocket	DINT	Return value from 'FdiSockCreate'
pstSockAddr	POINTER	POINTER TO ST_SOCK_ADDR Pointer to the structure which contains the IP address and port number to be added. This structure must be created in the network byte sequence.

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>enFlags</td><td>ENUM</td><td>EN_CTRL_FLAGS
Control flags
<table>
<tr><th>Range</th><th>Meaning</th></tr>
<tr><td>MSG_OOB</td><td>Process out-of-band data</td></tr>
<tr><td>MSG_PEEK</td><td>View incoming message</td></tr>
<tr><td>MSG_ DONTROUTE</td><td>Send without using routing table</td></tr>
<tr><td>MSG_EOR</td><td>All data accepted</td></tr>
<tr><td>MSG_TRUNC</td><td>Data deleted prior to transfer</td></tr>
<tr><td>MSG_CTRUNC</td><td>Control data lost prior to transfer</td></tr>
<tr><td>MSG_WAITALL</td><td>Wait for complete query or error</td></tr>
<tr><td>MSG_QUEUE</td><td>Buffer cannot send, in queue (target systems AS-PL12 and AS-PL14 only)</td></tr>
</table></td></tr>
<tr><td>uiSize</td><td>UINT</td><td>Maximum data length available to accommodate the information to be read.
uiSize ≤ SIZEOF(variable) referenced by 'pbyData'!</td></tr>
<tr><td>pbyBuffer</td><td>POINTER</td><td>POINTER TO BYTE
Pointer variable referencing the receive buffer in which the data received is written</td></tr>
</table>

Output variables

Name	Type	Description
FdiSockRecvFrom	DINT	The number of bytes located in the buffer is returned. At the end of the process, 'FdiSockRecv' =0 (TCP only) or 'SOCK_INVALID (-1) if an error has occurred. In the event of an error, the 'FuiSockGetLastError' function must be called to obtain the error code.

In the blocked state, the function is permanently blocked. Use 'FboSockSelect' to apply a timeout.

For UDP: A packet is removed from the input queue every time 'FdiSockRecvFrom' is called. If the packet contains several data items, the call is retrieved and the remaining data is lost.

The function comes to an end once the other side is closed (e.g. sends a FIN) and the input window is empty.

16.2.12 FdiSockSend (F)

The 'FdisockSend' functions sends data from the send buffer to the connected device. The socket must be connected.

If the socket is in blocking mode, it is reset once all data has been sent to and acknowledged by the remote host.

If the socket is not blocked, the data is in the output window queue and sends to the remote host for as long as possible. If there is no space available in the output window, the function is returned with an error.

User interface

Input variables

Name	Type	Description
diSocket	DINT	Return value from 'FdiSockCreate'

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>enFlags</td><td>ENUM</td><td>EN_CTRL_FLAGS
Control flags
<table>
<tr><td>Range</td><td>Meaning</td></tr>
<tr><td>MSG_OOB</td><td>Process out-of-band data</td></tr>
<tr><td>MSG_PEEK</td><td>View incoming message</td></tr>
<tr><td>MSG_DONTROUTE</td><td>Send without using routing table</td></tr>
<tr><td>MSG_EOR</td><td>All data accepted</td></tr>
<tr><td>MSG_TRUNC</td><td>Data deleted prior to transfer</td></tr>
<tr><td>MSG_CTRUNC</td><td>Control data lost prior to transfer</td></tr>
<tr><td>MSG_WAITALL</td><td>Wait for complete query or error</td></tr>
<tr><td>MSG_QUEUE</td><td>Buffer cannot send, in queue (target systems AS-PL12 and AS-PL14 only)</td></tr>
</table></td></tr>
<tr><td>uiSize</td><td>UINT</td><td>Maximum data length of the information to be written.
uiSize ≤ SIZEOF(variable) referenced by 'pbyData'!</td></tr>
<tr><td>pbyBuffer</td><td>POINTER</td><td>POINTER TO BYTE
Pointer variable referencing the send buffer which contains the data to be sent</td></tr>
</table>

Output variables

Name	Type	Description
FdiSockSend	DINT	The bytes to be sent located in the buffer queue are returned. Otherwise, SOCK_INVALID (-1) is returned. In the event of an error, the 'FuiSockGetLastError' function must be called to obtain the error code.

If the socket is not in blocking mode, the 'FboSockSelect' function is called in order to reserve space in the output window. The 'FdiSockIoCtl' function is called to determine the number of free bytes in the output window.

For 'TCP': In the event of a timeout, the port is closed and the function is returned SOCK_INVALID (-1).

The application must call the 'FboSockClose' function for free resources.

For 'UDP' and 'TCP': If the data size exceeds the maximum size of a packet, only one packet is sent. The data cannot be split or segmented.

If segmentation is permitted, the packet is segmented and sent. 'FdiSockSend' is not locked if the block is not in blocking mode. 'FdiSockSend' is locked until the packet is sent. 'FdiSockSend' returns SOCK_INVALID (-1) in the event of a failure at an ARP timeout or if the device fails during sending.

For 'TCP' and 'UDP': Successful completion of the 'FdiSockSend' is no guarantee that data has actually been sent.

For 'TCP' not in blocking mode: 'FdiSockSend' has completed successfully if notification is received that all data is in the output window queue. Call 'FboSockSelect' to receive notification of sent and confirmed data.

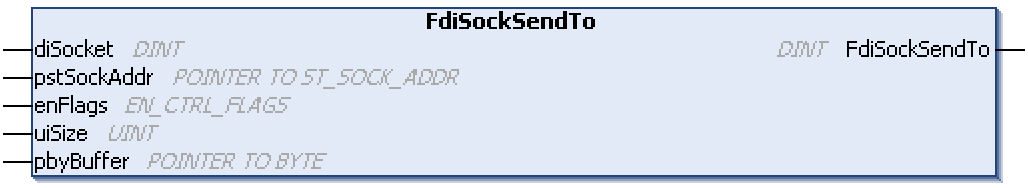
For 'TCP' in blocking mode: Successful completion of the send operation indicates that the remote host has received all data.

In blocking mode: The function blocks use permanently. Use 'FboSockSelect' to apply a timeout.

16.2.13 FdiSockSendTo (F)

This function sends data from the send buffer to a specific address.

User interface



Input variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>diSocket</td><td>DINT</td><td>Return value from 'FdiSockCreate'</td></tr>
<tr><td>pstSockAddr</td><td>POINTER</td><td>POINTER TO ST_SOCK_ADDR
Pointer to the structure which contains the IP address and port number to be added. This structure must be created in the network byte sequence.</td></tr>
<tr><td>enFlags</td><td>ENUM</td><td>EN_CTRL_FLAGS
Control flags
<table>
<tr><th>Range</th><th>Meaning</th></tr>
<tr><td>MSG_OOB</td><td>Process out-of-band data</td></tr>
<tr><td>MSG_PEEK</td><td>View incoming message</td></tr>
<tr><td>MSG_DONTROUTE</td><td>Send without using routing table</td></tr>
<tr><td>MSG_EOR</td><td>All data accepted</td></tr>
<tr><td>MSG_TRUNC</td><td>Data deleted prior to transfer</td></tr>
<tr><td>MSG_CTRUNC</td><td>Control data lost prior to transfer</td></tr>
<tr><td>MSG_WAITALL</td><td>Wait for complete query or error</td></tr>
<tr><td>MSG_QUEUE</td><td>Buffer cannot send, in queue (target systems AS-PL12 and AS-PL14 only)</td></tr>
</table></td></tr>
<tr><td>uiSize</td><td>UINT</td><td>Maximum data length of the information to be written.
uiSize ≤ SIZEOF(variable) referenced by 'pbyData'!</td></tr>
<tr><td>pbyBuffer</td><td>POINTER</td><td>POINTER TO BYTE
Pointer variable referencing the send buffer which contains the data to be sent</td></tr>
</table>

Output variables

Name	Type	Description
FdiSockSendTo	DINT	The bytes to be sent located in the buffer queue are returned. Otherwise, SOCK_INVALID (-1) is returned. In the event of an error, the 'FuiSockGetLastError' function must be called to obtain the error code.

For 'TCP': The socket must be connected. The 'stSockAddr' structure is not taken into account.

For 'UDP': The socket can be connected. If the socket is not connected, a connection is established with a unique port and the local IP address of the interface. The data is sent via the IP address.

For 'UDP' and 'RAW': Connected or not connected, the IP address and the port (not taken into account for 'RAW') of the end point must be sent to the 'stSockAddr' structure. If 'FdiSockSendTo' is called by 'IPADDR_BROADCAST', the packet is transferred to all interfaces. The interfaces with loop are an exception to this rule.

Use 'FdiSockSendto' or 'FboSockConnect' with remote addresses to set the broadcast ('IPADDR_BROADCAST') or multicast addresses

For more information: Siehe 'FdiSockSend (F)' auf Seite 484.

16.3 ConversionFunctions

FdwSockHtonl	Converts a double word from host byte order format (Little Endian) into network byte order format (Big Endian)
FdwSockInetAddr	Converts an IP address from decimal format with decimal points (e.g. 192.168.2.2) into double word format
FdwSockNtohl	Converts a double word from network byte order format (Big Endian) into host byte order format (Little Endian)
FstrSockInetNtoa	Converts an IP address from ST_INADDR format into decimal format with decimal points (e.g. 192.168.2.2)

FwSockHtons	Converts a word from host byte order format (Little Endian) into network byte order format (Big Endian)
FwSockNtohs	Converts a word from network byte order format (Big Endian) into host byte order format (Little Endian)

16.3.1 FdwSockHtonl (F)

Converts a double word from host byte order format (Little Endian) into network byte order format (Big Endian).

User interface

Input variables

Name	Type	Description
dwHost	DWORD	Double word in host byte order (32 bits)

Output variables

Name	Type	Description
FdwSockHtonl	DWORD	Conversion of input parameters into network byte order format

16.3.2 FdwSockInetAddr (F)

Converts an IP address from decimal format with decimal points (e.g. 192.168.2.2) into double word format.

User interface

Input variables

Name	Type	Description
strIP	STRING	STRING(SOCK_IP_STR_MAX) String containing the IP address (dotted decimal notation) of the node with which a connection is to be established. If this input is not assigned, the controller can connect to a communication partner with any IP address Default: ‘192.168.0.1’

Output variables

Name	Type	Description
FdwSockInetAddr	DWORD	Return of IP address as double word. 'IPADDR_NONE' is returned in the event of an error.

16.3.3 FdwSockNtohl (F)

Converts a double word from network byte order format (Big Endian) into host byte order format (Little Endian).

User interface

Input variables

Name	Type	Description
dwNet	DWORD	Double word in network byte order (32 bits)

Output variables

Name	Type	Description
FdwSockNtohl	DWORD	Conversion of input parameters into host byte order format.

16.3.4 FstrSockInetNtoa (F)

Converts an IP address from 'ST_INADDR' format into decimal format with decimal points (e.g. 192.168.2.2).
The string is returned when the entire length of the IP address is reached (at least 15 characters).

User interface

Input variables

Name	Type	Description
dwIP	DWORD	IP address in network byte sequence

Output variables

Name	Type	Description
FstrSockInetNtoa	STRING	Returns the IP address from the STRING format in decimal format with decimal points.

16.3.5 FwSockHtons (F)

Converts a word from host byte order format (Little Endian) into network byte order format (Big Endian).

User interface

Input variables

Name	Type	Description
wHost	WORD	Word in host byte order (16 bits).

Output variables

Name	Type	Description
FwSockHtons	WORD	Conversion of input parameters into network byte order format

16.3.6 FwSockNtohs (F)

Converts a word from network byte order format (Big Endian) into host byte order format (Little Endian).

User interface

Input variables

Name	Type	Description
wNet	WORD	Word in network byte order.

Output variables

Name	Type	Description
FwSockNtohs	WORD	Conversion of input parameters into host byte order format.

16.4 SupportFunctions

FboSockIsConnect	Checks if the socket is connected
FboSockIsRead	Checks if permissible data can be read by a socket
FboSockIsWrite	Checks if data can be written to a socket
FboSockListClr	Utility function for 'FboSockSelect'. Deletes a socket from a list in the ST_SOCKET_FD_SET structure
FboSockListIsSet	Utility function for 'FboSockSelect'. This function can be used to determine if a socket appears in a list in the ST_SOCKET_FD_SET structure
FboSockListSet	Utility function for 'FboSockSelect'. Adds a socket to a list in the ST_SOCKET_FD_SET structure
FboSockListZero	Utility function for 'FboSockSelect'. Deletes a list in the ST_SOCKET_FD_SET structure
FdwSockGetOwnIP	Reads out the IP address of the device
FuiSockGetLastError	Returns the last error message

16.4.1 FboSockIsConnect (F)

Checks if the socket is connected. This function can be used if, after the identified 'FboSockSelect' has been called, a socket is available and complete once the identified connection has been made.

'FboSockSelect' identifies the subsequent operation of the socket and applies a block.

User interface

Input variables

Name	Type	Description
diSocket	DINT	Return value from 'FdiSockCreate'

Output variables

Name	Type	Description
FboSockIsConnect	BOOL	Return =TRUE if the socket is a connected TCP socket. Return =FALSE if the socket is not a connected TCP socket.

16.4.2 FboSockIsRead (F)

Checks if permissible data can be read by a socket. This function cannot be used until after the 'FboSockSelect' function has been called. 'FboSockSelect' must be set to read in order to activate the 'FboSockIsRead' socket.

User interface

Input variables

Name	Type	Description
diSocket	DINT	Return value from 'FdiSockCreate'

Output variables

Name	Type	Description
FboSockIsRead	BOOL	Return =TRUE if the socket is a TCP socket and legal data is being read from the socket. Return =FALSE if the socket is not a TCP socket and illegal data is being read from the socket.

16.4.3 FboSockIsWrite (F)

Checks if data can be written to a socket. This function can be used once the identified 'FboSockSelect' function has been called if the socket is ready to write (writing is not blocked). This function cannot be used until after the 'FboSockSelect' function has been called. 'FboSockSelect' must be set to write in order to activate the 'FboSockIsWrite' function.

User interface

Input variables

Name	Type	Description
diSocket	DINT	Return value from 'FdiSockCreate'

Output variables

Name	Type	Description
FboSockIsWrite	BOOL	Return =TRUE if the socket is a TCP socket and legal data is being written to the socket. Return =FALSE if the socket is not a TCP socket and illegal data is being written to the socket.

16.4.4 FboSockListClr (F)

The 'FboSysSockListClr' function deletes a socket from a list in the 'ST_SOCK_LIST' structure

User interface

Input variables

Name	Type	Description
diSocket	DINT	Return value from 'FdiSockCreate'

Name	Type	Description
pstList	POINTER	POINTER TO ST_SOCK_LIST Pointer to a structure which contains a list of sockets. Siehe 'ST_SOCK_LIST' auf Seite 495.

Output variables

Name	Type	Description
FboSockListClr	BOOL	Return always =TRUE

16.4.5 FboSockListIsSet (F)

The 'FboSockListIsSet' function identifies whether a socket appears in a list in the 'ST_SOCK_LIST' structure.

User interface

Input variables

Name	Type	Description
diSocket	DINT	Return value from 'FdiSockCreate'
pstList	POINTER	POINTER TO ST_SOCK_LIST Pointer to a structure which contains a list of sockets. Siehe 'ST_SOCK_LIST' auf Seite 495.

Output variables

Name	Type	Description
FboSockListIsSet	BOOL	Return always =TRUE

16.4.6 FboSockListSet (F)

The 'FboSockListSet' function adds a socket to a list in the 'ST_SOCK_LIST' structure.

User interface

Input variables

Name	Type	Description
diSocket	DINT	Return value from 'FdiSockCreate'
pstList	POINTER	POINTER TO ST_SOCK_LIST Pointer to a structure which contains a list of sockets. Siehe 'ST_SOCK_LIST' auf Seite 495.

Output variables

Name	Type	Description
FboSockListSet	BOOL	Return always =TRUE

16.4.7 FboSockListZero (F)

The 'FboSockListZero' function deletes a list in the 'ST_SOCK_LIST' structure.

User interface

Input variables

Name	Type	Description
pstList	POINTER	POINTER TO ST_SOCK_LIST Pointer to a structure which contains a list of sockets. Siehe 'ST_SOCK_LIST' auf Seite 495.

Output variables

Name	Type	Description
FboSockListZero	BOOL	Return from 'FboSockListZero' always =TRUE

16.4.8 FdwSockGetOwnIP (F)

The 'FdwSockGetOwnIP' function calls the IP address of the device when the program is executed.

User interface

Input variables

Name	Type	Description
diInterface	DINT	Interface instance

Output variables

Name	Type	Description
FdwSockGetOwnIP	DWORD	Return of IP address as a double word

16.4.9 FuiSockGetLastError (F)

The 'FuiSockGetLastError' function returns the last error message generated by a function from the socket library. Ordinarily, the function is called by other functions which return FALSE or SOCK_INVALID.

There are two functions for returning errors. The 'FuiSockGetLastError' function is called after the second function; it returns the last error that occurred.

If the 'FuiSockGetLastError' function is called after the first error occurs, it always returns the valid error code. However, this does not mean that an error is pending there. Call the 'FuiSockGetLastError' function block only if some functions are returned as FALSE or SOCK_INVALID.

User interface

Input variables

Name	Type	Description
diDummy	DINT	Parameter is not used. CoDeSys requires a function to take over at least one parameter.

Output variables

Name	Type	Description
FuiSockGetLastError	UINT	Return of an error code. If no errors are pending, 'FuiSockGetLastError' returns a value of 0. Siehe 'Error numbers' auf Seite 497.

16.5 TCPSpecific

FboSockListen	Initializes a server device to monitor connection requests
FdiSockAccept	Confirms a connection request from a host device

16.5.1 FboSockListen (F)

Initializes a server device to monitor connection requests.

The 'FboSockListen' function identifies backlog in the buffer content of a socket. Once the function has been called, the server application calls the 'FdiSockAccept' function to identify a client session and generates a new task for the session. The function called permits the client connection backlog and sends a request to the server to process the content of the buffer. Without 'FboSockListen', the called client connection cannot be established.

If 'uiMaxConnect' is set to an invalid value (higher than defined for OS), the highest valid value is set.

This function is only valid for TCP

User interface

Input variables

Name	Type	Description
diSocket	DINT	Return value from 'FdiSockCreate'
uiMaxConnect	UINT	Maximum number of dependent permissible connections Range: 0...10

Output variables

Name	Type	Description
FboSockListen	BOOL	Return successful =TRUE. Return not successful =FALSE. In the event of an error, the 'FuiSockGetLastError' function must be called to obtain the error code.

16.5.2 FdiSockAccept (F)

Confirms a connection request from a host device

The 'FdiSockAccept' function is a specific TCP function. It accepts a connection call from a remote host. When establishing a connection via a remote host, 'FboSockBind' and 'FboSockListen' must be called before the 'FdiSockAccept' function. The return of the function is accepted immediately by a virtual socket. If a connection is not accepted and the socket is in blocking mode, the 'FdiSockAccept' will block the connection until it is accepted or an error occurs. Siehe 'FdiSockIoCtl (F)' auf Seite 481. If a connection is not accepted and the socket is not in blocking mode, the function returns an error message immediately.

User interface

Input variables

Name	Type	Description
diSocket	DINT	Return value from 'FdiSockCreate' is linked to 'FboSockBind'; it is defined when 'boSockListen' is called.
pstSockAddr	POINTER	POINTER TO ST_SOCK_ADDR Pointer to the structure which contains the IP address and port number to be added. This structure must be created in the network byte sequence.

Output variables

Name	Type	Description
FdiSockAccept	DINT	If successful, a socket assigned to the connection is returned. Not successful, then SOCK_INVALID (-1). In the event of an error, the 'FuiSockGetLastError' function must be called to obtain the error code.

If an established connection exists, the defined function 'FboSockSelect' can be called. The 'FdiSockAccept' returns a new socket to use the connection; the original socket remains assigned. This socket is required for closing if no more original sockets are accepted and closed. The socket returned is closed by the acceptance if it is no longer required.

The function is permanently blocked. The 'FboSockSelect' function is used to apply a timeout.

16.6 DataTypes

16.6.1 ST_OPT_DATA

An 'ST_OPT_DATA' type structure is used as a parameter for the 'FboSockSetOption' and 'FboSockGetOption' functions. The structure contains the information about the socket options. The Socket options table describes how the corresponding values must be entered in the structure based on the option supported.

Siehe 'Socket options' auf Seite 496.

Structure elements

Name	Type	Description
dwVal1	DWORD	Option value 1 - specific option
dwVal2	DWORD	Option value 2 - specific option

Structure definition

```
TYPE ST_OPT_DATA:
    STRUCT
        dwVal1:DWORD;
        dwVal2:DWORD;
    END_STRUCT
END_TYPE
```

16.6.2 ST_SOCK_ADDR

The 'ST_SOCK_ADDR' type structure contains complete address information for the socket

Structure elements

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>enFamily</td><td>ENUM</td><td>EN_ADDR_FAMILY
Address format (family)
<table><tr><td>Default</td><td>AF_INET</td></tr></table>
<table>
<tr><td>Range</td><td>Meaning</td></tr>
<tr><td>AF_UNSPEC</td><td>Undefined protocol</td></tr>
<tr><td>AF_LOCAL</td><td>local to host</td></tr>
<tr><td>AF_INET</td><td>Internet transfer: UDP, TCP, etc.</td></tr>
<tr><td>AF_IMPLINK</td><td>ARPANET IMP - protocol</td></tr>
<tr><td>AF_PUP</td><td>PUP - protocol, e.g. BSP</td></tr>
<tr><td>AF_CHAOS</td><td>MIT CHAOS - protocol</td></tr>
<tr><td>AF_NS</td><td>Xerox NS - protocol</td></tr>
<tr><td>AF_ISO</td><td>ISO - protocol</td></tr>
<tr><td>AF_ECMA</td><td>ECMA - protocol (European Computer Manufacturers Association)</td></tr>
<tr><td>AF_DATAKIT</td><td>DataKit - protocol</td></tr>
<tr><td>AF_CCITT</td><td>CCITT - protocol, e.g. X.25</td></tr>
<tr><td>AF_SNA</td><td>IBM SNA - protocol</td></tr>
<tr><td>AF_DEC_NET</td><td>DECNet - protocol</td></tr>
<tr><td>AF_DLI</td><td>Direct Data Link - protocol</td></tr>
<tr><td>AF_LAT</td><td>LAT - protocol</td></tr>
<tr><td>AF_HYLINK</td><td>NSC HyperChannel - protocol</td></tr>
<tr><td>AF_APPLETALK</td><td>Apple Talk - protocol</td></tr>
<tr><td>AF_ROUTE</td><td>Internal routing - protocol</td></tr>
<tr><td>AF_LINK</td><td>Link Layer Interface</td></tr>
<tr><td>AF_PSEUDO_XTP</td><td>eXpress Transfer - protocol (not AF)</td></tr>
<tr><td>AF_INET6</td><td>IP - protocol version 6: UDP, TCP, etc.</td></tr>
</table></td></tr>
<tr><td>wPort</td><td>WORD</td><td>Port of the node with which the connection is being established
<table><tr><td>Default</td><td>1500</td></tr></table></td></tr>
<tr><td>dwIP</td><td>DWORD</td><td>IP address in network byte sequence
<table><tr><td>Default</td><td>192.168.0.1</td></tr></table></td></tr>
</table>

Structure definition

```
TYPE ST_SOCK_LIST:
	STRUCT
		enFamily:EN_ADDR_FAMILY;
		wPort:WORD;
		dwIP:DWORD;
	END_STRUCT
END_TYPE
```

Some specific functions are defined such as 'IPADDR_ANY' (receive all data) and 'IPADDR_BROADCAST' (send to broadcast address). The function can be used through the IP address in the 'dwIP' variable.

16.6.3 ST_SOCK_LIST

An 'ST_SOCK_LIST' type structure is defined to support socket directories. These directories are used by functions such as 'FboSockSelect', 'FboSockClr', 'FboSockIsSet', 'FboSockListZero', and 'FboSockSet'.

Structure elements

Name	Type	Description
udCount	UDINT	Number of elements in 'diSocket'
diSocket	DINT	ARRAY [0..SOCK_LIST_SIZE_MAX] OF DINT An ARRAY containing sockets. The maximum value of SOCK_LIST_SIZE_MAX is set to 63

Structure definition

```
TYPE ST_SOCK_LIST:
    STRUCT
        udCount:UDINT;
        diSocket:ARRAY [0..SOCK_LIST_SIZE_MAX] OF DINT;
    END_STRUCT
END_TYPE
```

16.7 Appendix

16.7.1 Socket options

All supporting options and their corresponding value from the 'ST_OPT_DATA' are listed in the table below.

Option	Description	Values in 'stOptData'	Default
SO_KEEPALIVE	Monitoring of the TCP KeepAlive packets to be sent	dwVal1: 0 = off; not equal to 0 = on	off
SO_LINGER	Activate/deactivate lingering of closed TCP	dwVal1: 0 = off; not equal to 0 = on dwVal2: linger time in seconds	off
SO_MAX_UDP_QUE	Maximum number of incoming UDP packets in the socket queue at the same time. Some additional packets are rejected	dwVal1: greater than 0 = maximum size of queue; less than 0 = off; 0 = not permitted	No limit
SO_NAGLE	Prevention of sending of small TCP packets despite the presence of some outstanding output data that has not yet been confirmed	dwVal1: 0 = off; not equal to 0 = on	on
SO_DELAYED_ACK	Acknowledging delay of up to 200 ms for sending of TCP. In a stream with full-size packets, every other packet is detected.	dwVal1: 0 = off; not equal to 0 = on	on
SO_REUSESOCK	Ability to reuse a socket if no other sockets are available in wait mode.	dwVal1: 0 = off; not equal to 0 = on	off
SO_TCP_NO_COPY	Monitoring of the copying of TCP input and output data directly to input and output packets	dwVal1: 0 = off; not equal to 0 = on	Copy
SO_TCP_ TIMESTAMP	Send timestamp in original SYNC	dwVal1: 0 = off; not equal to 0 = on	off
SO_SELECT_SIZE	Wake-up call if the specified space is available in the TCP window	dwVal1: 0 = off; not equal to 0 = on	off
SO_UDPCKSUM_IN	Checksum check of input packets	dwVal1: 0 = off; not equal to 0 = on	on
SO_UDPCKSUM_ OUT	Checksum check of output packets	dwVal1: 0 = off; not equal to 0 = on	on
SO_IP_TTL	Set IP (time-to-live) for outgoing packets	dwVal1: 0 – 255 = time-to-live value	60
SO_REUSEADDR	Allows the socket to use the same address, e.g. IP address and port number which is already being used by another socket with the same protocol type.	dwVal1: 0 = off; not equal to 0 = on	off
SO_BROADCAST	Approval to send broadcast messages.	dwVal1: 0 = off; not equal to 0 = on	off

Option	Description	Values in 'stOptData'	Default
SO_SNDBUF	Set maximum send buffer size in bytes	dwVal1: 2048 – minimum size; 104448 maximum size	104448
SO_RCVBUF	Set maximum receive buffer size in bytes	dwVal1: 2048 – minimum size; 104448 maximum size	104448
SO_RCVLOWAT	Set the minimum number of received bytes until the data has been forwarded to the receiver.	dwVal1: Greater than 0	1
IP_ADD_ MEMBERSHIP	Join a multicast group	dwVal1: Multicast address to join group dwVal2: Local IP address for joining	None
IP_DROP_ MEMBERSHIP	Leave a multicast group	dwVal1: Multicast address for leaving the group dwVal2: Local IP address to join	None
IP_MULTICAST_IF	Specification of a default interface for outgoing multicast packets	dwVal1: IP address of the interface when using a default multicast interface	off
IP_MULTICAST_ LOOP	Return loop monitoring for multicast data packets	dwVal1: 0 = off; 1 = on	on
IP_MULTICAST_TTL	Set IP (time-to-live) for outgoing multicast packets	dwVal1: 0 – 255 = time-to-live value	1
SO_802_2	Packets shall be sent to this socket as 802.2 packets	dwVal1: 0 = off; not equal to 0 = on	off
SO_TOS	Write value in a ToS (type of service) field of the IP header for outgoing packets	dwVal1: 0 = off; not equal to 0 = on	0

16.7.2 Error numbers

The error numbers are return values of the 'FuiSockGetLastError' function for the target systems AS-PL12 and AS-PL14.

Error number	Description
101	End point of the address is not available
102	Address is already in use
103	Family is not supported
104	ARP table full
105	Invalid baud rate
106	Invalid communication transport
107	Invalid device type
108	Invalid interface number
109	Invalid mask
110	Invalid ping reaction
111	End point of connection rejected
112	Target address is required
113	Target cannot be accessed (ICMP)
114	Invalid parameter (pointer is 0)
115	Interface closed
116	Interface table full
117	Opening of interface failed
118	Establishing connection (mode: non-blocking)
118	Non-blocking socket, but the function is blocked
119	Invalid function call (parameter)
120	Socket is already connected
121	Multicast table full
122	Multicast address not found

Error number	Description
123	Outside ports
124	Network failure (sending failed)
125	Network cannot be accessed (KeepAlive failed)
126	Outside DCUs (packets)
127	Parameter option is invalid
128	Socket is not connected
129	RTIP is not initialized
130	Invalid socket descriptor
131	Not enough devices
132	Socket type or specific operation is not supported for this function
133	Sending failed due to list being full.
134	Unable to identify device
135	Error, no reentrancy
136	Routing table input not found
137	Routing table full
138	Resource initialization failed
139	Illegal operation as a result of socket deactivation
140	Timeout
141	Type not supported (only 'SOCKET_STREAM' and 'SOCKET_DGRAM' are supported)
142	Sending to ARP is necessary. However, ARP is not available
143	Not enough local process memory to assign request
144	Table full
145	Invalid packet size
146	Opening of block failed

16.7.3 Error numbers

The error numbers are return values of the 'FuiSockGetLastError' function for the target systems AS-PL15 and AS-Cxx.

Error number	Description
1	1. An attempt has been made to establish a connection to a broadcast address without the broadcast flag being enabled. 2. The connection attempt failed due to the local firewall settings
4	The function was interrupted by a signal.
5	Input / output error occurred during read or write access to the file system.
9	The socket input variable is invalid.
11	The socket is non-blocking but the required operation is blocked. FdiSockAccept: The socket is non-blocking and no accepted connections are available. FdiSockSend, FdiSockSendTo: The non-blocking socket does not have sufficient memory capacity for the send request. FdiSockRecv, FdiSockRecvFrom: No data is available when a non-blocking socket is read.
12	There is not enough memory capacity available to complete the request.
13	Write access to the socket has been refused or permission to search one of the directories on the path has been refused.
14	The socket address structure is invalid.

Error number	Description
22	FboSockSetOption: The specified option is invalid for the specified socket level or the socket is closed. FboSockGetOption: The specified option is invalid for the specified socket level FboSockBind: The socket is connected to an address and the protocol does not permit a connection to a different address, or the socket is closed. FboSockConnect: Invalid address structure in the 'ST_SOCK_ADDR' family. FboSockListen: The socket is already connected or closed. FdiSockAccept: The socket will not accept connections. FboSockSelect: An invalid timeout time has been specified FdiSockRecv, FdiSockRecvFrom: The MSG_OOB flag has been set and there is no 'out of band' data. FboSockShutDown: The 'EN_SHTD_MODE' argument is invalid. FdiSockIoCtl: The request or the argument is invalid for the device.
23	The maximum number of data is already open
24	No more data is available for this process
32	The local end has been shut down for a connection-based socket.
33	The timeout times for send and receive operations are too long and do not fit into the timeout time fields in the socket structure.
88	The input variable does not reference a socket.
89	FboSockListen: The socket is not connected to a local address and the protocol does not support listening to an unconnected socket. FdiSockSend, FdiSockSendTo: The socket is not in connection mode and neither a participant address nor a target address has been specified. FdiSockSend, FdiSockSendTo: The socket is not in connection mode and a participant address has not been entered.
92	The option is not supported by the protocol.
93	The protocol is not supported by the address family or is not implemented.
95	FdiSockSend, FdiSockSendTo: Some bits in the flag are not compatible with this socket type.
97	The specified address is not valid for this address family at the specified socket
98	The specified address is already in use.
101	There is no connection to the network.
103	A connection has been aborted.
105	Not enough resources are available to complete the operation. The operation cannot be executed due to the absence of a connection to the system.
106	The socket is already connected The option cannot be executed while the socket is connected.
107	FdiSockRecv, FdiSockRecvFrom: An attempt has been made to receive data at a socket in connection mode but the socket is not connected. FdiSockSend, FdiSockSendTo: The socket is not connected or a participant address has not been specified. FboSockShutDown: The socket is not connected.
110	FboSockConnect: The connection attempt failed at the timeout time before a connection could be established. FdiSockRecv, FdiSockRecvFrom: The connection was lost whilst being established due to a timeout or a transfer time timeout.
111	The target address is not responding to a connection attempt or rejects the connection request.
114	A connection request is already in progress for the specified socket.
115	The socket is not blocking and the connection cannot be established immediately. It would be better to establish the connection asynchronously.

17 AmkTcp - Communication interface specific to AMK

The function blocks in the 'AmkTCP' communication library provide the user with an easy way of sending and receiving via TCP. Accordingly, the user does not need to have in-depth background knowledge of network transfer. Only basic knowledge of IP addresses and the use of port numbers is required.

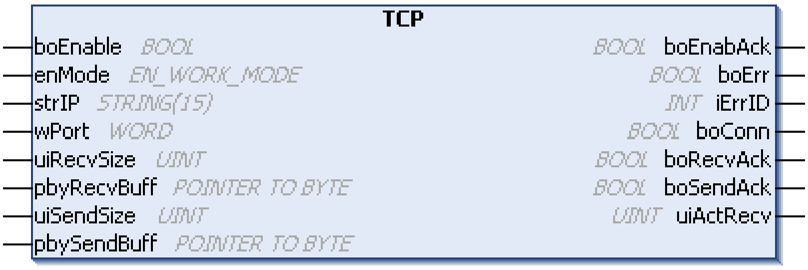
Like the 'TCP_1' function block, the 'TCP' function block should normally only be used to transfer data packets smaller than 1448 bytes. If the data packets are larger, either a header with length information must be included with the transfer or the 'TCP_2' function block must be used.

TCP	TCP communication interface specific to AMK
TCP_1	TCP communication interface specific to AMK
TCP_2	TCP communication interface specific to AMK

17.1 POUs

17.1.1 TCP (FB)

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.
enMode	ENUM	EN_WORK_MODE Selection mode for establishing communication Default: AUTO_ AUTO_: Automatic mode selection, where the node with the higher IP address becomes ACTIVE and the node with the lower IP address becomes PASSIVE. Suitable for a connection between two controllers ACTIVE_: In this mode, the function block attempts to establish a connection with the specified node. In the context of client/server connections, this mode is suitable for the client PASSIVE_: In this mode, the function block waits for a connection request from another node. In the context of client/server connections, this mode is suitable for the server. In this mode, the IP address can be "empty"
strIP	STRING (15)	String containing the IP address (dotted decimal notation) of the node with which a connection is to be established. If this input is not assigned, the controller can connect to a communication partner with any IP address Default: ‘192.168.0.1’

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>wPort</td><td>WORD</td><td>Port of the node with which the connection is being established<table><tr><td>Default</td><td>1500</td></tr></table></td></tr>
<tr><td>uiRecvSize</td><td>UINT</td><td>Size of the receive buffer</td></tr>
<tr><td>pbyRecvBuff</td><td>POINTER</td><td>POINTER TO BYTE
Pointer variable referencing the receive buffer in which the data received is written</td></tr>
<tr><td>uiSendSize</td><td>UINT</td><td>Size of the send buffer</td></tr>
<tr><td>pbySendBuff</td><td>POINTER</td><td>POINTER TO BYTE
Pointer variable referencing the send buffer which contains the data to be sent</td></tr>
</table>

Output variables

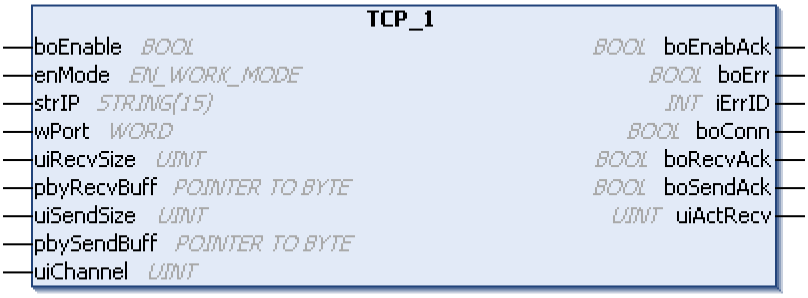
<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state<table><tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr><tr><td>TRUE</td><td>Error</td></tr></table></td></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output<table><tr><td colspan="2">iErrID = 0</td><td>No error</td></tr><tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr><tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr></table>Error<table><tr><th>Range</th><th>Meaning</th></tr><tr><td>0</td><td>No error</td></tr><tr><td>1 - 999</td><td>Error numbers are described in the library documentation for AmkSockets.lib</td></tr></table>For AmkTCP.lib up to Version 02.04 2008/25, the following also applies:<table><tr><th>Range</th><th>Meaning</th></tr><tr><td>101 -</td><td>Illegal pointer to the receive buffer</td></tr><tr><td>102 -</td><td>Illegal pointer to the send buffer</td></tr><tr><td>103 -</td><td>Incorrect communication mode</td></tr></table>For AmkTCP.lib as of Version > 02.04 2008/25, the following also applies:<table><tr><th>Range</th><th>Meaning</th></tr><tr><td>1001 -</td><td>Illegal pointer to the receive buffer</td></tr><tr><td>1002 -</td><td>Illegal pointer to the send buffer</td></tr><tr><td>1003 -</td><td>Incorrect communication mode</td></tr><tr><td>1004</td><td>Incorrect instance 'uiChannel' or invalid IP address</td></tr><tr><td>1005</td><td>Read/write operation interrupted, connection closed</td></tr><tr><td>1006</td><td>Internal error</td></tr></table></td></tr>
<tr><td>boConn</td><td>BOOL</td><td>Signal for the communication status. TRUE means that a connection to the specified communication partner exists.</td></tr>
<tr><td>boRecvAck</td><td>BOOL</td><td>Signal indicating that data has been received successfully. The signal is set if the data has been received in full in the buffer and is reset, if the 'uiRecvSize' input is set to “0”.</td></tr>
<tr><td>boSendAck</td><td>BOOL</td><td>Signal indicating that data has been sent successfully. The signal is set if all of the data has been sent from the buffer and is reset, if the 'uiSendSize' input is set to “0”.</td></tr>
</table>

Name	Type	Description
uiActRecv	UINT	Length of data currently received

17.1.2 TCP_1 (FB)

The 'TCP_1' function block is based on the 'TCP' function block. The difference is the addition of the 'uiChannel' input signal. The communication interface (communication instance) can be selected with this signal. Therefore, if there is more than one Ethernet interface, this function block must be used instead of the 'TCP' function block

User interface



Input variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnable</td><td>BOOL</td><td>Enable signal: With a positive edge, the initialisation of the block starts.
As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC.
In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.</td></tr>
<tr><td>enMode</td><td>ENUM</td><td>EN_WORK_MODE
Selection mode for establishing communication
<table><tr><td>Default</td><td>AUTO</td></tr></table>
<table>
<tr><th>Range</th><th>Meaning</th></tr>
<tr><td>AUTO_</td><td>Automatic mode selection, where the node with the higher IP address becomes ACTIVE and the node with the lower IP address becomes PASSIVE. Suitable for a connection between two controllers</td></tr>
<tr><td>ACTIVE_</td><td>In this mode, the function block attempts to establish a connection with the specified node. In the context of client/server connections, this mode is suitable for the client</td></tr>
<tr><td>PASSIVE_</td><td>In this mode, the function block waits for a connection request from another node. In the context of client/server connections, this mode is suitable for the server. In this mode, the IP address can be "empty"</td></tr>
</table></td></tr>
<tr><td>strIP</td><td>STRING (15)</td><td>String containing the IP address (dotted decimal notation) of the node with which a connection is to be established. If this input is not assigned, the controller can connect to a communication partner with any IP address
<table><tr><td>Default</td><td>‘192.168.0.1’</td></tr></table></td></tr>
<tr><td>wPort</td><td>WORD</td><td>Port of the node with which the connection is being established
<table><tr><td>Default</td><td>1500</td></tr></table></td></tr>
<tr><td>uiRecvSize</td><td>UINT</td><td>Size of the receive buffer</td></tr>
<tr><td>pbyRecvBuff</td><td>POINTER</td><td>POINTER TO BYTE
Pointer variable referencing the receive buffer in which the data received is written</td></tr>
</table>

Name	Type	Description
uiSendSize	UINT	Size of the send buffer
pbySendBuff	POINTER	POINTER TO BYTE Pointer variable referencing the send buffer which contains the data to be sent
uiChannel	UINT	Selection of communication instance Default: 4

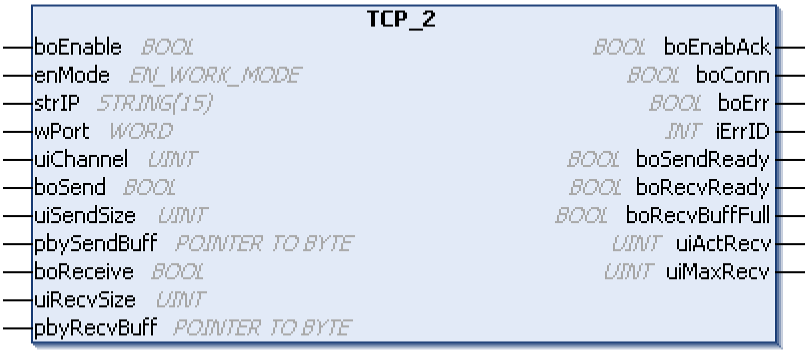
Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table></td></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error
<table>
<tr><th>Range</th><th>Meaning</th></tr>
<tr><td>0</td><td>No error</td></tr>
<tr><td>1 - 999</td><td>Error numbers are described in the library documentation for AmkSockets.lib</td></tr>
</table>
For AmkTCP.lib up to Version 02.04 2008/25, the following also applies:
<table>
<tr><th>Range</th><th>Meaning</th></tr>
<tr><td>101 -</td><td>Illegal pointer to the receive buffer</td></tr>
<tr><td>102 -</td><td>Illegal pointer to the send buffer</td></tr>
<tr><td>103 -</td><td>Incorrect communication mode</td></tr>
</table>
For AmkTCP.lib as of Version > 02.04 2008/25, the following also applies:
<table>
<tr><th>Range</th><th>Meaning</th></tr>
<tr><td>1001 -</td><td>Illegal pointer to the receive buffer</td></tr>
<tr><td>1002 -</td><td>Illegal pointer to the send buffer</td></tr>
<tr><td>1003 -</td><td>Incorrect communication mode</td></tr>
<tr><td>1004</td><td>Incorrect instance 'uiChannel' or invalid IP address</td></tr>
<tr><td>1005</td><td>Read/write operation interrupted, connection closed</td></tr>
<tr><td>1006</td><td>Internal error</td></tr>
</table></td></tr>
<tr><td>boConn</td><td>BOOL</td><td>Signal for the communication status. TRUE means that a connection to the specified communication partner exists.</td></tr>
<tr><td>boRecvAck</td><td>BOOL</td><td>Signal indicating that data has been received successfully. The signal is set if the data has been received in full in the buffer and is reset, if the 'uiRecvSize' input is set to “0”.</td></tr>
<tr><td>boSendAck</td><td>BOOL</td><td>Signal indicating that data has been sent successfully. The signal is set if all of the data has been sent from the buffer and is reset, if the 'uiSendSize' input is set to “0”.</td></tr>
<tr><td>uiActRecv</td><td>UINT</td><td>Length of data currently received</td></tr>
</table>

17.1.3 TCP_2 (FB)

Like the 'TCP_1' function block, the 'TCP_2' function block can be used to select the communication interface (communication instance). Furthermore, data packets of any size can be sent easily.

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.
enMode	ENUM	EN_WORK_MODE Selection mode for establishing communication Default: AUTO_ AUTO_: Automatic mode selection, where the node with the higher IP address becomes ACTIVE and the node with the lower IP address becomes PASSIVE. Suitable for a connection between two controllers ACTIVE_: In this mode, the function block attempts to establish a connection with the specified node. In the context of client/server connections, this mode is suitable for the client PASSIVE_: In this mode, the function block waits for a connection request from another node. In the context of client/server connections, this mode is suitable for the server. In this mode, the IP address can be "empty"
strIP	STRING (15)	String containing the IP address (dotted decimal notation) of the node with which a connection is to be established. If this input is not assigned, the controller can connect to a communication partner with any IP address Default: ‘192.168.0.1’
wPort	WORD	Port of the node with which the connection is being established Default: 1500
uiChannel	UINT	Selection of communication instance Default: 4
boSend	BOOL	Sends the number of 'uiSendSize' data located in the send buffer 'pbySendBuff'
uiSendSize	UINT	Size of the send buffer
pbySendBuff	POINTER	POINTER TO BYTE Pointer variable referencing the send buffer which contains the data to be sent

Name	Type	Description
boReceive	BOOL	Receive data up to the maximum size set in 'uiRecvSize' for writing to receive buffer 'pbyRecvBuff'
uiRecvSize	UINT	Size of the receive buffer
pbyRecvBuff	POINTER	POINTER TO BYTE Pointer variable referencing the receive buffer in which the data received is written

Output variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boEnabAck</td><td>BOOL</td><td>Acknowledgement: Function block is initialised and enabled</td></tr>
<tr><td>boConn</td><td>BOOL</td><td>Signal for the communication status. TRUE means that a connection to the specified communication partner exists.</td></tr>
<tr><td>boErr</td><td>BOOL</td><td>The function block is in an error state
<table>
<tr><td>FALSE</td><td>No error (permitted commanding or warning)</td></tr>
<tr><td>TRUE</td><td>Error</td></tr>
</table></td></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error
<table>
<tr><th>Range</th><th>Meaning</th></tr>
<tr><td>0</td><td>No error</td></tr>
<tr><td>1 - 999</td><td>Error numbers are described in the library documentation for AmkSockets.lib</td></tr>
</table>
For AmkTCP.lib up to Version 02.04 2008/25, the following also applies:
<table>
<tr><th>Range</th><th>Meaning</th></tr>
<tr><td>101 -</td><td>Illegal pointer to the receive buffer</td></tr>
<tr><td>102 -</td><td>Illegal pointer to the send buffer</td></tr>
<tr><td>103 -</td><td>Incorrect communication mode</td></tr>
<tr><td>104 -</td><td>Incorrect instance (uiChannel) or invalid IP address</td></tr>
<tr><td>105 -</td><td>Read/write operation interrupted, connection closed</td></tr>
<tr><td>106 -</td><td>Internal error</td></tr>
</table>
For AmkTCP.lib as of Version > 02.04 2008/25, the following also applies:
<table>
<tr><th>Range</th><th>Meaning</th></tr>
<tr><td>1001 -</td><td>Illegal pointer to the receive buffer</td></tr>
<tr><td>1002 -</td><td>Illegal pointer to the send buffer</td></tr>
<tr><td>1003 -</td><td>Incorrect communication mode</td></tr>
<tr><td>1004 -</td><td>Incorrect instance (uiChannel) or invalid IP address</td></tr>
<tr><td>1005 -</td><td>Read/write operation interrupted, connection closed</td></tr>
<tr><td>1006 -</td><td>Internal error</td></tr>
</table></td></tr>
<tr><td>boSendReady</td><td>BOOL</td><td>Signal indicating that data has been sent successfully. The signal is TRUE if all of the data has been sent from the send buffer. 'boSendReady' changes to FALSE when 'boSend' is set to FALSE</td></tr>
<tr><td>boRecvReady</td><td>BOOL</td><td>Signal indicating that data has been received successfully. The signal changes to TRUE if all data has been received. 'boRecvReady' changes to FALSE when 'boReceive' is set to FALSE</td></tr>
</table>

Name	Type	Description
boRecvBuffFull	BOOL	The signal changes to TRUE if the quantity of data received is greater than the quantity specified in 'uiRecvSize'
uiActRecv	UINT	Length of data currently received
uiMaxRecv	UINT	Maximum size of the data packet expected

17.2 Application

The TCP function blocks are for data exchange between two network nodes via TCP (peer-to-peer connection). An additional instance of this function block is required for each additional connection to another network node.

To start the function block, 'boEnable' must be set from FALSE to TRUE. Before starting the function block, the 'strIP', 'wPort', 'enMode', 'pbyRecvBuff', and 'pbySendBuff' inputs must be assigned the corresponding values.

The 'enMode' input can be declared with the following predefined values:

- AUTO (default value) – this setting allows the function block to decide automatically whether to establish a connection or to wait for an incoming connection request. This communication mode is suitable for a connection between two CoDeSys applications. Once the function block has started, it compares the dedicated IP address of the controller with the IP address with which a connection is to be established 'strIP'. If the dedicated IP address is smaller, the controller is set to PASSIVE mode and waits for an incoming connection request. Otherwise it is ACTIVE and attempts to establish a connection with the specific IP address 'strIP'.
- ACTIVE – The function block actively attempts to establish a connection with the specified IP address. This attempt is repeated every ten seconds (if a connection has not yet been established) or a check is made for an existing connection (if a connection has been established further to a previous attempt). In the context of client/server connections, this communication mode is suitable for the client.
- PASSIVE – The function block waits for a connection request from another controller on the network. If a connection request is received, it is compared with the specified IP address 'strIP'. If the two addresses are not identical, the request is rejected; if they are identical, a connection is established. This communication mode should be used for more complex network structures (a client/server application between two controllers, for example). In PASSIVE mode, the 'strIP' input cannot be declared ("empty"); accordingly, multiple controllers can access the controller that is in PASSIVE mode with only one function block being necessary to do this.

Once the function block has been started ('boEnable'=TRUE), the 'enMode', 'strIP', 'wPort' inputs must not be changed. During sending and receiving, the 'pbySendBuff', 'uiSendSize', 'pbyRecvBuff', and 'uiRecvSize' inputs must not be changed, as this can lead to transmission errors.

The 'wPort' input can be freely selected. Please note: The port number is 16 bits; i.e. values between 0 and 65535 are possible. The assignments of port numbers 0 through 1023 are fixed (e.g. port 21 for FTP); therefore, they should not be used. Using these port numbers can result in network traffic complications. Default value: 1500

17.2.1 Max. user data per TCP message frame

TCP data is transferred via Ethernet. The maximum block size of an Ethernet stream is 1500 bytes; this is also referred to as the MTU (maximum transmission unit). The MTU for AMK controllers is set to 1500 bytes. The MTU can be smaller for third-party manufacturer devices.

In the TCP message frame, a header of 20 bytes (+ 12 bytes for options) is transferred to the Ethernet stream. As TCP runs over the Internet protocol (IP), a header of 20 bytes is also transferred by the IP message frame. This results in the following:

1500 bytes Ethernet stream

- 32 bytes TCP header (including options)

- 20 bytes IP header

= 1448 bytes for user data (AMK controllers).

For connections with third-party manufacturer devices, the following applies:

MTU (device) – 52 bytes = maximum number of user data in one message frame

If the dataset to be sent exceeds the maximum number of user data (MTU – 52) of one of the two communication partners, the data is automatically broken down (segmentation) into 1448 bytes (e.g. AMK controller). As a result, the sent data can also be received in segments on the receiving side, depending on the temporal sequences. As the full data length is not entered in the TCP and IP header, the receiving side does not know how much data has to be sent or received. If data blocks larger than MTU – 52 bytes are received, the length must be checked on receipt. This can be done with a prefixed header, for example, in which the quantity of data received is indicated. The receiver then knows how much data belongs to this packet. It must repeat the receive operation until all data has been received.

Example:

- The sender sends 3800 bytes
- The 'uiRecvSize' input variable on the receiver has a setting of 2200
- Depending on the temporal sequences, either only the first 1448 bytes are received (the maximum number of user data per message frame is 1448) or 2200 bytes
- Set the 'uiRecvSize' input variable from 0 to 2200
- The next 1448 bytes or the remaining 1600 bytes are received
- If not all data has been received 'uiRecvSize' input variable from 0 to 2200
- The remaining 904 bytes are received

At the end of data exchange, the function block must be deactivated ('boEnable' = FALSE) in order to release the resources used.

17.2.2 Sending and receiving data

17.2.2.1 TCP and TCP_1

To send the data to a connected communication partner, the function block must be called with a pointer to the data to be sent and with the quantity of data to be sent 'pbySendBuff' and 'uiSendSize'. If a number that is not equal to zero is set at the 'uiSendSize' of the function block, it starts sending automatically. Once the data has been sent, the 'boSendAck' output is set to TRUE. The signal changes to FALSE if zero is set at the 'uiSendSize' input. To send again, 'uiSendSize' must first be set to zero and then set back to the quantity of the data to be sent. If 'uiSendSize' is equal to zero, the 'pbySendBuff' input variable can be changed.

If data is to be received, the function block must be called with a pointer to the receive buffer and the size of the receive buffer 'pbyRecvBuff' and 'uiRecvSize'. If 'uiRecvSize' does not equal zero, the data received is saved in the receive buffer 'pbyRecvBuff' but only up to the number specified in 'uiRecvSize', and 'boRecvAck' changes to TRUE. The quantity of the received data is displayed in 'uiActRecv'. So that new data can be received, 'uiRecvSize' must be set to zero. This sets 'boRecvAck' to FALSE. Now no further data is saved in the buffer and the following options are available:

- Process the received data (copy, edit, etc.)
- Change the receive buffer (pointer)

Once the user has finished processing the data in the receive buffer, the size of the next receive buffer can be specified in 'uiRecvSize' once again.

17.2.2.2 TCP_2

To send the data to a connected communication partner, the function block must be called with a pointer to the data to be sent and with the quantity of data to be sent 'pbySendBuff' and 'uiSendSize'. If the 'boSend' input is set to TRUE, the data from 'pbySendBuff' is sent with the size 'uiSendSize'. Once the data has been sent, the 'boSendReady' output is set to TRUE. A reset at the 'boSend' input changes the 'boSendReady' signal to FALSE. So, to send again, 'boSend' must first be set to FALSE and then set back to TRUE. If 'boSend' is FALSE, the 'pbySendBuff' and 'uiSendSize' input variables can be changed.

If data is to be received, the function block must be called with a pointer to the receive buffer and the size of the receive buffer 'pbyRecvBuff' and 'uiRecvSize'. If the 'boReceive' input is set to TRUE, the data received is saved in the receive buffer 'pbyRecvBuff'. If 'uiMaxRecv' ≤ 'uiRecvSize', the data is saved in full in the receive buffer and 'boRecvReady' changes to TRUE. If 'uiMAxRecv' > 'uiRecvSize', the data received is saved in the receive buffer up to a size 'uiRecvSize' and 'boRecvBuffFull' changes to TRUE. The user can create a new receive buffer 'pbyRecvBuff'. Another positive edge at the 'boReceive' input triggers the reception of the remaining data. Once all data has been received, 'boRecvBuffFull' changes to FALSE and 'boRecvReady' to TRUE.

18 AmkUdp - UDP communication interface specific to AMK

The UDP function block provides the user with an easy way of sending and receiving data via 'UDP'. Accordingly, the user does not need to have in-depth background knowledge of network transfer. Only basic knowledge of IP addresses and the use of port numbers is required.

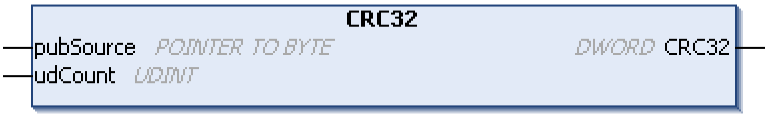
UDP	UDP communication interface specific to AMK
CRC32	Calculation of a 32-bit CRC checksum

18.1 POUs

18.1.1 CRC32 (F)

The 'CRC32' function calculates a 32-bit CRC checksum (CRC = cyclic redundancy check). The function is used, for example, in the 'UDP' block. It helps to check for data transfer problems.

User interface



Input variables

Name	Type	Description
pubSource	POINTER	POINTER TO BYTE Address of the memory range from which calculation of the CRC checksum commences
udCount	UDINT	Number of bytes which (starting from 'pubSource') are included in the CRC checksum calculation

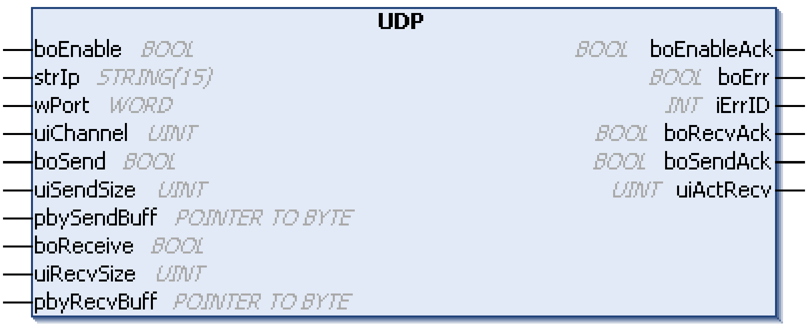
Output variables

Name	Type	Description
CRC32	DWORD	32-bit CRC checksum as return value of function CRC32

18.1.2 UDP (FB)

The 'UDP' function block provides the user with an easy way of sending and receiving data. Accordingly, the user does not need to have in-depth background knowledge of network transfer. Only basic knowledge of IP addresses and the use of port numbers is required.

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.
strIp	STRING (15)	String containing the IP address (dotted decimal notation) of the node with which a connection is to be established. If this input is not assigned, the controller can connect to a communication partner with any IP address Default: '192.168.0.1'
wPort	WORD	Port of the node with which the connection is being established Default: 1500
uiChannel	UINT	Selection of communication instance Default: 4
boSend	BOOL	Sends the number of 'uiSendSize' data located in the send buffer 'pbySendBuff'
uiSendSize	UINT	Size of the send buffer: user data + transfer information (8 bytes)
pbySendBuff	POINTER	POINTER TO BYTE Pointer variable referencing the send buffer which contains the data to be sent. At the end of the send buffer, 8 bytes must be left available for the transfer information. The size of the send buffer must be at least equal to 'uiSendSize'.
boReceive	BOOL	Receive data up to the maximum size set in 'uiRecvSize' for writing to receive buffer 'pbyRecvBuff'
uiRecvSize	UINT	Size of the receive buffer, expected user data + transfer information (8 bytes)
pbyRecvBuff	POINTER	POINTER TO BYTE Pointer variable referencing the receive buffer in which the data received is written (user data + transfer information (8 bytes). The size of the receive buffer must be at least equal to 'uiRecvSize'

Output variables

Name	Type	Description
boEnableAck	BOOL	Acknowledgement: Function block is initialised and enabled
boErr	BOOL	The function block is in an error state FALSE: No error (permitted commanding or warning) TRUE: Error

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>
Error
<table>
<tr><th>Range</th><th>Meaning</th></tr>
<tr><td>0</td><td>No error</td></tr>
<tr><td>1-999</td><td>Error numbers are described in the library documentation for AmkSockets.lib</td></tr>
</table>
For AmkTCP.lib up to Version 02.01 2008/25, the following also applies:
<table>
<tr><th>Range</th><th>Meaning</th></tr>
<tr><td>101</td><td>Illegal pointer to the receive buffer</td></tr>
<tr><td>102</td><td>Illegal pointer to the send buffer</td></tr>
<tr><td>103</td><td>Incorrect CRC32 value in current data packet</td></tr>
</table>
For AmkTCP.lib as of Version > 02.04 2008/25, the following also applies:
<table>
<tr><th>Range</th><th>Meaning</th></tr>
<tr><td>1001</td><td>Illegal pointer to the receive buffer</td></tr>
<tr><td>1002</td><td>Illegal pointer to the send buffer</td></tr>
<tr><td>1003</td><td>Incorrect CRC32 value in current data packet</td></tr>
<tr><td>1004</td><td>Incorrect instance 'uiChannel' or invalid IP address</td></tr>
<tr><td>1005</td><td>Read/write operation interrupted, connection closed</td></tr>
<tr><td>1006</td><td>Internal error</td></tr>
</table></td></tr>
<tr><td>boRecvAck</td><td>BOOL</td><td>Signal indicating that data has been received successfully. The signal is set if the data has been received in full in the buffer and is reset, if 'boReceive' is set to FALSE</td></tr>
<tr><td>boSendAck</td><td>BOOL</td><td>Signal indicating that data has been sent successfully. The signal is set if all of the data has been sent from the buffer and is reset, if 'boSend' is set to FALSE</td></tr>
<tr><td>uiActRecv</td><td>UINT</td><td>Length of data currently received: user data + transfer information (8 bytes)</td></tr>
</table>

18.2 Application

The 'UDP' function block facilitates data exchange between two or more network nodes via UDP (user data protocol).

Properties of UDP:

- Faster and more efficient than TCP (lower overhead)
- Supports broadcast
- Not reliable
- Connectionless
- No jam control

'UDP' can be used to transfer cyclic data or for time-intensive applications, for example.

With 'UDP', no checks are made to ascertain if the data has arrived at the receiver (not reliable) or whether it is in the correct order. As 'UDP' is connectionless, data can be exchanged between network nodes more quickly than with a TCP connection, for example. The advantage of this is of particular interest in the context of transferring smaller data packets.

Before starting the 'UDP' function block via the 'boEnable' input, the 'strIP',and 'wPort' inputs must be assigned the corresponding values.

The 'strIP' input can be assigned the following values:

- "(empty string): default value. The controller can then receive data from any network node sending data on the same port. The IP address of the sender of the received data is buffered. This can be used to send a response (confirming receipt, for example). (Server application)
- 255.255.255.255: The controller uses this IP address to send the data to all participants on the network (broadcast). Every network node which has set the same port can receive the data. In this mode, the controller can also receive data from any network node.
- 192.168.0.1: (fixed assigned network address) With this setting, data can only be received from and sent to the specified network nodes.

If the function block is set with the IP address 255.255.255.255, the gateway address of the Ethernet interface in the controller must not be 255.255.255.255 (ID34056; default setting). The gateway address must be set to a node located on the network.

The 'wPort' input can be freely selected. Please note the following:

The port number is 16 bits; i.e. values between 0 and 65535 are possible. The assignments of port numbers 0 through 1023 are fixed (e.g. port 21 for FTP); therefore, they should not be used. Using these port numbers can result in network traffic complications.

Default value: 1500

18.2.1 Max. user data per UDP datagram

UDP data is transferred via Ethernet. The maximum block size of a UDP datagram is 1500 bytes; this is also referred to as the MTU (maximum transmission unit). The MTU for AMK controllers is set to 1500 bytes. The MTU can be smaller for third-party manufacturer devices.

As UDP runs over the Internet protocol (IP), a header of 20 bytes is transferred by the IP message frame. The UDP message frame adds an 8-byte header to the UDP datagram and the 'UDP' function blocks adds another 8 bytes of transfer information. This results in the following:

1500 bytes UDP datagram

-20 bytes IP header

-8 bytes UDP header

-8 bytes transfer information

= 1464 bytes for user data (AMK controllers).

For connections with third-party manufacturer devices, the following applies:

MTU (device) – 32 bytes = maximum number of user data in one message frame.

If the dataset to be sent exceeds the maximum number of user data (MTU – 32) of one of the two communication partners, the data is automatically broken down into data packets (segmentation). As the 'UDP' transfer service is unreliable, there is an increased chance of data packets being lost in the event of segmentation. Lost data packets are detected from the transfer information at the receiver; they are output as errors (data not written to receive buffer).

Therefore, the maximum number of user data should not exceed MTU – 32 bytes.

18.2.2 Sending and receiving data

To send data, the function block must be called with a pointer to the data to be sent and with the quantity of data to be sent 'pbySendBuff' and 'uiSendSize'. If the 'boSend' input is set to TRUE, the data from 'pbySendBuff' is sent with the size 'uiSendSize'. Once the data has been sent, the 'boSendAck' output is set to TRUE. A reset at the 'boSendAck' input changes the 'boSend' signal to FALSE. So, to send again, 'boSend' must first be set to FALSE and then set back to TRUE. If 'boSend' is FALSE, the 'pbySendBuff' and 'uiSendSize' input variables can be changed.

If data is to be received, the function block must be called with a pointer to the receive buffer and the size of the receive buffer ('pbyRecvBuff' and 'uiRecvSize'). If the 'boReceive' input is set to TRUE, the data received is saved in the receive buffer 'pbyRecvBuff', but only up to the volume specified in 'uiRecvSize', and 'boRecvAck' changes to TRUE. The quantity of the received data is displayed in 'uiActRecv'. More data can be received once another positive edge has occurred at the 'boReceive' input. If 'pbyRecvBuff' has not changed, the data received previously is overwritten.

The 'UDP' function block appends 8 bytes of transfer information to the user data to be sent. This transfer information is used to check the data packet for completeness at the receiver.

The sender must ensure that the specified size of the 'uiSendSize' input variable equals that of the user data + transfer information (8 bytes). In the same way, on the receiver, the specified size of the 'uiRecvSize' input variable equals that of the user data received + transfer information (8 bytes).

Example:

	uiSendSize = 500		uiRecvSize = 500	
1 . . . 492	492 bytes user data	Send →	492 bytes user data	1 . . . 492
493 . . . 500	8 bytes transfer information		8 bytes transfer information	493 . . . 500

If the data received (user data + transfer information) is greater than 'uiRecvSize', an error ('boErr'=TRUE; 'iErrID'=103) is output. The receiver cannot find any transfer information in the data packet received and, therefore, has no information indicating whether the data packet has been received in full. The data is subsequently not written to the receive buffer.

19 AmkSm3Drive - Sm3Drive blocks specific to AMK

'AmkSm3Drive' is an internal AMK library which contains blocks for the implementation of the SoftMotion bus interface. It is divided into:

PLCopen project	Creation and use of a PLCopen project
SoftMotion	SoftMotion bus interface

19.1 Creation and use of a PLCopen project

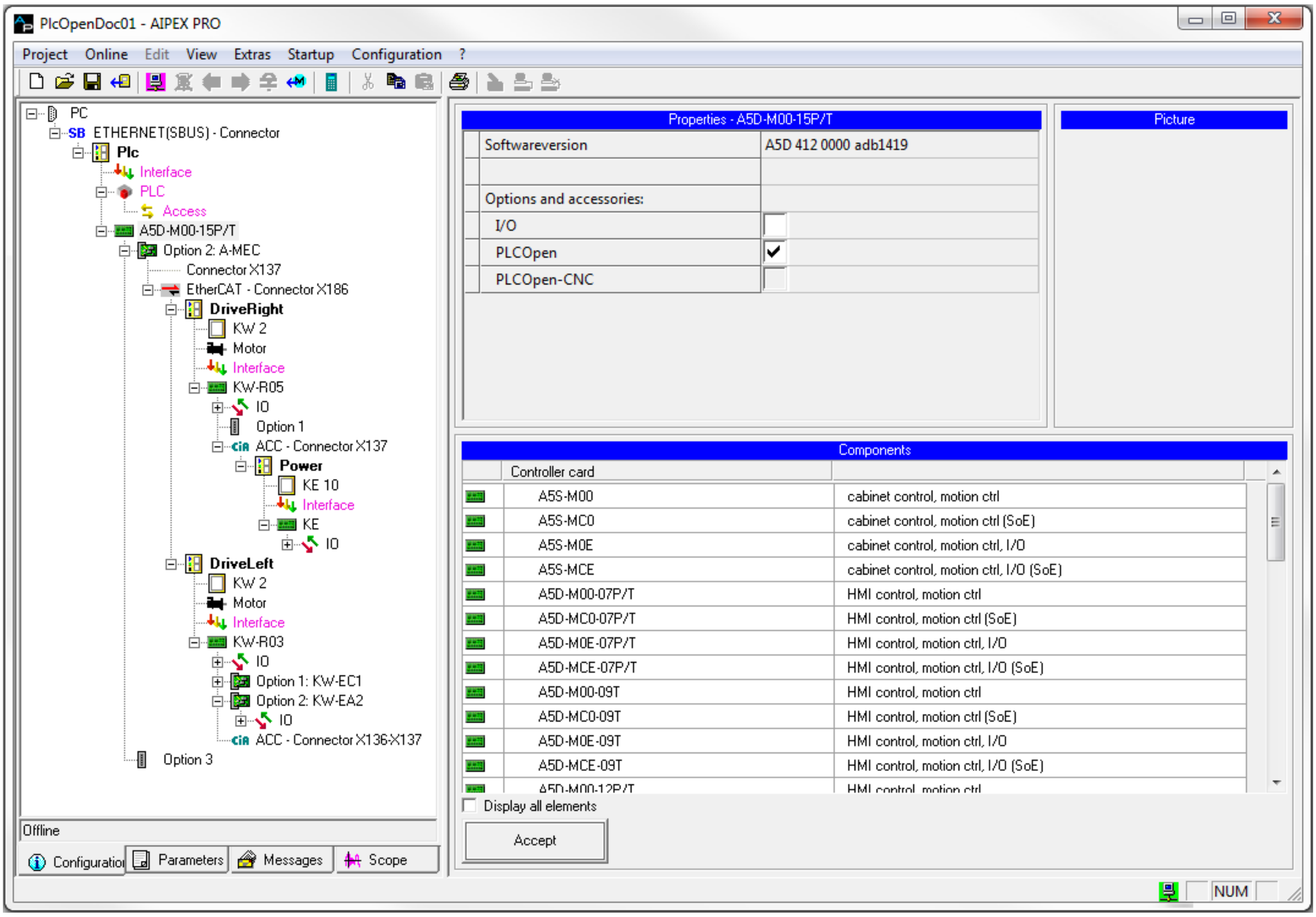
CODESYS V3 supports the additional options "PCO = PLCopen" and "PNC =PLCopen CNC". In conjunction with an A5x-Mxx PLC module, this enables the PLCopen blocks from the SM3_Basic library by 3S to be used (PLCopen function figure).

Alternatively, the "PLCopen CNC" property can also be selected. This makes the SM3_CNC 3S library available, with the CNC function implemented by 3S.

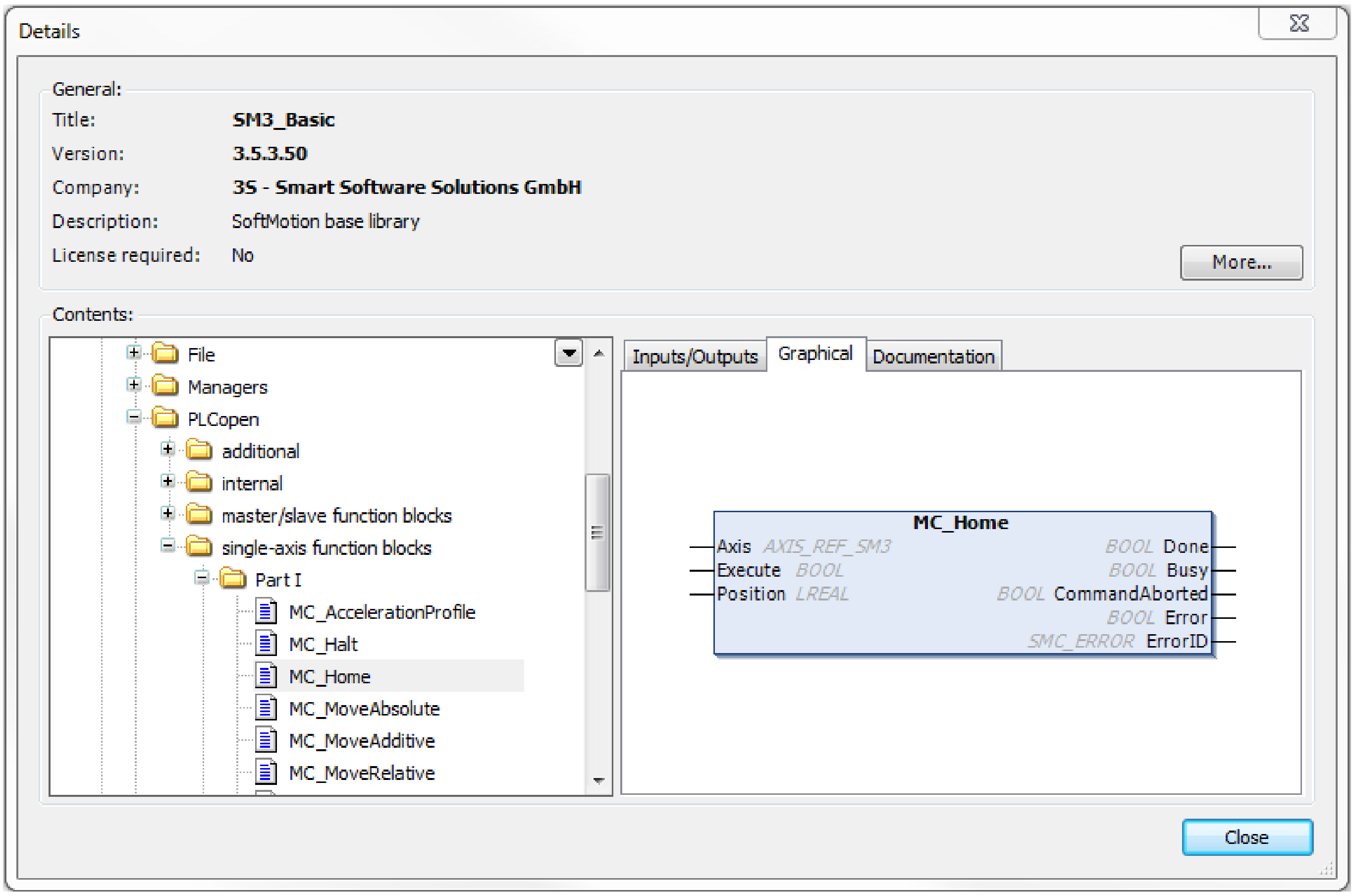
Prerequisite: The PLC option (PLCopen CNC) must be enabled in the target system!

Creation	Creation of the PLCopen project
Configuration	Configuration of the PLCopen project

Selection of the PLCopen function:



Add PLCopen function!

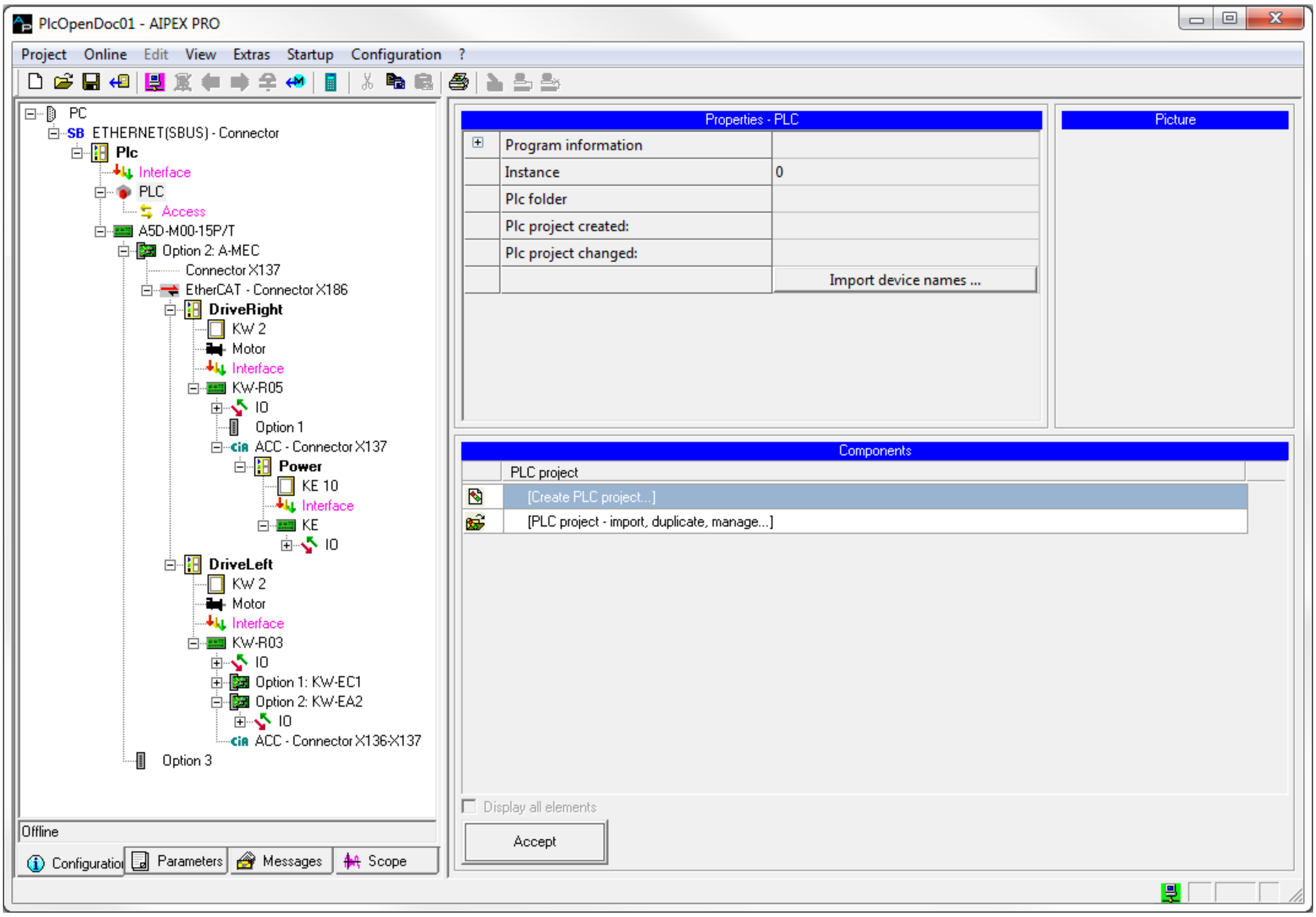


19.1.1 Creation of the PLCopen project

If an A5 display controller with the PLCopen property is selected (e.g. A5D-MC0-15P/T), "Create new PLC project" can be used to create a suitable base project (template).

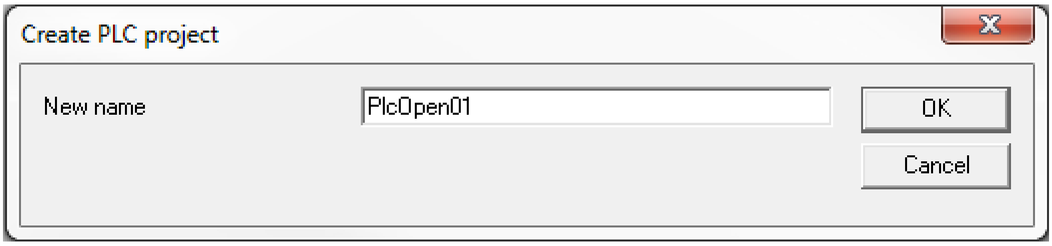
The A5x-Mxx target system software version must be ≥ AS V4.11 2013/50.
The "PLCopen" option must be enabled in the A5S or A5D controller!

Create new PLC project:



The project name is specified after selecting "Create new PLC project" (triggered by double-clicking).

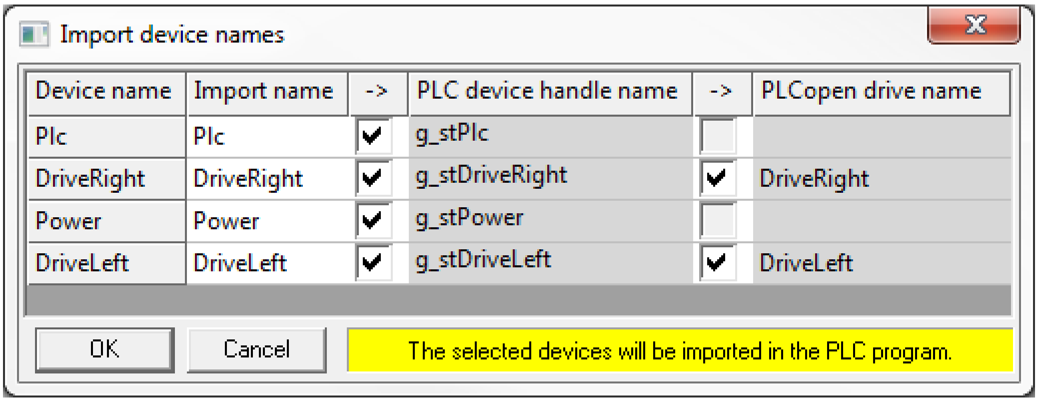
Specify PLC project name:



If the "AIPEX PRO" option "Device name when creating a new PLC project" is activated, the device names that can be derived from the "AIPEX PRO" device tree are suggested as the basis for the necessary PLC handle and any PLCopen device names that might be required.

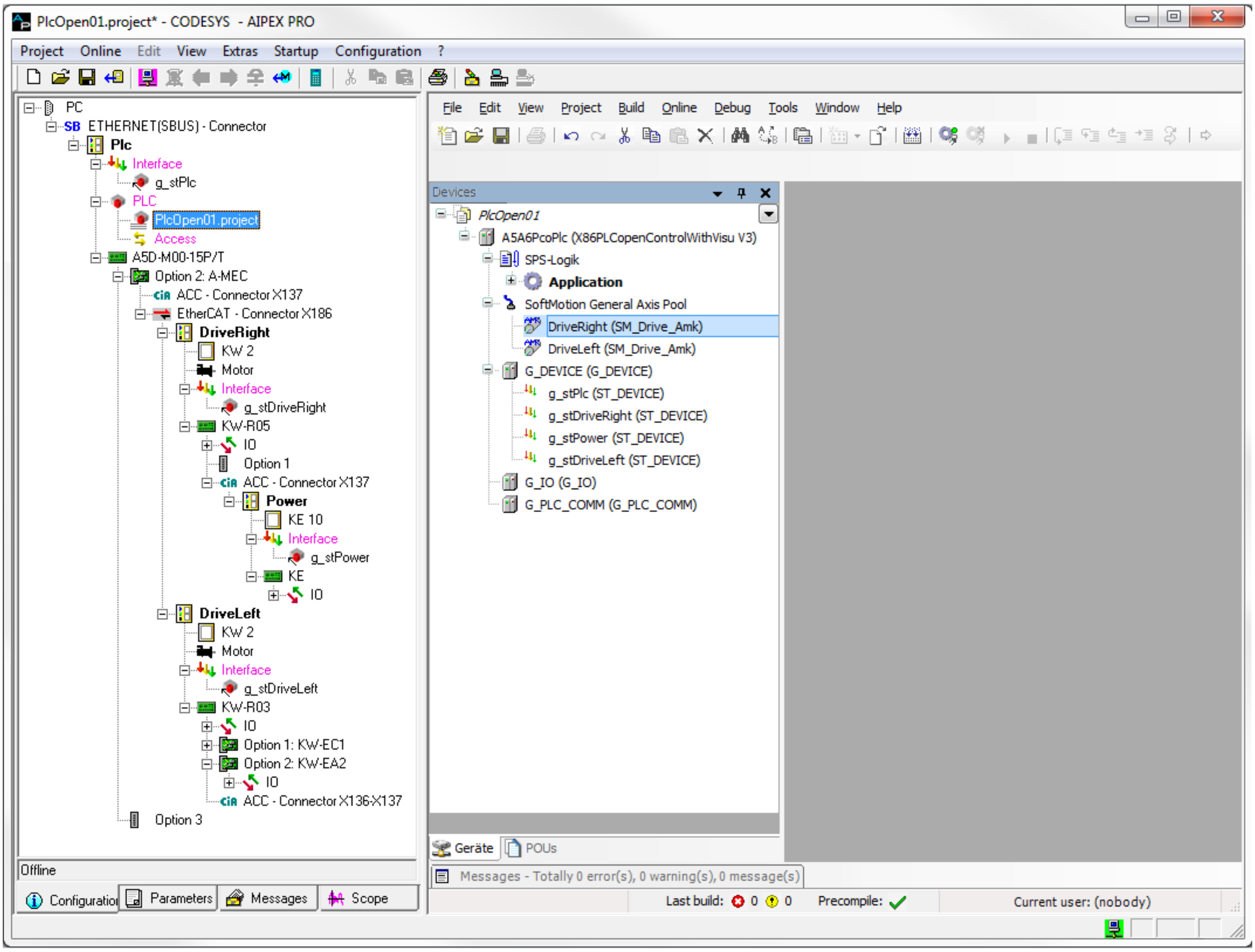
Independent of this option, the dialog can also be opened by selecting "Import device names".
The import name, which does not have to be the same as the device name, can be specified in the dialog.

Import device name into PLC project:



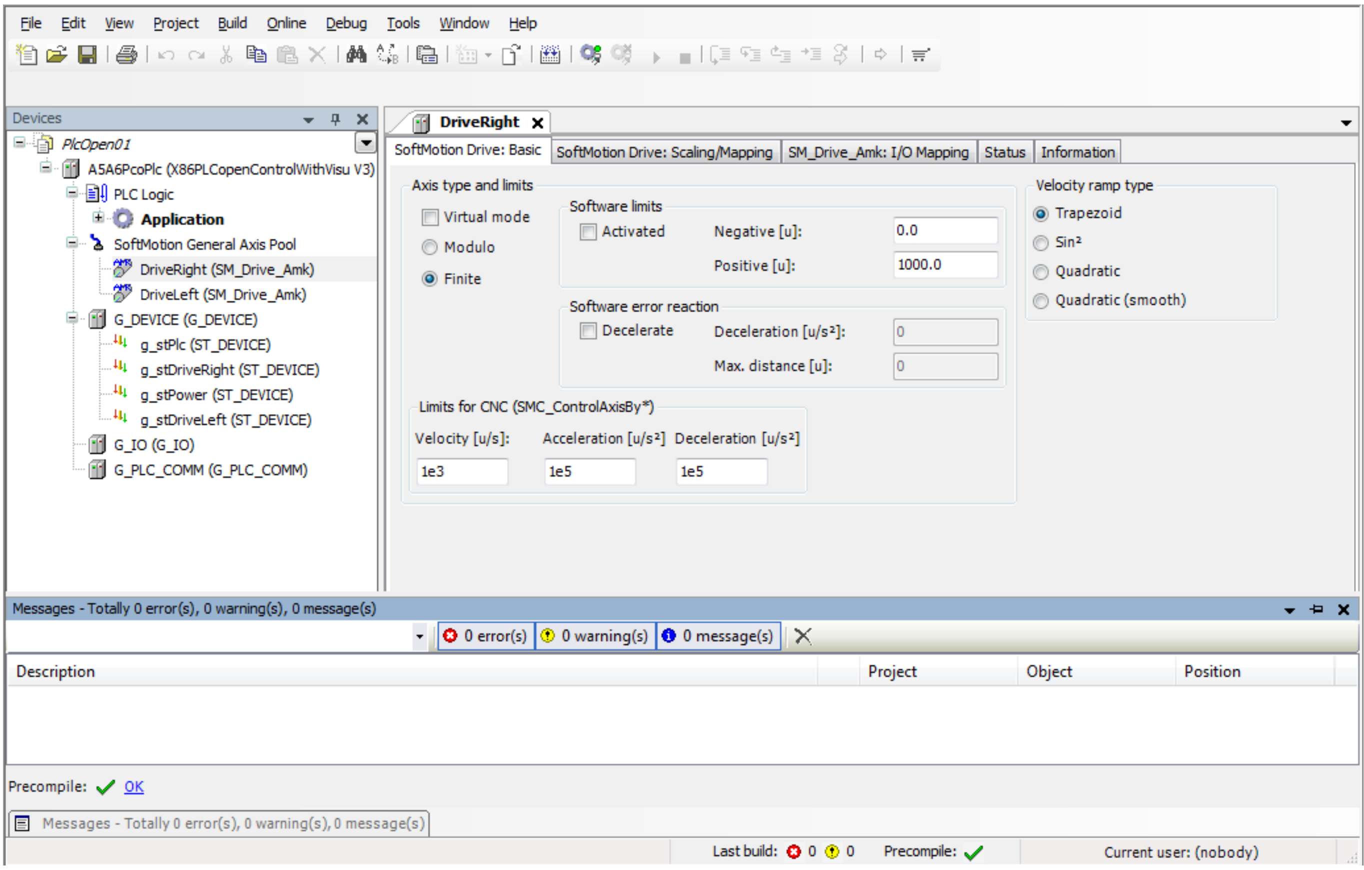
Once the "Import device name" dialog has been confirmed, the suggested PLC handle and PLCopen device name are imported into the PLC project and are available in the device tree of the CODESYS project (G_DEVICE or SoftMotion General Axis Pool).

Controller configuration based on device name:

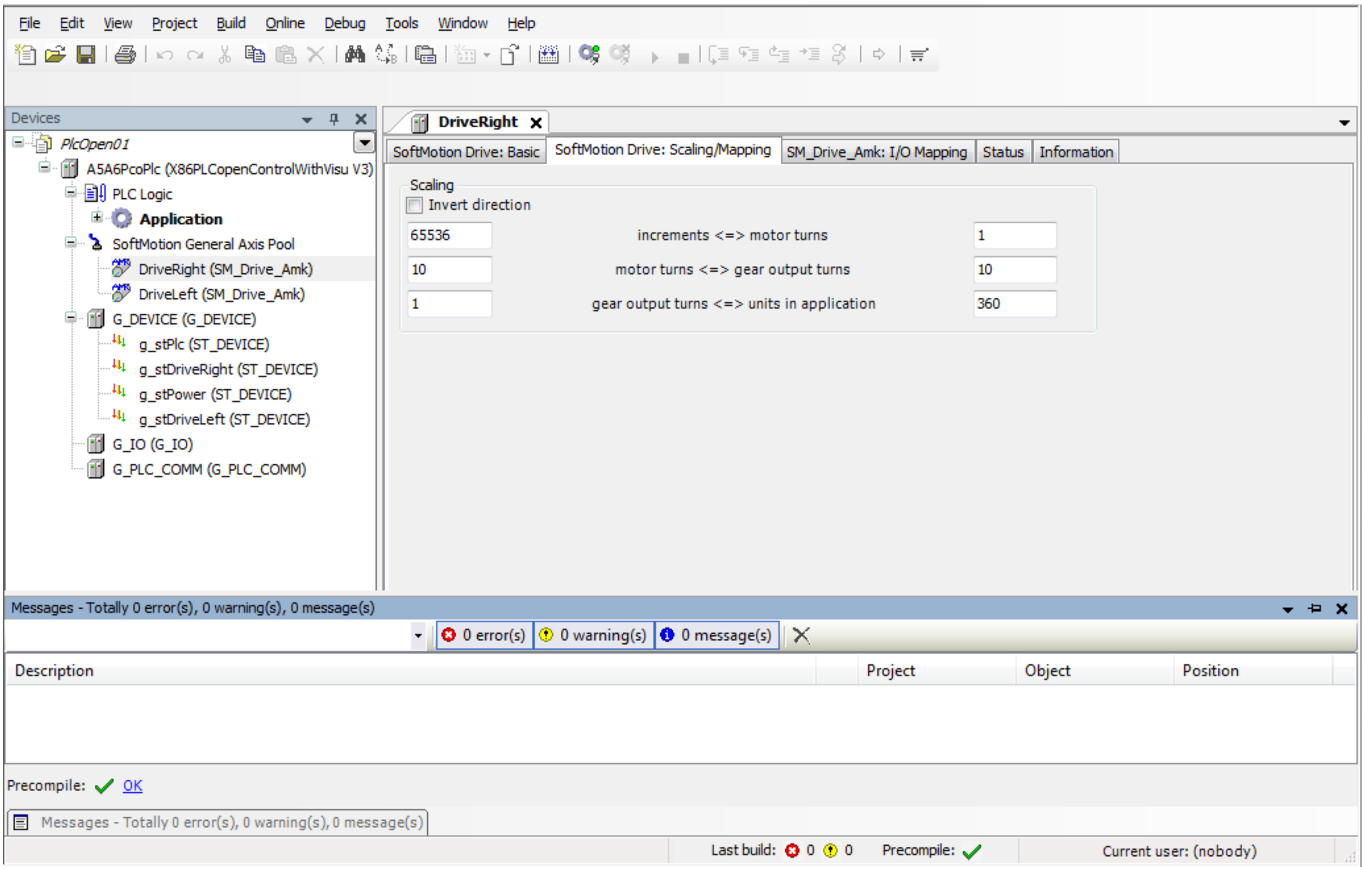


Next, the base parameters and conversion factors (scalings) relevant to PLCopen can be set in the dialog for drive parameters.

Base parameters



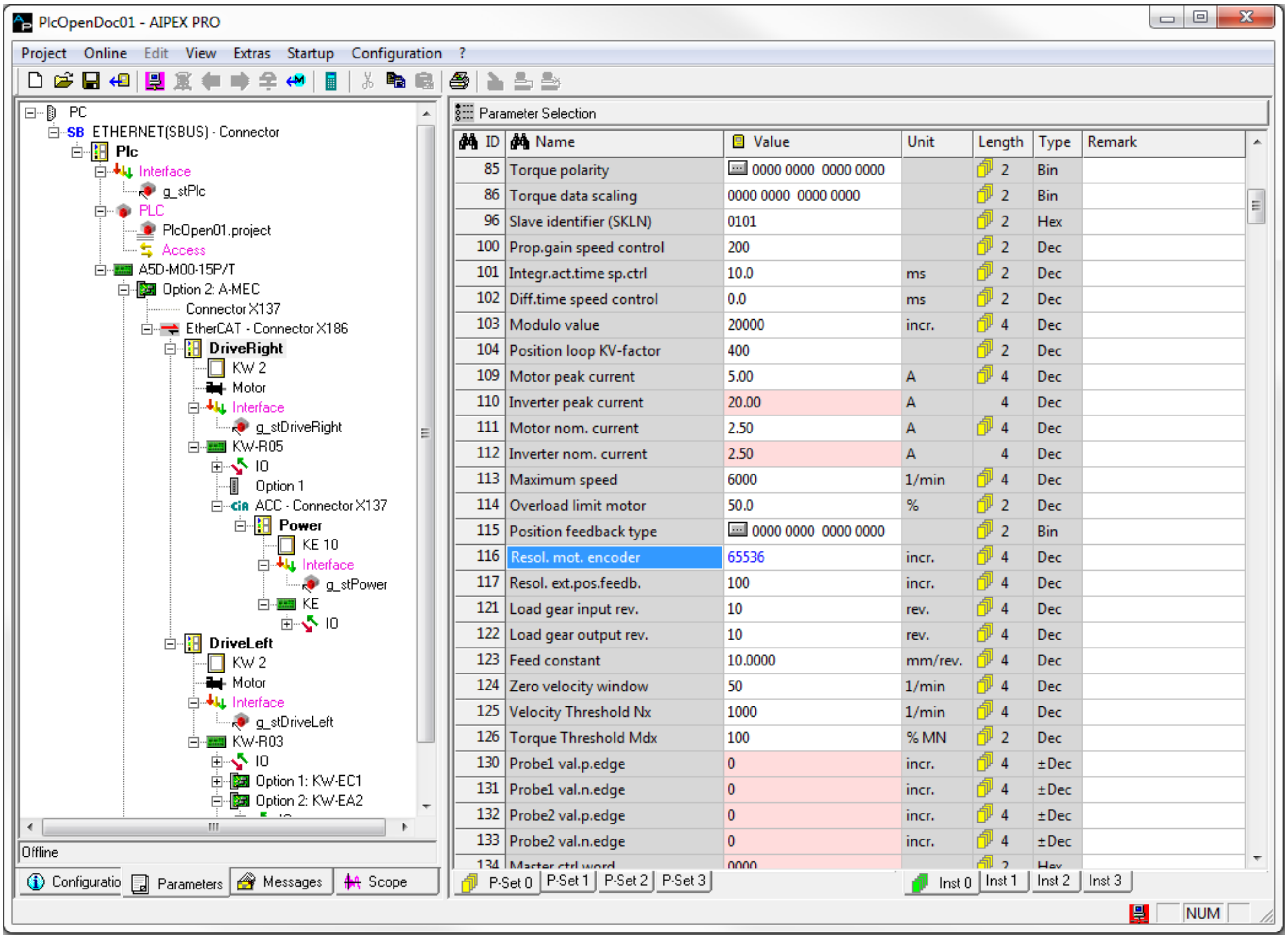
Scalings:



The settings for "increments / revolution" und "gear input revolutions / gear output revolutions" must match the parameterization of the corresponding drive (ID116 'Resolution motor encoder', ID121 'Load gear input revolution', ID122 'Load gear output revolution').

The setting for "gear output revolutions / SoftMotion units" can be selected by the user; it specifies the SoftMotion unit. The resolution of "1 / 360" set in the 'Scaling' figure, for example, results in a SoftMotion unit of 1 degree.

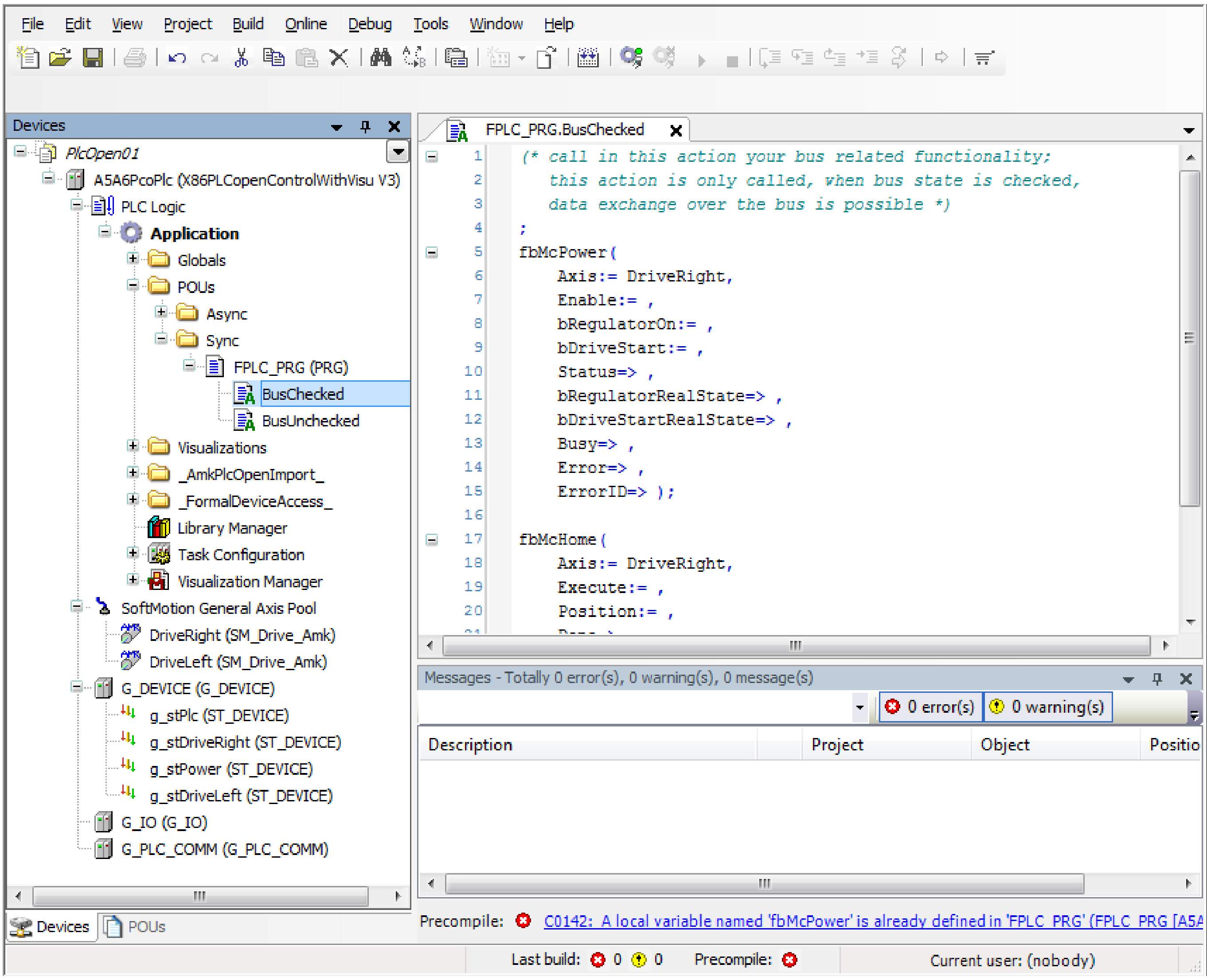
Encoder resolution:



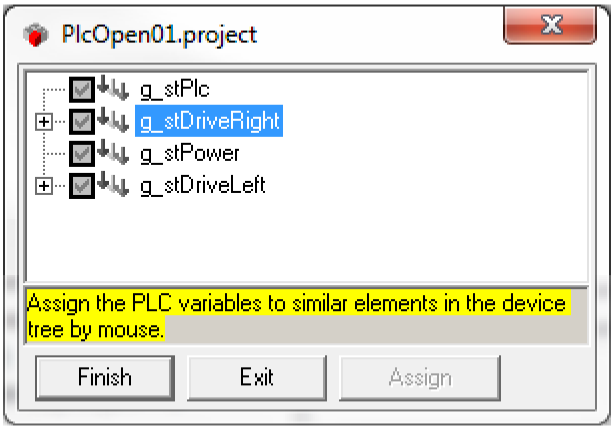
In the CODESYS project, programming is carried out based on instances of PLCopen blocks (see the 'PLCopen function' figure). The assignment to the devices (drives) is based on the use of the PLCopen device names derived from the PLC handle identifier (see the 'Importing device names into the project' figure).

The bus is configured automatically in "Create configuration" (using the corresponding PLC handle; see 'Device handle assignment' figure).

PLCopen block instances:

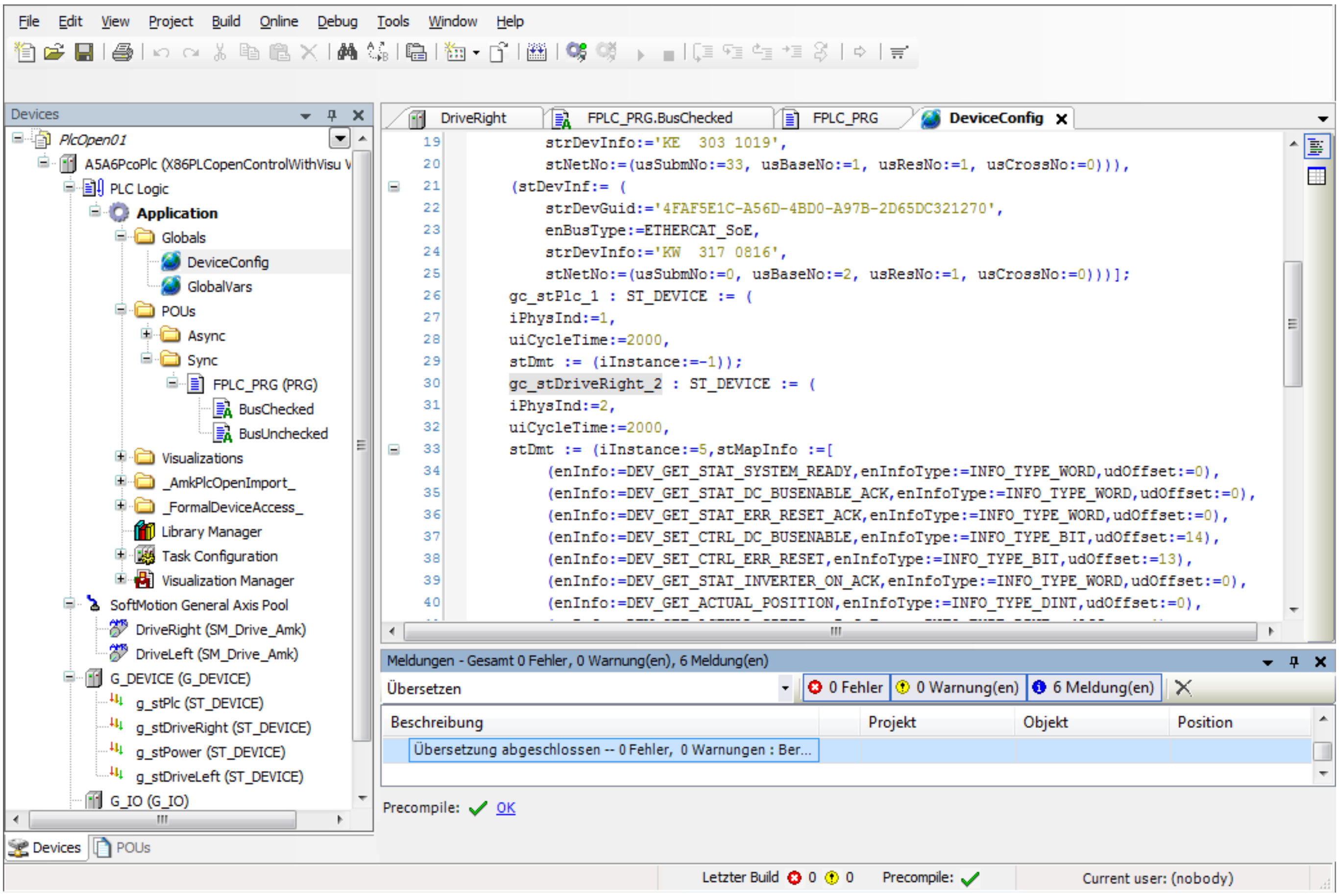


Device handle assignment:



Once the dialog has been confirmed with "Done", the necessary bus configuration information is both transferred to the controller (in online mode) and imported into the CODESYS project. The PLC project can now be uploaded to the controller during "Login".

'Device_Configuration:



19.1.2 Configuration of the template

The PLCopen template shown in the 'PLCopen template' figure initially comprises 3 basic organizational units:

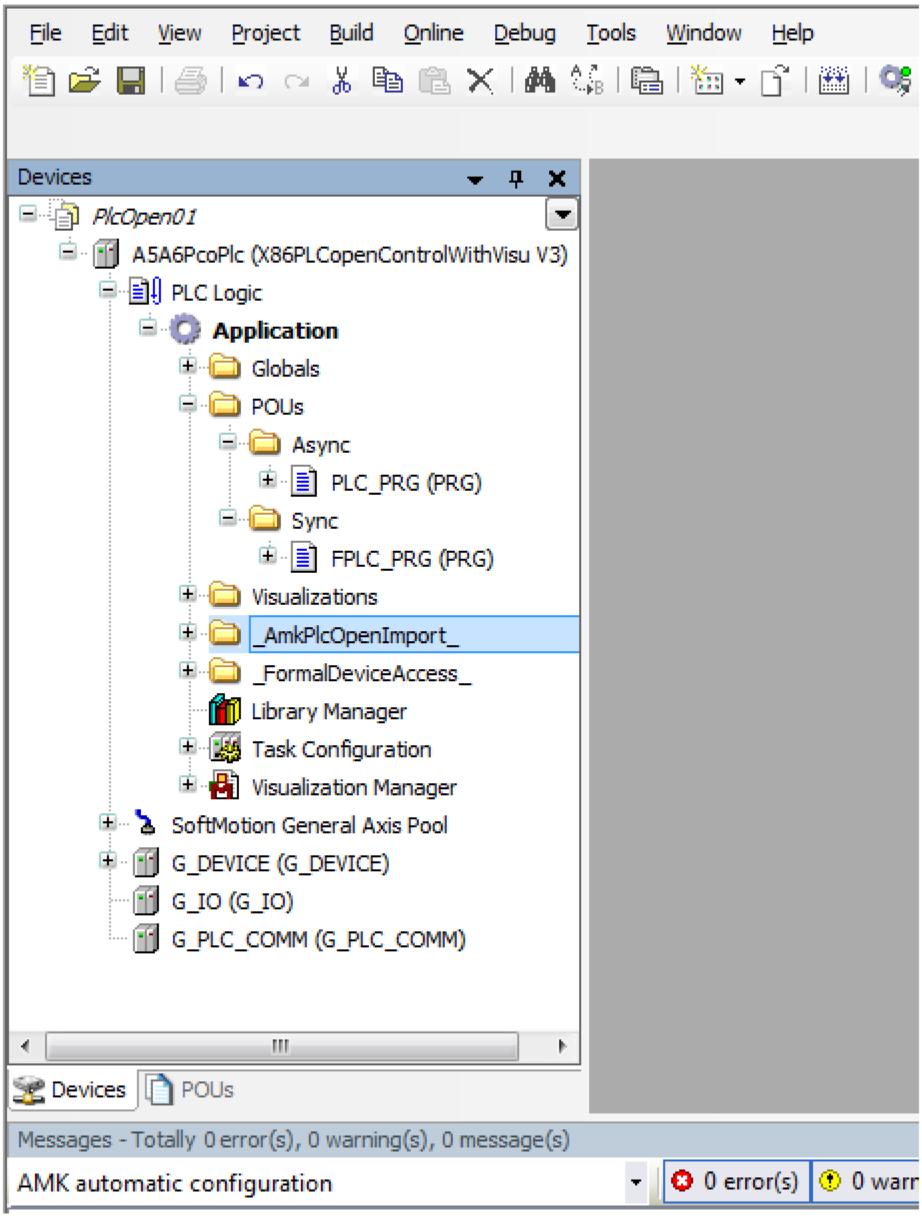
• The "_AmkPlcOpenImport_" folder, which is imported into the template in the context of "Import device name". The blocks in this folder are generated automatically and must not (cannot) be changed by the user.

• The "FPLC_PRG" program block which is embedded in the task configurator with the "externally event-driven" PGT task (see the 'FPLC_PRG' figure). This block is thus called synchronized with the central system cycle (PGT = Peripherie Grund Takt (peripheral basic cycle)). It is used to process blocks for synchronous access to drive movement information.

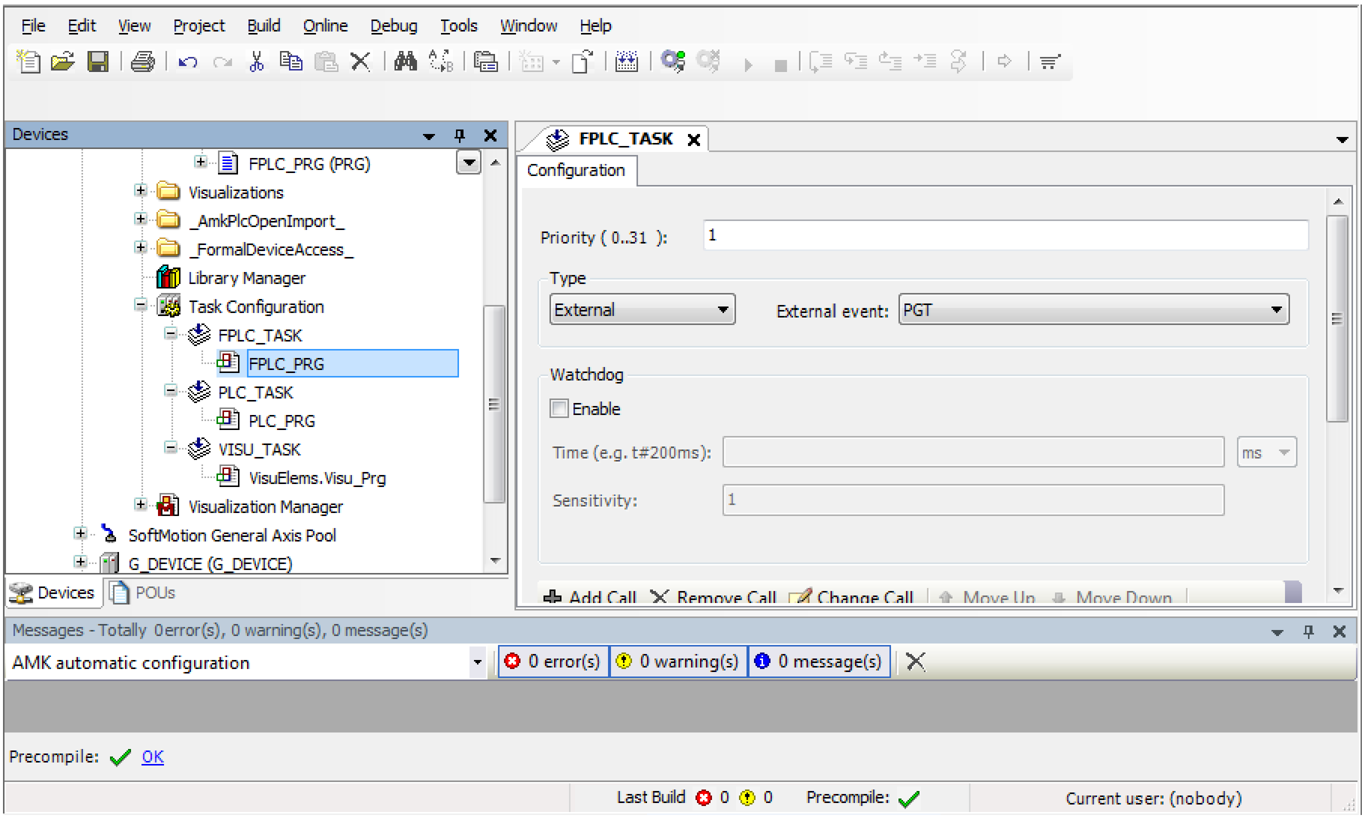
Notice: All PLCopen blocks must be processed in this block (in its BusChecked action; see the 'PLCopen block instances figure).

• The "PLC_PRG" program block which is embedded in the task configurator with a "free-running" task (see the 'PLC_PRG' figure). This block is thus called asynchronous to the central system clock (PGT). Therefore, it can only be used to process blocks with asynchronous access to drive movement functions. Any other non-time-equidistant function can also be implemented here.

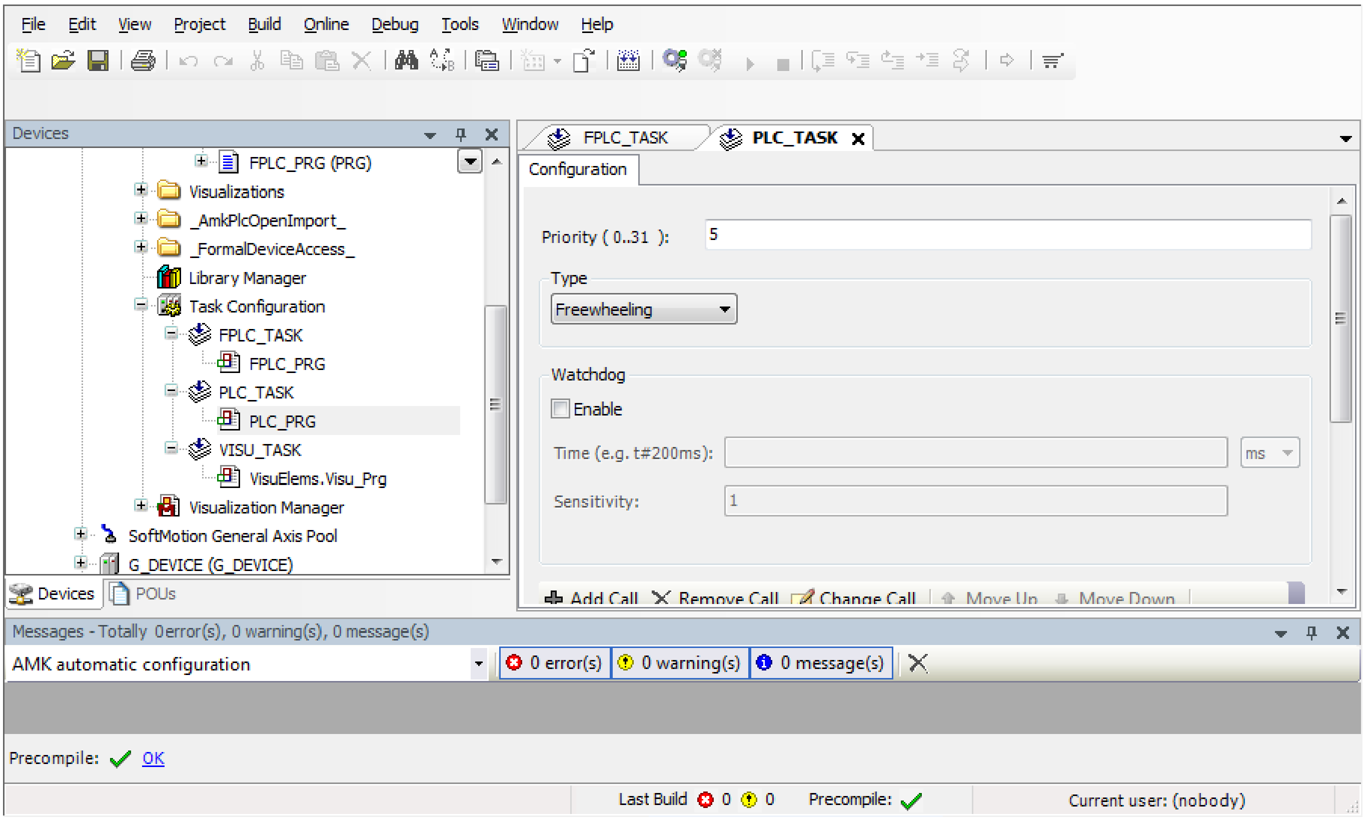
PLCopen template:



Task configuration (FPLC_PRG):



Task configuration (PLC_PRG):



Each of the two program blocks FPLC_PRG and PLC_PRG has a "BusChecked" and a "BusUnchecked" action.

So, as shown in the 'PLC template (PLC_PRG)' and 'PLC template (FPLC_PRG)' figures:

• The blocks called in the "BusUnchecked" action are always processed.

• The blocks called in the "BusChecked" action are only processed if the "AmkSm3Drive.g_enInitState<>AmkSm3Drive.AMK_INIT_DONE" state has been adopted. In turn, this is only the case if "Bus startup" for all buses (EtherCAT and/or ACC) has been completed (see the 'PLC template (PLC_PRG, InitSystem action) figure').

Note: The condition for "Bus starting up" Siehe 'Performance features' auf Seite 529.
"(UINT_TO_WORD (FuiGetNetSatus()) AND 16#0013) = 16#0013".

• The "PLC_PRG.InitSystem()" action is called automatically in PLC_PRG. In collaboration with PLCopen initialization, it organizes the state graph as shown in the 'PLC template (PLC_PRG, InitSystem action)' figure.
• The "PLC_PRG.InitGlobals()" action is generated automatically; it does not have to be called explicitly.

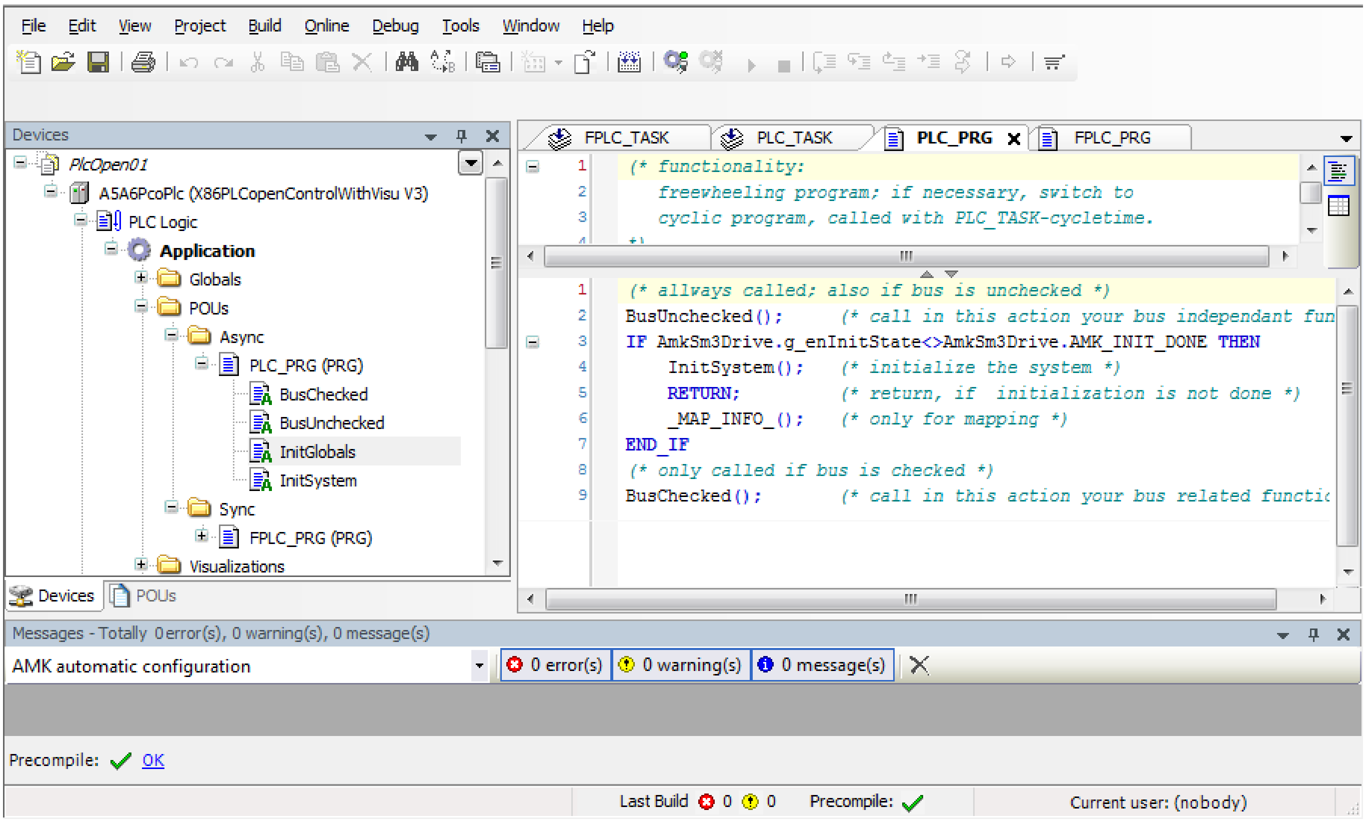
Use of template actions

Program function	FPLC_PRG. BusUnchecked	FPLC_PRG. BusChecked	PLC_PRG. BusUnchecked	PLC_PRG. BusChecked
Processing in PGT raster, no access via the bus	X			
Processing in PGT raster, no access via the bus		X		
Asynchronous processing; no access via the bus			X	
Asynchronous processing; no access via the bus				X

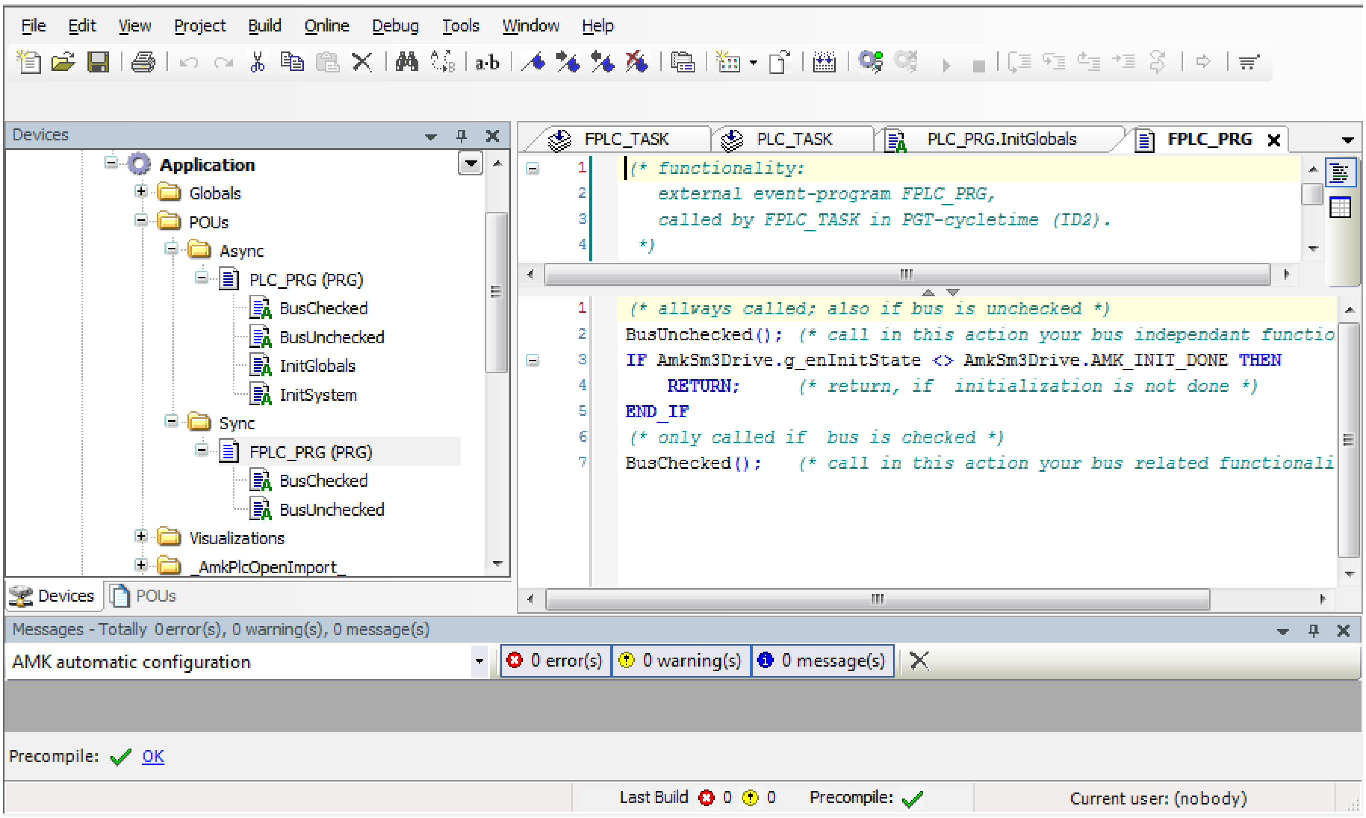
The user function should be called during the course of one of these four actions. Which action is selected is based on the information in the table.

PLCopen blocks must always be processed in the PGT raster.

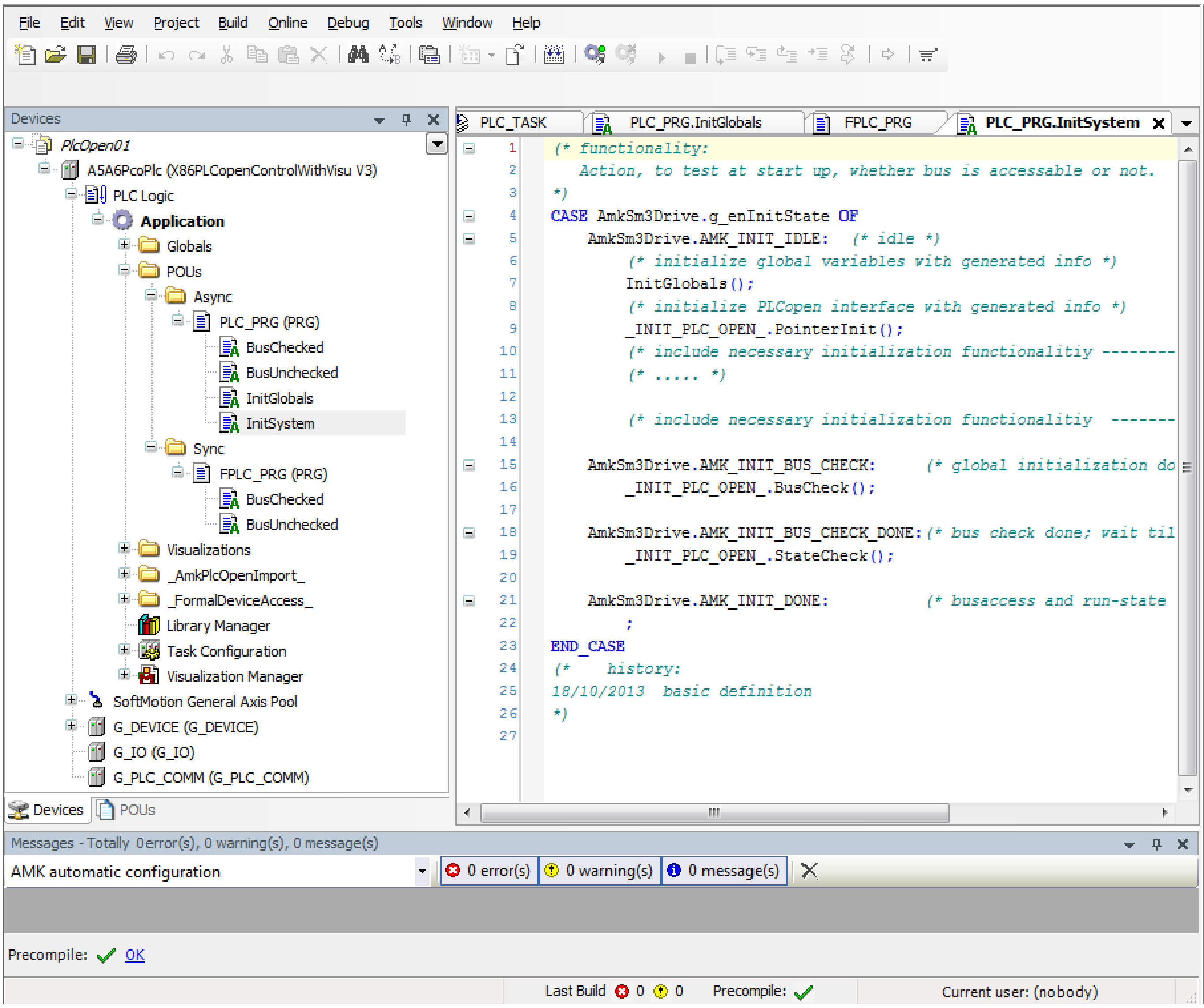
PLC template (PLC_PRG):



PLC template (FPLC_PRG):



PLC template (PLC_PRG, InitSystem action):



19.2 SoftMotion - specific bus interface

Access to the drives is essentially based on blocks from the AmkEasyDev library or the AmkDevAccess library. All drives that can be reached with these blocks can be accessed. Currently, these are drives which support the necessary functional scope as defined in the EtherCAT standard or AMK's ACC/AFP protocol. The two bus systems can be operated in parallel. For the implementation of the AMK-specific interface, in the context of the AmkSm3 drive library, the specific AMK function has been added to blocks from the SM3 Basic library. This results in the following derived AMK blocks, as listed in the table:

	EXTENDS
AXIS_REF_AMK_SM3	AXIS_REF_SM3
AXIS_REF_VIRTUAL_AMK_SM3	AXIS_REF_VIRTUAL_SM3
AXIS_REF_LOGICAL_AMK_SM3	AXIS_REF_LOGICAL_SM3
FREE_ENCODER_REF_AMK	FREE_ENCODER_REF

Detailed knowledge of these blocks is not necessary, as they are called implicitly by the controller system software when a device is selected (see the 'PLCopen devices by AMK, part 1' figure or the 'PLCopen devices by AMK, part 2' figure).

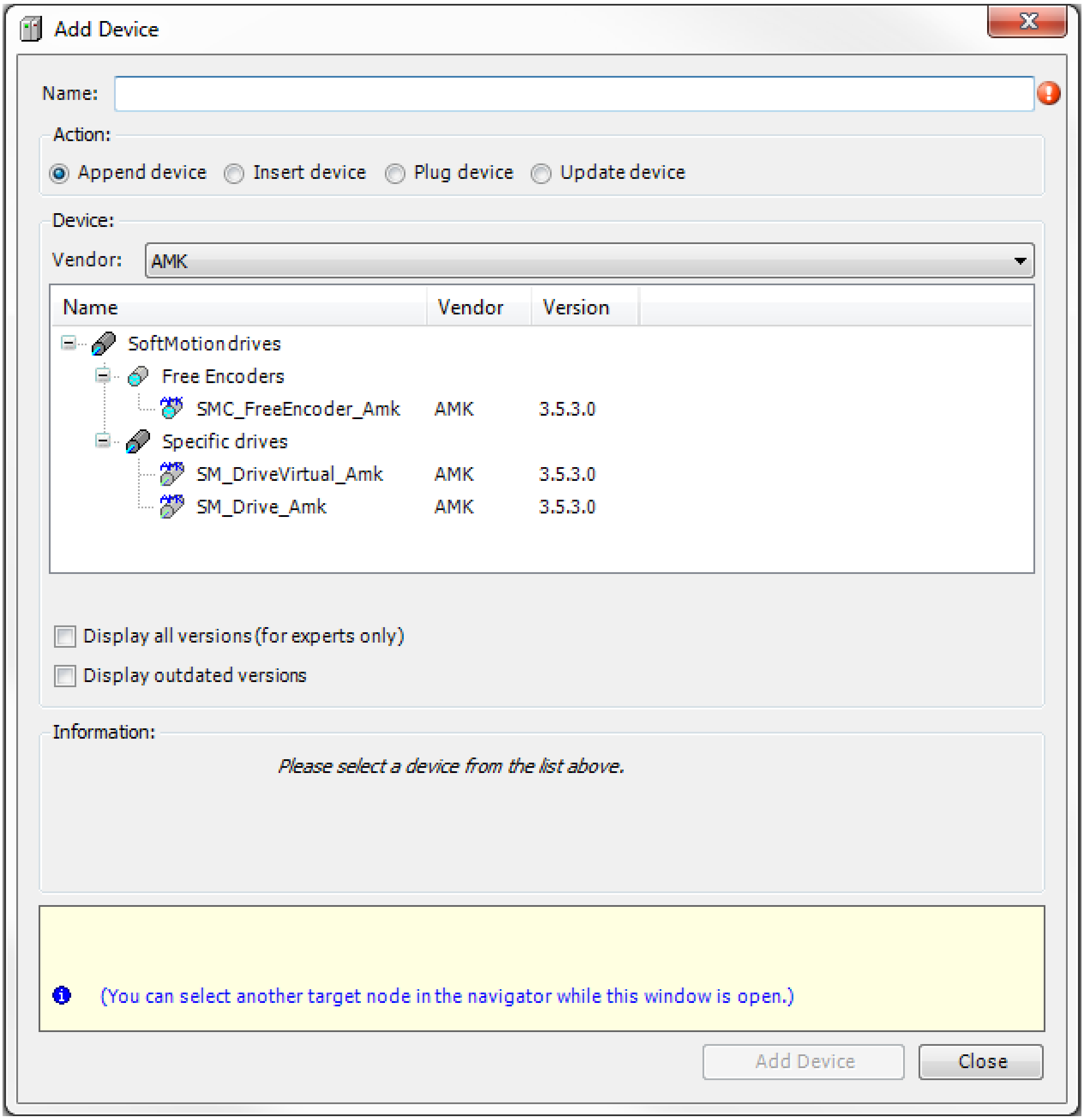
The assignments of the devices to the corresponding AMK blocks, along with their function, are listed in the following table.

Device identifier	Block name	Comment
SM_Drive_Amk	AXIS_REF_AMK_SM3	Physical axis (drive)
SM_DriveVirtual_Amk	AXIS_REF_VIRTUAL_AMK_SM3	Virtual axis

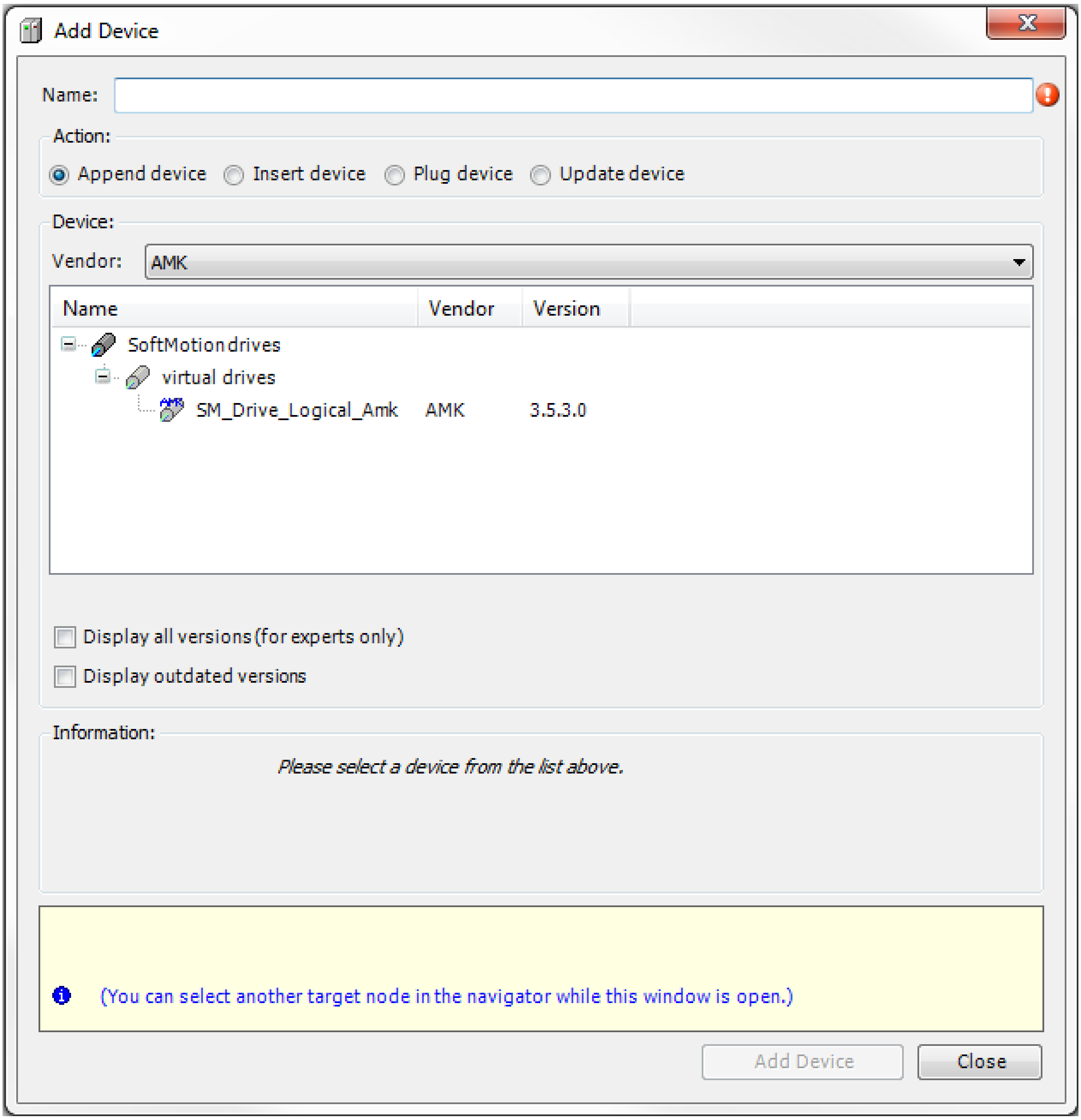
Device identifier	Block name	Comment
SM_Drive_Logical_Amk	AXIS_REF_LOGICAL_AMK_SM3	Logical axis
SMC_FeeEncoder_Amk	FREE_ENCODER_REF_AMK	Free encoder

A physical axis can also be operated virtually for test purposes by activating "virtual mode" in the base parameters (see the 'Base parameters' figure).

PLCopen devices by AMK, part 1:



PLCopen devices by AMK, part 2:



19.2.1 Variables

19.2.2 AXIS_REF_AMK_SM3

19.2.2.1 Extended local variables

Name	Type	Comment
iHomingState	INT	Internal use
iErrorState	INT	Internal use
reWait	LREAL	Internal use
reHomingDelay	LREAL	Default: 0.5 [s]
boUseProbe	BOOL	Default: TRUE
fbTon	TON	Internal use
fbEasyDevice	EASY_DEVICE	Internal use
fbEasyControl	EASY_CONTROL	Internal use

Name	Type	Comment
fbAmkProbeAccess	AmkProbeAccess	Internal use
fbReadNIdsDint	READ_N_IDS_DINT	Internal use
stInitIdValues	ST_N_ID_VALUES	Internal use
usiAcyclicCommand	USINT	Internal use
usiAcyclicState	USINT	Internal use
fbFtrig	F_TRIG	Internal use

19.2.2.2 Global variables

Name	Type	Comment
g_enInitState	EN_INIT_STATE	Internal use

19.2.3 User blocks

19.2.3.1 AMK_GetSpecialInfo

The 'AMK_GetSpecialInfo' block is used to display information special 'AXIS_REF_AMK_SM3' information.

The 'idle' state is the prerequisite for the reactivation of UE through MC_Power.bRegulaterOn (e.g. through MC_Reset.Execute after error reset).

User interface

Input variables

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>Enable</td><td>BOOL</td><td>Enable signal: With a positive edge, the initialisation of the block starts.
As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC.
In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.</td></tr>
<tr><td>iSelect</td><td>INT</td><td>Information selection
<table>
<tr><th>Range</th><th>Meaning</th></tr>
<tr><td>0</td><td>Get QUE (where QUE = DC bus enable acknowledge). The "BOOL" type information is displayed at the bStatus output.</td></tr>
<tr><td>1</td><td>Get UE graphs (0 = idle). The "DINT" type information is displayed at the diStatus output.</td></tr>
</table></td></tr>
</table>

Output variables

Name	Type	Description
bStatus	BOOL	Display binary information
diStatus	DINT	Displays integer values
fStatus	LREAL	Display of floating point values
Error	BOOL	Error signal to indicate errors

<table>
<tr><th>Name</th><th>Type</th><th colspan="3">Description</th></tr>
<tr><td rowspan="4">ErrorID</td><td rowspan="4">INT</td><td colspan="3">Error identity number: Diagnostic number is output</td></tr>
<tr><td colspan="2">iErrID = 0</td><td>No error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = TRUE</td><td>Error</td></tr>
<tr><td>iErrID ≠ 0</td><td>boErr = FALSE</td><td>Warning</td></tr>
</table>

Input and output variables

Name	Type	Description
Axis	AXIS	AXIS_REF_AMK_SM3

19.2.4 Performance features

• bRegulaterOn (MC_Power), to switch UE (DC bus enable)

Note: If the drive is functioning as the ACC master for the KE, bRegulaterOn automatically switches UE (DC bus enable) for the KE.

• bDriveStart (MC_Power), to switch RF (inverter on).

• Status (MC_Power) changes to:

TRUE for "bRegulaterOn AND QUE AND QRF" (where QUE = acknowledge DC bus enable and QRF = acknowledge inverter on); FALSE otherwise.

• Detecting and clearing drive errors.

• Drive errors can be cleared with MC_Reset.

Prerequisite: "bRegulaterOn = FALSE" (MC_Power).
“bDriveStart = FALSE" (MC_Power).

• The 'AMK_GetSpecialInfo' block queries information specific to AMK.

• Reading and writing of SoftMotion parameters.

Overview of SoftMotion parameters

Parameter	Axis variable
1030	bError
1031	wErrorID
1032	bErrorAckn
1091	byControllerMode
1092	byRealControllerMode
1, 1100	fSetPosition
1101	fActPosition
11, 1110	fSetVelocity
10, 1111	fActVelocity
9, 1112	fMaxVelocity
1120	fSetAcceleration
1121	fActAcceleration
13, 1122	fMaxAcceleration
1130	fSetDeceleration
1131	fActDeceleration
15, 1132	fMaxDeceleration
1140	fSetJerk
1141	fActJerk
16, 1142	fMaxJerk
1151	fActCurrent

Parameter	Axis variable
1152	fMaxCurrent
1153	fSWMaxCurrent
1160	fSetTorque
1161	fActTorque
1162	fMaxTorque
1202	fCaptPosition
1206	bHWLimitEnable
1207	bCaptureOccured
1208	bStartCapturing
1210	bStartReference
1211	fReference
1220	fFirstCapturePosition
1221	fLastCapturePosition
1223	bCaptureWindowActive

• Reading and writing of drive parameters. Siehe 'Specific drive parameter access' auf Seite 531.

• Support of linear and rotary axes.

• Drive-internal homing cycle, based on homing cycle parameters according to ID41 'Homing velocity', ID147 'Homing parameter', ID150 'Homing offset 1', ID32926 'AMK homing cycle parameter'

The input variable position of the 'MC_Home' block is applied temporarily to ID153 'Spindle angle position'.

• Touch probe support, based on 'MC_TouchProbe' (see the 'Example for program-based detection of touch probe 1' figure)

ID169 'Probe control parameter' must be preassigned (<> 0); however, only one edge (positive or negative) per touch probe may be selected in each case (see bit assignment table for ID169 'Probe control parameter').

Bit assignment of ID169 'Probe control parameter'

Bit3	Bit2	Bit1	Bit0
Negative edge touch probe 2	Positive edge touch probe 2	Negative edge touch probe 1	Positive edge touch probe 1

iTriggerNumber :=1 touch probe 1 (if supported by the hardware!)

iTriggerNumber :=2 touch probe 2 (if supported by the hardware!)

Example for program-based detection of touch probe 1:

```
(* touch probe *)
stTriggerInput: TRIGGER_REF := (
    bFastLatching:=TRUE,
    iTriggerNumber:=1);
fbMC_TouchProbe: MC_TouchProbe;
```

```
(* touch probe *)
fbMC_TouchProbe(
    Execute:= ,
    WindowOnly:= ,
    FirstPosition:= ,
    LastPosition:= ,
    Axis:= Axis,
    TriggerInput:=stTriggerInput ,
    Done=> ,
    Error=> ,
    ErrorID=> ,
    RecordedPosition=> ,
    CommandAborted=> );
```

The prerequisites for the touch probe function are:

ID32980 'Port 3 Bit 2'= 401 (touch probe 1)

ID32979 'Port 3 Bit 1'= 402 (touch probe 2; only available for KW-R05 / KW-R06)

ID169 'Probe control parameter', ID32979 'Port 3 Bit 1' (or ID32980 'Port 3 Bit 2') must always be set, i.e.even if the 'MC_TouchProbe' block is not being used!

Exception: The touch probe function is deselected with the property "ProbeEnable:=FALSE".

19.2.5 Initialization

19.2.5.1 wCommunicationState

wCommunicationState	Comment
0	Initial state: Initialization not yet underway. Waiting for: "g_enInitState = AMK_INIT_BUS_ CHECK_DONE"
1	Activate reading of relevant drive ID (ID116 'Resolution motor encoder', ID121 'Load gear input revolution', ID122 'Load gear output revolution', ID169 'Probe control parameter').
2	Wait for reading of drive ID to be completed.
3	Wait for reading of drive ID to be completed; in the event of an error.
10	Transition to 'wCommunicationState': =80.
80	Definition of standardization factors for velocity and torque.
99	Transition to "operational" state.
100	"Operational" state.
200-210	Rest axis group.

19.2.5.2 Error

wCommunicationState	Comment
0	Initial state is not exited and "g_enInitState = AMK_INIT_BUS_CHECK ": Wait for "Bus starting up" is not exited: The bus does not switch to "data exchange mode" (bus error).
1001	"pstDevice" not initialized.
1003	Error reading ID.

19.2.6 Specific drive parameter access

In the context of the 'MC_ReadParameter' and 'MC_WriteParameter' blocks, the corresponding (positive) drive ID value can be read and written with a negative 'ParameterNumber'. The temporary value (data) of the ID is used.

Independently of this, the blocks in the 'AmkSystem' library can be used to gain full access to the drive parameters (including, for example, access to listen IDs, reading all ID elements in a Sercos, ID, etc.).

20 Appendix

20.1 Error bit information

Regardless of the type of access (READ_SDO / WRITE_SDO or READ_ID / WRITE_ID blocks), the following error codes describe the errors during data transport:

Error code	Error code from PLC (iErrID)	Description
0x00000002	0x0002	General error message
0x00000003	0x0003	Source module not available
0x00000004	0x0004	The addressed destination does not exist (routing address is incorrect)
0x00000005	0x0005	Memory errors
0x00000006	0x0006	Wrong module number
0x00000007	0x0007	Wrong element
0x00000008	0x0008	Resource error
0x00000009	0x0009	Protocol error (command)
0x0000000A	0x000A	Unused
0x0000000B	0x000B	Timeout
0x0000000C	0x000C	Internal error
0x0000000D	0x000D	Unknown command
0x0000000E	0x000E	Unused
0x0000000F	0x000F	Internal error
0x00000016	0x0016	No connection to target
0x00000017	0x0017	Error in 'Login', device already used

Valid for EtherCAT SOE

ID access (blocks READ_ID / WRITE_ID)

The error codes from the SOE slave device have the following meaning:

Error code	Error code from PLC (iErrID)	Description
0x00000000	0x0000	No error
0x00001001	0x1001	ID number not available
0x00001009	0x1009	Invalid access to element 1
0x00002001	0x2001	Name does not exist
0x00002002	0x2002	Name transmitted too short
0x00002003	0x2003	Name transmitted too long
0x00002004	0x2004	Name can not be changed
0x00002004	0x2004	Name is currently write protected
0x00003001	0x3001	Attribute does not exist
0x00003002	0x3002	Attribute transmitted too short
0x00003003	0x3003	Attribute transmitted too long
0x00003004	0x3004	Attribute can not be changed
0x00003005	0x3005	Attribute is currently write protected
0x00004001	0x4001	Unit not available
0x00004002	0x4002	Unit transmitted too short
0x00004003	0x4003	Unit transmitted too long
0x00004004	0x4004	Unit can not be changed
0x00004005	0x4005	Unit is currently write protected

Error code	Error code from PLC (iErrID)	Description
0x00005001	0x5001	Minimum input value not available
0x00005002	0x5002	Minimum input value transmitted too short
0x00005003	0x5003	Minimum input value transmitted too long
0x00005004	0x5004	Minimum input value can not be changed
0x00005005	0x5005	Minimum input value is currently write protected
0x00006001	0x6001	Maximum input value not available
0x00006002	0x6002	Maximum input value transmitted too short
0x00006003	0x6003	Maximum input value transmitted too long
0x00006004	0x6004	Maximum input value can not be changed
0x00006005	0x6005	Maximum input value is currently write protected
0x00007002	0x7002	Operating date transmitted too short
0x00007003	0x7003	Operating date transmitted too long
0x00007004	0x7004	Operating date can not be changed
0x00007005	0x7005	Operating date is currently write protected
0x00007006	0x7006	Operating date is less than the minimum input value
0x00007007	0x7007	Operating date is greater than the maximum input value
0x00007008	0x7008	Invalid operating date
0x00007009	0x7009	Operating date is write protected by password.
0x0000700A	0x700A	Operating date is write protected as a result of cyclic usage
0x0000700B	0x700B	Unauthorized indirect addressing
0x0000700C	0x700C	Operation date write protected as a result of other defaults (e.g., operating mode, ..)
0x0000700D	0x700D	Invalid floating number
0x0000700E	0x700E	Operating date write protected during 'parameterization level'
0x0000700F	0x700F	Operating date write protected during 'operating level'
0x00007010	0x7010	Procedure command already active
0x00007011	0x7011	Procedure command can not be interrupted
0x00007012	0x7012	Procedure command can not be executed at this time
0x00007013	0x7013	Procedure command can not be executed (invalid or incorrect parameters)
0x00008009	0x8009	General access error

Valid for EtherCAT COE and ACC

Index / sub-index access (blocks READ_SDO / WRITE_SDO)

The error codes from the COE / ACC slave device have the following meaning:

Error code	Error code from PLC (iErrID)	Description
0x05030000	0x5300	Toggle bit not changed
0x05040000	0x5400	SDO protocol timed out
0x05040001	0x5401	SDO Command Specifier invalid or unknown
0x05040002	0x5402	Invalid block size (Block Transfer mode only)
0x05040003	0x5403	Invalid sequence number (Block Transfer mode only)
0x05030004	0x5304	CRC error (Block Transfer mode only)
0x05030005	0x5305	Out of memory
0x06010000	0x6100	Access to this object is not supported
0x06010001	0x6101	Attempt, to write to a Write_Only parameter
0x06010002	0x6102	Attempt, to write to a Read_Only parameter
0x06020000	0x6200	Object is not present in the object directory
0x06040041	0x6441	Object can not be mapped to PDO

Error code	Error code from PLC (iErrID)	Description
0x06040042	0x6442	The number and / or the length of the mapped objects would exceed the PDO length
0x06040043	0x6443	General parameters Incompatibility
0x06040047	0x6447	General internal error in the device
0x06060000	0x6600	Access due to hardware failure aborted
0x06070010	0x6710	Data type or parameter length do not match or are unknown
0x06070012	0x6712	Data type does not match, parameter length too long
0x06070013	0x6713	Data type does not match, parameter length too short
0x06090011	0x6911	Sub-index not available
0x06090030	0x6930	General value range error
0x06090031	0x6931	Value range error: Parameter value too large
0x06090032	0x6932	Value range error: Parameter value too small
0x06090036	0x6936	Maximum value is less than minimum value
0x060A0023	0x6A23	Resource not available
0x08000000	0x8000	General error
0x08000020	0x8020	Data cannot be transferred or stored to the application
0x08000021	0x8021	Access not possible due to local application
0x08000022	0x8022	Can not access due to current device status
0x08000023	0x8023	Object Dictionary dynamic generation fails or no Object Dictionary is present (e.g. Object dictionary is generated from file and generation fails because of a file error)

20.2 Table 1: Global AmkFile function block error codes

Error code	Meaning
9	Invalid path or device
11	Global error - a more detailed description of this error cause does not exist
12	Global error - a more detailed description of this error cause does not exist
15	Too many files - there is no more memory capacity available in the file table
16	No more files found - the 'FIND_FILE' cannot find any files with this search criterion
19	File not found - the file name used could not be found in the system
26	Access denied - access is not possible at the current time, e.g. due to file access from another source
28	Directory is empty
29	Invalid directory
31	Data medium full - there is no more memory capacity available on the data medium
32	Disk full - there is no more memory capacity available
46	Directory already exists

Glossary

A

A1
Analog input 1

Ax-PCO
PLCopen

Ax-VIS
Web visualization

A-SIP
EtherNET/IP slave interface

A-SCN
CAN /ACC bus slave interface

A-SPN
Profinet IO Device interface

A-SEC
EtherCAT slave interface

A-MEC
EtherCAT master interface

Ax-PNC
Numerical Control Motion

A-SPB
Profibus DP slave interface

ASCII
American Standard Code for Information Interchange

ARRAY
List with equal format elements

AIPEX
AMK startup and parameterizing software (PC software): Programming, parameterization, configuration, diagnosis, oscilloscope, status information

AFP
AMK fieldbus protocol for drive control (e.g. homing, relative Positioning, digital speed control etc.)

ACC
AMK CAN Communication (CAN bus interface with standard CANopen protocol DS301 and additional hardware synchronization signal)

A4 / A5 / A6
AMKAMAC controller A4 / A5 / A6

C

CAN
Controller Area Network

CRC
Cyclic redundancy check (Checksum)

D

Default
Factory setting

E

EtherCAT
Real-time Ethernet bus

F

FB
Function block

FL
Command (Causes a new system run-up)

FPLC_PRG
Real-time PLC task, synchronized to device cycle

G

g_yourDevice
Symbolic name of a device in a PLC project. The name is defined in CoDeSys configuration: devices

I

iSA-VIS
Web visualization

iSA-PNC
Numerical Control Motion

ID
Parameter identification numbers acc. to SERCOS Standard

i²t
Integral of the squared current over time

I/O
Input / output

iSA-PCO
PLCopen

iSA
AMKASMART decentralized controller with power supply

K

KP
Proportional gain (speed control, PID controller)

L

Latched
'To latch a value' means: 'to save a value'

latch
'To latch a value' means: 'to save a value'

O

Operational
In state operational, data are transferred cyclically via fieldbus

P

Pre-operational
In pre-operational state, the controller can access the bus participants via the service channel. No cyclic data is exchanged.

POU
Program organization unit (PLC program elements; types program, function or function block)

PMC
Printing mark control

PM
Printing mark

PLC_PRG
Task which is not synchronized to the device cycle

PDK_xxxxxx_abcdefgh
Product documentation; xxxxxx - AMK part no. , abcdefgh - name

Parameter
Identification number acc. to SERCOS standard

PGT
Periphery basic clock Fetch cycle in the basic device to which the drive controller is synchronized (The cycle time is according to ID2)

Q

QUE
Acknowledgment DC bus on; shows that DC bus is loaded

QRF
Acknowledgment controller enable; the drive is controlled in the activated operation mode

QFL
Acknowledgment clear error; the command clear error was executed

R

RF
Command 'Controller enable'; the drive is energized and will be controlled depending on the selected operation mode. Controller enable can only be set if the device is error-free (SBM = TRUE) and acknowledgement DC bus on is set (QUE = TRUE).Acknowledgement controller enable (QRF) is set.

S

SBM
System ready message; shows that the device is error-free In case of error. SBM will be reset

SDO
Service Data Object

U

UE
Command 'DC bus on' control signal to load the DC bus e.g. in KE. DC bus on can only be set if the device is error-free (SBM = TRUE). After the DC bus is loaded, the acknowledgement message QUE is set.

Your opinion is important!

With our documentation we want to offer you the highest quality support in handling the AMKmotion products.
That is why we are now working on optimizing our documentation.
Your comments or suggestions are always of interest to us.
We would be grateful if you take a bit of time and answer our questions. Please return a copy of this page to us.

e-mail: Documentation@amk-motion.com
or
fax no.: +49 7021/50 05-199

Thank you for your assistance.
Your AMKmotion documentation team

1. How would you rate the layout of our AMKmotion documentation?

   (1) very good (2) good (3) satisfactory (4) less than satisfactory (5) poor

2. Is the content structured well?

   (1) very good (2) good (3) moderate (4) hardly (5) not at all

3. How easy is it to understand the documentation?

   (1) very easy (2) easy (3) moderately easy (4) difficult (5) extremely difficult

4. Did you miss any topics in the documentation?

   (1) no (2) if yes, which ones:

5. How would you rate the overall service at AMKmotion?

   (1) very good (2) good (3) satisfactory (4) less than satisfactory (5) poor

AMKmotion GmbH + Co KG
Phone : +49 7021/50 05-0, fax: +49 7021/50 05-199
E-Mail: info@amk-motion.com
Homepage: www.amk-motion.com