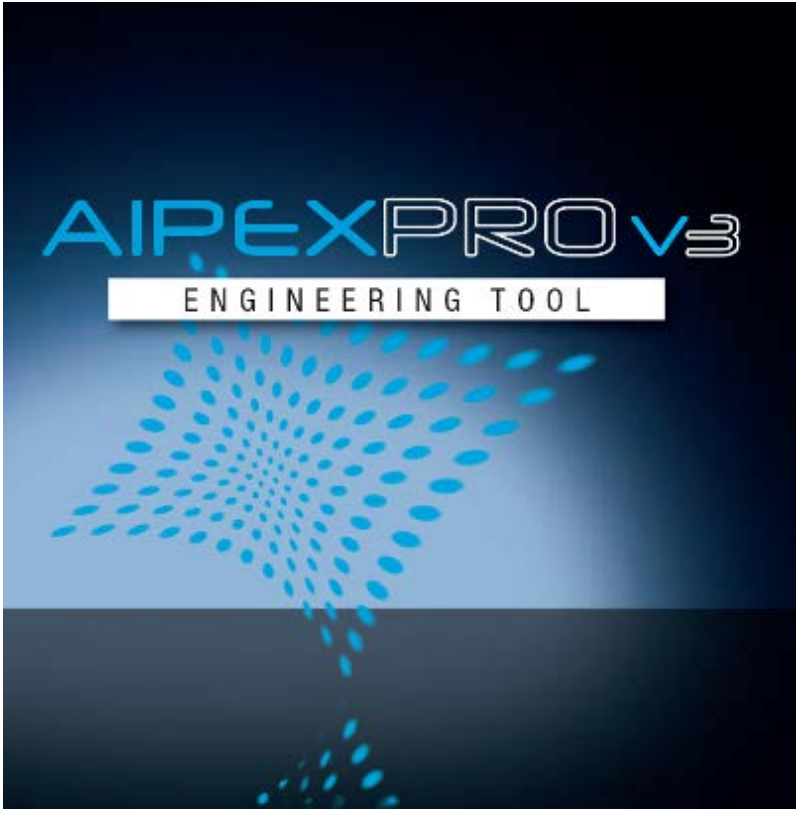


AMKmotion
Software description
AIPEX PRO V3
AFL AMKmotion function libraries

Version: 2023/09
Part no.: 204979
Translation of the "Original Dokumentation"

AMKmotion
MEMBER OF THE ARBURG FAMILY

Imprint

Name: PDK_204979_Software_AIPEX_PRO_V3_en

Version:

Version: 2023/09	
Change	**Letter symbol**
• new AMKmotion Design	LeS

Previous version: 2019/45

Product version:

Product	Firmware Version (Part no.)
PC Software	AIPEX PRO V3.04 2018/47 (207211)
Software AFL for CODESYS V2	AFL V3 Version 3.14 2015/41 (206005)
Software AFL for CODESYS V3	AFL V4 Version 3.5.5.0 2015/41 (206004)

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 About this documentation

1.1 Structure of this document

Content	Title	Chapter no.
Validity, usage and purpose of this documents	Imprint	-
	About this documentation	1
For your Safety	Basic notes	2
	Presenting safety messages	
	Class of hazard	
	Intended use	
AIPEX PRO Description of the software with practical applications.	Installation manual	3
	Service pack	
	Program overview	
	Communication PC - AMK device	
	Taps (configuration, parameter, messages, scope, diagnose)	
	Menus	
	Functions	
AFL Function Libraries Includes the description of AFL Standard and AFL Application blocks.	AFL AMK Function Libraries	4
AFL Standard blocks With AFL Standard Blocks the functionality of the AMK devices can be integrated into a PLC project. The AFL Standard blocks consist of AFL Application blocks.	Introduction AFL Standard blocks for CODESYS V3	4.1
	Introduction AFL Standard blocks for CODESYS V2	
	Description Structures (ST)	
	Description Standard blocks (FB)	
AMK AFL Application blocks AMK support with AFL Application blocks complex functions. They consists of PLC Basic blocks (separate documentation).	Library description AmkAfl.lib	4.2
	Library description AmkAflConfig.lib	
FIRST STEPS with AMK AFL Standard blocks Programming examples for CODESYS V3 and CODESYS V2 with AFL Standard blocks.	FIRST STEPS with CODESYS V3 and AFL Standard Blocks	4.3
	FIRST STEPS with CODESYS V2 and AFL Standard Blocks	
Prefabricated visualization elements for AFL function blocks.	Visualization of AFL blocks	4.4
Abbreviations and explanation of terms	Glossary	-

1.2 Keeping this document

This document must permanently be available and readable at the place where the product is in use. If the product is used at another place or changed the owner, the document must be passed on.

1.3 Target group

Any person that is qualified and intends to work with this product must read, understand and follow this document:

- Projecting
- Connection
- Parameterization
- Startup
- Service and repair

- Replacement
- Software installation
- PLC programming

1.4 Purpose

This document is addressed to any person who handles the product. It gives information about the following topics:

- Startup and operation
- Diagnosis
- Software installation
- FIRST STEPS description

1.5 Display conventions

Display	Meaning
	This symbol points to parts of the text to which particular attention should be paid!
	The red hand symbol indicates the button or menu item to click on. The red hand symbol indicates the option to be selected.
 RMB	Click the right mouse button
0x	0x followed by a hexadecimal number, e. g. 0x500A
'Names'	Names are represented with apostrophes e. g. parameters, variables, etc.
'Text'	Menu items and buttons in a software or on a controller, e. g.: Click the **'OK'** button in the **'Options'** menu to call up the 'Delete PLC program' function
→	Task procedure / operating sequence, e. g. **'Start'** → **'All programs'** → **'Additional'** → **'Editor'** e. g. 0 → 1 edge
See 'chapter name' on page x	Executable cross-reference in electronic output media

1.6 Appendant documents

Startup descriptions

AMK part-no.	Title
204539	Initial startup KE/KW
204737	Initial startup of decentralized drives
204665	Connection of third-party controllers to AMK drives

Software descriptions

AMK part-no.	Title
205210	Software description AmkLibraries
204072	AIPEX PRO add in gateway for TwinCAT
203771	Software description ATF - AMK Tool Flasher
205171	AIPEX PRO Startup

1.7 Translation help for screenshots

German	English
Abbrechen	Cancel
Beenden	Exit
Fertig stellen	Finish
Ja	Yes

German	English
Löschen	Delete
Nein	No
Öffnen	Open
Schließen	Close
Übernehmen	Accept
Weiter	Next
Zurück	Back
Zuweisen	Assign

2 For your safety

2.1 Basic notes for your safety

- At electrical drive systems, hazards are present in principle that can result in death or fatal injuries:
  - Electrical hazard (e. g. electric shock due to touch on electrical connections)
  - Mechanical hazard (e. g. crush, retract due to the rotation of the motor shaft)
  - Thermal hazard (e. g. burns due to touch on hot surfaces)
- These hazards are present while starting up and operating the unit, and also during servicing or maintenance work.
- Safety instructions in the documentation and on the product warn about the hazards.
- Personnel must have read and understood the safety instructions before installing and operating the product. In the documentation about the product the usage warnings pertain to direct hazards and must therefore be followed directly when operating or handling the product by the operator.
- AMKmotion products must be kept in their original order, that means it is not allowed to do a significant constructional change on hardware side and software is not allowed to be decompiled and change the source code.
- Damaged or faulty products are not allowed to be integrated or put into operation.
- Do not start the system in which the AMKmotion products are installed (begin of intended use) until you can determine that all relevant standards, laws, and directives have been complied with, e. g. low voltage directive, EMC directive, and the machinery directive, and possible further product standards. The plant manufacturer is responsible for the compliance with the laws, directives, and standards.
- The devices must be installed, electrically connected and operated as shown in the device description documentation. The technical data and the required environmental conditions must be observed at all times.

2.2 Presenting safety messages

Any safety information is configured as follows:

⚠ SIGNAL WORD	
	**Type and source of risk** Consequence(s) of non-observance **Steps to prevent:** • ...

2.3 Class of hazard

Safety and warning messages are graduated into classes of hazard (according to ANSI Z535). The class of hazard defines the potential risk of harm and is described by a single word, if the safety information is ignored. The signal word is followed by a safety alert symbol (ISO 3864, DIN EN ISO 7010). In accordance with ANSI Z535, the following signal words are used to define the class of hazard.

Safety alert symbol and signal word	Class of hazard and its meaning
⚠ DANGER	DANGER indicates a hazardous situation which, if not avoided, **will** result in death or serious injury
⚠ WARNING	WARNING indicates a hazardous situation which, if not avoided, **could** result in death or serious injury
⚠ CAUTION	CAUTION, used with the safety alert symbol, indicates a hazardous situation which, if not avoided, **could** result in minor or moderate injury
NOTICE	NOTICE is used to address preventions to avoid material damage, but not related to personal injury.

2.4 Intended use

The AMK 'AIPEX PRO V3' software supports the startup of AMK drive systems. AMK drive systems connected with an electronic nameplate are detected and identified automatically. At the same time, all the important data of the devices are displayed. With the AMK 'AIPEX PRO V3' software and integrated 'CODESYS V3' software, you can configure, optimize, diagnose, program and start up AMK drive systems.

3 AIPEX PRO

3.1 Installation manual

3.1.1 Requirement

Recommended PC requirement

- Processor: Pentium V, Centrino > 3,0 GHz, Pentium M > 1,5 GHz
- Available hard disc space 1 GB or higher
- RAM 1024 MB or higher

Supported operation systems

- Windows 10
- Windows 8 and 8.1
- Windows 7
- Windows XP with SP3 (SP1 must be installed before SP3 become installed)

Necessary .NET versions

SafePMT needs Microsoft .NET Framework 3.5
CODESYS V3 needs Microsoft .NET Framework 4.0

Administrator rights are required for the installation.

3.1.2 Important hints

An old version of 'SafePMT ' must first be manually removed.

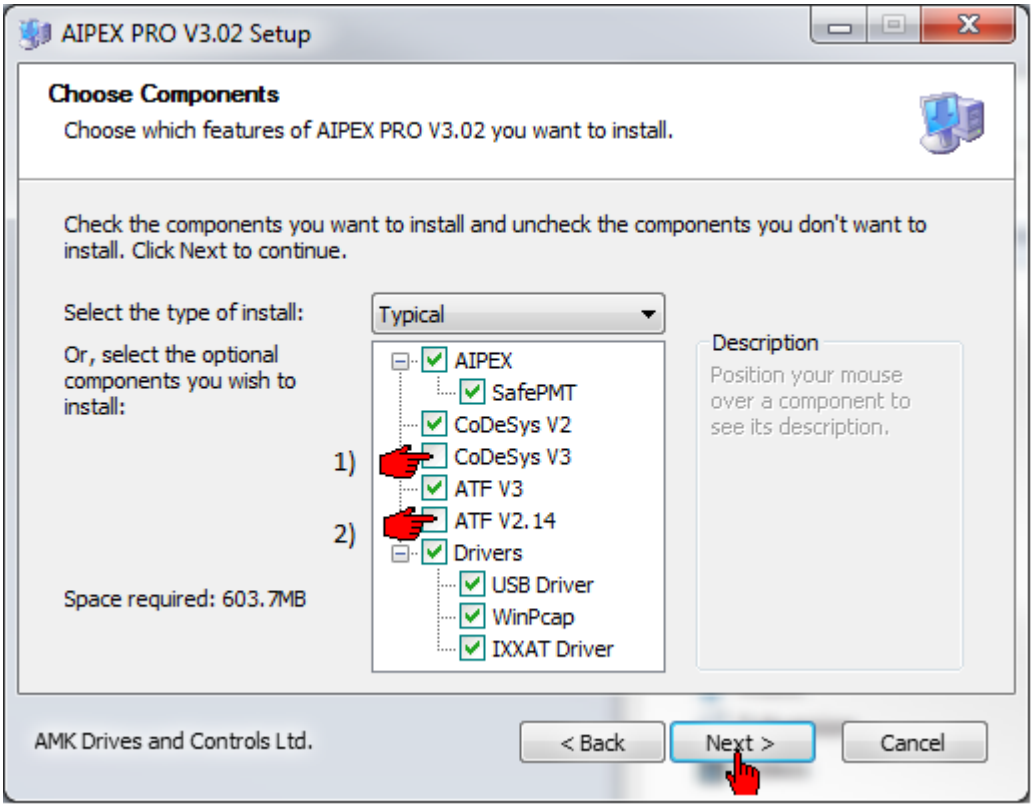
If you deinstall AIPEX PRO from your PC, all installed drivers (WinPcap, USBCOM driver and IXXAT driver) will be removed.

3.1.3 Installation

The 'AMK AIPEX PRO Installer' starts automatically after insertion the CD.
Call manually the file 'SetupAipexPro.exe' if the automatically CD start is deactivated.

Follow the 'AIPEX PRO Installer instructions'


AIPEX PRO V3.00 Setup
Welcome to the AIPEX PRO V3.00 Setup Wizard
This wizard will guide you through the installation of AIPEX PRO V3.00.
It is recommended that you close all other applications before starting Setup. This will make it possible to update relevant system files without having to reboot your computer.
Click Next to continue.
Next >
Cancel


AIPEX PRO V3.02 Setup
Choose Components
Choose which features of AIPEX PRO V3.02 you want to install.
Check the components you want to install and uncheck the components you don't want to install. Click Next to continue.
Select the type of install:
Typical
Or, select the optional components you wish to install:
AIPEX
SafePMT
CoDeSys V2
1)
CoDeSys V3
ATF V3
2)
ATF V2.14
Drivers
USB Driver
WinPcap
IXXAT Driver
Description
Position your mouse over a component to see its description.
Space required: 603.7MB
AMK Drives and Controls Ltd.
< Back
Next >
Cancel

1) From version AIPEX PRO V3.02

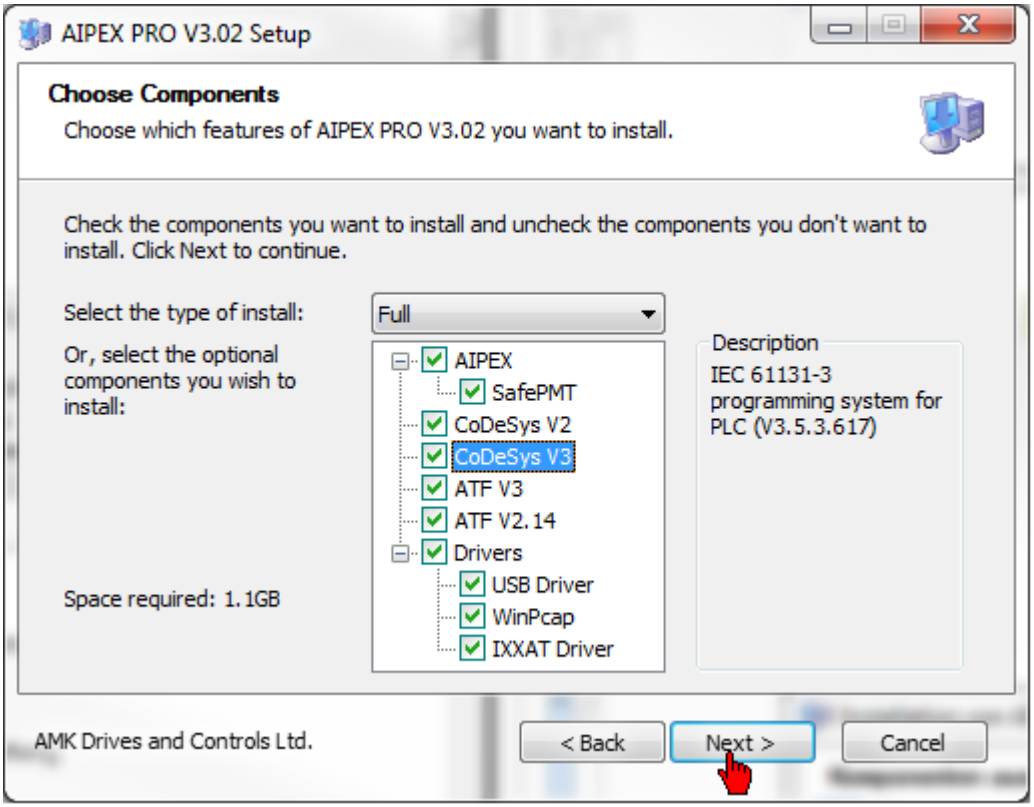
CODESYS V3 is not installed automatically. Please select CODESYS V3 if you want to use.

2) Software ATF version 2.14

IDT4 and KWZ requires the ATF version V2.14. The version 2.14 is part of the installation package of AIPEX PRO but must be selected explicit during installation process.

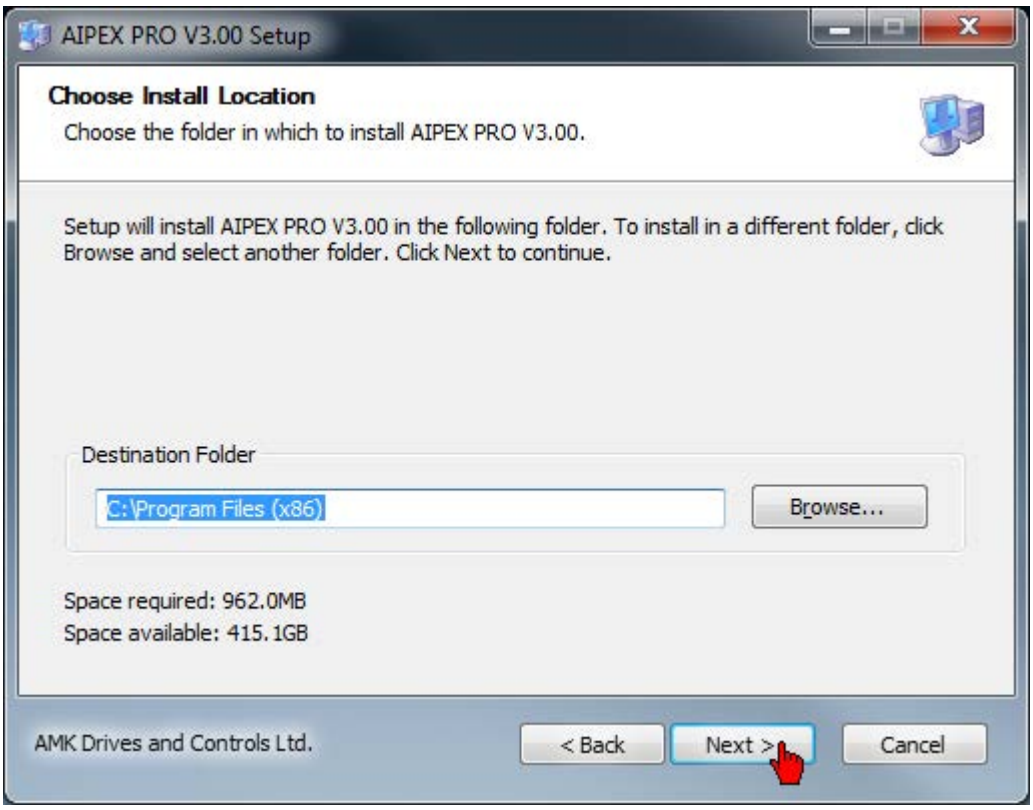
The ATF version 2.14 and 3.x can be installed both. If ATF 2.14 is started, no software ATF up from V3.0 and other parts of AIPEX PRO 3.x can be started at the same time.

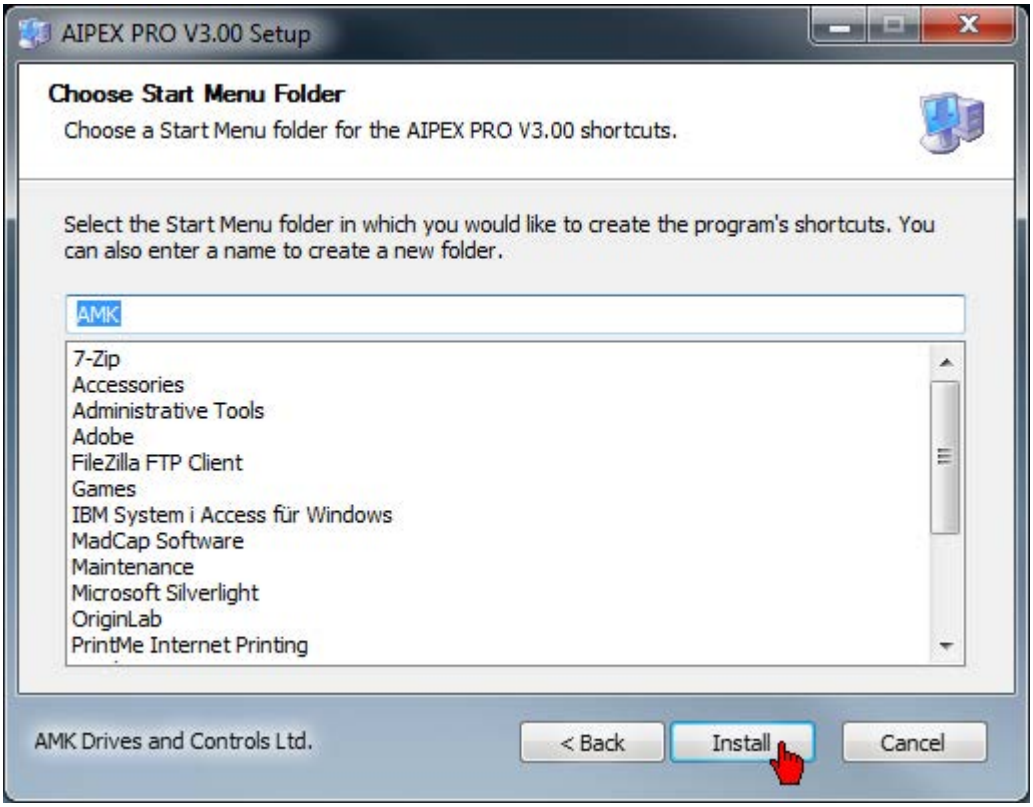
All components of AFT V2.14 are installed inside the local folder of this programm (C:\Program Files (x86)\ATF V2.14) and not into the windows folder (C:\Windows\System32)

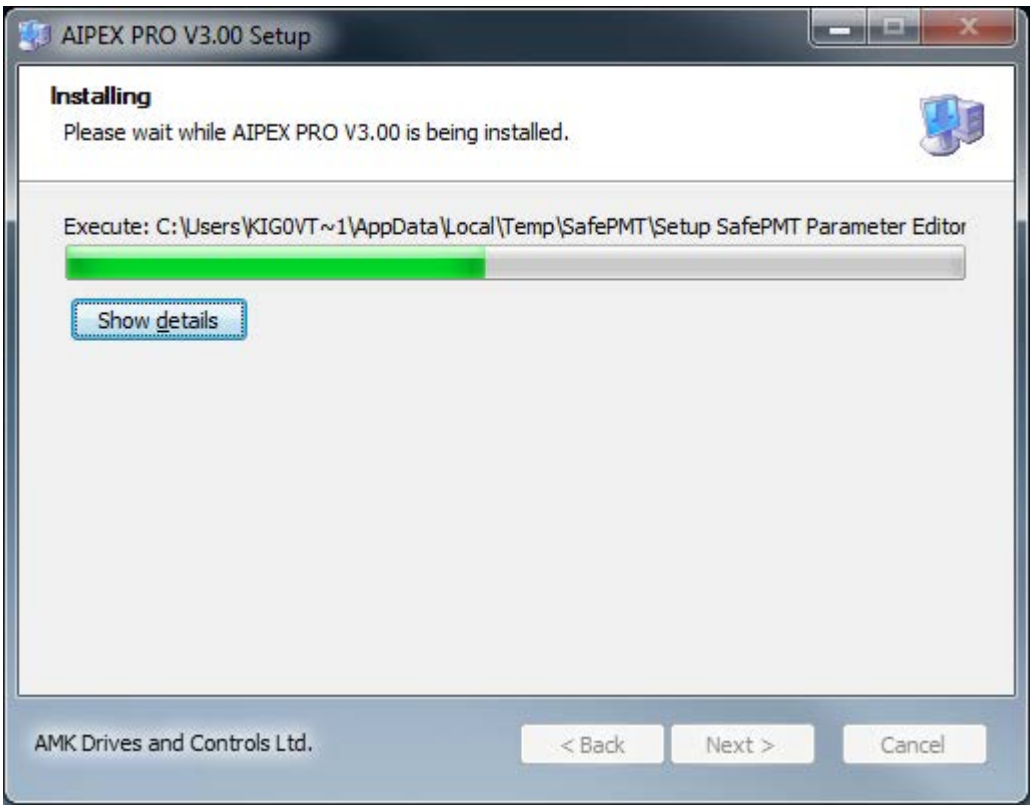


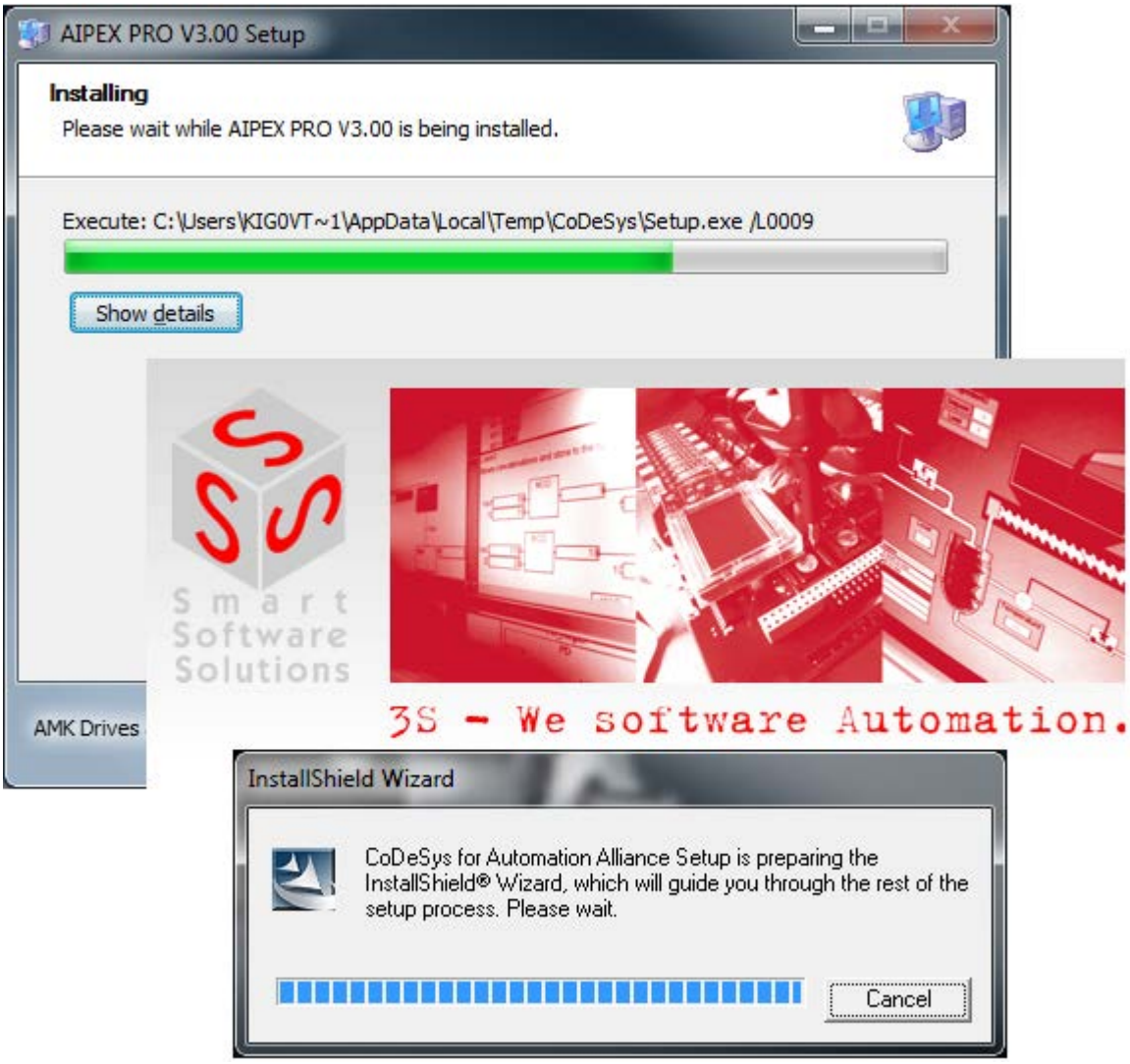
AIPEX: Commissioning and parameter explorer
SafePMT: Safe parameter editor for devices with 'Safety functions'
CODESYS V2: IEC 61131-3 programming system for PLC (CODESYS V2)
CODESYS V3: IEC 61131-3 programming system for PLC (CODESYS V3)
ATF: Tool for firmware update
USBCOM: Driver for serial communication over USB
WinPcap: Windows packet capture library for access over EtherCAT
IXXAT: Driver for USB-to-CAN addapter from IXXAT

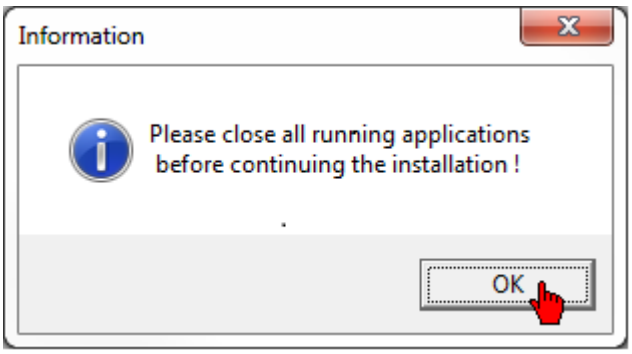
If you deinstall AIPEX PRO from your PC, all installed drivers (WinPcap, USBCOM driver and IXXAT driver) will be removed.

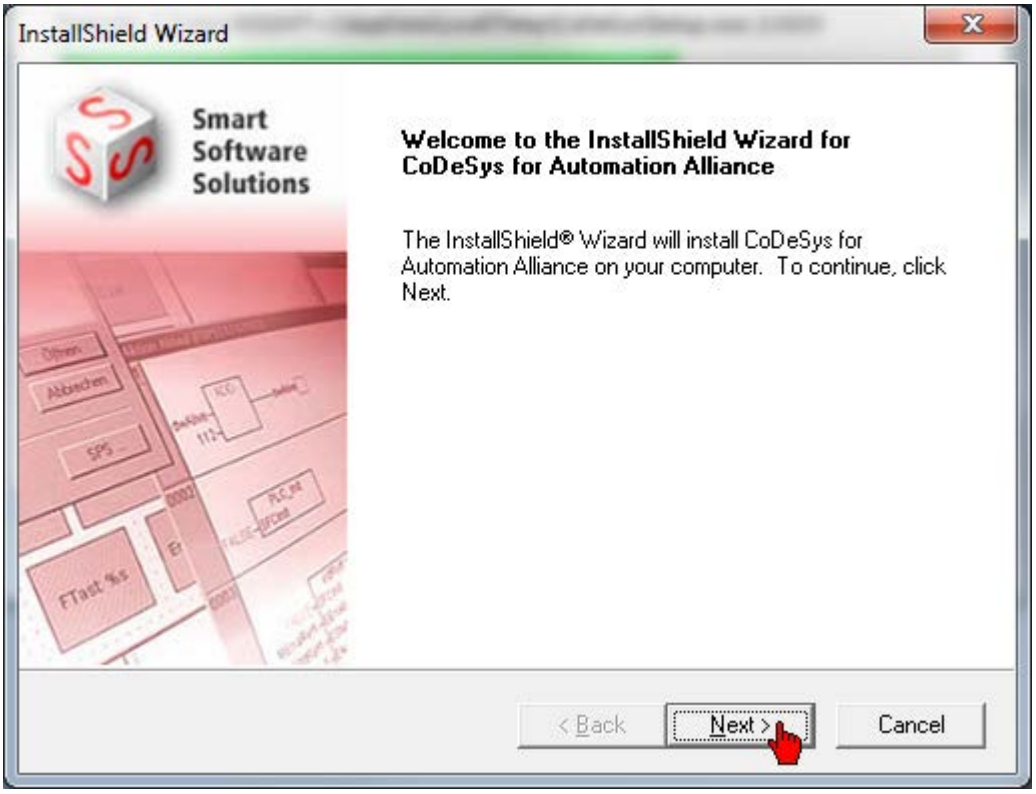

AIPEX PRO V3.00 Setup
Choose Install Location
Choose the folder in which to install AIPEX PRO V3.00.
Setup will install AIPEX PRO V3.00 in the following folder. To install in a different folder, click Browse and select another folder. Click Next to continue.
Destination Folder
C:\Program Files (x86)
Browse...
Space required: 962.0MB
Space available: 415.1GB
AMK Drives and Controls Ltd.
< Back
Next >
Cancel

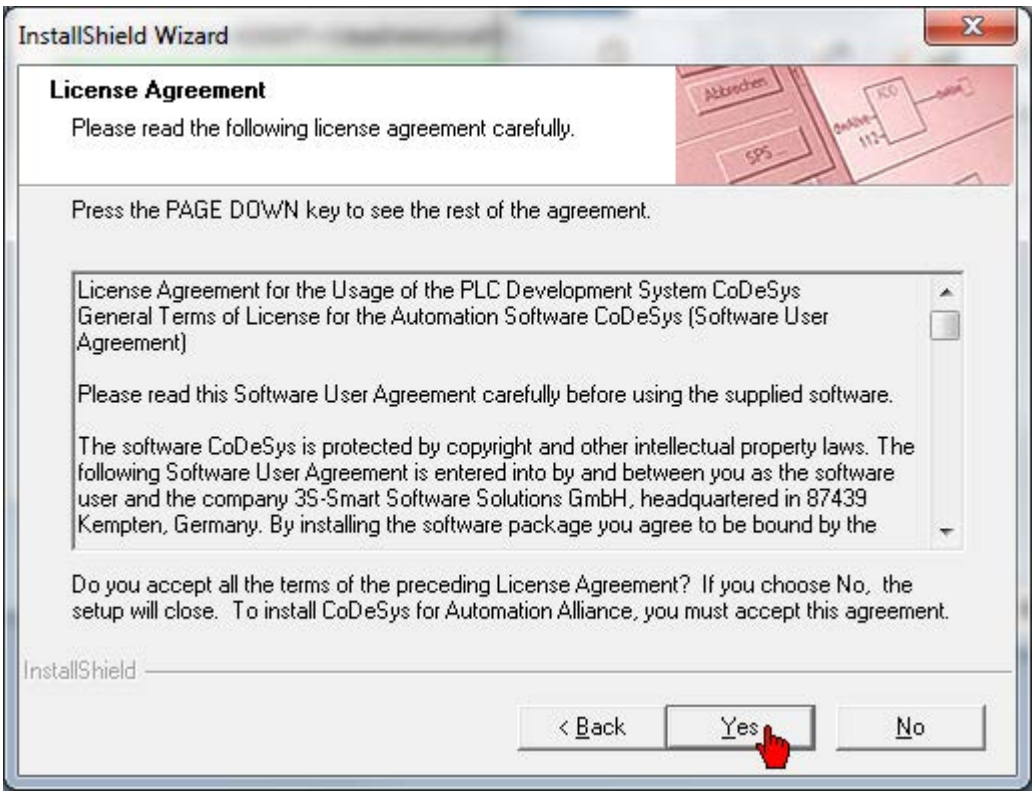

AIPEX PRO V3.00 Setup
Choose Start Menu Folder
Choose a Start Menu folder for the AIPEX PRO V3.00 shortcuts.
Select the Start Menu folder in which you would like to create the program's shortcuts. You can also enter a name to create a new folder.
AMK
7-Zip
Accessories
Administrative Tools
Adobe
FileZilla FTP Client
Games
IBM System i Access für Windows
MadCap Software
Maintenance
Microsoft Silverlight
OriginLab
PrintMe Internet Printing
AMK Drives and Controls Ltd.
< Back
Install
Cancel

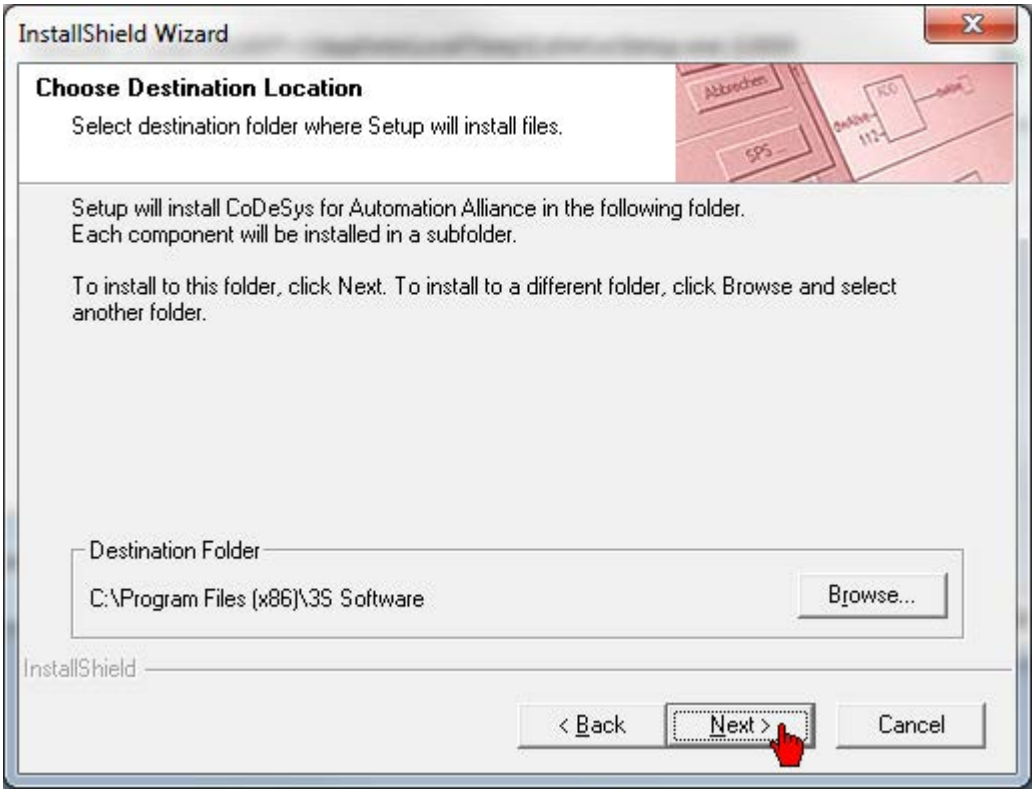

AIPEX PRO V3.00 Setup
Installing
Please wait while AIPEX PRO V3.00 is being installed.
Execute: C:\Users\KIG0VT~1\AppData\Local\Temp\SafePMT\Setup SafePMT Parameter Editor
Show details
AMK Drives and Controls Ltd.
< Back
Next >
Cancel

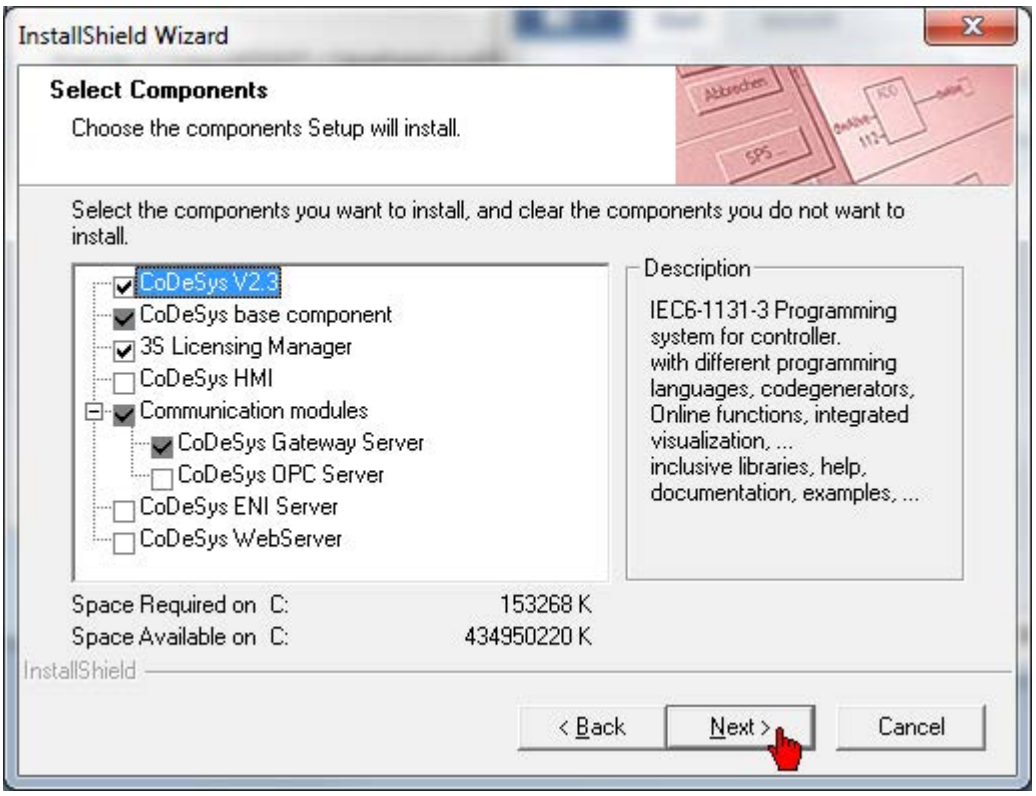

AIPEX PRO V3.00 Setup
Installing
Please wait while AIPEX PRO V3.00 is being installed.
Execute: C:\Users\KIG0VT~1\AppData\Local\Temp\CoDeSys\Setup.exe /L0009
Show details
Smart Software Solutions
3S - We software Automation.
AMK Drives
InstallShield Wizard
CoDeSys for Automation Alliance Setup is preparing the InstallShield® Wizard, which will guide you through the rest of the setup process. Please wait.
Cancel

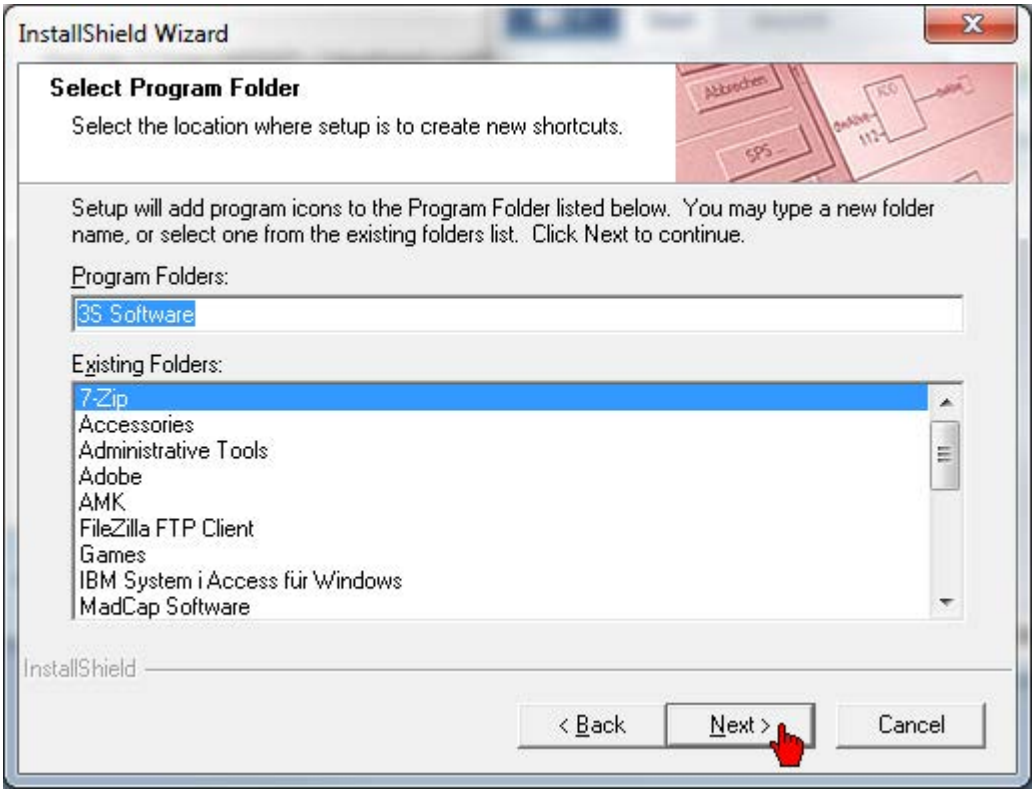

Information
Please close all running applications
before continuing the installation !
OK

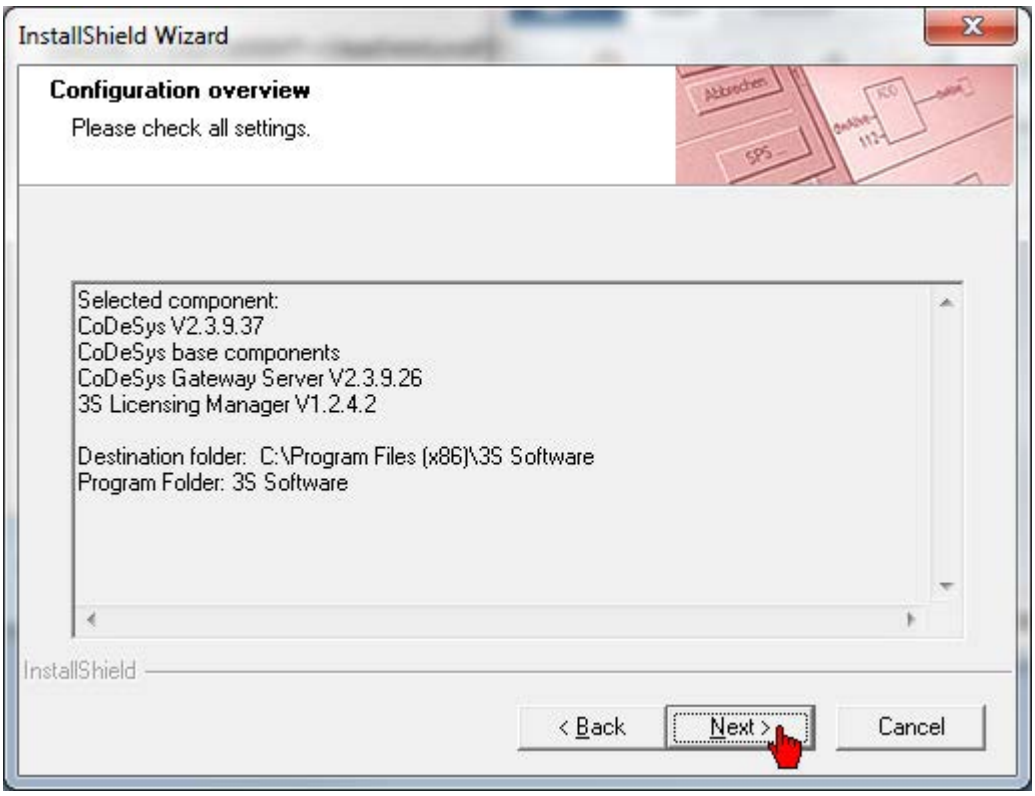

InstallShield Wizard
Smart
Software
Solutions
Welcome to the InstallShield Wizard for
CoDeSys for Automation Alliance
The InstallShield® Wizard will install CoDeSys for
Automation Alliance on your computer. To continue, click
Next.
< Back
Next >
Cancel

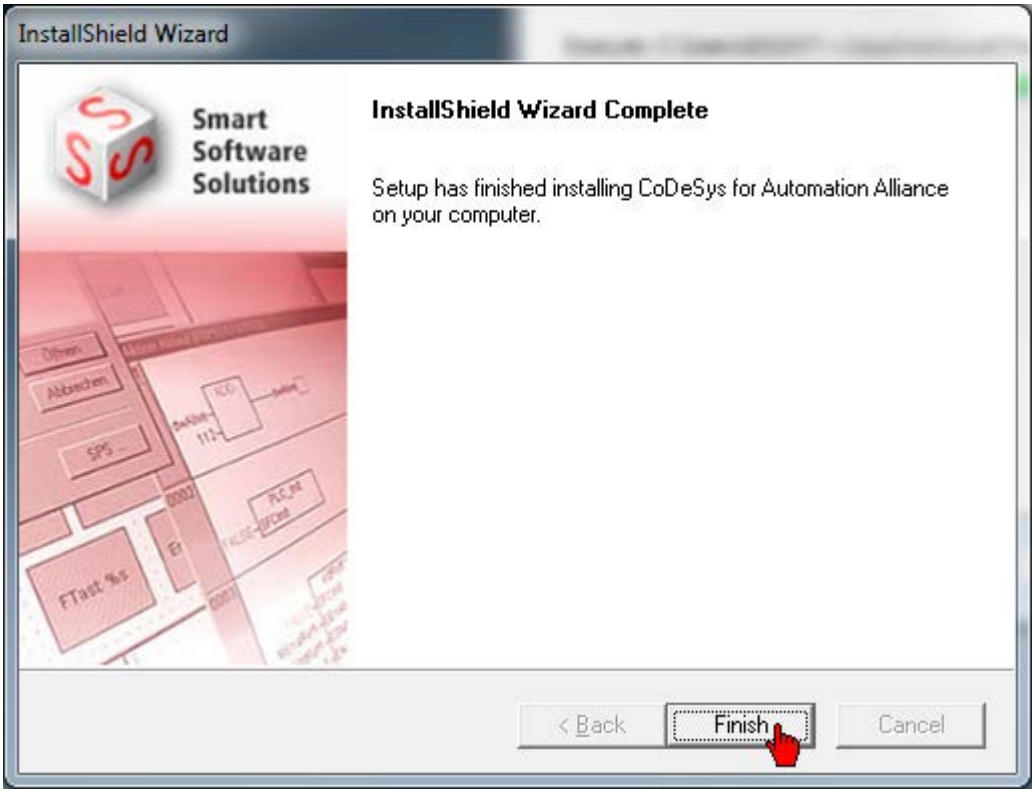

InstallShield Wizard
License Agreement
Please read the following license agreement carefully.
Press the PAGE DOWN key to see the rest of the agreement.
License Agreement for the Usage of the PLC Development System CoDeSys
General Terms of License for the Automation Software CoDeSys (Software User Agreement)
Please read this Software User Agreement carefully before using the supplied software.
The software CoDeSys is protected by copyright and other intellectual property laws. The following Software User Agreement is entered into by and between you as the software user and the company 3S-Smart Software Solutions GmbH, headquartered in 87439 Kempten, Germany. By installing the software package you agree to be bound by the
Do you accept all the terms of the preceding License Agreement? If you choose No, the setup will close. To install CoDeSys for Automation Alliance, you must accept this agreement.
InstallShield
< Back
Yes
No

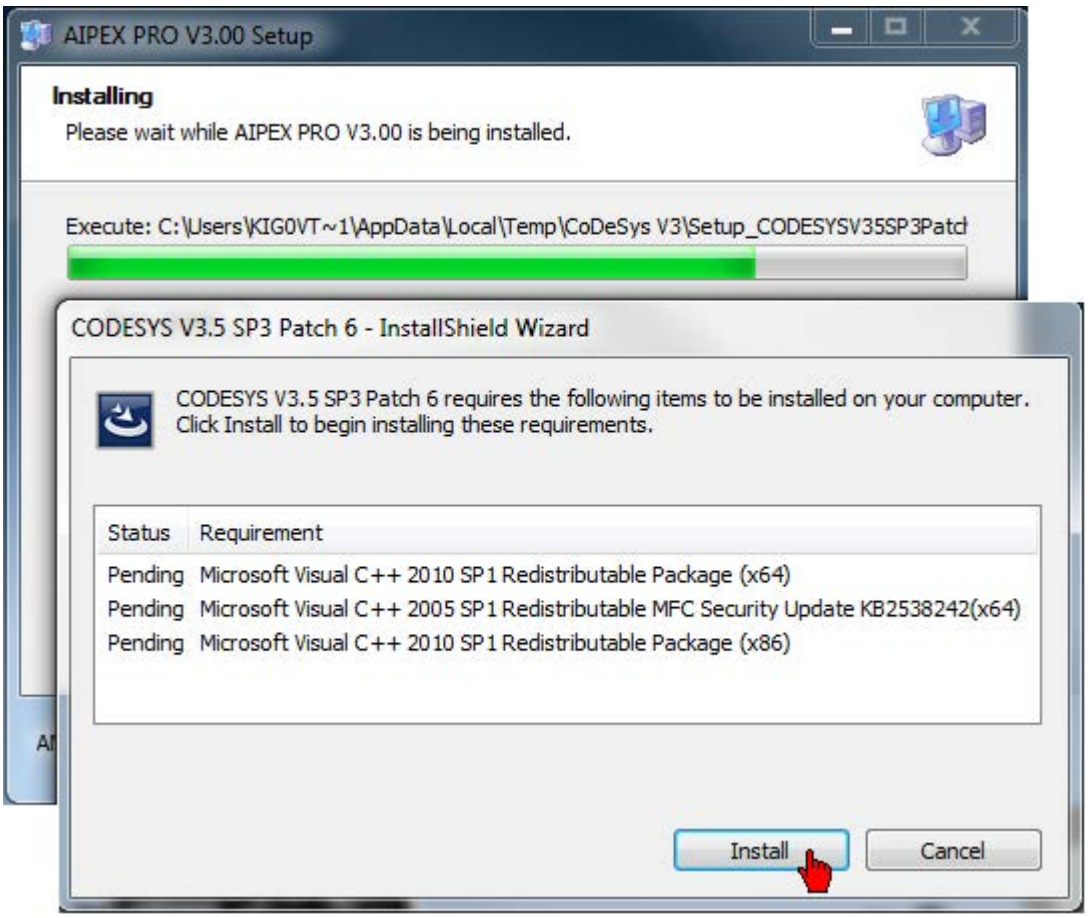

InstallShield Wizard
Choose Destination Location
Select destination folder where Setup will install files.
Setup will install CoDeSys for Automation Alliance in the following folder.
Each component will be installed in a subfolder.
To install to this folder, click Next. To install to a different folder, click Browse and select another folder.
Destination Folder
C:\Program Files (x86)\3S Software
Browse...
InstallShield
< Back
Next >
Cancel

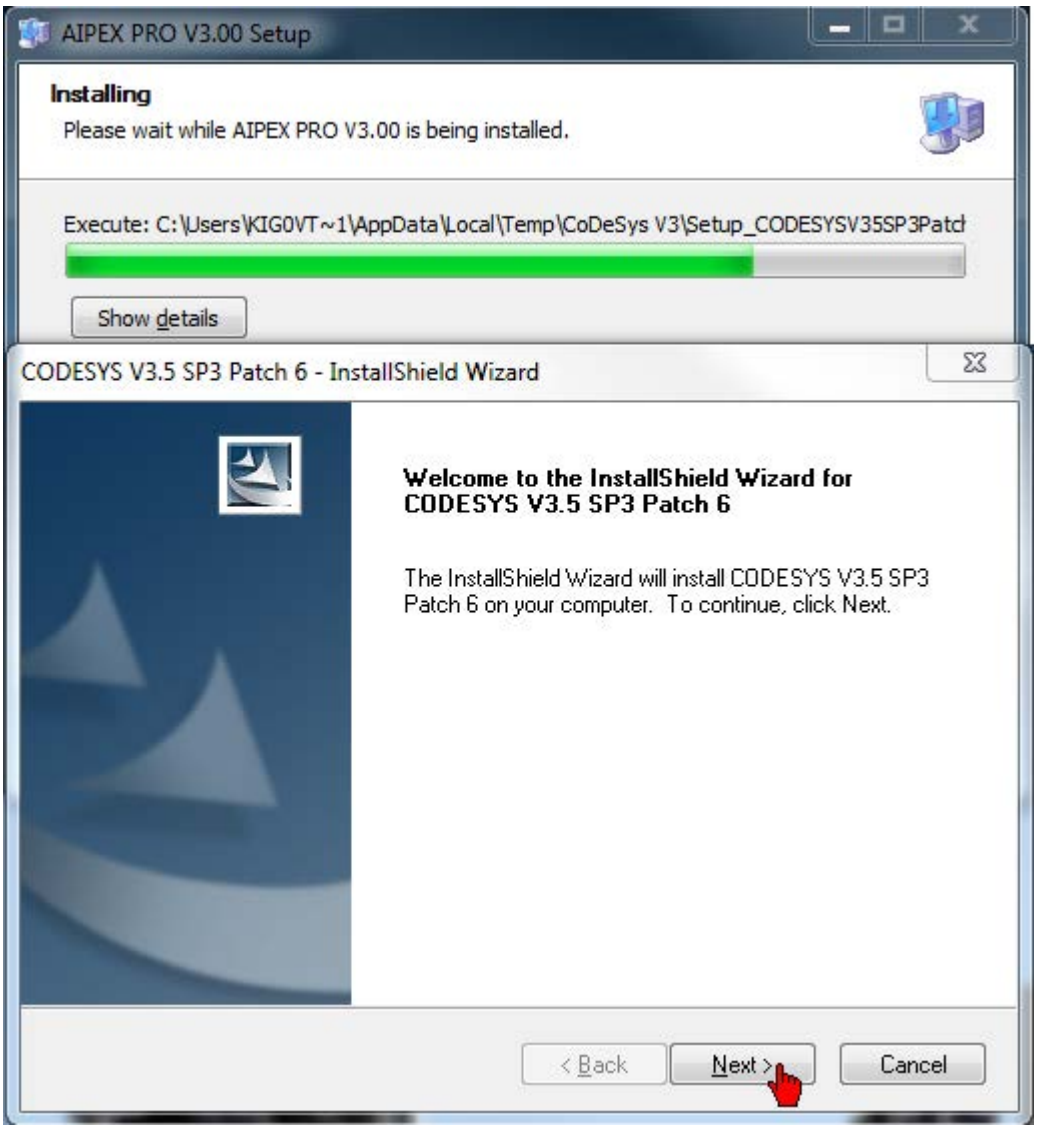

InstallShield Wizard
Select Components
Choose the components Setup will install.
Select the components you want to install, and clear the components you do not want to install.
CoDeSys V2.3
CoDeSys base component
3S Licensing Manager
CoDeSys HMI
Communication modules
CoDeSys Gateway Server
CoDeSys OPC Server
CoDeSys ENI Server
CoDeSys WebServer
Description
IEC6-1131-3 Programming system for controller.
with different programming languages, codegenerators,
Online functions, integrated visualization, ...
inclusive libraries, help, documentation, examples, ...
Space Required on C: 153268 K
Space Available on C: 434950220 K
InstallShield
< Back
Next >
Cancel

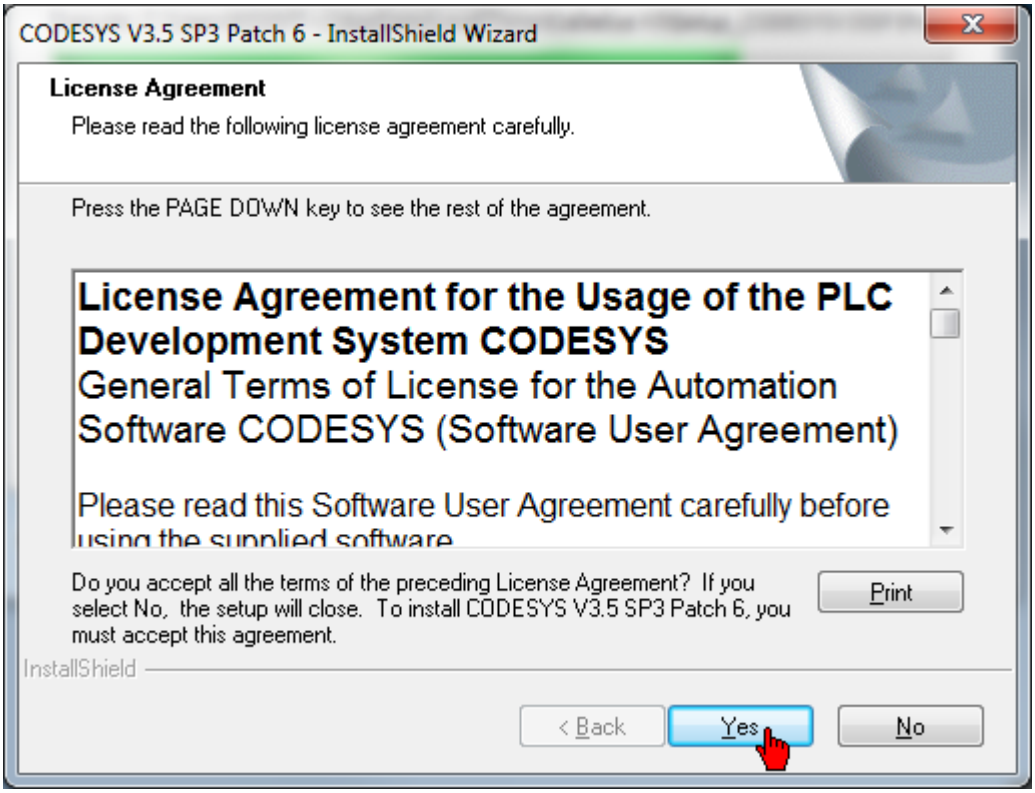

InstallShield Wizard
Select Program Folder
Select the location where setup is to create new shortcuts.
Setup will add program icons to the Program Folder listed below. You may type a new folder name, or select one from the existing folders list. Click Next to continue.
Program Folders:
3S Software
Existing Folders:
7-Zip
Accessories
Administrative Tools
Adobe
AMK
FileZilla FTP Client
Games
IBM System i Access für Windows
MadCap Software
InstallShield
< Back
Next >
Cancel

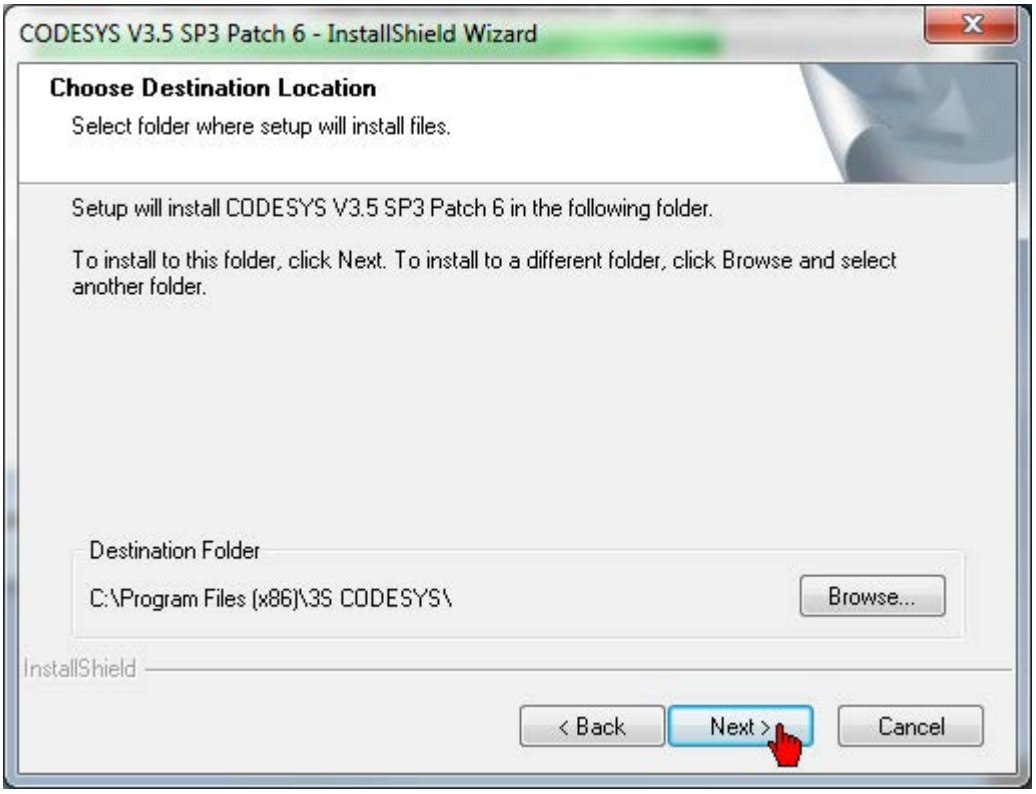

InstallShield Wizard
Configuration overview
Please check all settings.
Selected component:
CoDeSys V2.3.9.37
CoDeSys base components
CoDeSys Gateway Server V2.3.9.26
3S Licensing Manager V1.2.4.2
Destination folder: C:\Program Files (x86)\3S Software
Program Folder: 3S Software
InstallShield
< Back
Next >
Cancel

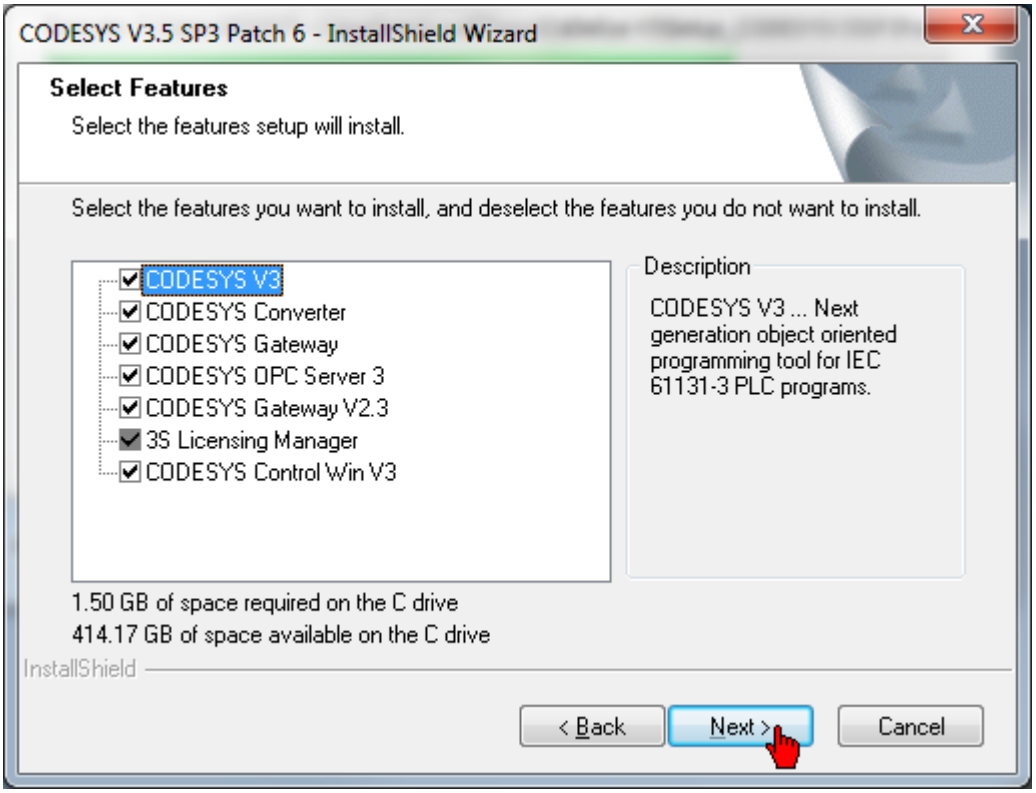

InstallShield Wizard
Smart
Software
Solutions
InstallShield Wizard Complete
Setup has finished installing CoDeSys for Automation Alliance on your computer.
< Back
Finish
Cancel

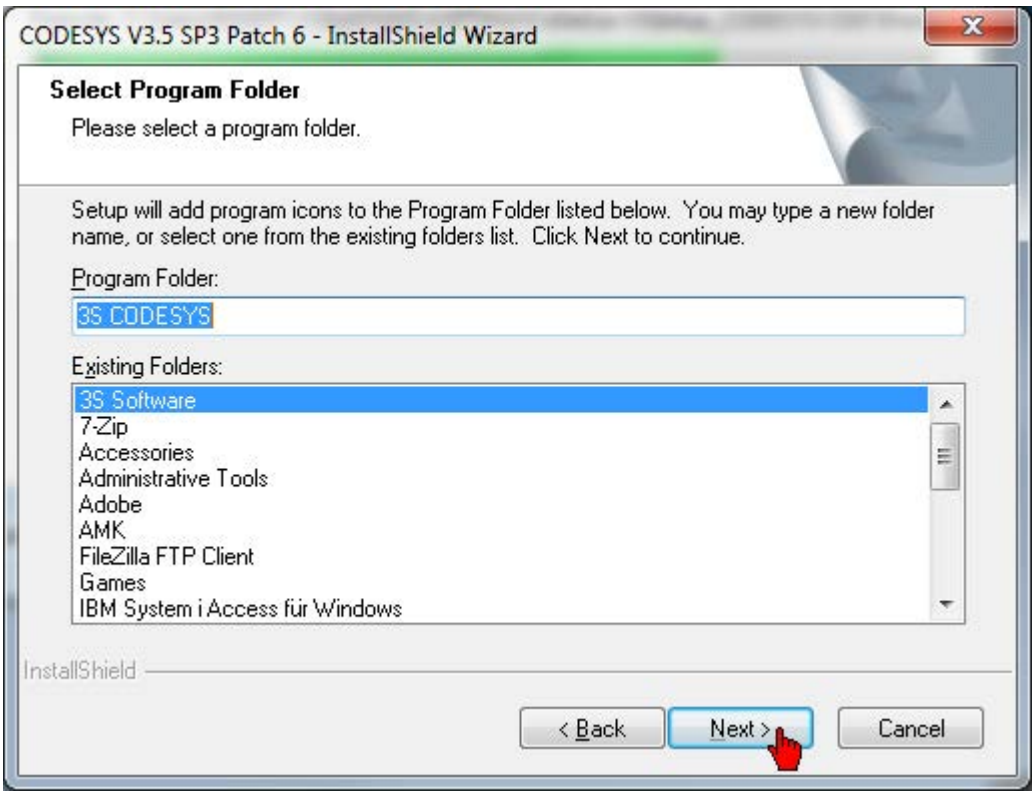

AIPEX PRO V3.00 Setup
Installing
Please wait while AIPEX PRO V3.00 is being installed.
Execute: C:\Users\KIG0VT~1\AppData\Local\Temp\CoDeSys V3\Setup_CODESYSV35SP3Patcl
CODESYS V3.5 SP3 Patch 6 - InstallShield Wizard
CODESYS V3.5 SP3 Patch 6 requires the following items to be installed on your computer.
Click Install to begin installing these requirements.
Status
Requirement
Pending
Microsoft Visual C++ 2010 SP1 Redistributable Package (x64)
Pending
Microsoft Visual C++ 2005 SP1 Redistributable MFC Security Update KB2538242(x64)
Pending
Microsoft Visual C++ 2010 SP1 Redistributable Package (x86)
Install
Cancel

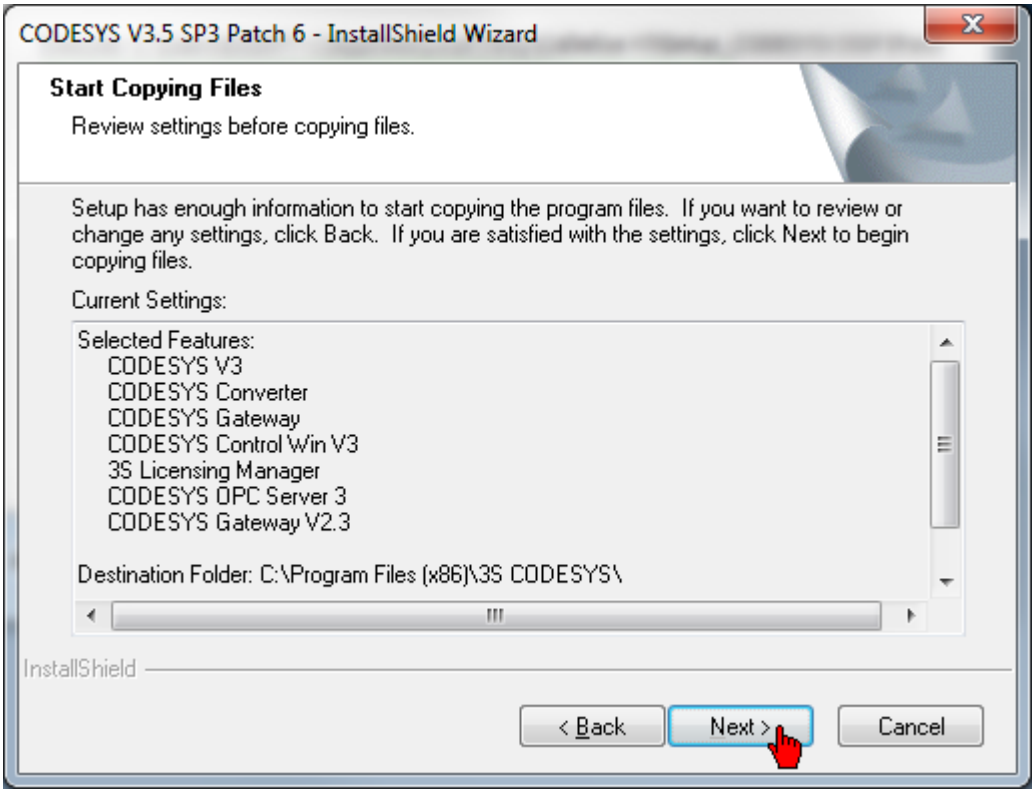

AIPEX PRO V3.00 Setup
Installing
Please wait while AIPEX PRO V3.00 is being installed.
Execute: C:\Users\KIG0VT~1\AppData\Local\Temp\CoDeSys V3\Setup_CODESYSV35SP3Patcl
Show details
CODESYS V3.5 SP3 Patch 6 - InstallShield Wizard
Welcome to the InstallShield Wizard for CODESYS V3.5 SP3 Patch 6
The InstallShield Wizard will install CODESYS V3.5 SP3 Patch 6 on your computer. To continue, click Next.
< Back
Next >
Cancel

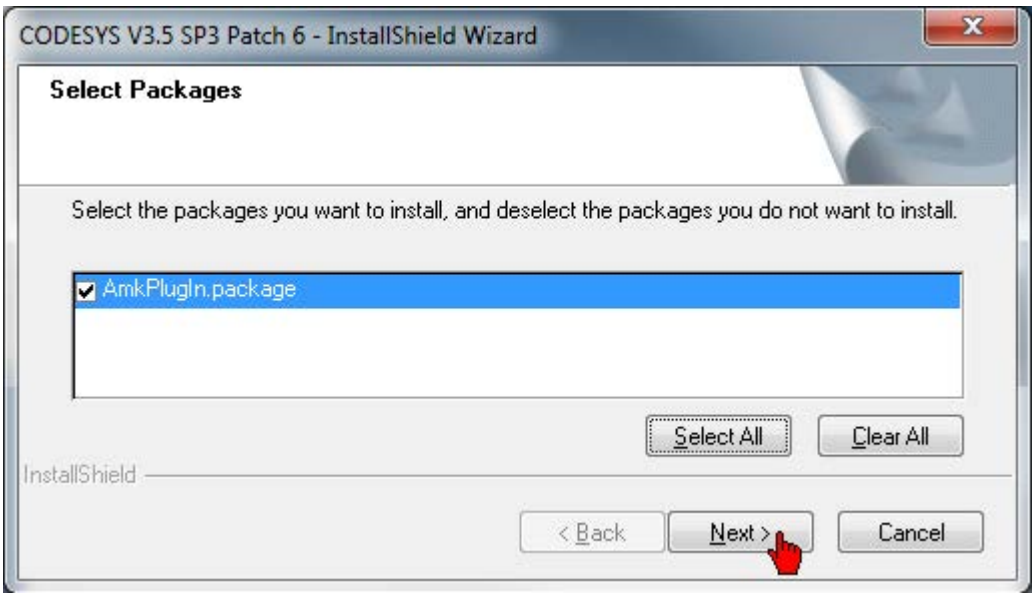

CODESYS V3.5 SP3 Patch 6 - InstallShield Wizard
License Agreement
Please read the following license agreement carefully.
Press the PAGE DOWN key to see the rest of the agreement.
License Agreement for the Usage of the PLC Development System CODESYS
General Terms of License for the Automation Software CODESYS (Software User Agreement)
Please read this Software User Agreement carefully before
Do you accept all the terms of the preceding License Agreement? If you select No, the setup will close. To install CODESYS V3.5 SP3 Patch 6, you must accept this agreement.
Print
InstallShield
< Back
Yes
No

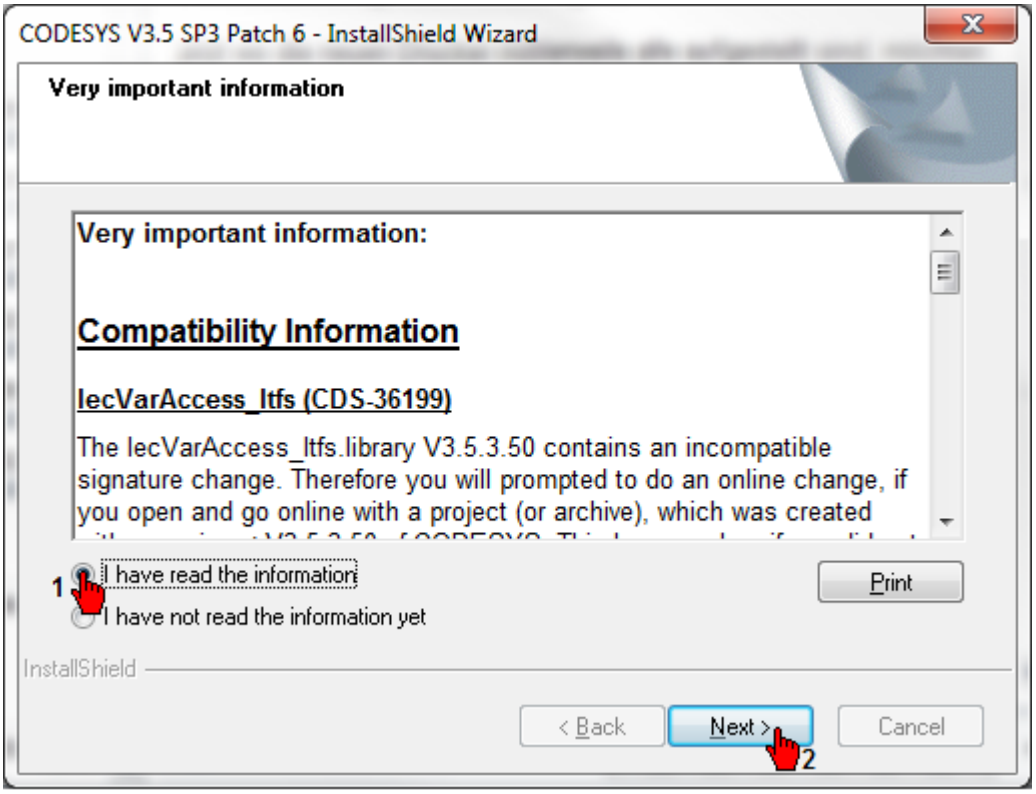

CODESYS V3.5 SP3 Patch 6 - InstallShield Wizard
Choose Destination Location
Select folder where setup will install files.
Setup will install CODESYS V3.5 SP3 Patch 6 in the following folder.
To install to this folder, click Next. To install to a different folder, click Browse and select another folder.
Destination Folder
C:\Program Files (x86)\3S CODESYS\
Browse...
InstallShield
< Back
Next >
Cancel

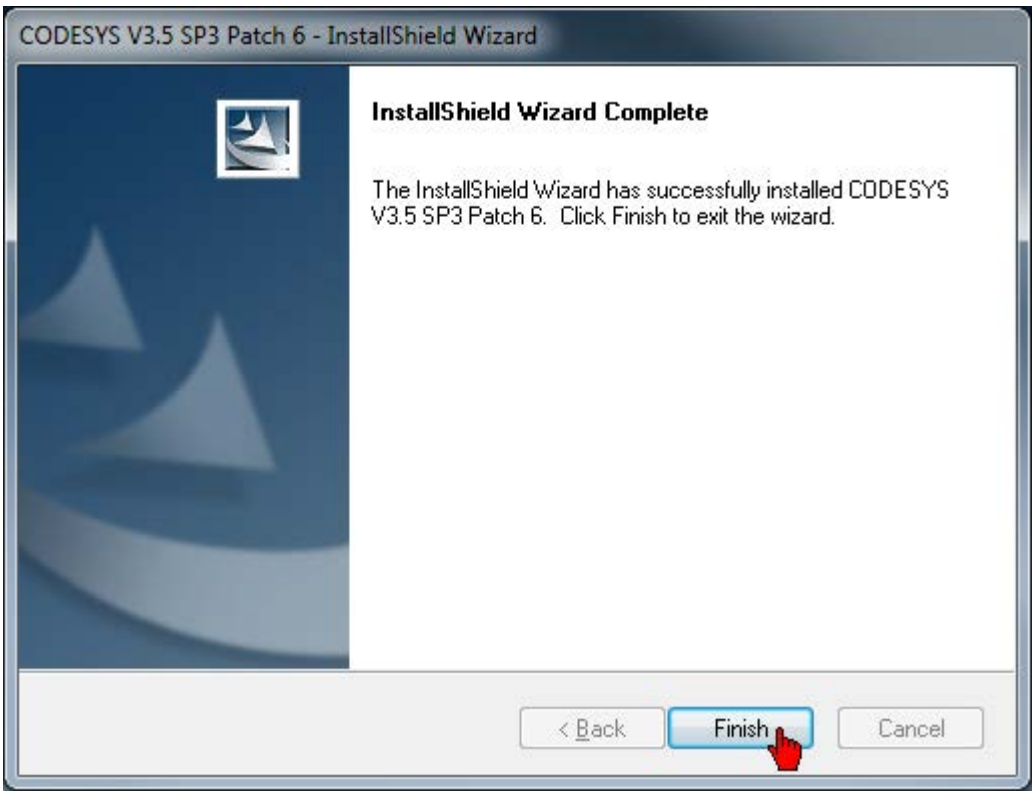

CODESYS V3.5 SP3 Patch 6 - InstallShield Wizard
Select Features
Select the features setup will install.
Select the features you want to install, and deselect the features you do not want to install.
CODESYS V3
CODESYS Converter
CODESYS Gateway
CODESYS OPC Server 3
CODESYS Gateway V2.3
3S Licensing Manager
CODESYS Control Win V3
Description
CODESYS V3 ... Next generation object oriented programming tool for IEC 61131-3 PLC programs.
1.50 GB of space required on the C drive
414.17 GB of space available on the C drive
InstallShield
< Back
Next >
Cancel

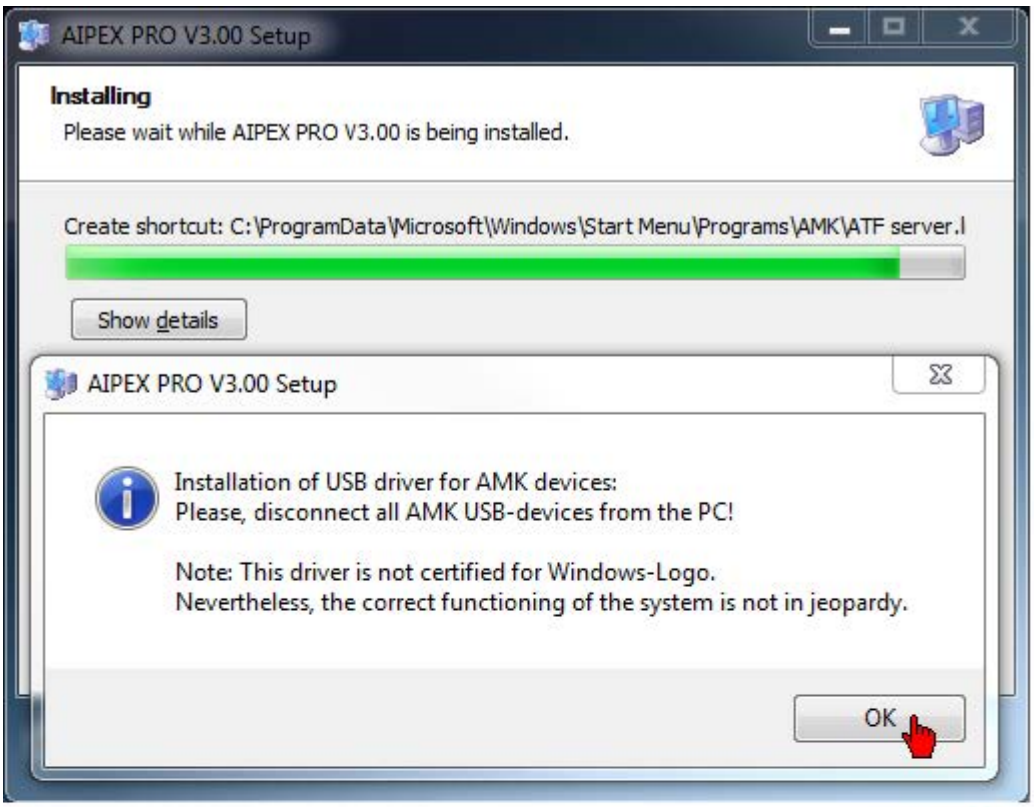

CODESYS V3.5 SP3 Patch 6 - InstallShield Wizard
Select Program Folder
Please select a program folder.
Setup will add program icons to the Program Folder listed below. You may type a new folder name, or select one from the existing folders list. Click Next to continue.
Program Folder:
3S CODESYS
Existing Folders:
3S Software
7-Zip
Accessories
Administrative Tools
Adobe
AMK
FileZilla FTP Client
Games
IBM System i Access für Windows
InstallShield
< Back
Next >
Cancel

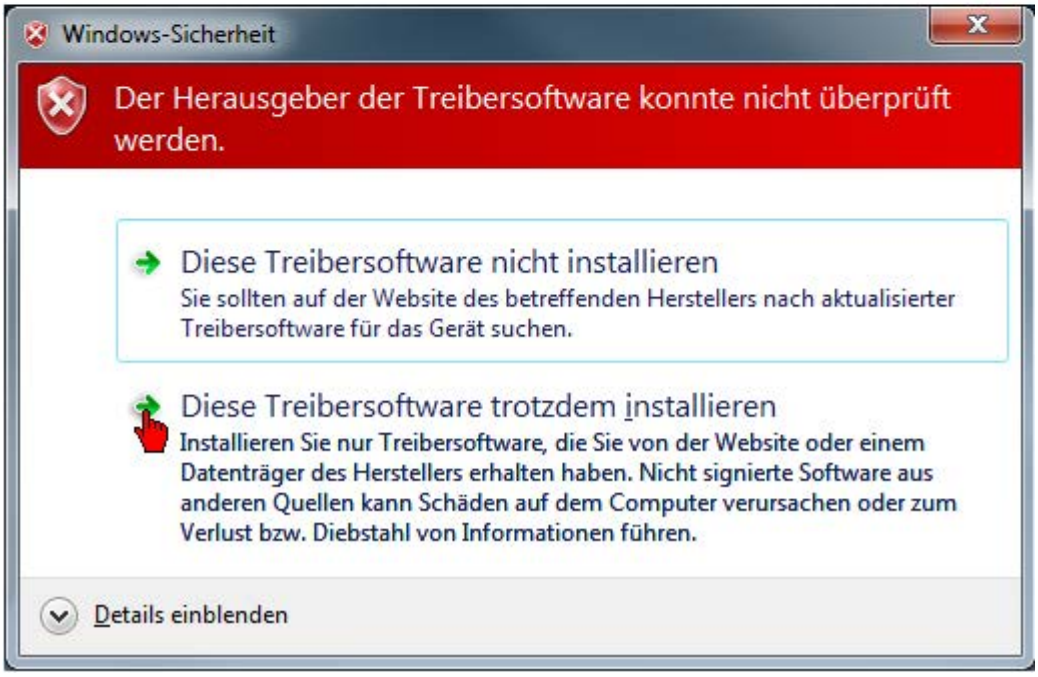

CODESYS V3.5 SP3 Patch 6 - InstallShield Wizard
Start Copying Files
Review settings before copying files.
Setup has enough information to start copying the program files. If you want to review or change any settings, click Back. If you are satisfied with the settings, click Next to begin copying files.
Current Settings:
Selected Features:
CODESYS V3
CODESYS Converter
CODESYS Gateway
CODESYS Control Win V3
3S Licensing Manager
CODESYS OPC Server 3
CODESYS Gateway V2.3
Destination Folder: C:\Program Files (x86)\3S CODESYS\
InstallShield
< Back
Next >
Cancel


CODESYS V3.5 SP3 Patch 6 - InstallShield Wizard
Select Packages
Select the packages you want to install, and deselect the packages you do not want to install.
AmkPlugIn.package
Select All
Clear All
InstallShield
< Back
Next >
Cancel


CODESYS V3.5 SP3 Patch 6 - InstallShield Wizard
Very important information
Very important information:
Compatibility Information
IecVarAccess_Itfs (CDS-36199)
The IecVarAccess_Itfs.library V3.5.3.50 contains an incompatible signature change. Therefore you will prompted to do an online change, if you open and go online with a project (or archive), which was created
I have read the information
I have not read the information yet
Print
1
2
InstallShield
< Back
Next >
Cancel


CODESYS V3.5 SP3 Patch 6 - InstallShield Wizard
InstallShield Wizard Complete
The InstallShield Wizard has successfully installed CODESYS V3.5 SP3 Patch 6. Click Finish to exit the wizard.
< Back
Finish
Cancel

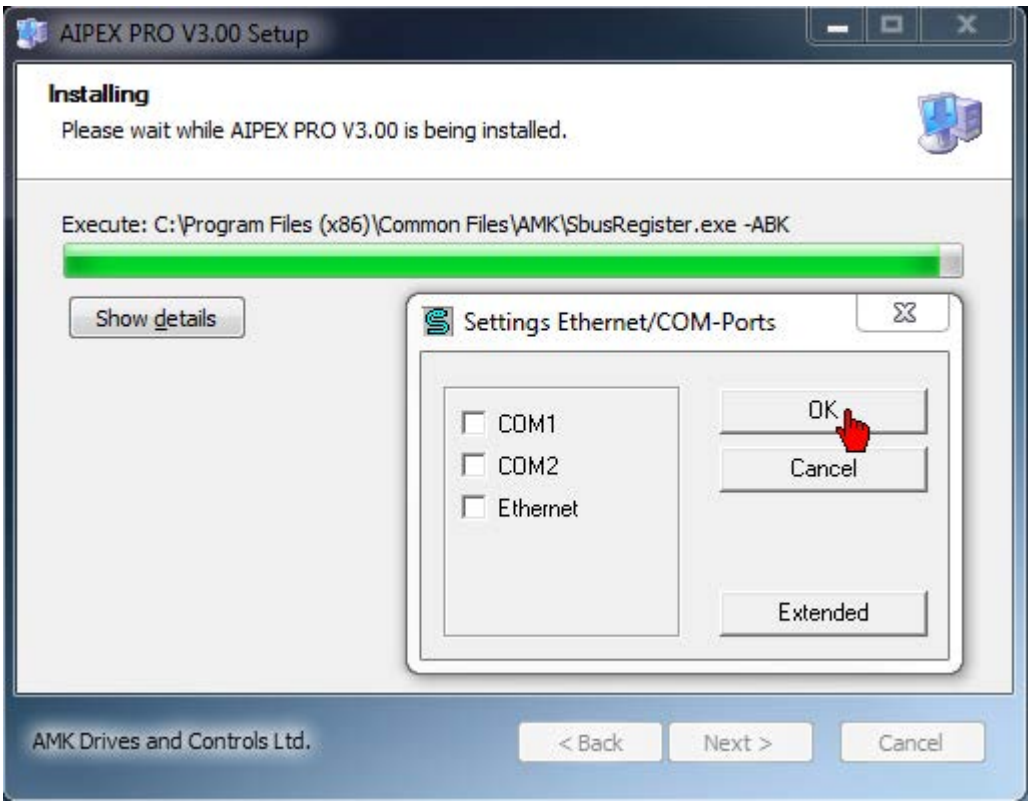

AIPEX PRO V3.00 Setup
Installing
Please wait while AIPEX PRO V3.00 is being installed.
Create shortcut: C:\ProgramData\Microsoft\Windows\Start Menu\Programs\AMK\ATF server.l
Show details
AIPEX PRO V3.00 Setup
Installation of USB driver for AMK devices:
Please, disconnect all AMK USB-devices from the PC!
Note: This driver is not certified for Windows-Logo.
Nevertheless, the correct functioning of the system is not in jeopardy.
OK


Windows-Sicherheit
Der Herausgeber der Treibersoftware konnte nicht überprüft werden.
Diese Treibersoftware nicht installieren
Sie sollten auf der Website des betreffenden Herstellers nach aktualisierter Treibersoftware für das Gerät suchen.
Diese Treibersoftware trotzdem installieren
Installieren Sie nur Treibersoftware, die Sie von der Website oder einem Datenträger des Herstellers erhalten haben. Nicht signierte Software aus anderen Quellen kann Schäden auf dem Computer verursachen oder zum Verlust bzw. Diebstahl von Informationen führen.
Details einblenden

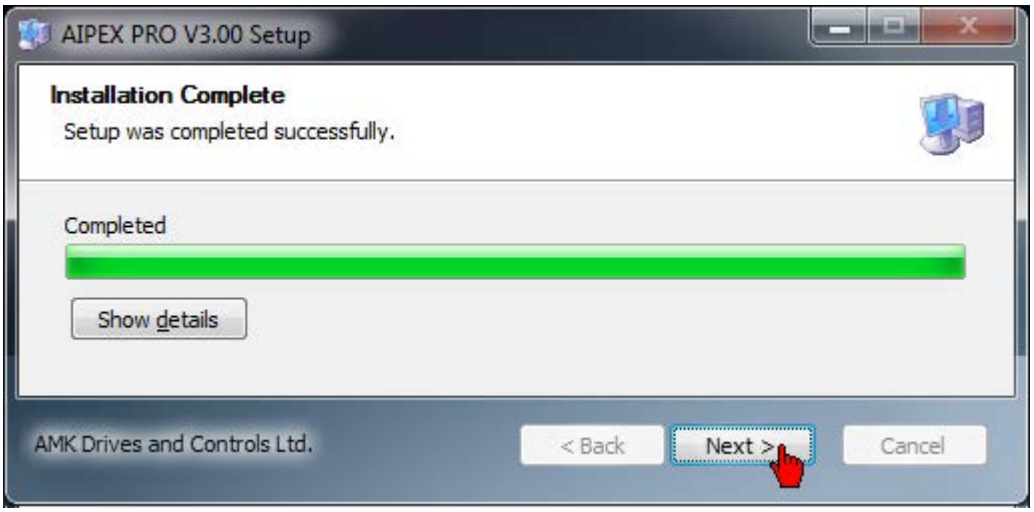
Customer specific input.

Communication interface between AIPEX PRO and AMK device. You can also choose the interface later in the running program.

Siehe 'Communication PC - AMK device' auf Seite 43.



Customer specific input.



3.2 Service pack

An AIPEX PRO service pack is needed if, after an AIPEX PRO release, "new", i.e. still unknown devices or device features for AIPEX PRO, have been released by AMK.

The installation is executed automatically by running the supplied Setup.exe.

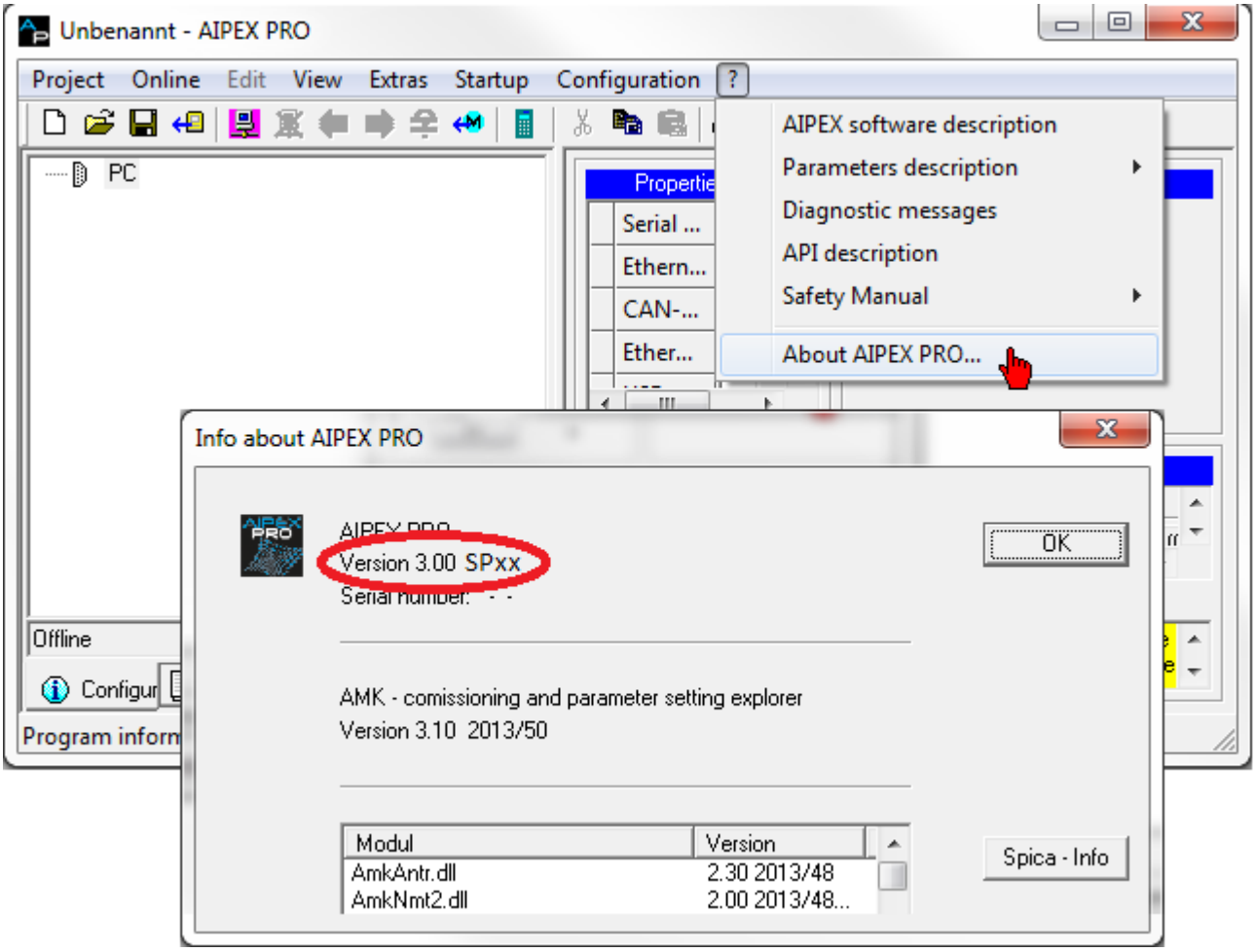
The installation of a service pack is possible only for the corresponding AIPEX PRO version. This means for example, a service pack AipexPro_105_SP04_1051_203405.zip can only be postinstalled for an AIPEX PRO version V1.05.

Each service pack contains all features of the preceding service pack. That is why it is sufficient to always install only the latest service pack.

Info about AIPEX PRO

Display of the AIPEX PRO version and the installed service pack.

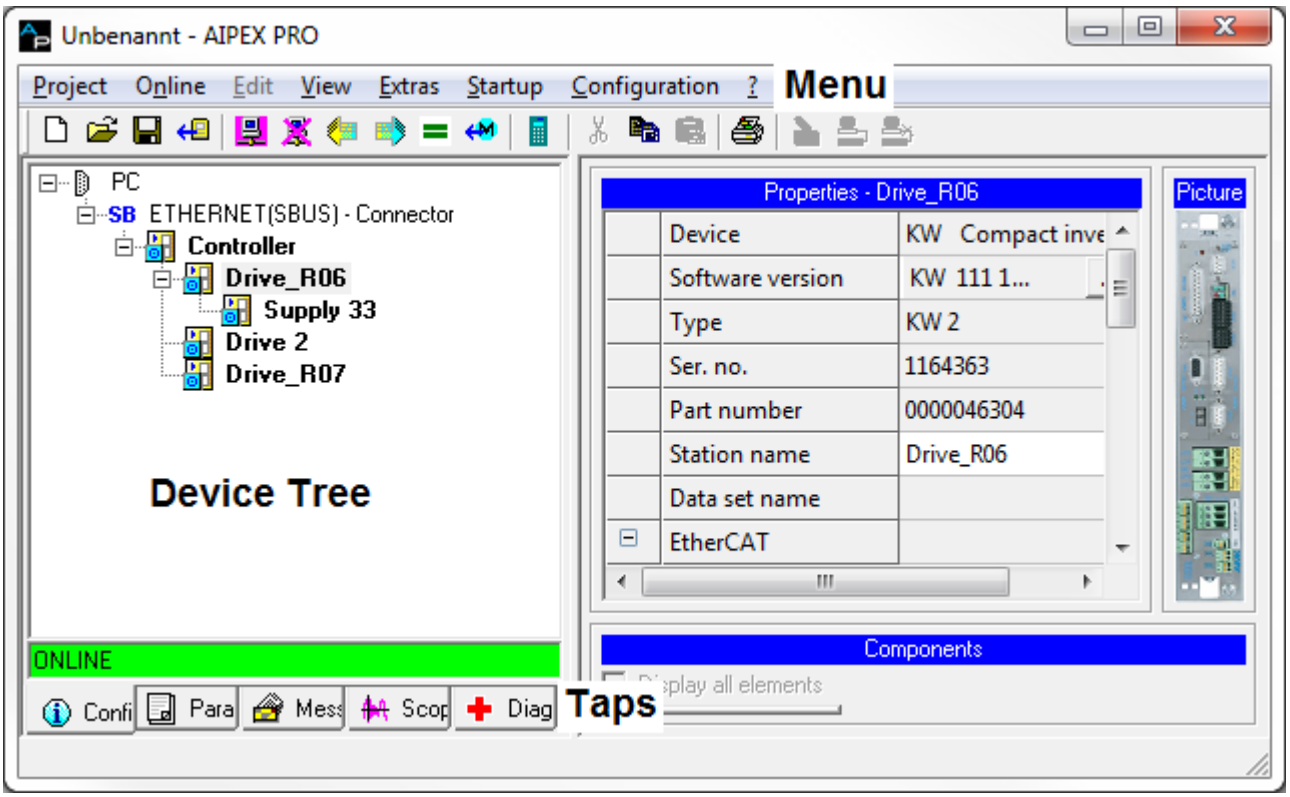
In the table, the versions of the corresponding components and data bases are listed.



3.3 Program overview

AIPEX PRO integrates all engineering tools needed during the life cycle of a machine, e.g., programming, parameter assignment, commissioning, optimization, and diagnostics. This saves you time-consuming efforts to coordinate, for example, between your PLC program containing drive parameters and the configured user data exchange via the fieldbus.

AIPEX PRO does this work for you automatically. The engineering tool also has a visualization editor with ready-made blocks.



Tab	Functionality
Configuration	Display and input possibility for device properties Siehe 'Configuration' auf Seite 70.
Parameters	Display and input possibility for parameter values Siehe 'Parameter' auf Seite 74.
Messages	Display and input possibility for network data transfer Siehe 'Messages' auf Seite 83.
Scope	Oscilloscope function to measure drive values Siehe 'Scope' auf Seite 89.
Diagnostics	Diagnostic module Siehe 'Diagnose' auf Seite 93.

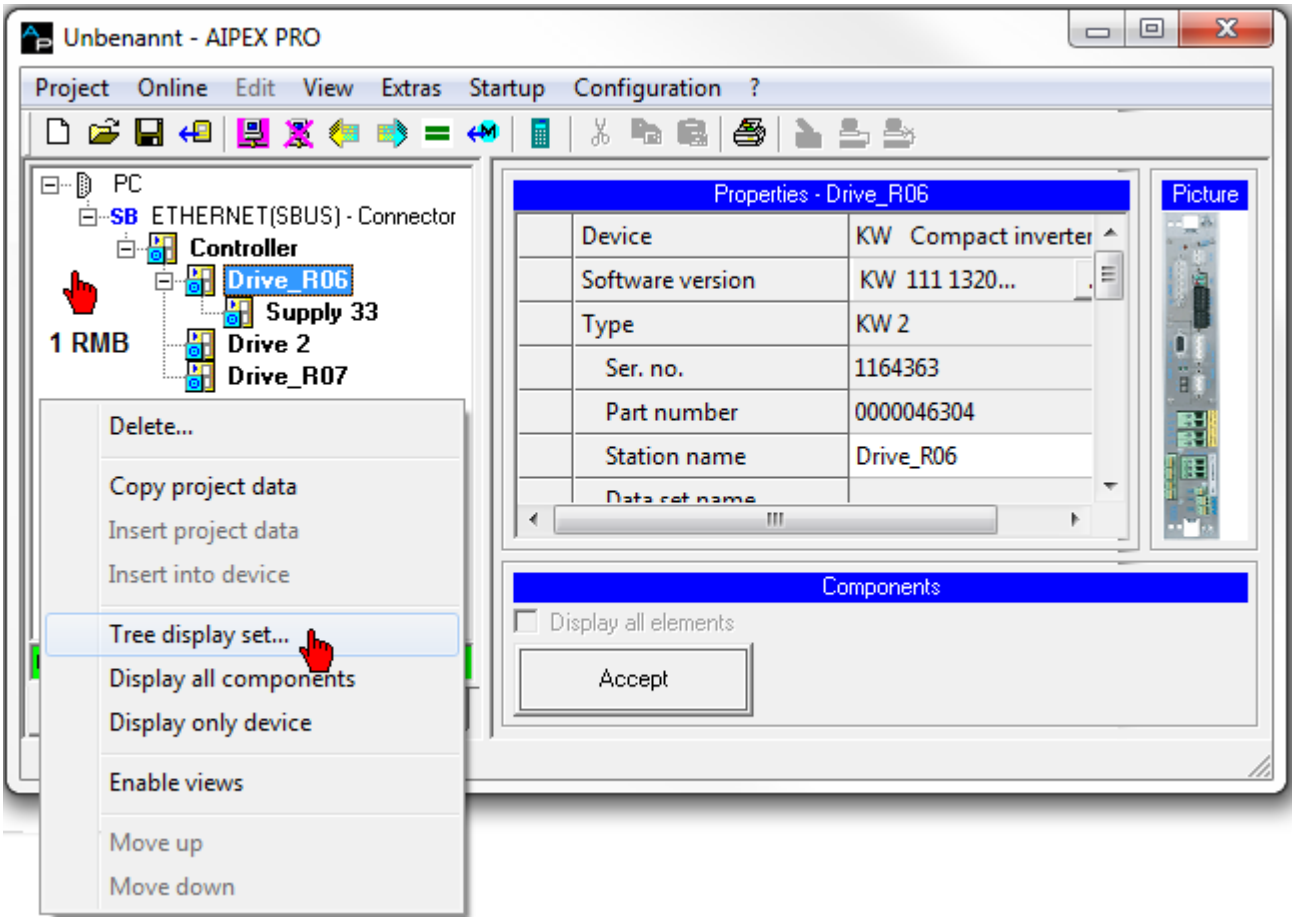
3.3.1 Program overview - Symbols and Icons

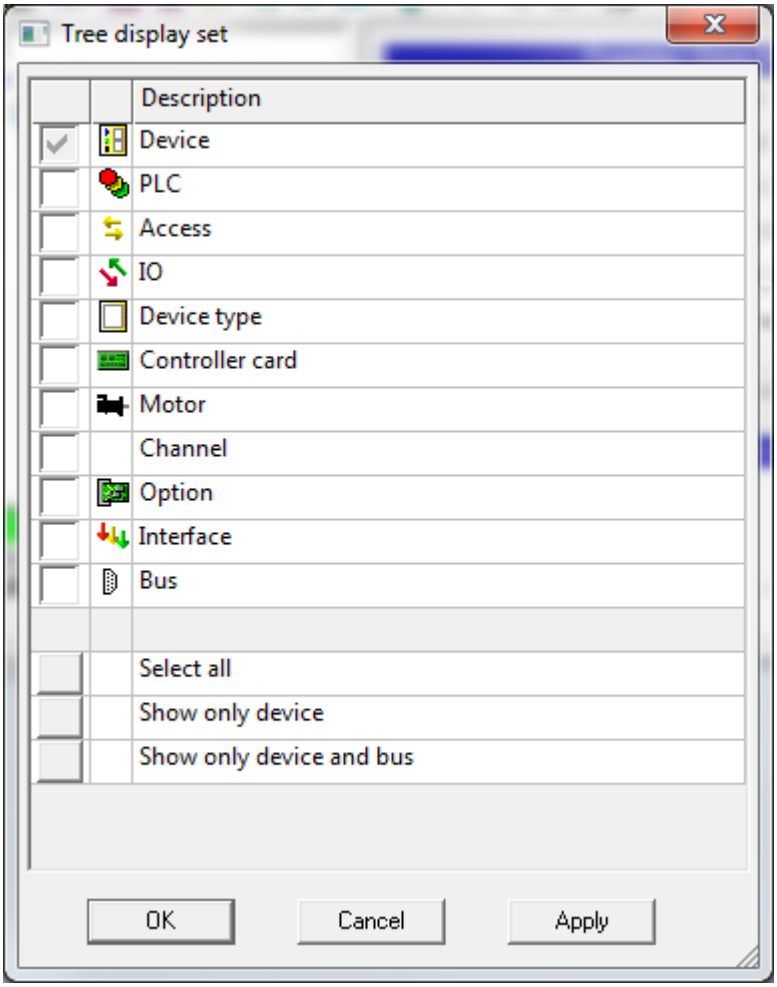
Symbol Icon	Meaning
	The hand symbol shows the user graphically in examples where to click with the mouse. RMB: Right mouse button
	Communication interface between PC and device is activated, physically existing and ready
	Communication interface between PC and device is activated, but not ready
	The opened project is closed and a new project is created. Siehe 'Project' auf Seite 94.
	AIPEX PRO and AIPEX/AIPAR files can be opened. Siehe 'Project' auf Seite 94.
	When you save a file for the fist time, the 'Save as' dialog box is displayed automatically. Siehe 'Project' auf Seite 94.
	Print Siehe 'Project' auf Seite 94.

Symbol Icon	Meaning
	Offline device No connection to a physically existing device
	Online Gerät With connection to a physically existing device
	Siehe 'Project' auf Seite 94.
	Siehe 'Online' auf Seite 96.
	Siehe 'Online' auf Seite 96.
	Parameters transfer from the selected device to the project
	Data transfer from the current project to the selected device
	Siehe 'Online' auf Seite 96.
	Siehe 'Online' auf Seite 96.
	Siehe 'Direct mode' auf Seite 122.
	Undo / Redo function (manual network configuration)
	Siehe 'Configuration' auf Seite 121.
	Siehe 'Configuration' auf Seite 121.
	Siehe 'Configuration' auf Seite 121.
	CODESYS Version 2
	CODESYS Version 3

3.3.2 Program overview - Display filter

Improve the design the device tree's structure by adapting the display filter to your application. For this, click in the 'device explorer' with the right mouse button.





A separate display filter can be used for each tab **Configuration**, **Parameters**, **Messages**, **'Scope'** and **'Diagnostics'**.

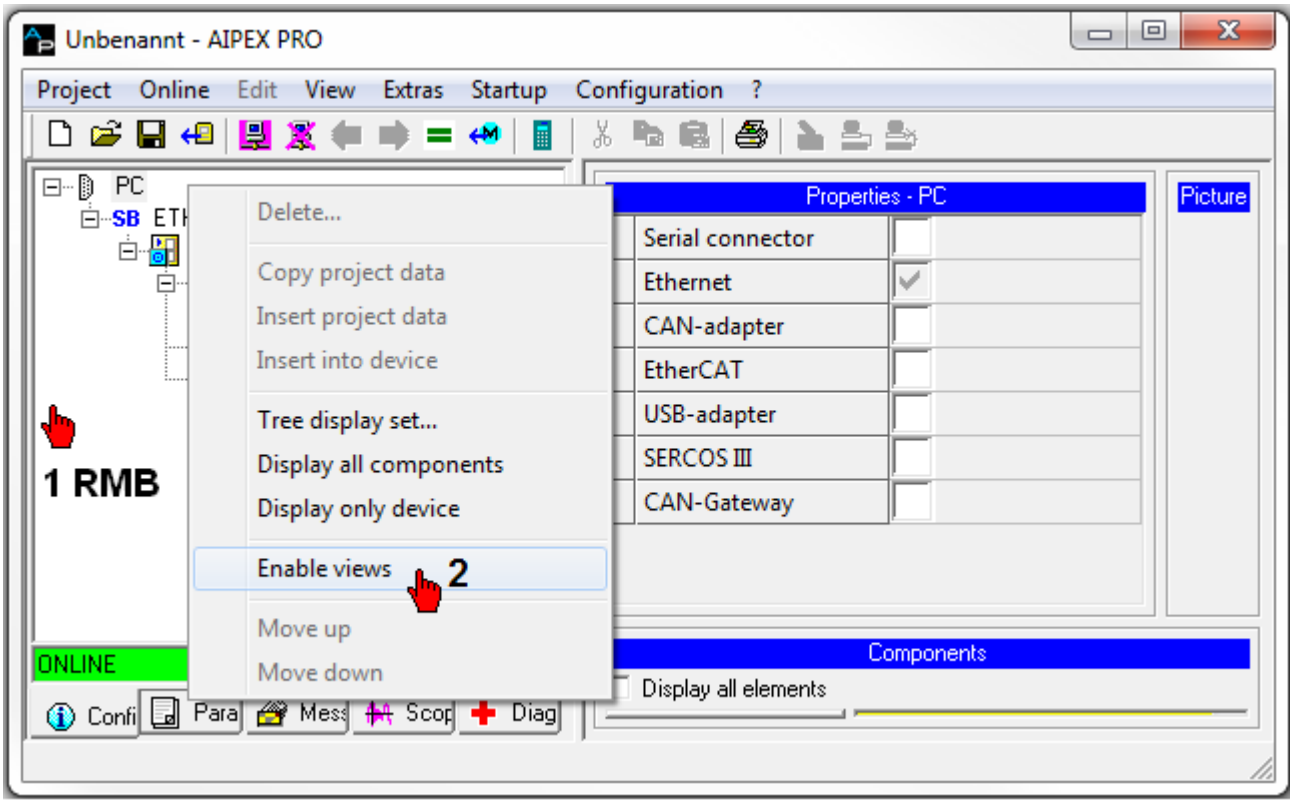
Name	Note
Device	Display of the station name
PLC	Display of the PLC component
I/O	Display of the binary inputs and outputs
Device type	Display of the device type
Controller card	Display of the controller card
Motor	Properties of the motor
Unassigned channel	Display of unassigned interfaces
Option	Display of option cards
Interface	Interface between AIPEX PRO and CoDeSys
Bus	AMK bus (e.g. ACC bus, CAN-S, EtherCAT...)

3.3.3 Program overview - Enable views

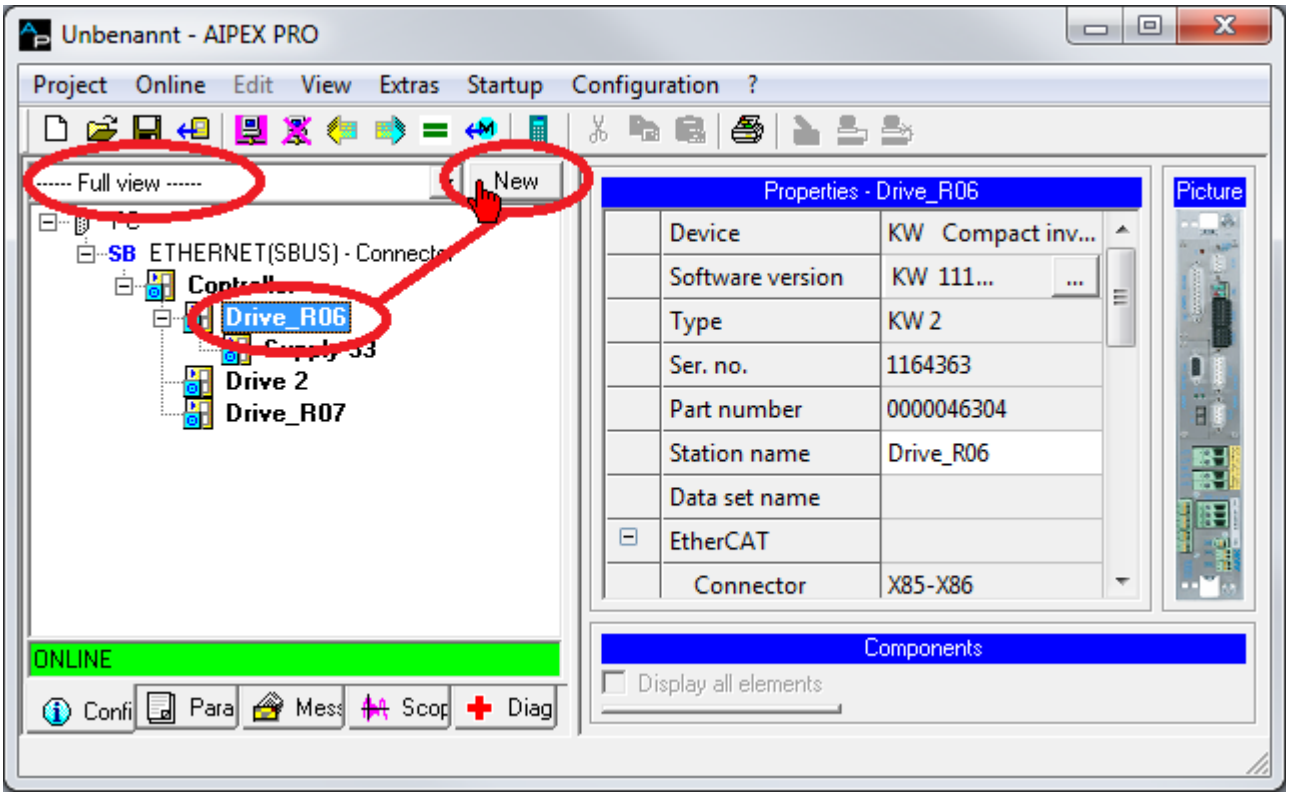
Using the function **'Enable views'**, you can hide parts of the entire device tree. The user can select which modules are displayed. By hiding individual modules, a better overview is provided for the user. In addition, access to data and devices is sped up by hiding the modules.

The partial view found between the start and end points. The start point is the selected module when pressing the **'New'** button. The end point is set in the dialogue box that opens, **'Generate partial view'**.

You activate the partial view by right-clicking in the device tree.
Then click in the dialog box on **'Enable views'**.

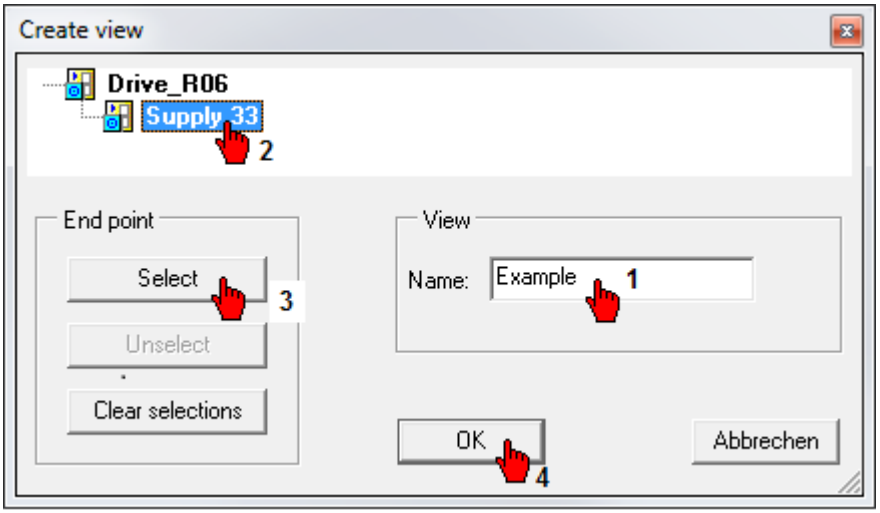


Using the selection box, you can select between **'Full view'** and (if it exists) the user-specific partial views. User-specific partial views are generated by selecting the start point and the clicking on the **'New'** button.

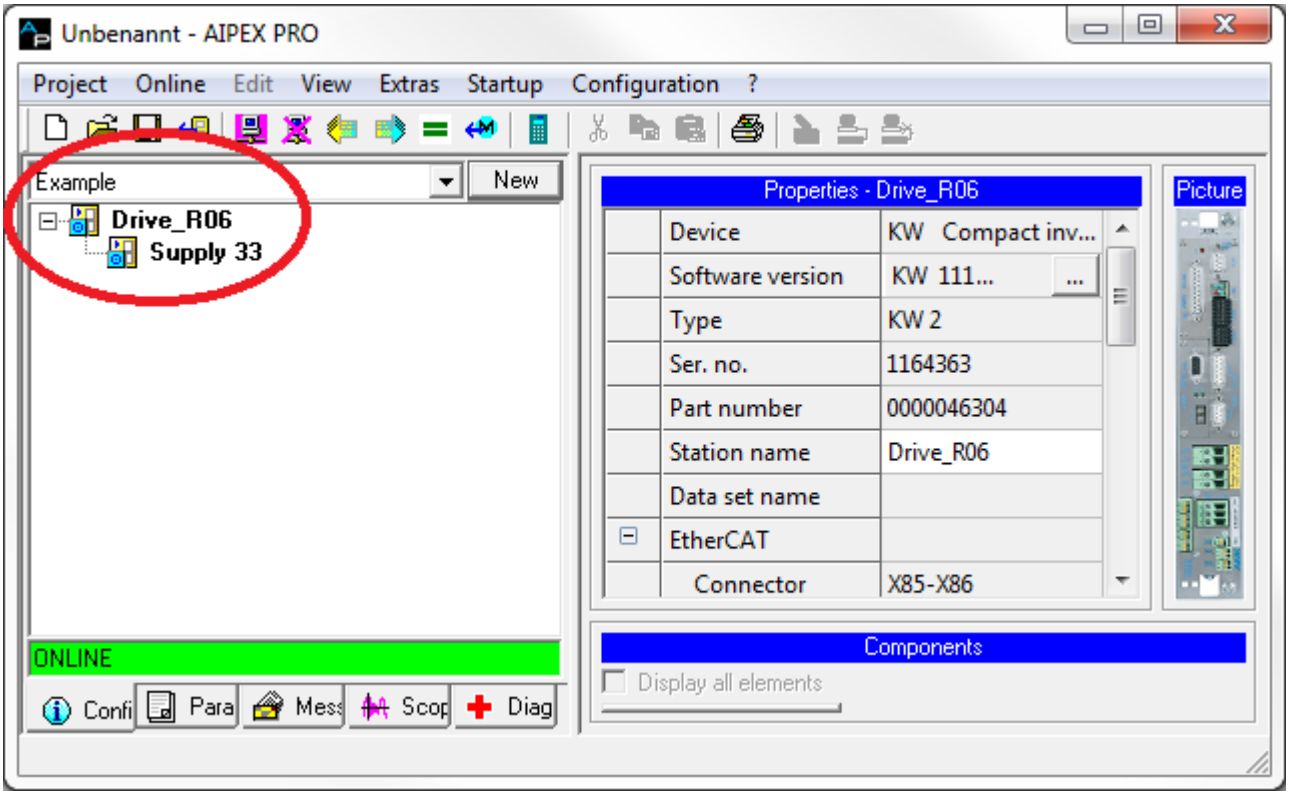


Generate partial view:

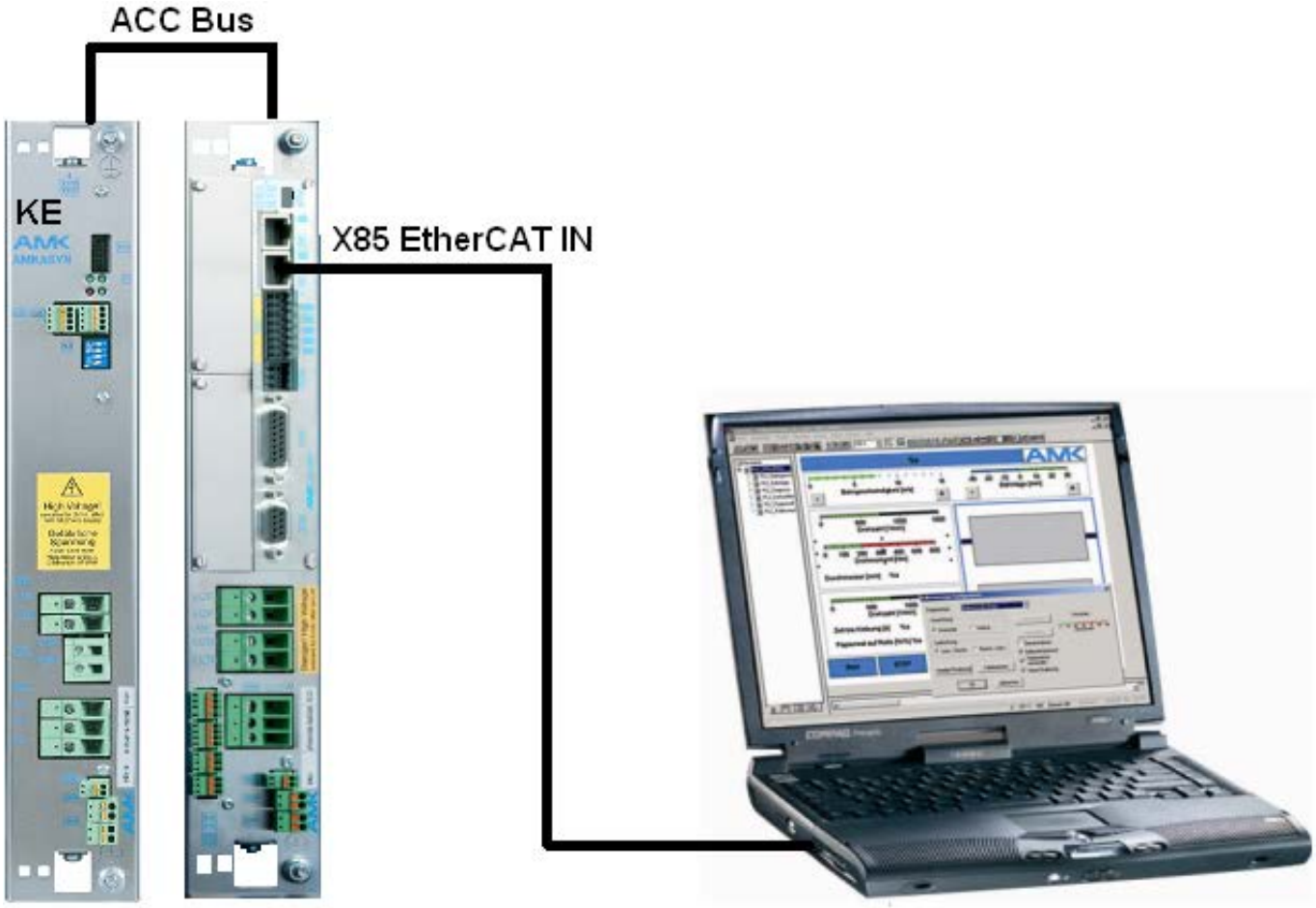
Assign a name, then select your end point (module). Press the Set button to define the end point.



Example partial view: Start point Drive 1 module, end point Supply 33 module.



3.4 Communication PC - AMK device



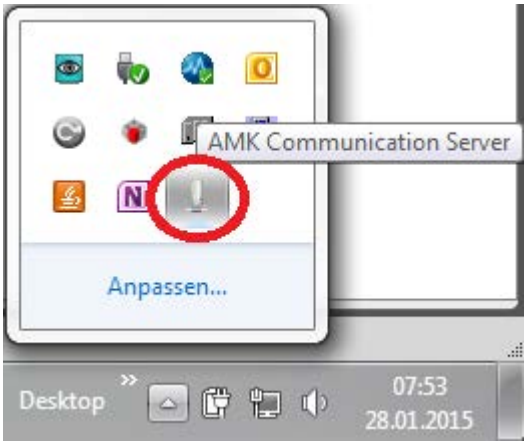
Siehe 'CAN interface' auf Seite 46.
Siehe 'COM interface' auf Seite 48.
Siehe 'EtherCAT interface' auf Seite 54.
Siehe 'Ethernet interface' auf Seite 57.
Siehe 'SERCOS III interface' auf Seite 66.
Siehe 'USB interface' auf Seite 69.
Siehe 'AIPEX PRO Add In Gateway for TwinCAT V2' auf Seite 69.
Siehe 'Testing the communication' auf Seite 69.

3.4.1 Communication server

The communication server (AMK Communication Server) is used to establish a connection (Ethernet, EtherCAT, CANCLIENT and USB) to the devices. With the communication server can be accessed from several programs (AIPEX PRO, AIPEX LITE, ATF,...) at the same time on the devices. In this way, AIPEX PRO and e. g. the Startup Tool 'AIPEX Startup' are executed in parallel.

It is not possible to run several instances of the sames software at the same time!

The program will start automatically when you call AIPEX PRO, AIPEX LITE, ATF,.. and closed again.
The started, 'AMK Communication Server' is visible in the Windows Task bar.

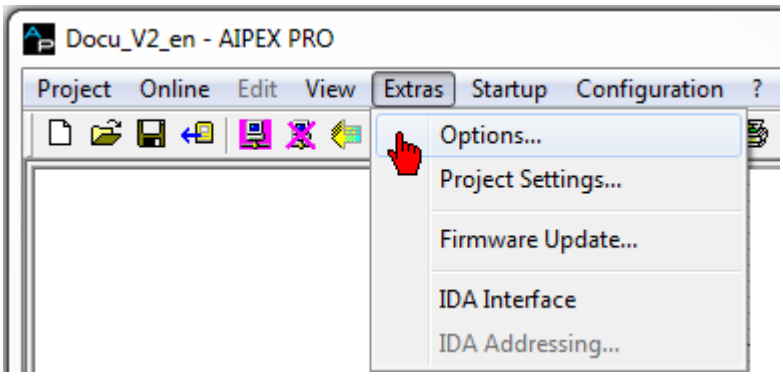


3.4.1.1 Communication server adjustments

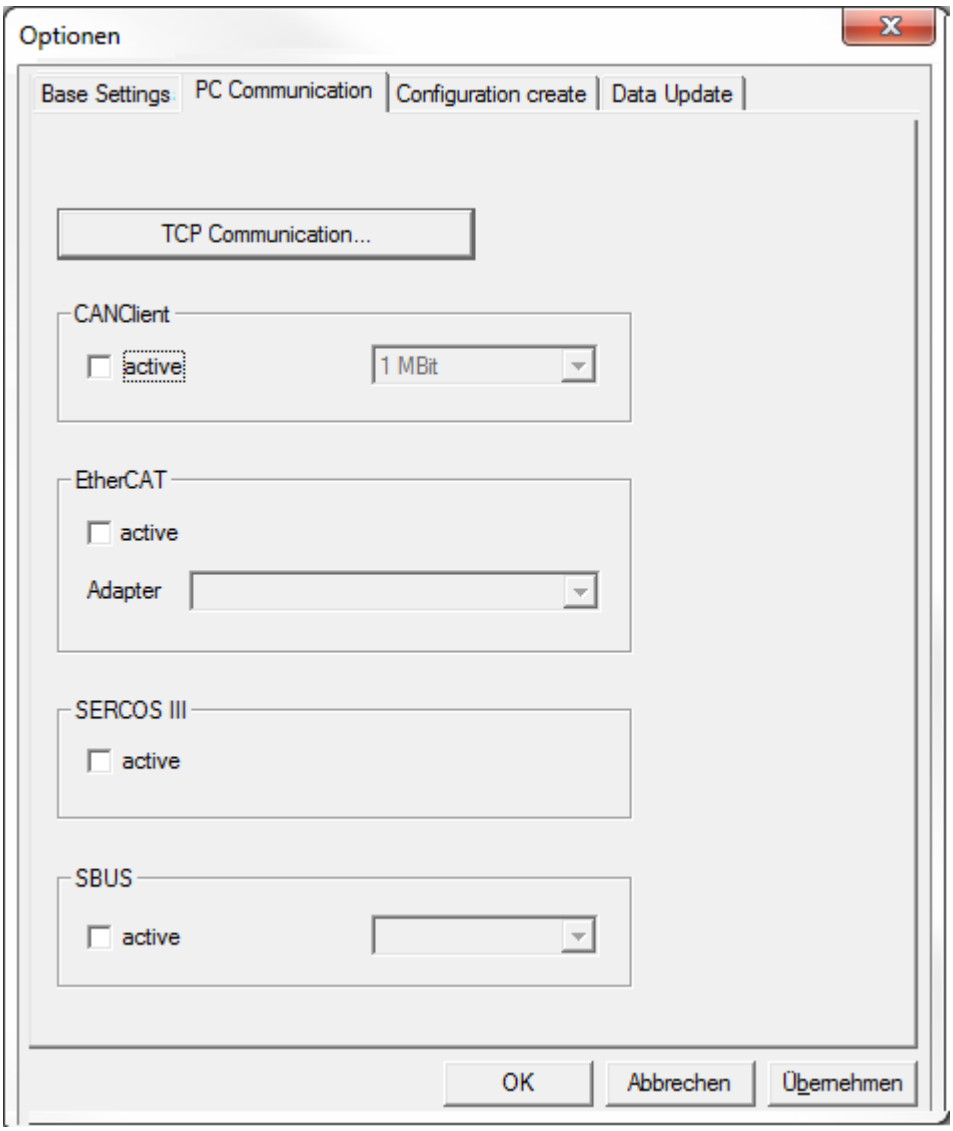
The following bus systems are supported by the communication server (Hardware components and drivers must be installed):

- Ethernet is always active
- USB is always active
- CANCLIENT has to be activated in AIPEX PRO
- EtherCAT has to be activated in AIPEX PRO

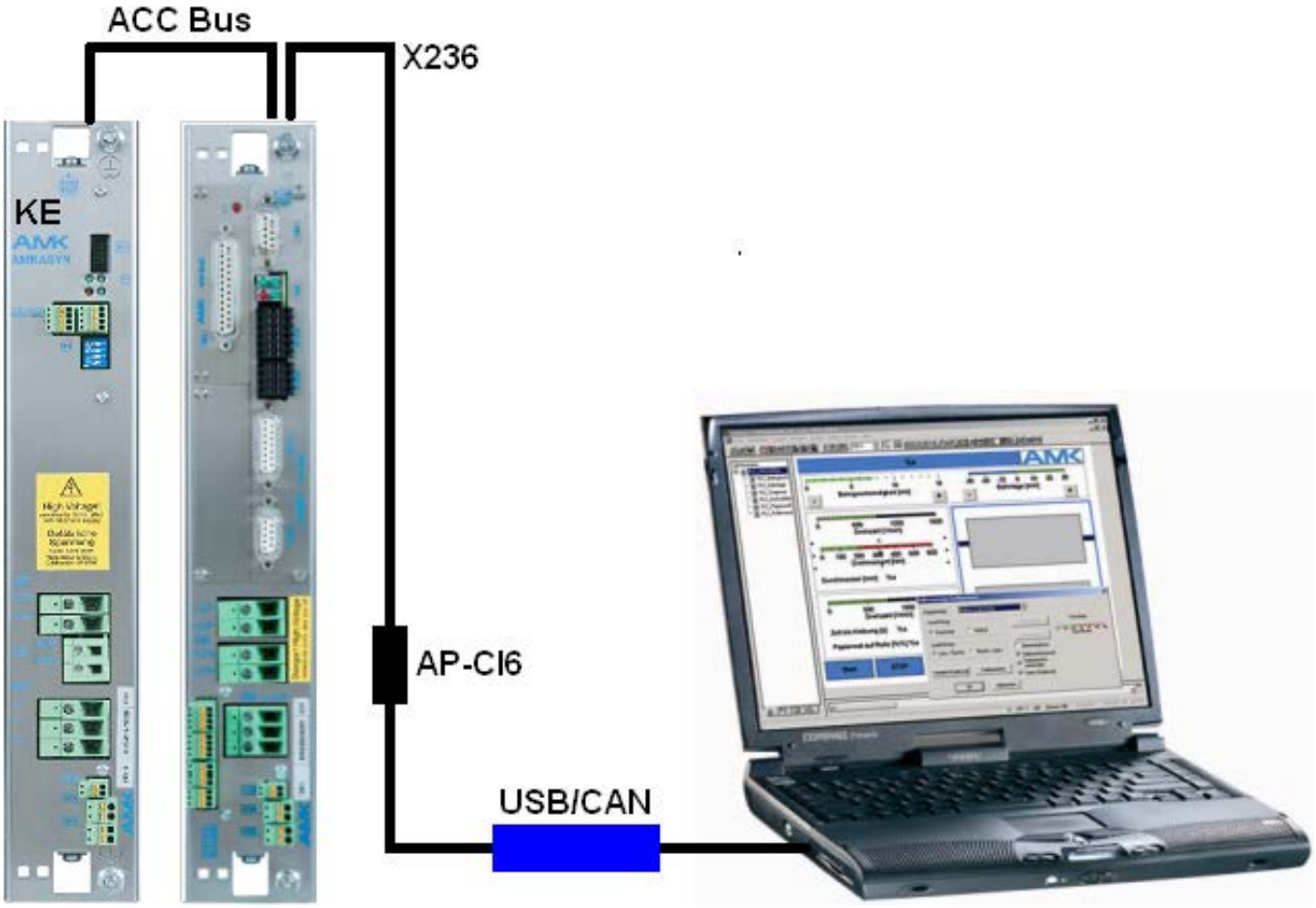
The interface must be activated in AIPEX PRO:



In register **'PC Communication'**, the interface can be activated and de-activated.


Optionen
Base Settings
PC Communication
Configuration create
Data Update
TCP Communication...
CANClient
active
1 MBit
EtherCAT
active
Adapter
SERCOS III
active
SBUS
active
OK
Abbrechen
Übernehmen

3.4.2 CAN interface

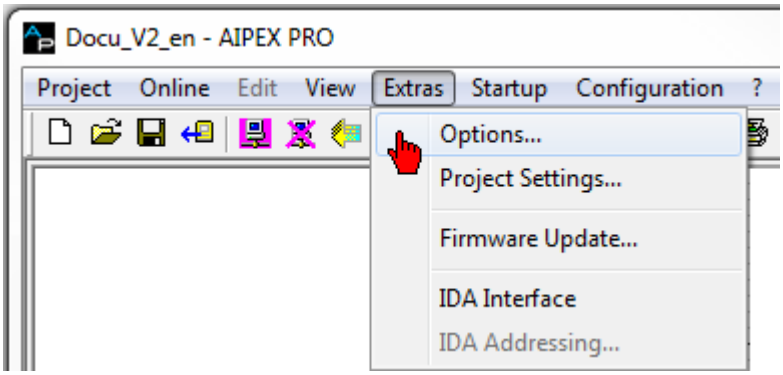


Example:
KE with ACC bus and KW with KW-R03 (comparable with KU-/KW-R03(P) and KU-/ KW-R04).

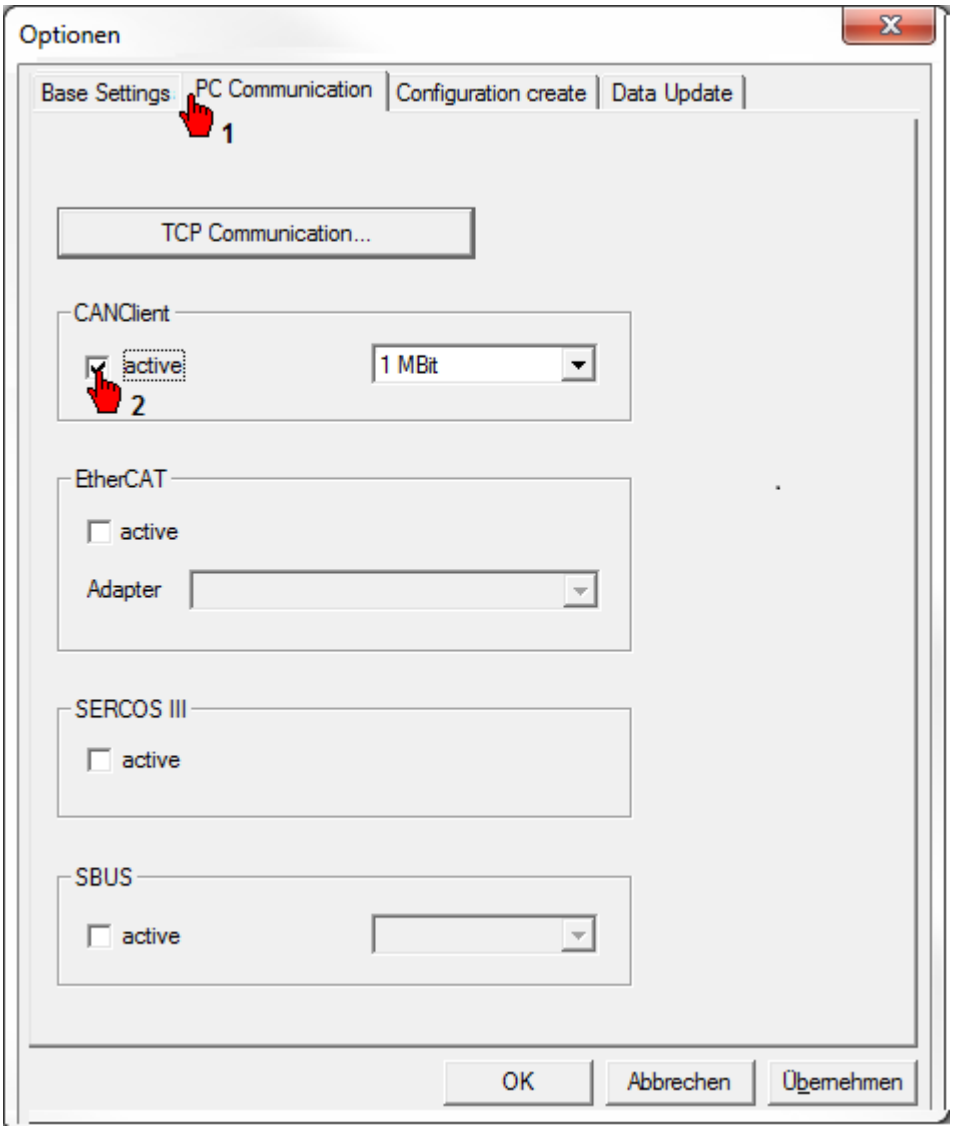
Converter:
AMK USB-CAN converter (part-no. O755). (Includes connection cables and termination resistors for 'KE/KW' series and also for 'Decentralized drive technology'.

Only ACC bus devices
CAN networks without NMT master (CAN/ACC bus master) must set:
ID34026 'BUS mode attribute' Bit 11 = 1

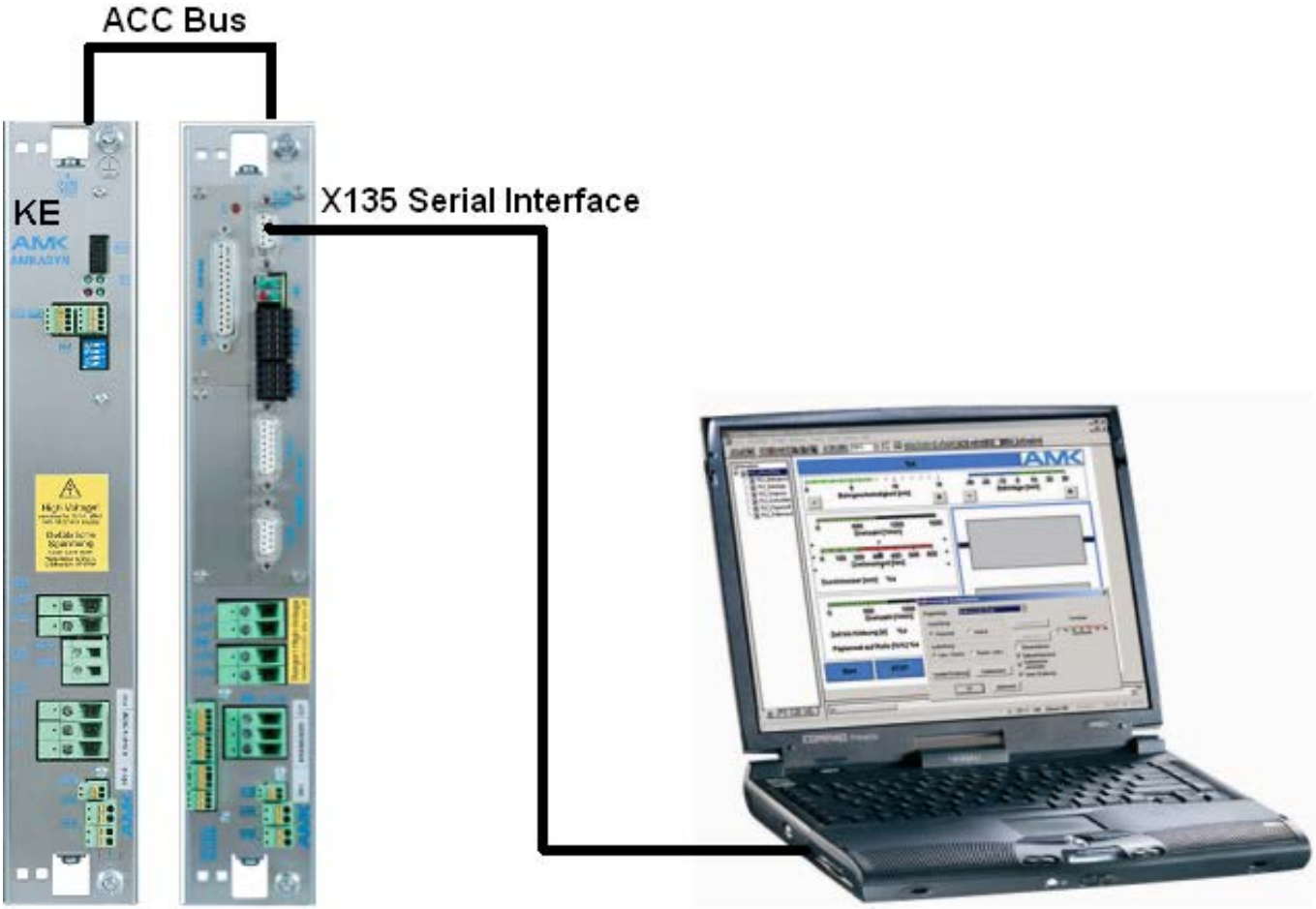
3.4.2.1 CAN adjustment



The IXXAT driver must be installed.
The USB-CAN converter (IXXAT device) must be linked with the PC.



3.4.3 COM interface



Example:

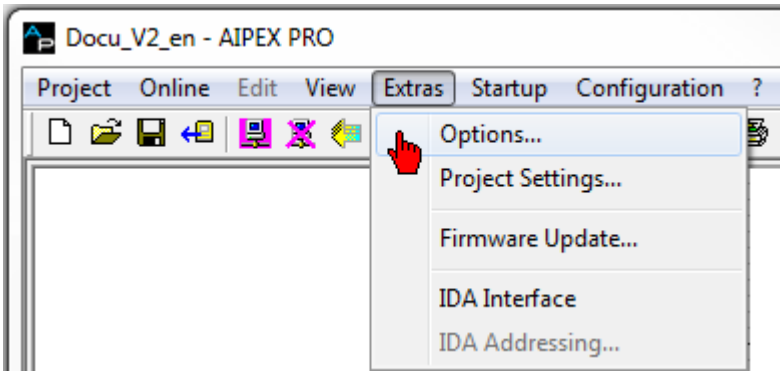
KE with ACC bus and KW with KW-R03 (comparable with KU-/KW-R03(P) and KU-/ KW-R04).

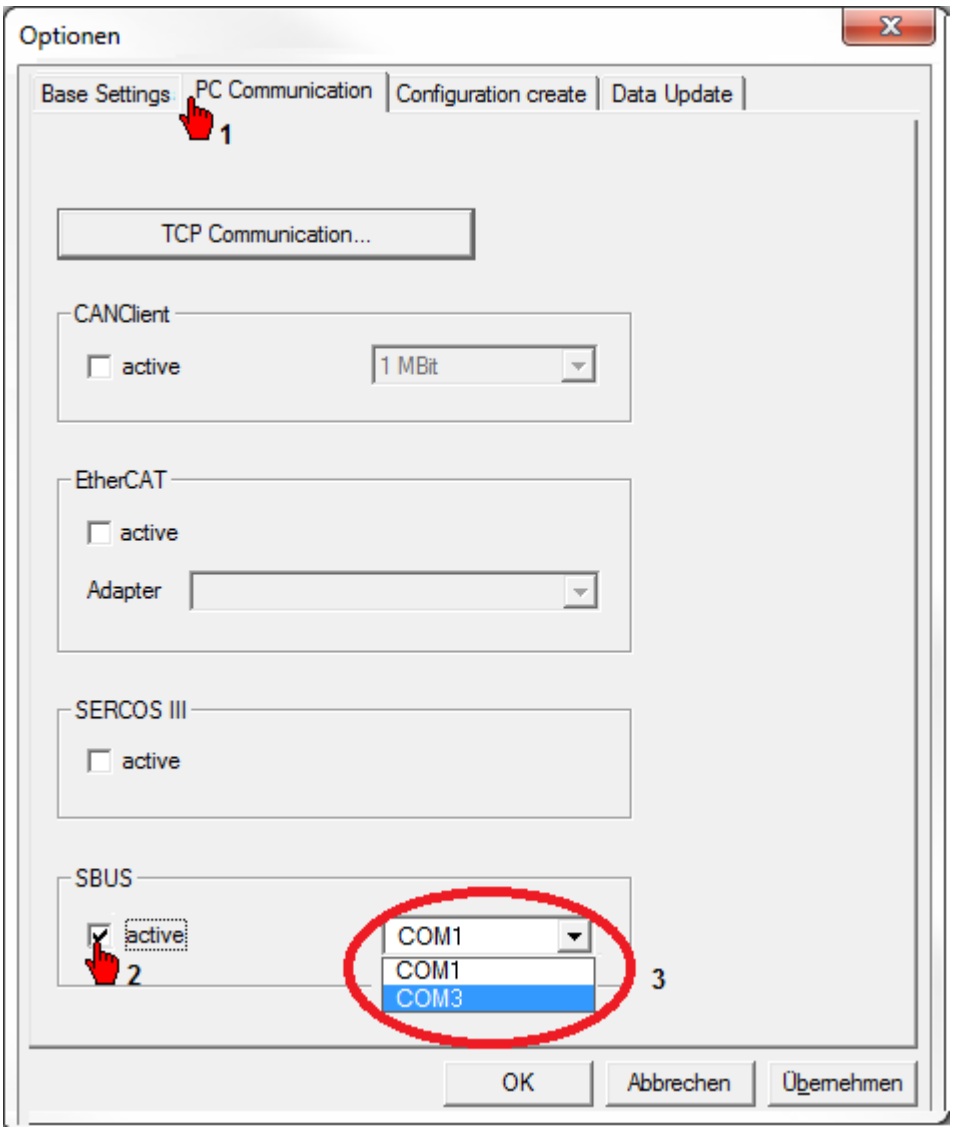
It is possible to get access over ACC bus to other devices. In this case the connected controller card must be defined as ACC bus master.

Cable:

- Serial interface cable AMK part-no. O576
- AMK USB-RS232 converter AMK part-no. 200770

3.4.3.1 COM adjustments

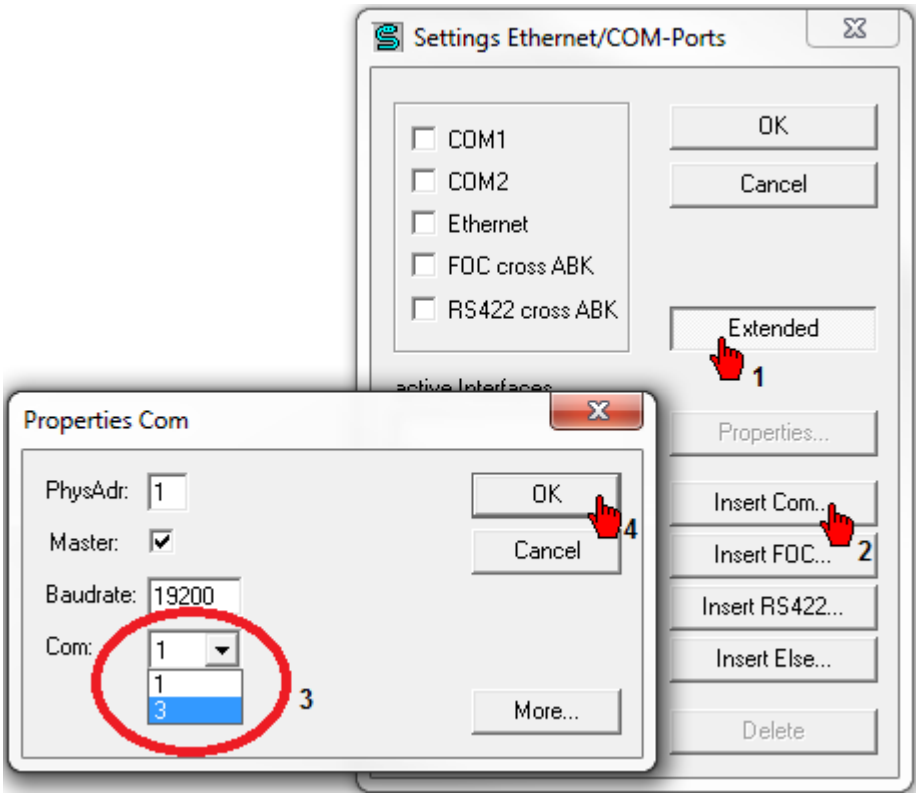



Optionen
Base Settings
PC Communication
Configuration create
Data Update
1
TCP Communication...
CANClient
active
1 MBit
EtherCAT
active
Adapter
SERCOS III
active
SBUS
active
2
COM1
COM1
COM3
3
OK
Abbrechen
Übernehmen

Set the desired port.

If you need a other port than COM1 or COM2, you can set this in **'SbusRegister'** (**'Start'** -> **'All Programs'** --> **'AMK'** --> **'SbusRegister'**) click the button **'Extended'** and than **'Insert Com…'**.

The predetermined properties may only be changed in connection with the AZ option card CNC or during a modem connection



To test the communication, connect a device with the PC and restart AIPEX PRO again.

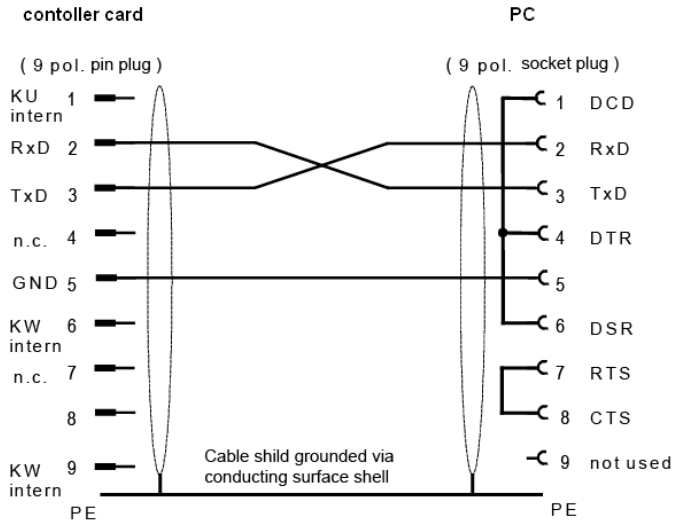
The change of the colour of the symbol from red to green at the bottom right edge indicates that the communication is functioning properly.

A green/red symbol indicates that there is more than one active interface.

3.4.3.2 Serial AMK cable RS232 (PC-AMK)

The AMK serial interface cable RS232 (AMK part-no. O576) connects the serial interface of the AMK KU-/ KW devices with the serial interface of the PC.

Property of serial interface cable:

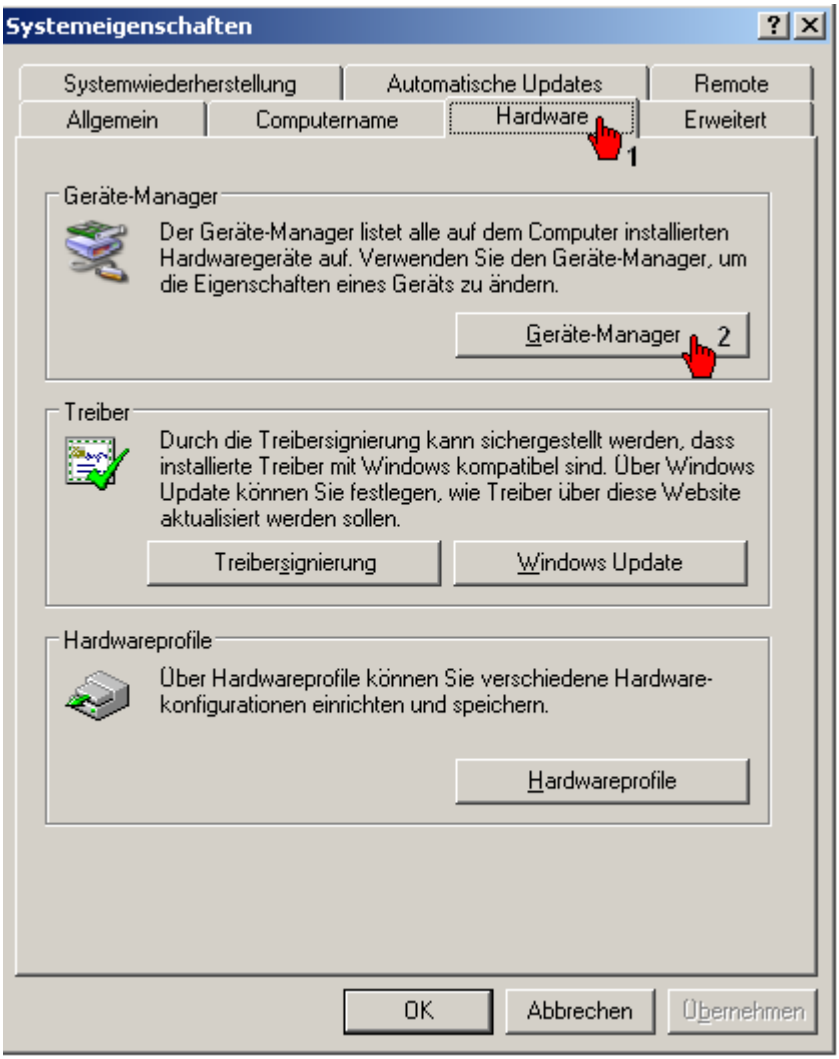


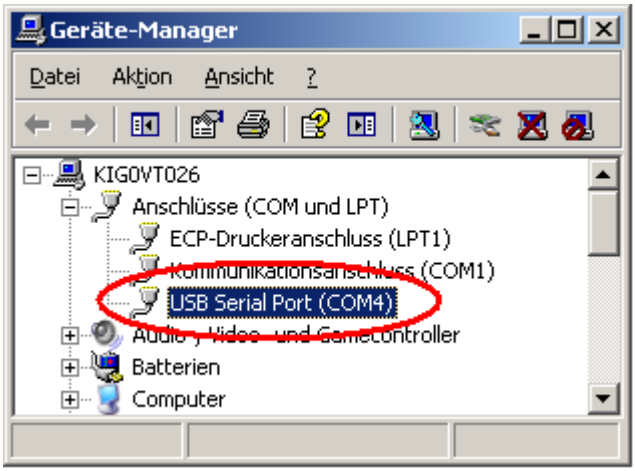
3.4.3.3 USB-RS232 Interface

The AMK USB-RS232 converter (AMK part-no. 200770) connects the serial interface of the AMK KU-/ KW devices with the USB interface of the PC.

Example Windows XP

During the driver installation of a AMK USB-RS232 converter, an additional virtual COM interface is generated by the PC. The interface number can be read out. Open **'Windows'** -> **'System Properties'** -> **'Hardware'** -> **'Device Manager'** -> **'Ports (COM and LPT)'**.

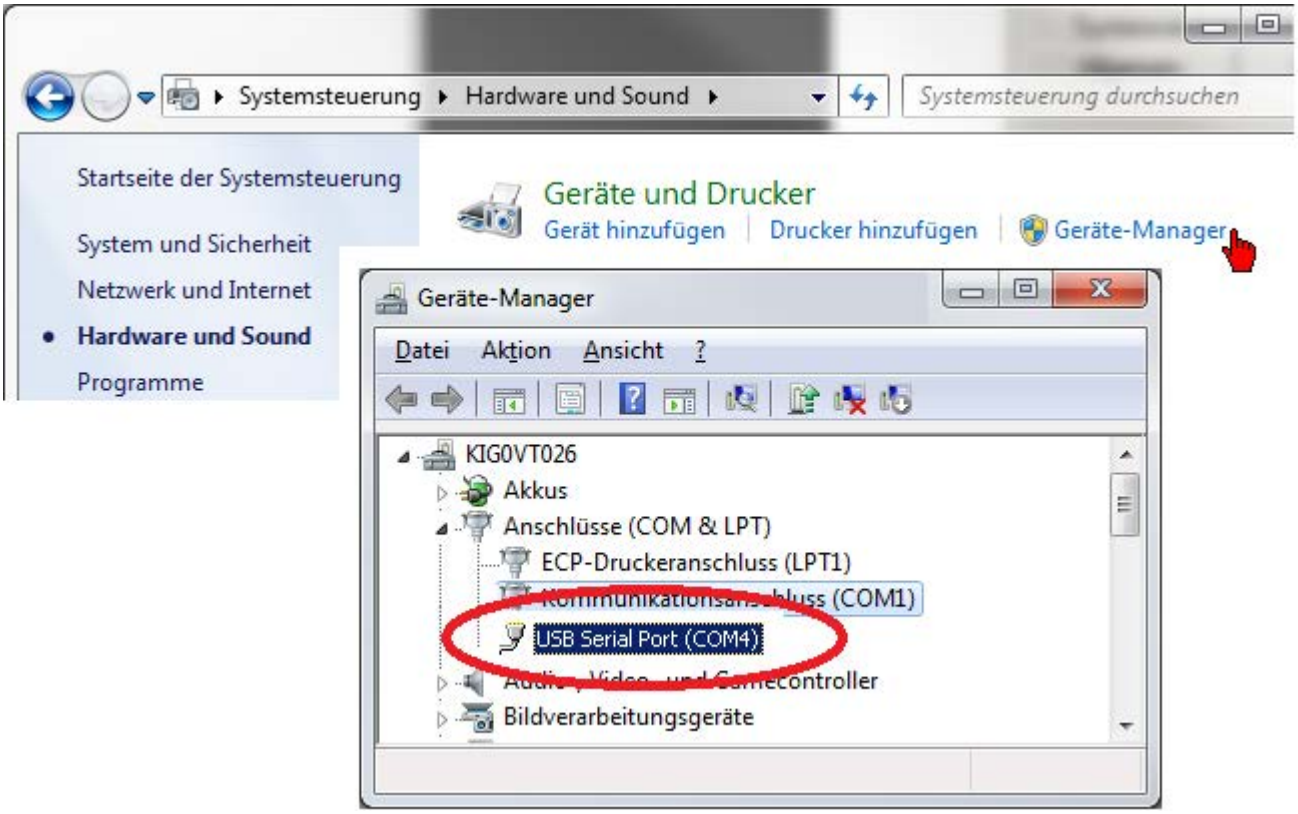




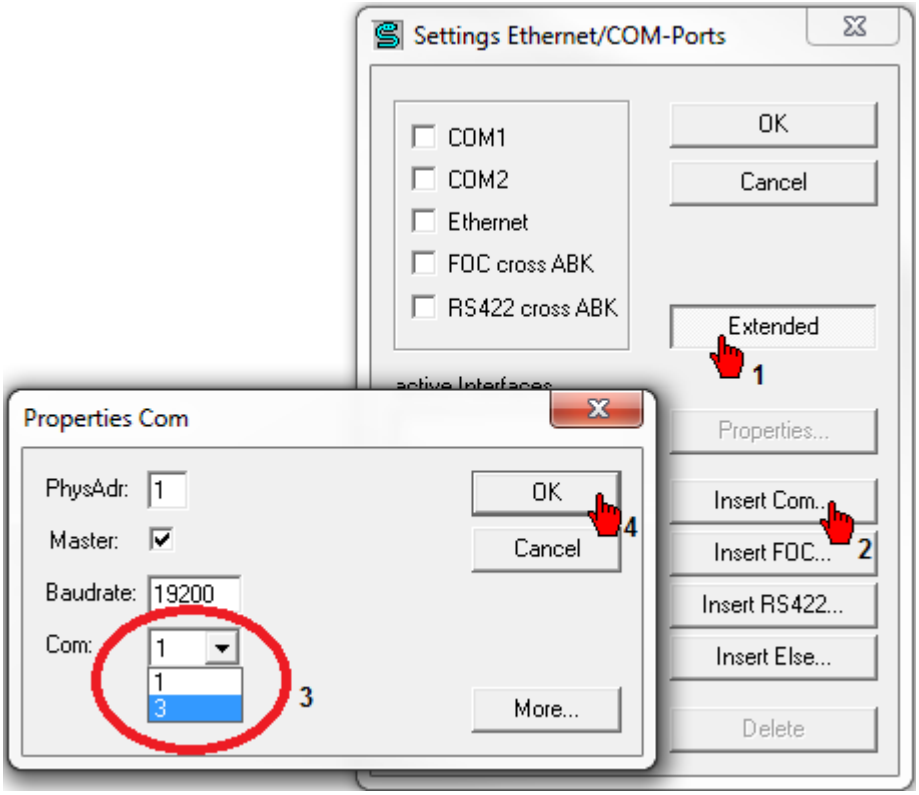
Example Windows 7

Invoke the Windows menu **'System control'** -> **'Hardware and Sound'**.

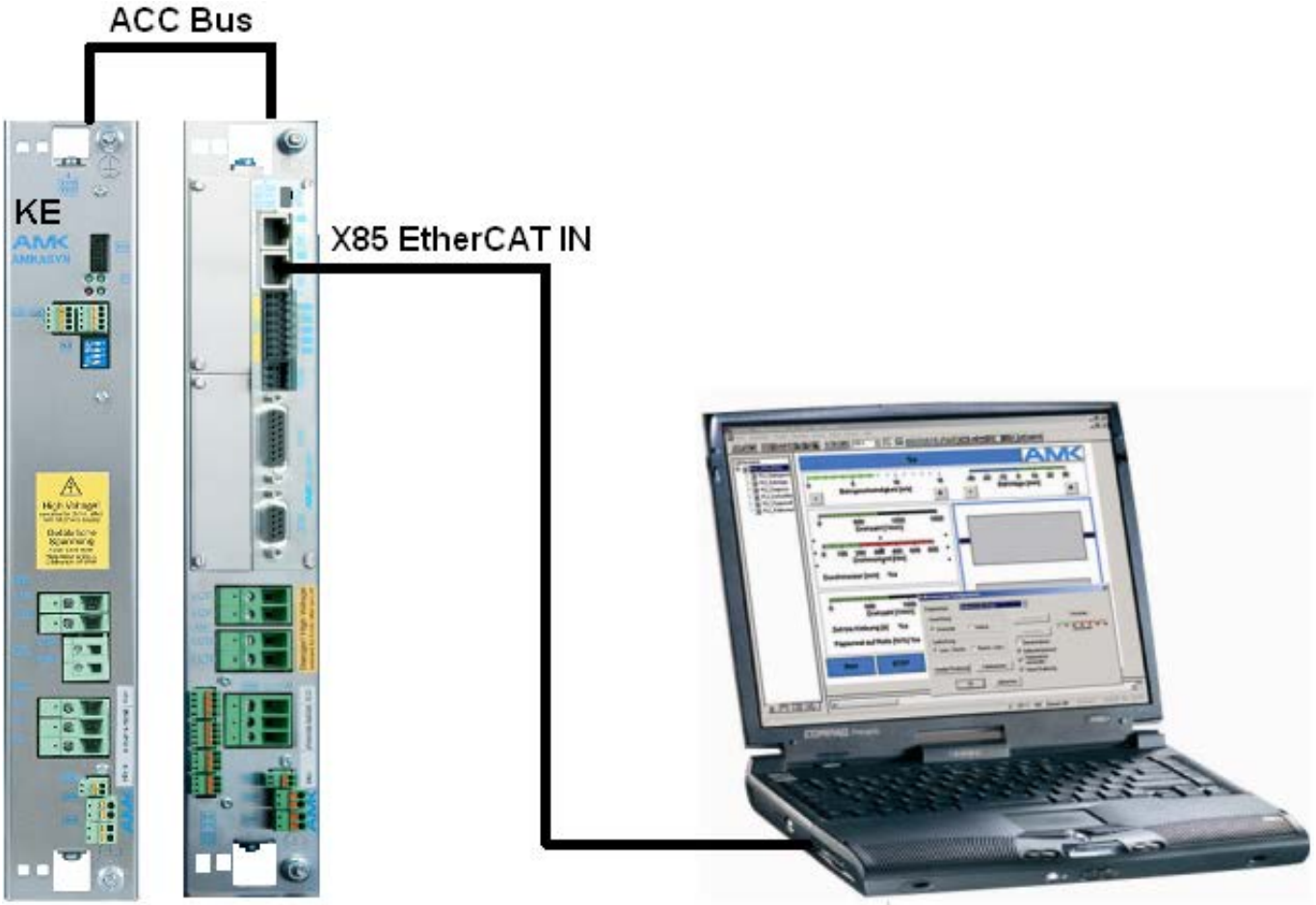
Invoke the **'Device-manager'**.



Enter the USB Serial Port interface number (in this example COM3) in the **'SbusRegister'** (**'Start'** -> **'All Programs'** --> **'AMK'** --> **'SbusRegister'**). Use for that the button **'Extended'** and the button **'Insert COM'**.



3.4.4 EtherCAT interface



Example:

KE with ACC bus and KW with KW-R06 (comparable with KW-R07, KW-R16 and KW-R17).

Access is possible via the EtherCAT connection to all connected modules. You can also access ACC bus slave devices via the ACC bus connection of the controller card.

Cable:

Ethernet Standard RJ45 Twisted Pair Patch cable.

An Ethernet cable with an M12 connector is required for the connection to the 'Decentralized drive technology':

See device description AMK Part no. 203445 Decentralized drive technology iC / iX / iDT5, chapter: Cables for EtherCAT connector [X85] and [X86].

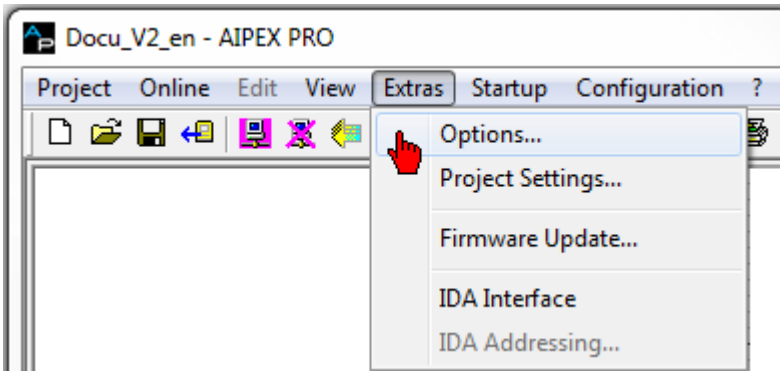
3.4.4.1 EtherCAT adjustments

EtherCAT master

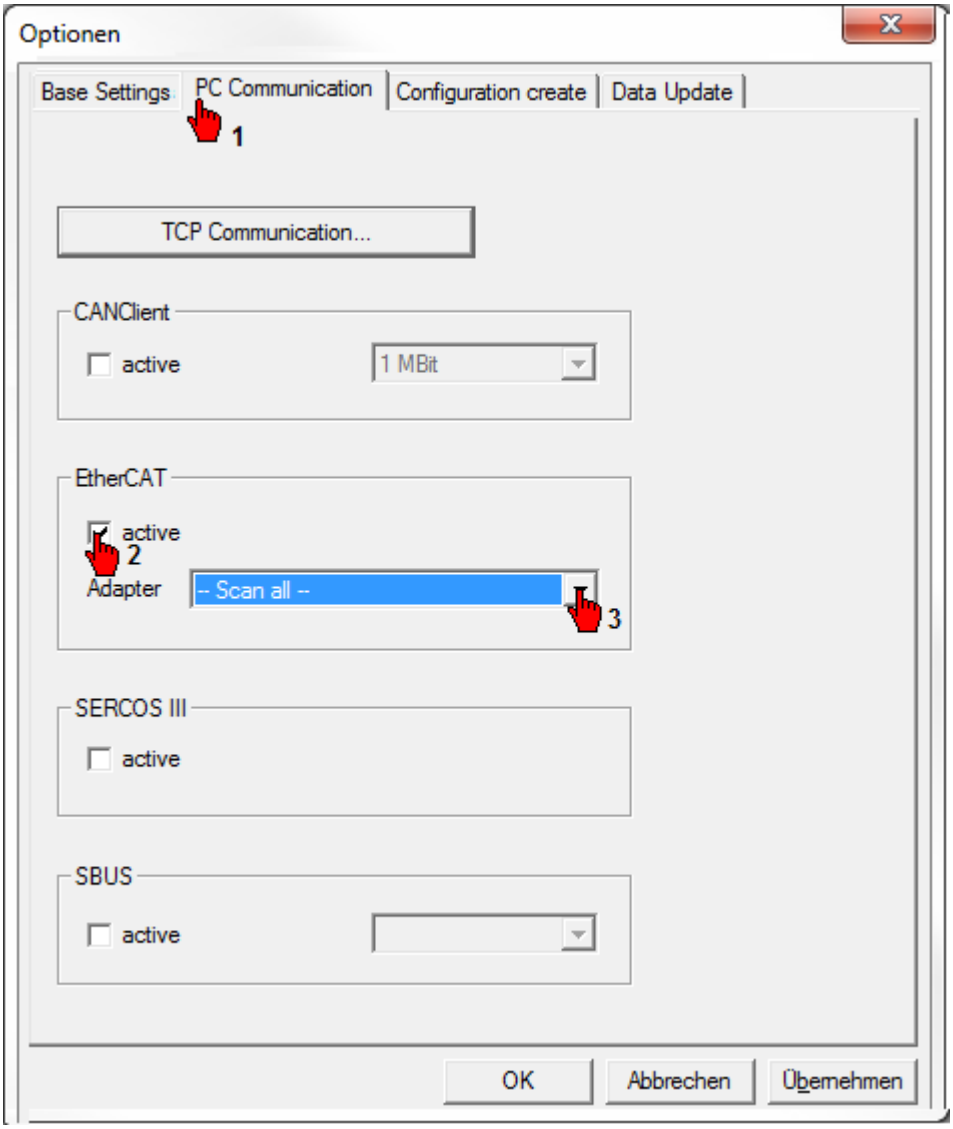
On a point to point connection via EtherCAT, the PC works as EtherCAT master.
The Ethernet network settings of the PC are not to be changed.

EtherCAT slave

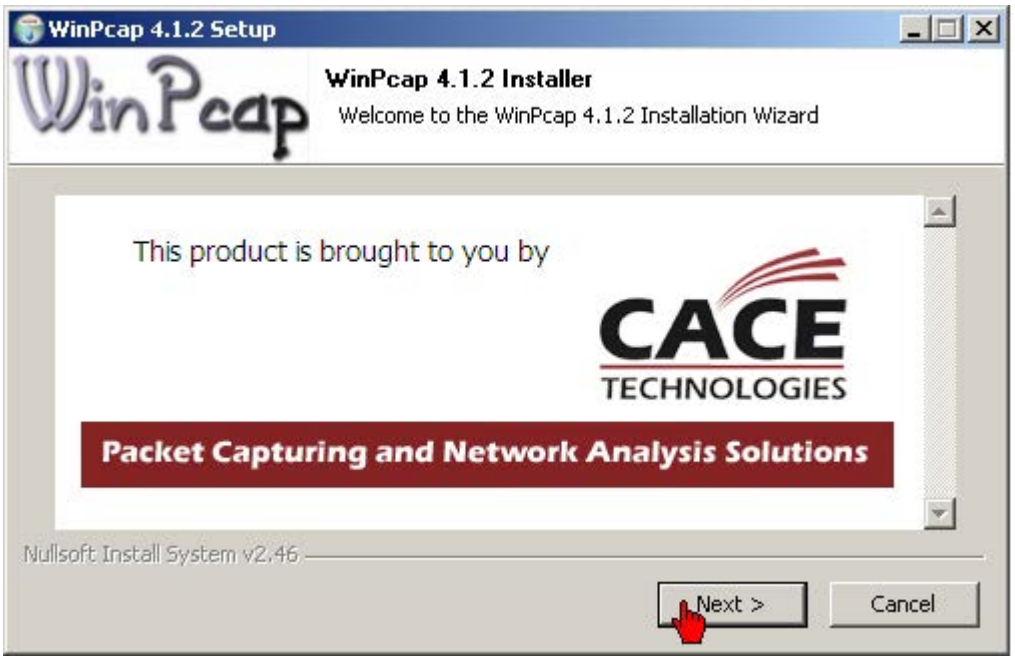
The connected drive (compact power supply or compact inverter) establishes the EtherCAT slave.
Its network settings must be adapted:



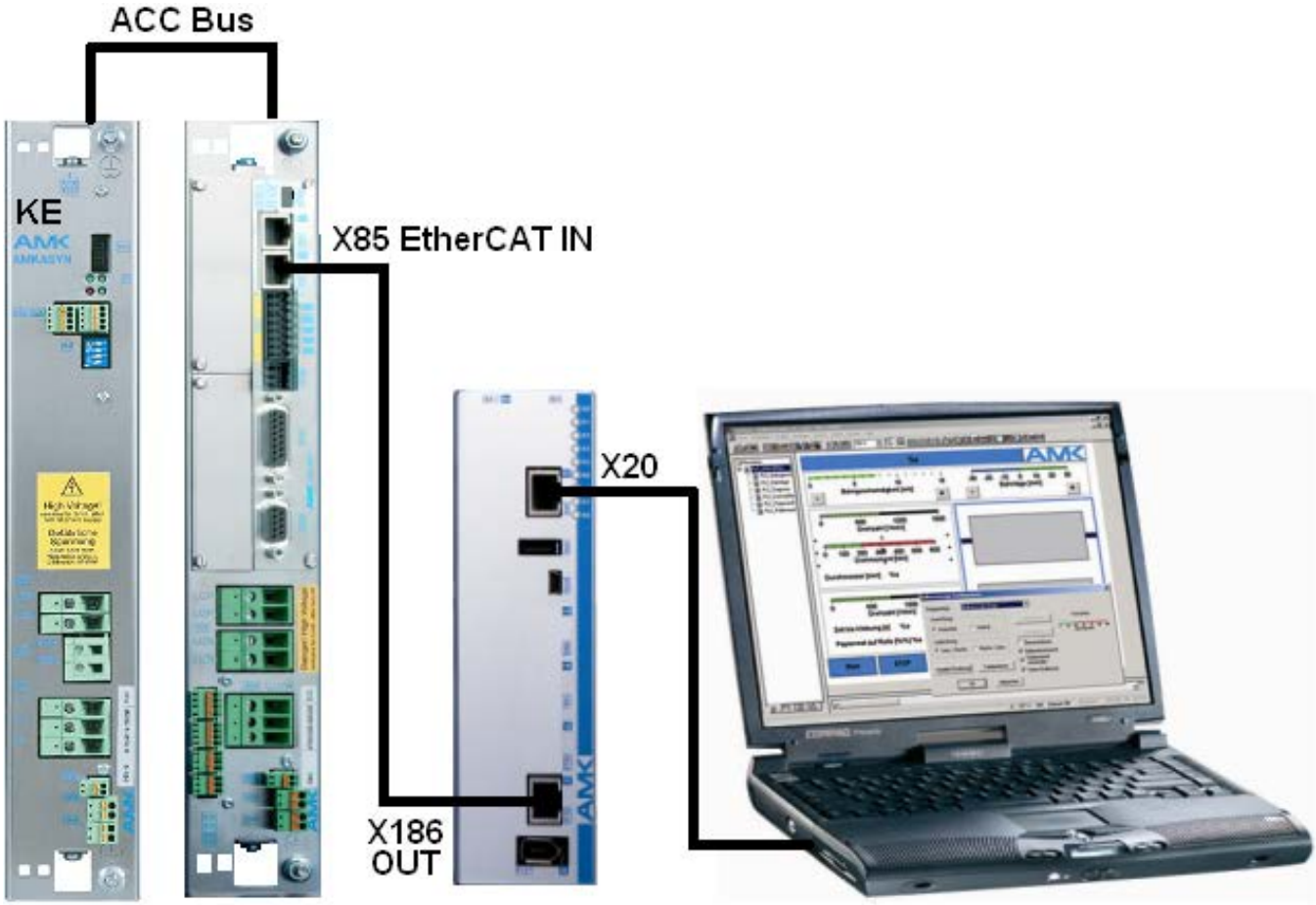
In register **'PC Communication'**, the EtherCAT communication can be activated and de-activated.
You can shorten the initialising phase by selecting the used Ethernet adapter of your interface.



Prerequisite:
WinPcap will be automatically installed with AIPEX PRO



3.4.5 Ethernet interface



Example:

Controller, KE with ACC bus and KW with KW-R06 (comparable with KW-R07, KW-R16 and KW-R17).

Access is possible via the EtherCAT connection to all connected modules. You can also access ACC bus slave devices via the ACC bus connection of the controller card.

Cable:

Ethernet Standard RJ45 Twisted Pair Patch cable.

If your PC not automatically crossed a Twisted Pair Cable, use a RJ45 Crosscable for the connection.

With an active firewall, the following releases are necessary:

- TCP Port 700
- TCP 50.001
- UDP Port 40.000
- Broadcast on

3.4.5.1 Ethernet adjustments

The Ethernet interface is always active.

PC adjustment for Point - to - Point connection

The standard address 192.168.0.1 is saved in the AMKAMAC compact controller. The address belongs to the net address range of the subnet mask 255.255.255.0.

For a successful communication, the PC and compact controller need to lie in the same address range.

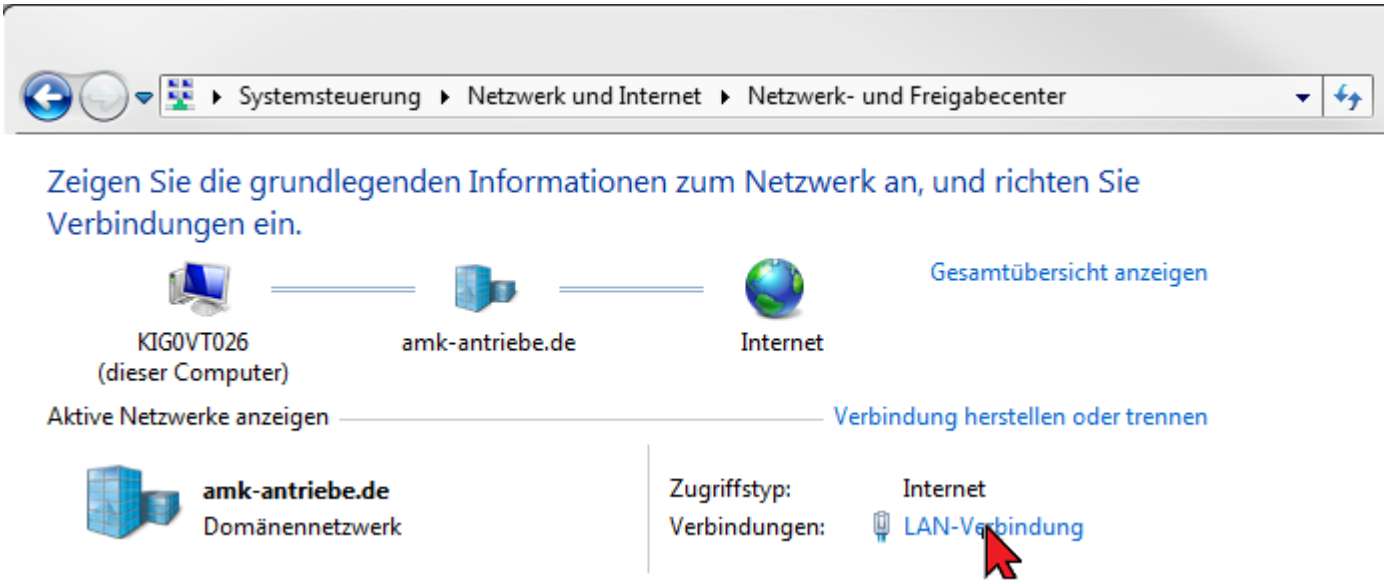
In the following, instructions are provided on how you save a fixed IP address in your PC along with the corresponding subnet mask.

If you use the combination PC -- (company-) network -- AMK Controller, you have to adjust the controller IP address onto the (company-)network adjustments.

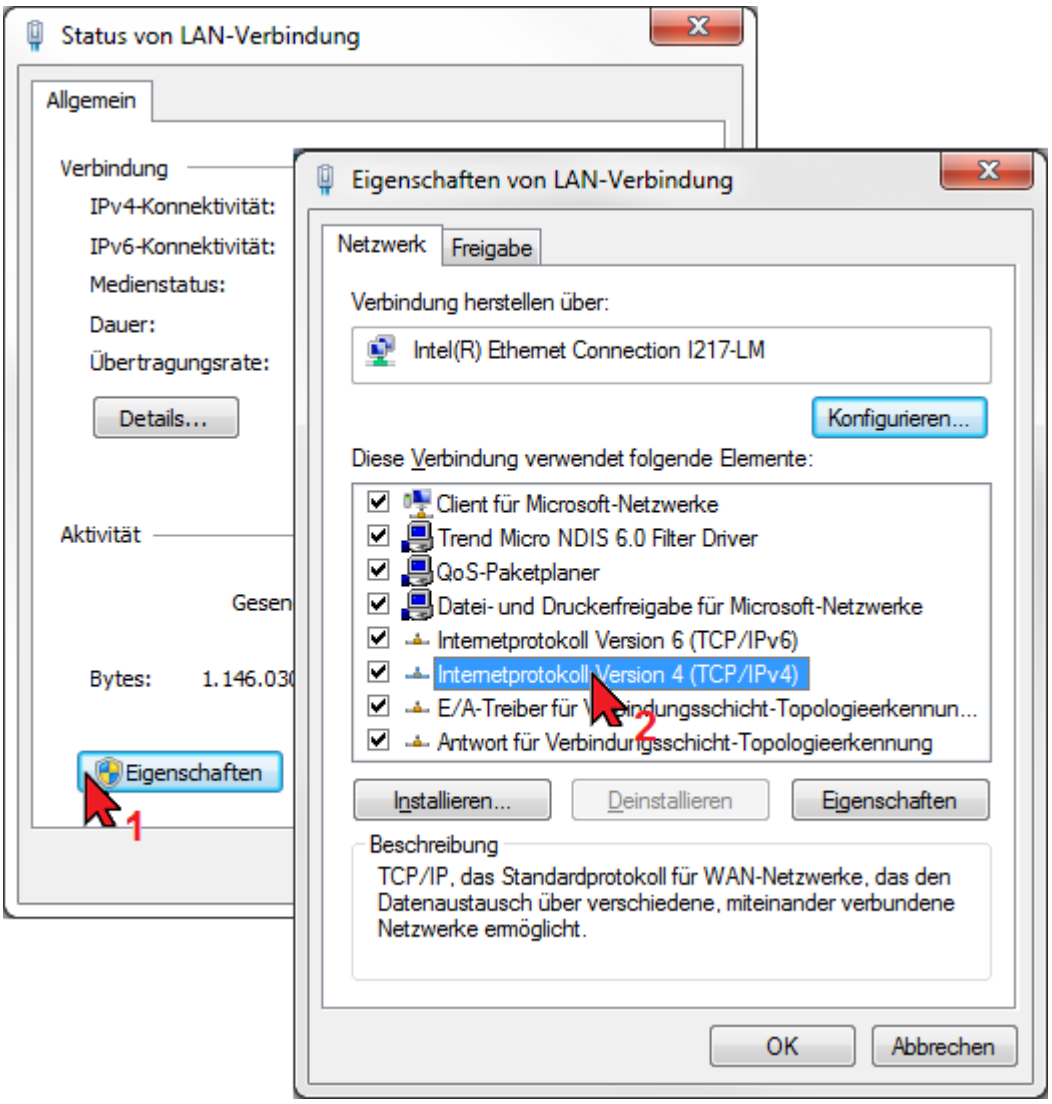
Example Windows 7

Invoke the Windows menu **'Systemsteuerung'** -> **'Netzwerk und Internet'** -> **'Netzwerk- und Freigabecenter'** Open your active LAN connection by clicking on it.

No active LAN connection: Invoke the menu **'Adaptereinstellungen ändern'** .



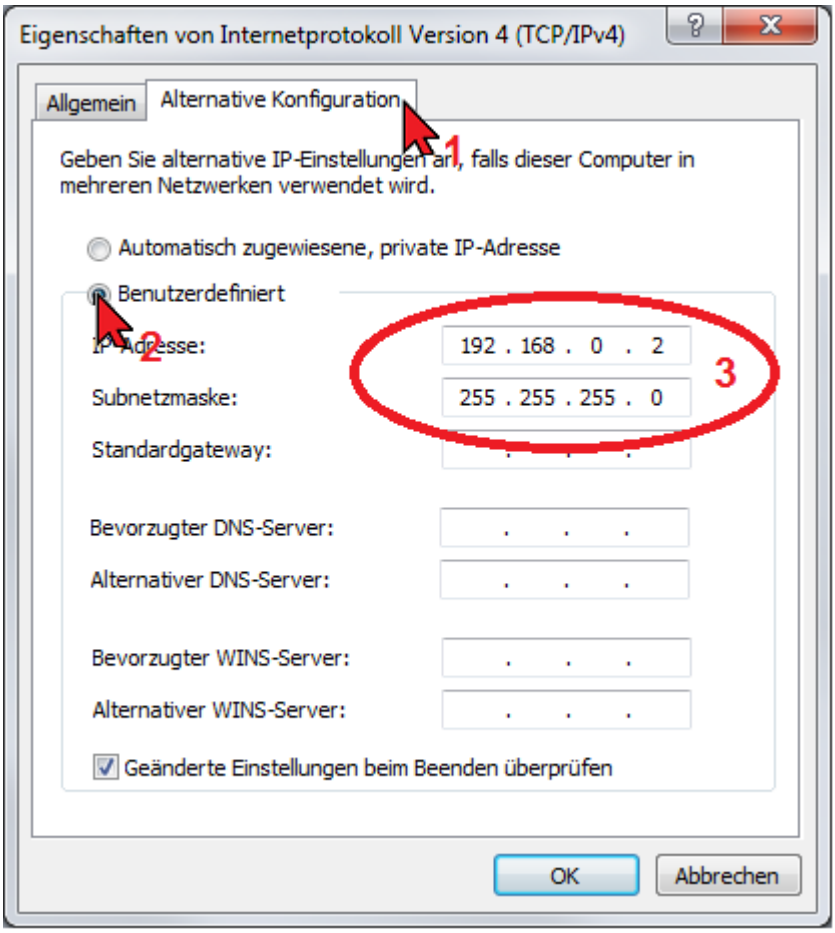
Open the Window **'Internetprotokoll Version 4 (TCP/IPv4)'**



Enter in the tab **'Alternative configuration'** under **'User defined'** the 'IP address 192.168.0.2' and the 'subnet mask 255.255.255.0'.

Confirm by pressing **'OK'**.

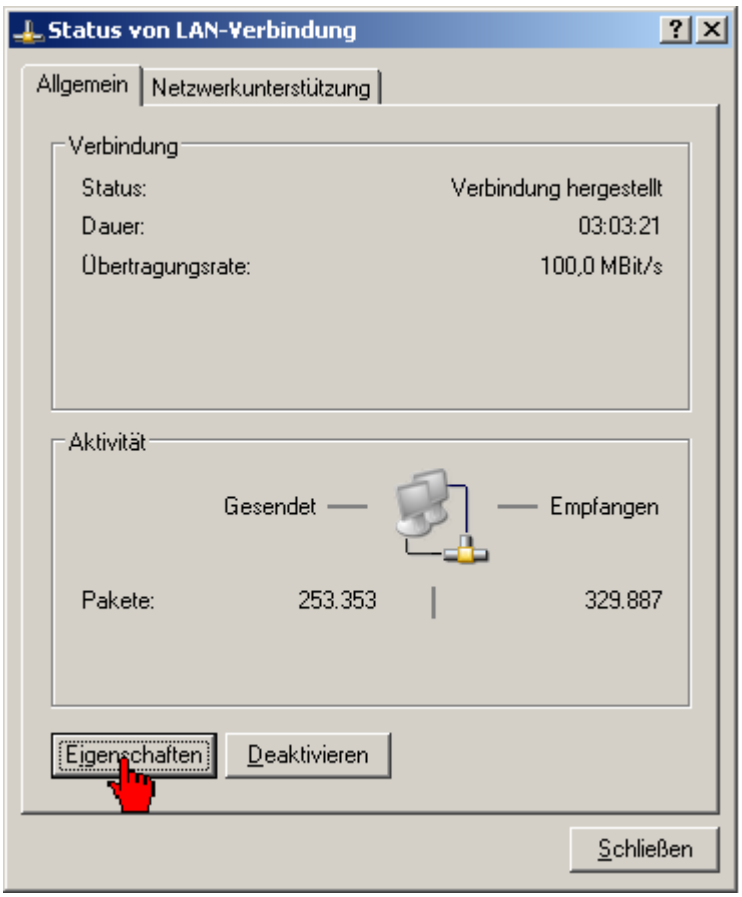
The connection initialisation goes faster if you use the tap 'Allgemein' to enter your IP address. But in this case you always have to change the IP address manually if you change between company network and controller.



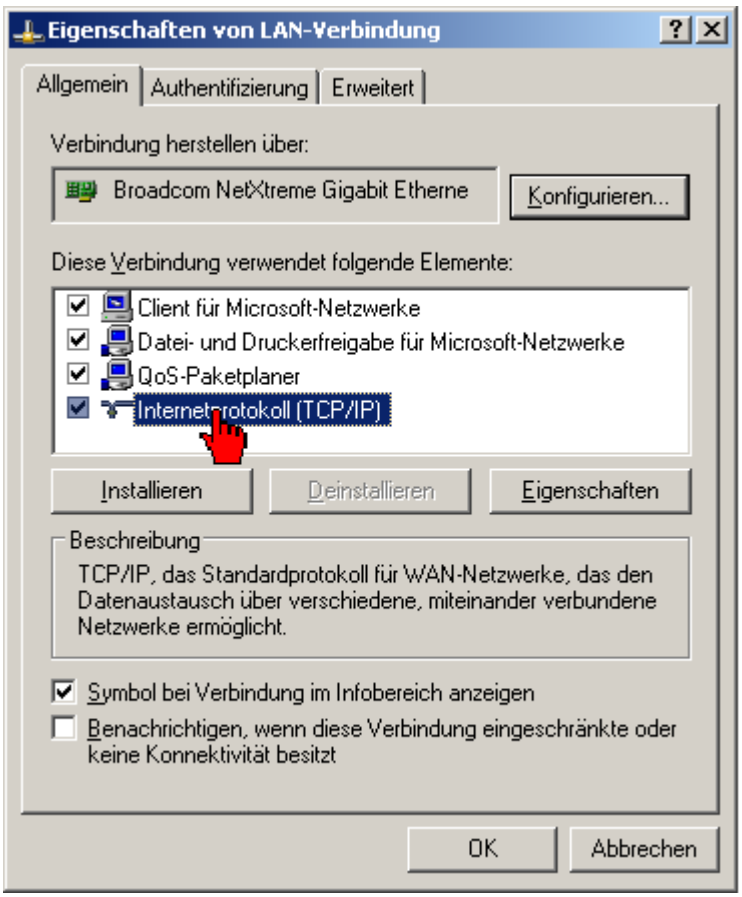
Example Windows XP

Invoke the Windows menu **'Network connections'**. Open your active LAN connection by clicking on it.

Select the button **'Properties'**.



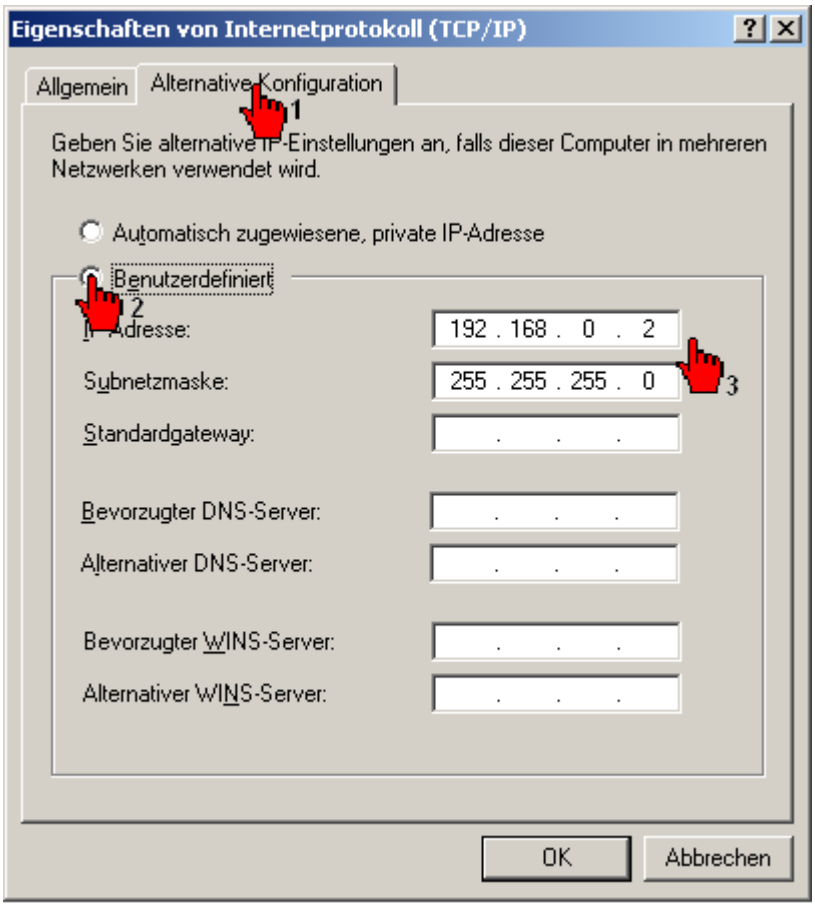
Open the properties of the **'Internet protocol TCP/IP'** by clicking on it.



Enter in the tab **'Alternative configuration'** under **'User defined'** the 'IP address 192.168.0.2' and the 'subnet mask 255.255.255.0'.

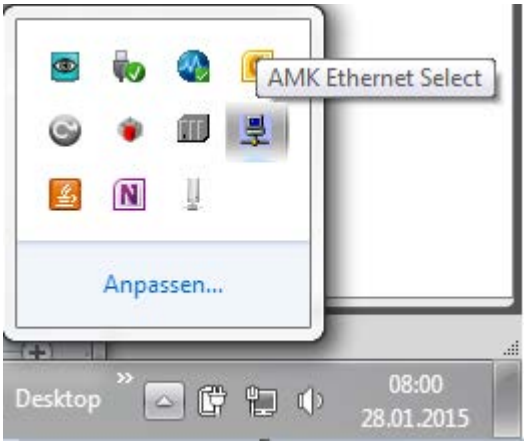
Confirm by pressing **'OK'**.

The connection initialisation goes faster if you use the tap 'Allgemein' to enter your IP address. But in this case you always have to change the IP address manually if you change between company network and controller.



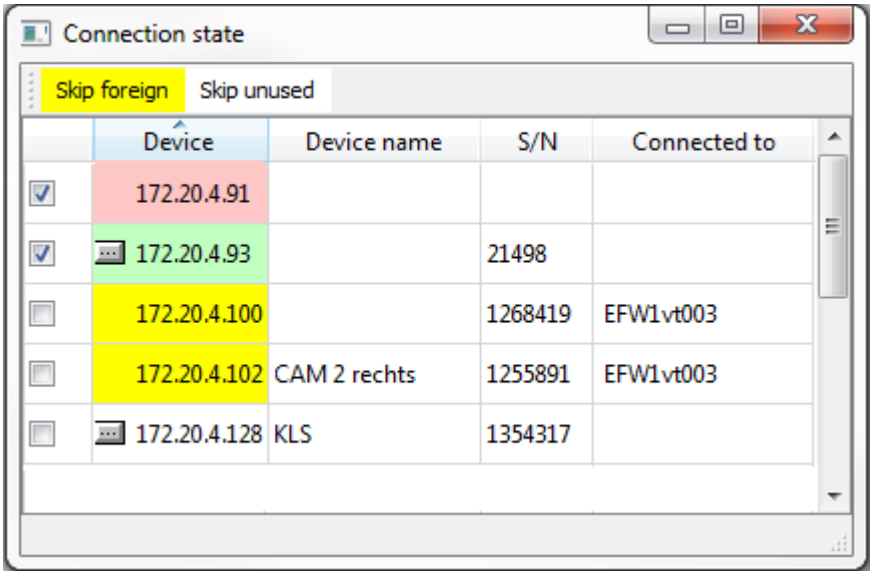
AMK Ethernet Select

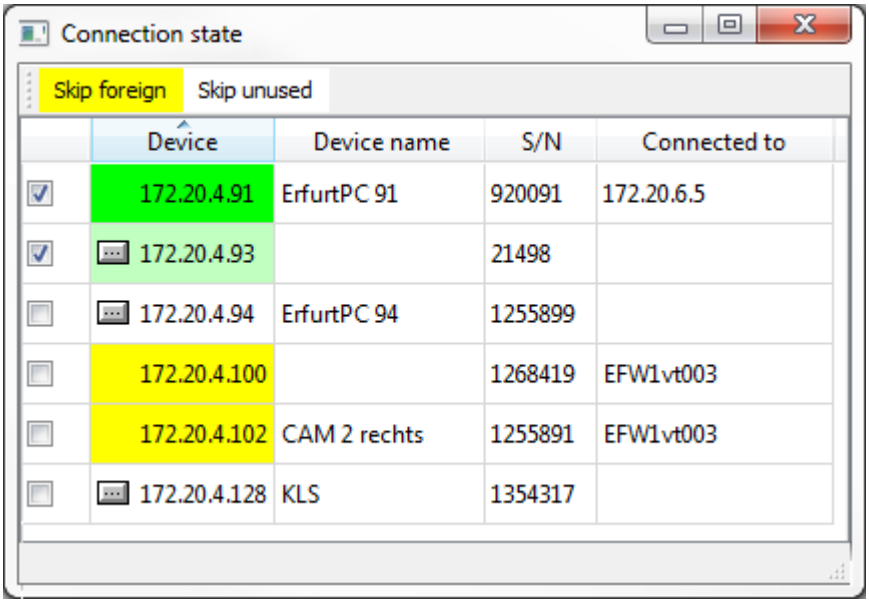
With active Ethernet communication, you will find **'AMK Ethernet Select'** in the Windows Task bar. Select the icon **'AMK Ethernet Select'** to open the **'Connection state'** dialog box.



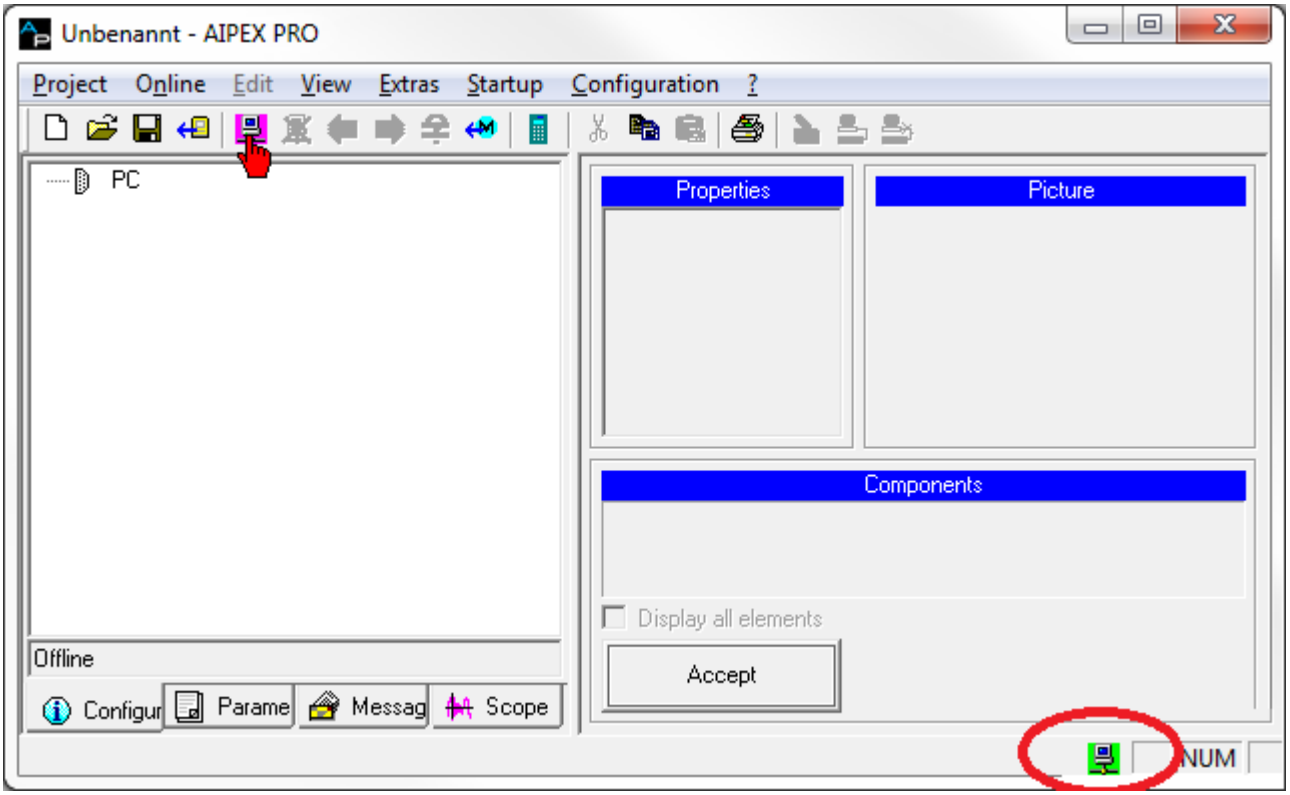
All active and manually created AMK Ethernet devices are displayed that were created in the title bar **device**. Select the device with which you want to establish a connection.

Colour status	Meaning
Red	Device cannot be reached via Ethernet
Yellow	Device is connected with a different PC
Light Green	Device is connected with your PC
Green	Device is connected with your PC and it is being actively accessed to the device
White	Device is not connected

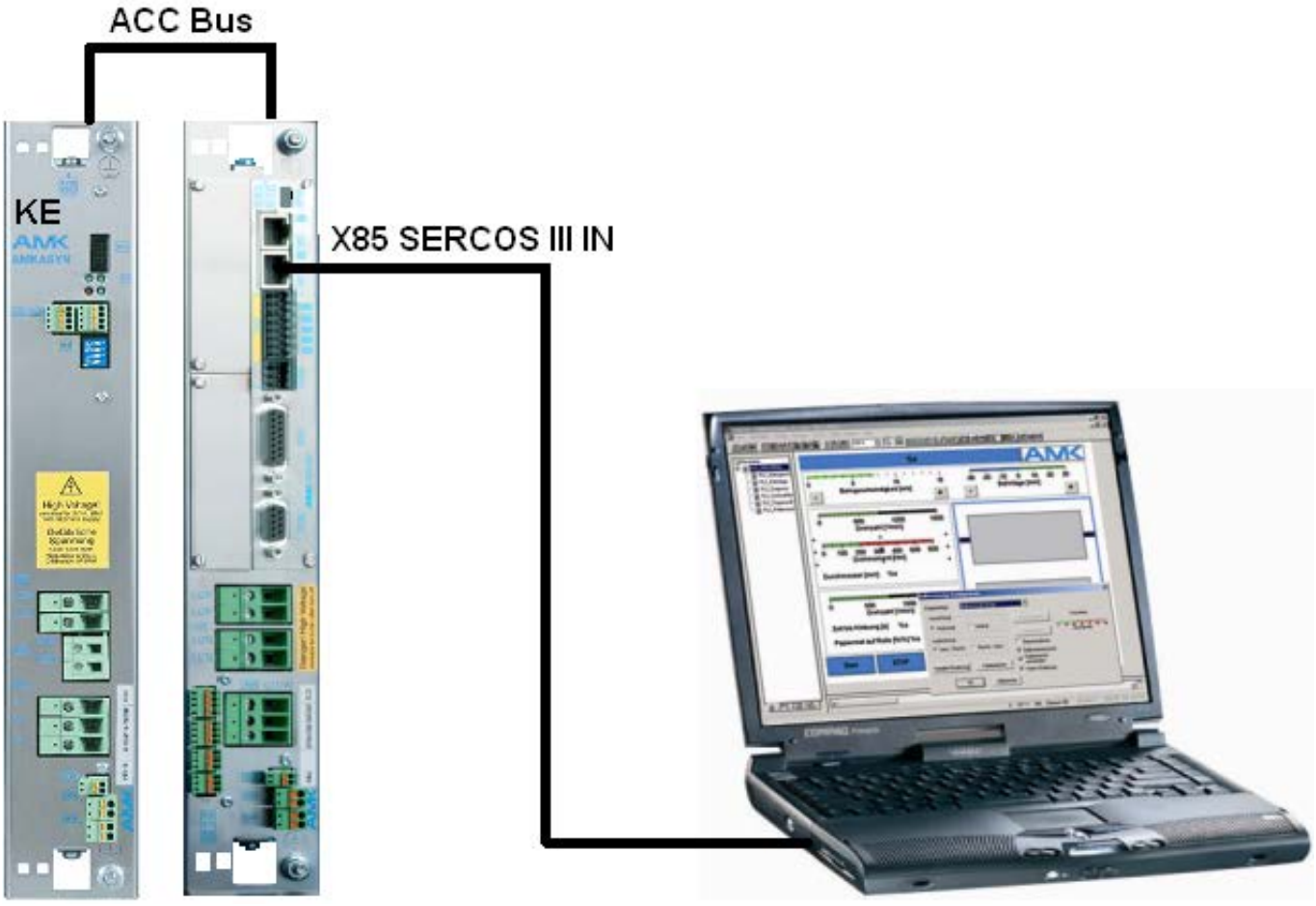




Close **'AMK Ethernet Select'**.
As soon as the status communication icon turns green/yellow, you can press the **'Logon'** button.



3.4.6 SERCOS III interface



Example:

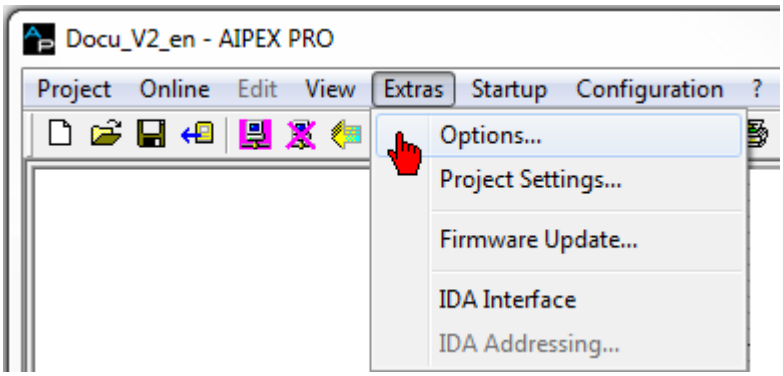
KE with ACC bus and KW with KW-R06 (comparable with KW-R07, KW-R16 and KW-R17).

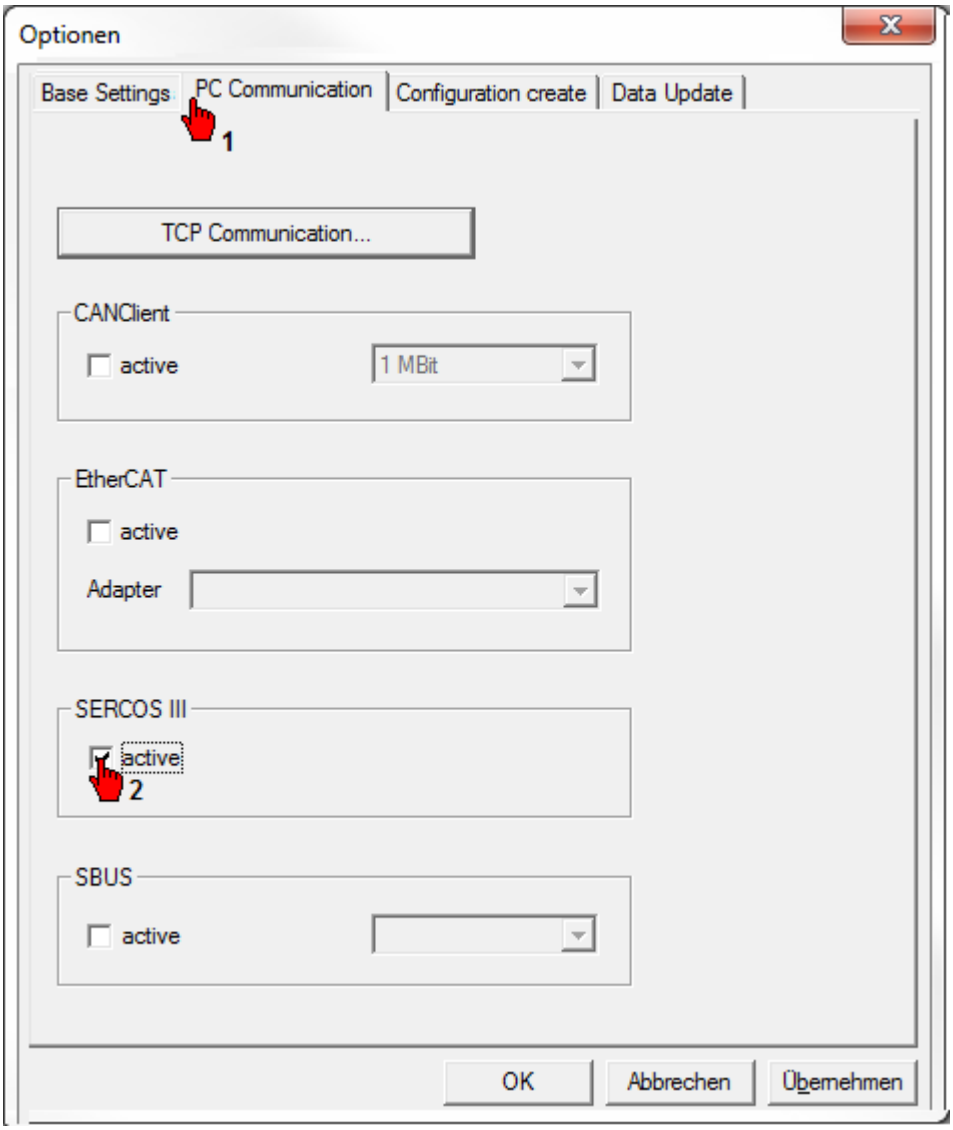
Access is possible via the SERCOS III connection to all connected modules. You can also access ACC bus slave devices via the ACC bus connection of the controller card.

Cable:

Ethernet Standard RJ45 Twisted Pair Patch cable.

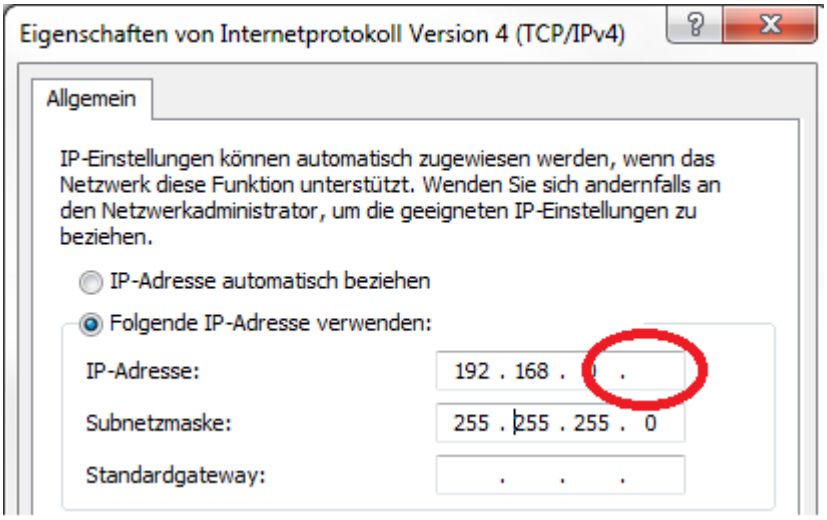
3.4.6.1 SERCOS III adjustment



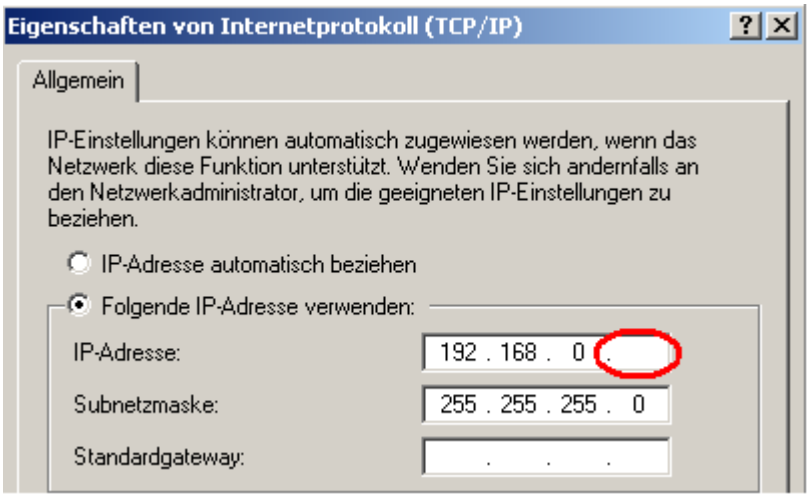


PC adjustment: Enter a fix IP address which is not used by a SERCOS III node.
The SERCOS III node addresses are fixed given to 192.168.0.X.

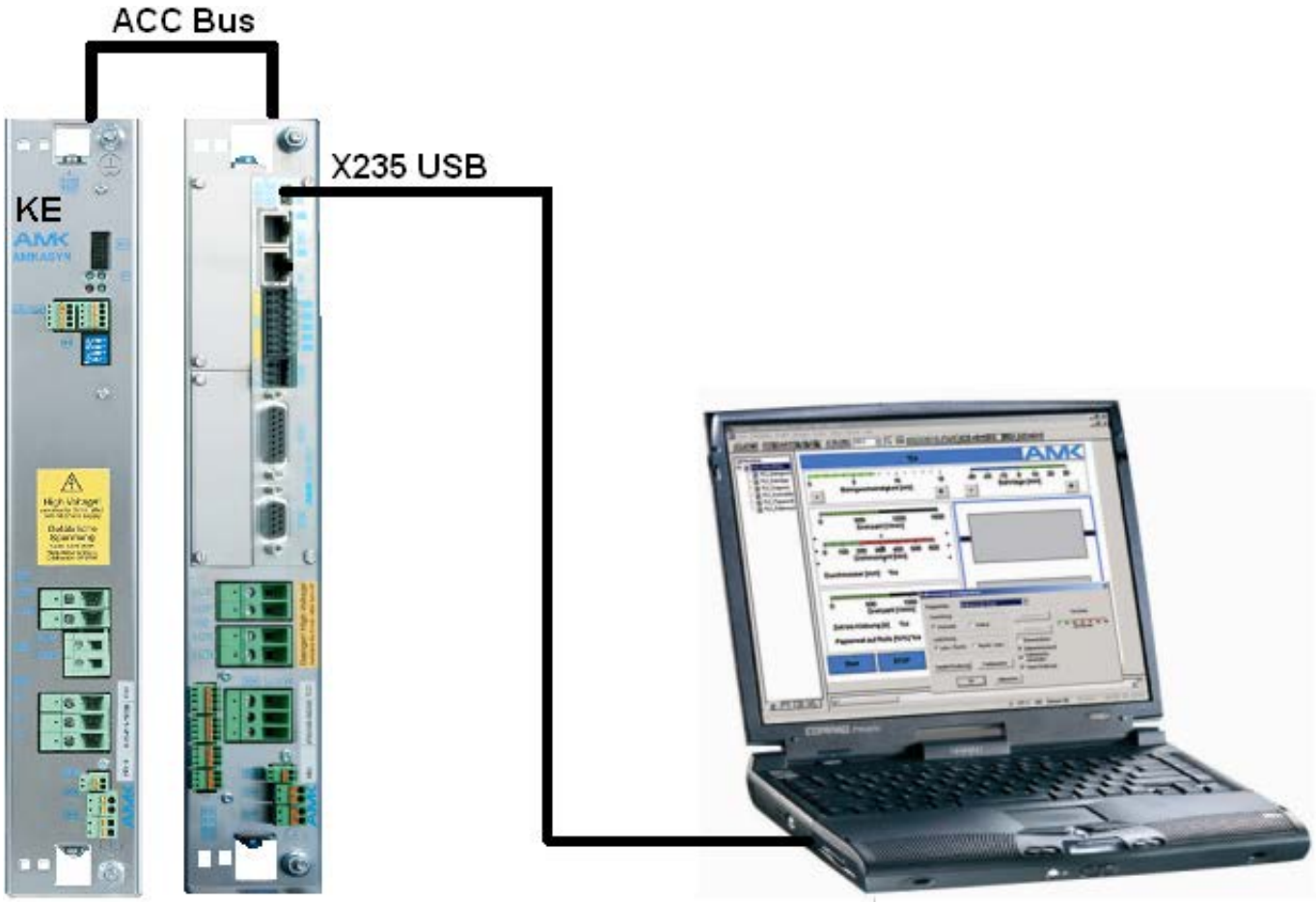
Windows 7



Windows XP



3.4.7 USB interface



Example:
KE with ACC bus and KW with KW-R06 (comparable with KW-R07, KW-R16 and KW-R17).
You get also access to ACC bus slave devices via the ACC bus connection of the controller card.

Cable:
AMK USB cable with ferrite core 3 m (AMK part-no. 47058).

3.4.7.1 USB adjustments

If you use the USB port, it is not necessary to adjust the communication settings.

3.4.8 AIPEX PRO Add In Gateway for TwinCAT V2

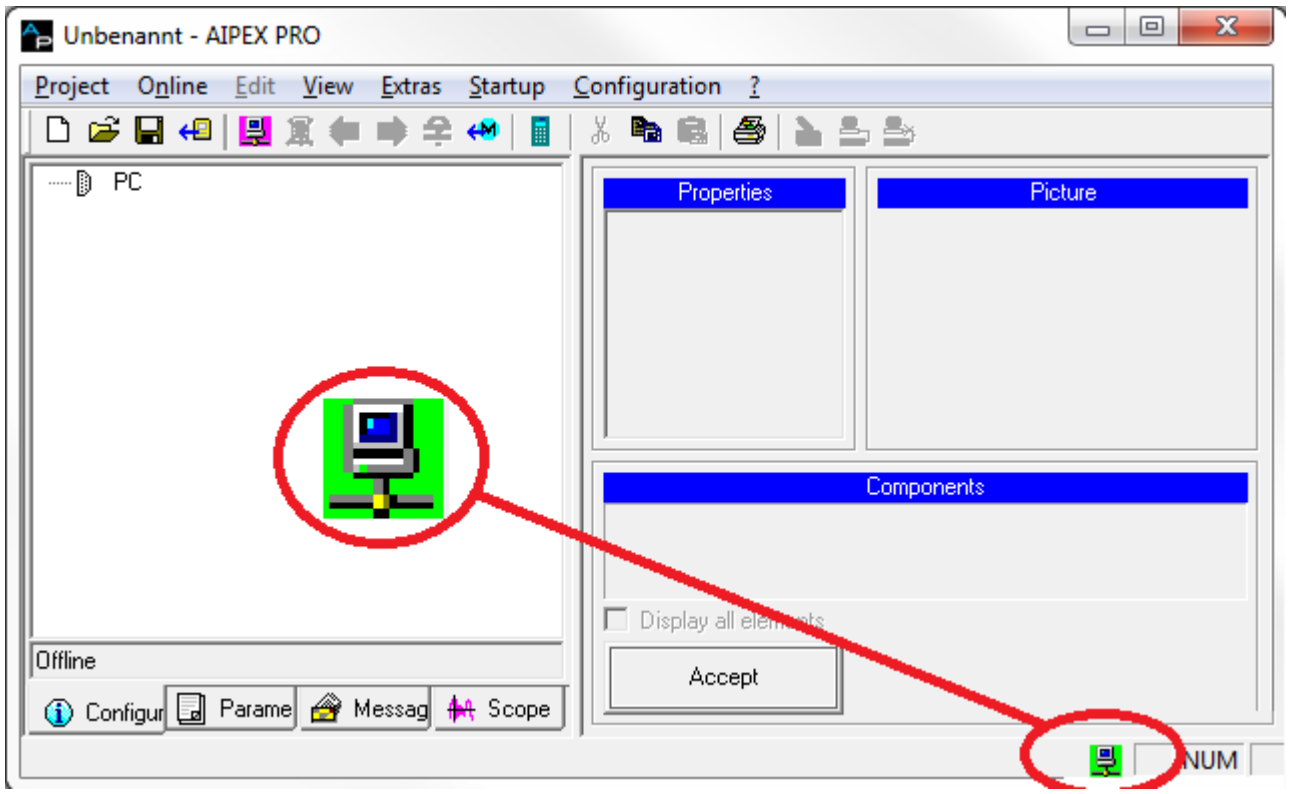
See documentation PDK_204072_AIPEXPRO_Add_In_TwinCAT_Gateway

3.4.9 Testing the communication

1. Connect the 24 VDC power supply to the device.
2. Start AIPEX PRO
3. After the initialisation is terminated, the green PC symbol in the status bar shows the activated connection between the AIPEX PRO PC and the AMK device.

Prerequisites:

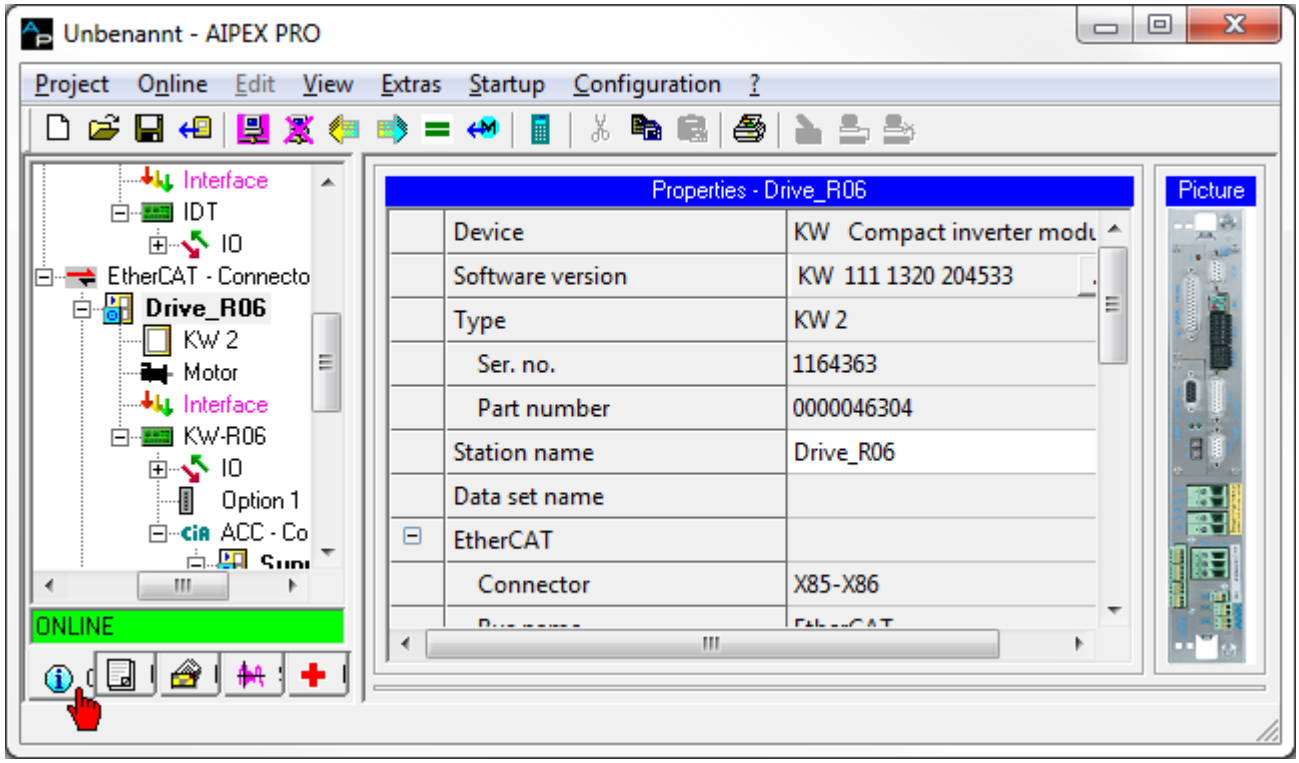
- communication is defined in AIPEX PRO
- connecting cable between PC and device
- AIPEX PRO window maximised (full screen)



green symbol	activated connection between the AIPEX PRO PC and the AMK device
no or red symbol	• no connection established • no interface defined • AIPEX PRO window not at full screen
two-coloured symbol (green / red)	multiple communication interfaces activated in AIPEX PRO

3.5 Tabs

3.5.1 Configuration



In the 'Device tree', you can find devices that exist offline and online.

The view of the device tree can be set specifically for each tab.Siehe 'Program overview - Enable views' auf Seite 40. und siehe 'Program overview - Display filter' auf Seite 39

On the right side you can find specific device properties. Use Properties to generate an offline project. Following you can transfer this offline project to an application.

3.5.1.1 Configuration - Manual

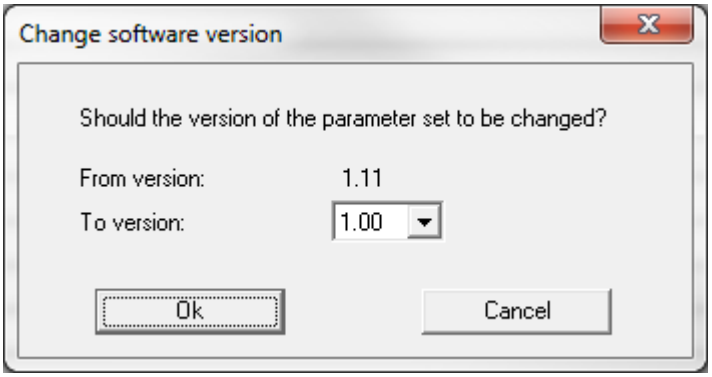
3.5.1.1.1 Properties - device

Properties devices

AIPEX PRO	Meaning	Example
Device	AMK device description	KW Compact inverter module
Software version	Installed software version	KW 111 1320 204533
Type	AMK type and nameplate ratings	KW 2
Ser. No.	Serial number	123456
Part number	Part number	46304
Station name	The station name can be changed by the user	Antrieb_R06
Data set name	Data set name of the actual opened AIPEX PRO project.	Demo

Button **'Software version'**

If you generate an offline project you can adjust the software version. Example: A project is generated offline and following the project will be transferred to an existing application with defined software version.



Fieldbus properties

AIPEX PRO	Meaning	Example
Connection	Interface on the device	X85-X86
Bus name	AMK bus description	EtherCAT
Bus physic	General bus description	ETHERCAT
Instance	Each bus will be parameterised in a own instance.	1
IP Address *1	Controller network adjustments	192.168.0.1
Network mask *1		255.255.255.0
Gateway Address *1		0.0.0.0
Address *12	Device address	1
Fix Addressing *1	Automatic assigned address by bus master or fix address	

AIPEX PRO	Meaning	Example
Optional *1	During the initialization of the bus system the device must not be present. Requirement: 'Fix Addressing' is active	
Master *1	Declaration bus master or bus salve	
Node Config file*1	Device description file (*.eds oder *.xml)	
Slave fix addressing *1	All slaves inside the bus system will be set to 'Fix Addressing'	

*1 Bus or device specific

***2 Differences offline / online project at EtherCAT**

Offline project

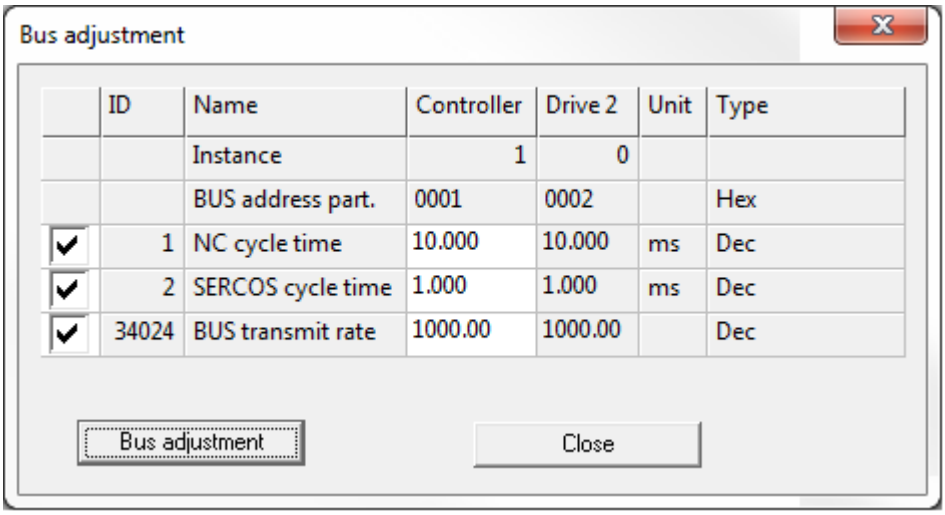
1. **Automatic assigned address by bus master**
   If you generate an offline project, you have to enter the future bus position as address.
   Requirement: ID34023 'BUS address participant' = 0.
2. **Fix address**
   Enter the address, following mark the check box 'Fix address'.

Online project

AIPEX PRO reads the actual (fix) address from the device. The address will be displayed.
After changing the address or check box 'Fix address' the device must be restart.

If you call the menu **'Configuration'** -> **'Configuration create'** the (fix) address will be overtaken to the XML configuration file.

Bus **'Bus adjustment...'**
The selected parameters will be set to the same value.

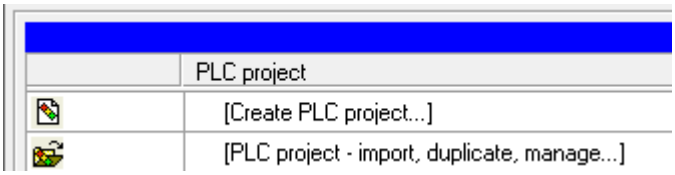


3.5.1.1.2 Interface

The 'Interface' is the interface between physical device and symbolic device name (variable) from PLC editor.
The allocation can be automatically (Siehe 'Automatic bus configuration' auf Seite 711.) or manual per drag and drop.

3.5.1.1.3 PLC

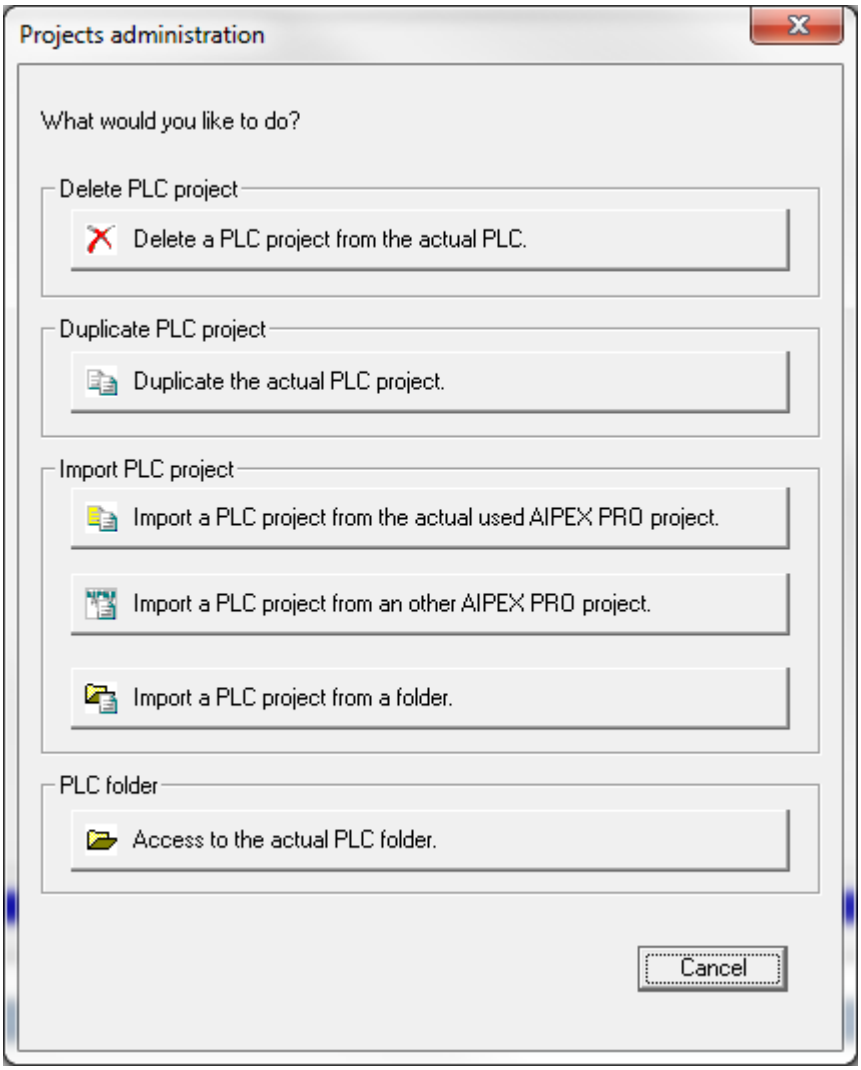
Click on 'PLC' to admin the plc project.



Create PLC project

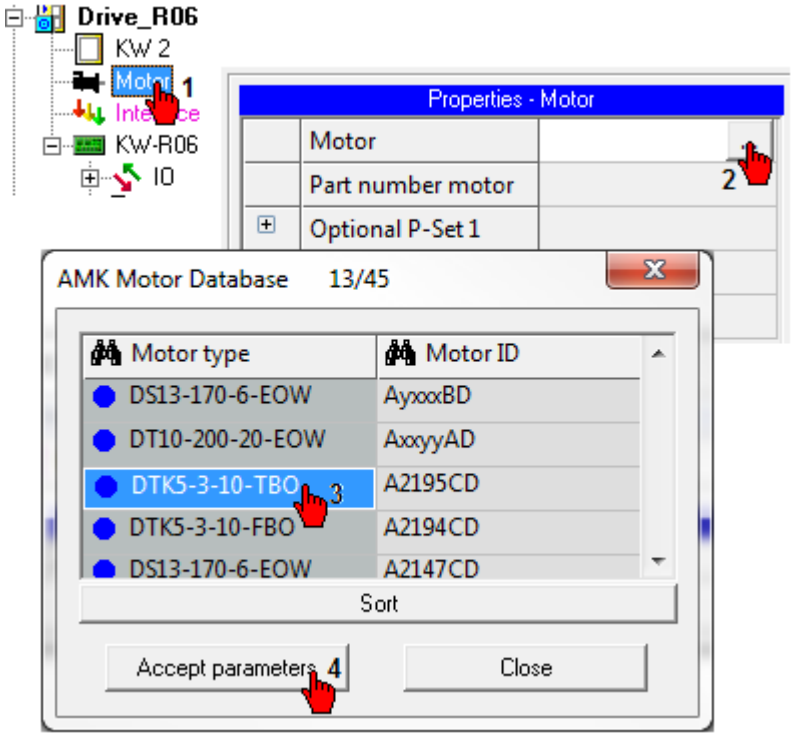
The plc editor will be started and a new plc program will be opened.

PLC project - import, duplicate, manage ...

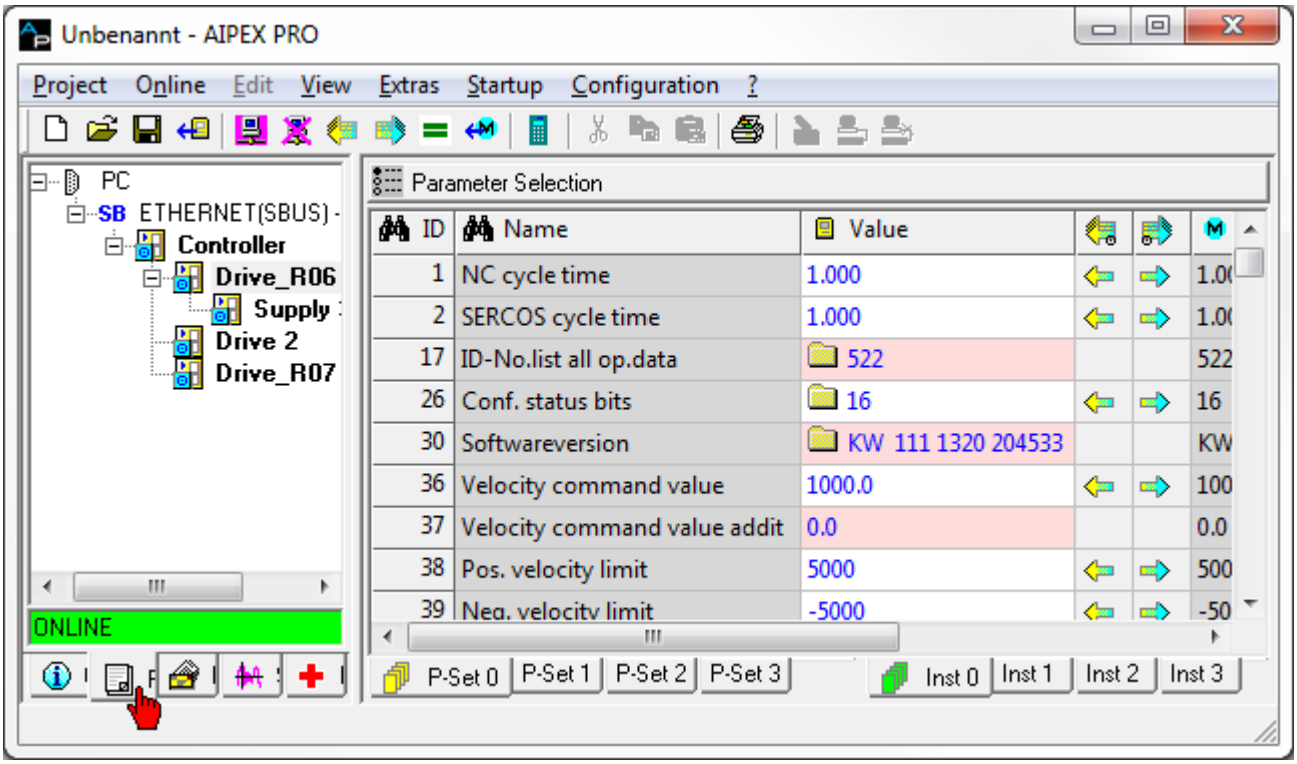


3.5.1.1.4 Motor

Manual selection of motor parameters.



3.5.2 Parameter



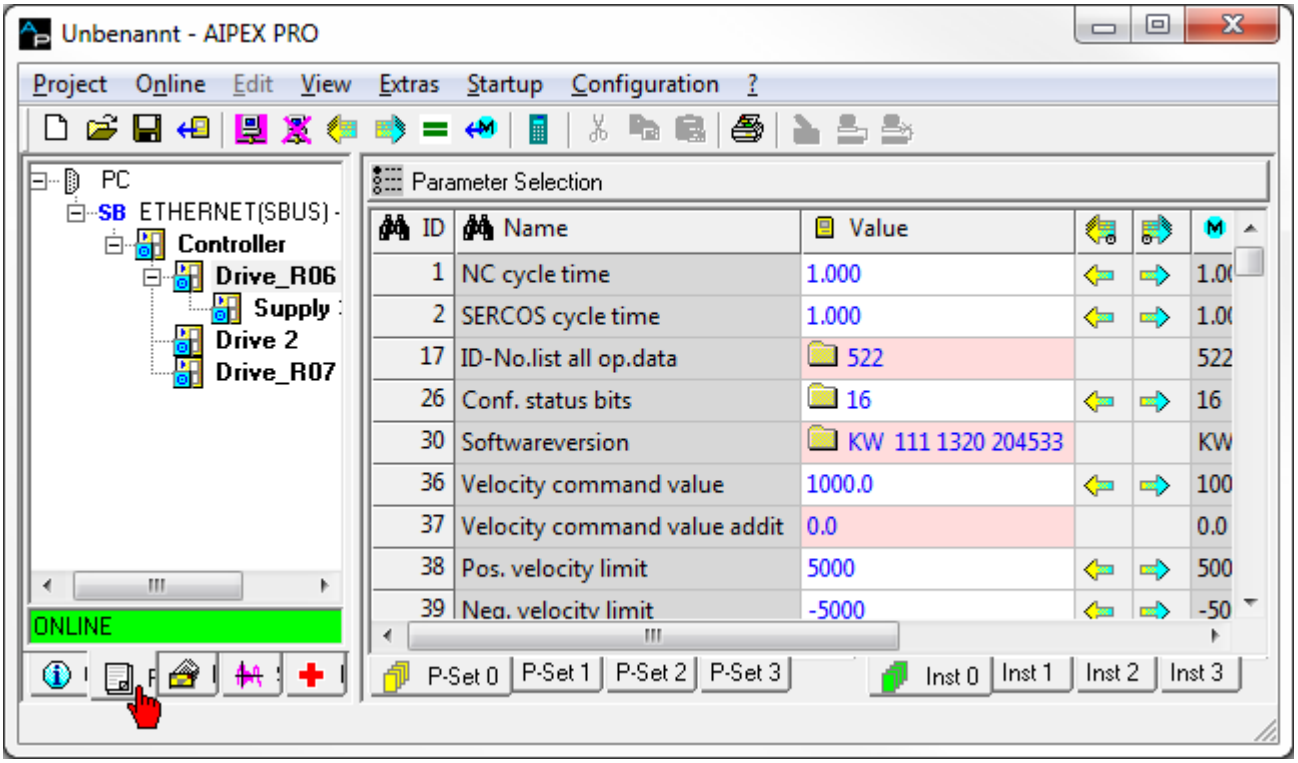
'Parameter' displays available offline and online values of the device selected in the device tree.

3.5.2.1 Parameter - Manual

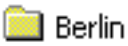
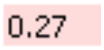
Changes in the 'Parameters' are not effective immediately in the drive. It belongs to the parameter group what you have to do to activate.

- Group Global and Instance: Logic Voltage (24 VDC) off/on
- Group Drive specific: Controller enable RF off/on

3.5.2.1.1 Overview

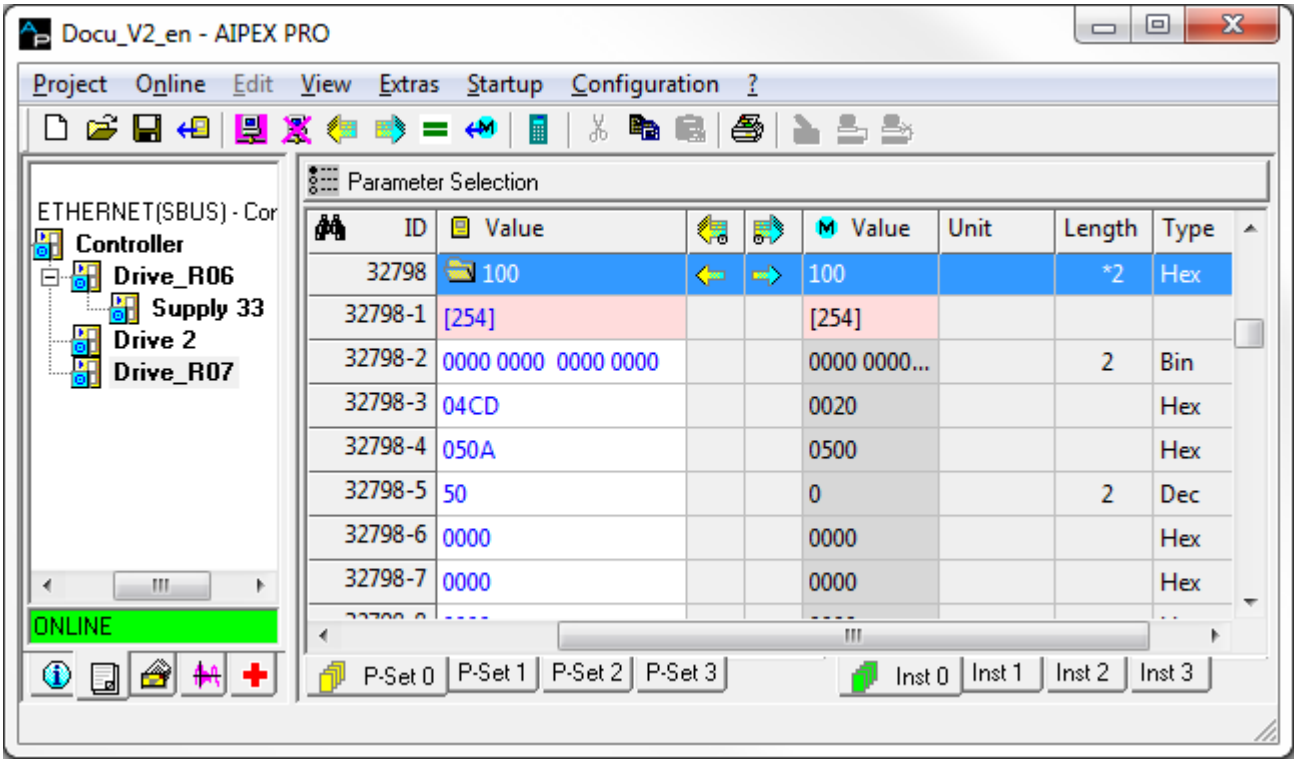


Icon	Meaning
Parameter Selection	Window: Parameter selection and System internal parameters
ID	Search: ID number
Name	Search: Parameter name
Value	Online value (value which is stored at the AMK device)
Value	Offline value (value which is stored at the PC)
	Upload (from device to PC) / Download (from PC to device) selected parameter
	Upload (from device to PC) / Download (from PC to device) complete parameter set
P-Set 0	Parameter set
Inst 0	Instance (used for bus parameters)
	Selection list input possibilities

Icon	Meaning
	List Parameter
	Pink background: Read only parameter

Field	Explanation
ID	Parameter number
Name	Designation of the parameter
Value	Current value of the parameter Folder icon = Parameter list
Unit	Unit of the parameter
Length	Data length of the parameter in byte • 1 byte • 2 byte • 4 byte
Type	Display of the parameter • Dec: decimal • ±Dec: decimal signed • Bin: binary • Hex: hexadecimal • Ascii: Ascii string
Remark	Text field, freely usable by the user
Background colour in the column "Value"	White: Value can be modified Pink: Value cannot be modified (write-protected or formal)
Text colour in the column "Value"	Blue: Value was modified, but not yet saved Black: Value is unchanged since the last saving Red: This is a system parameter (can be modified only if enabled)
Icon in column "Length"	Yellow: Parameter is specific for the parameter set Green: Parameter is specific for the instance

3.5.2.1.2 Display and structure of lists



Lists

- Lists are indicated by the folder icon in 'Value' column.
- For non-ASCII lists, the number of list elements is indicated next to the icon. For ASCII lists, the text content is displayed here.
- All lists can be displayed in expanded form by a double-click in the 'Value' column. The number of maximum possible list elements is shown in the first line underneath. All further lines display the list content by elements.
- The display of the list elements can be structured by entering own values in the columns lengths (1, 2, 4) and type (Dec, Hex, Bin, Ascii).

Entering values

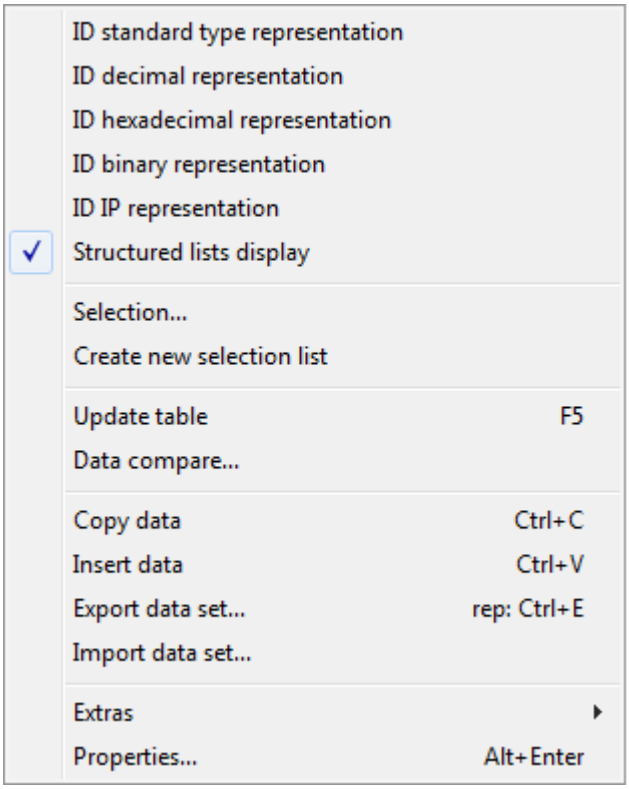
- The value of modifiable parameters can be edited directly in the table. Each input is completed by pressing the **'Enter'**.
- The input needs to be appropriate for the displayed data type.
- Some parameters have a minimum and a maximum limitation of the value. If these are breached during input, an error message is displayed.

Modifying lists

- For non-ASCII lists, entering a value next to the folder icon changes is current list length. The value may not be longer than the maximum list length.
- For ASCII lists, the input next to the folder icon is interpreted as a string and the list is modified appropriately.
- In expandable lists, the list elements can be modified directly. The input needs to be appropriate for the displayed data type of the element or in case none exists, fit the list.

3.5.2.1.3 Parameter context menu

Select the tab 'Parameter'. By selecting any parameter with the right mouse button, the context menu opens.

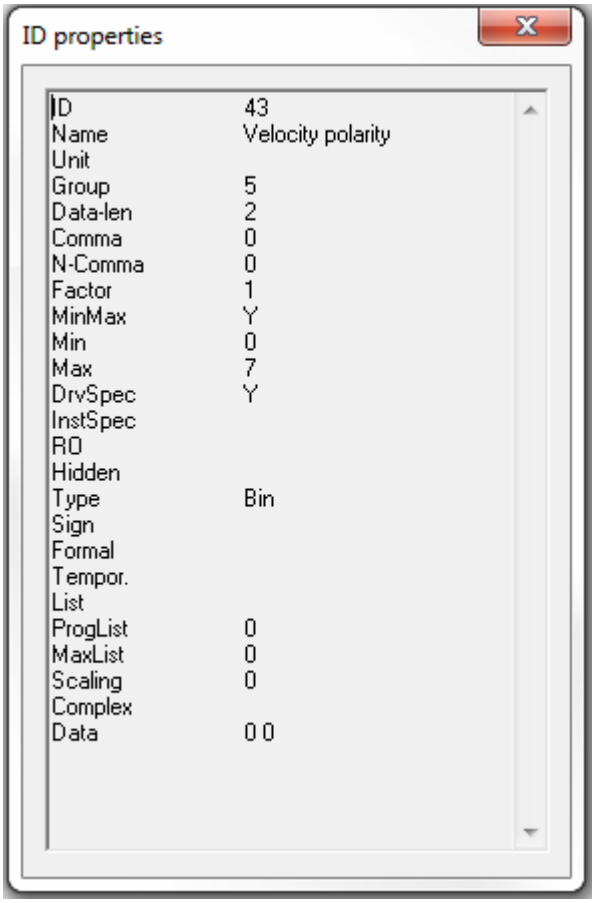


Field	Explanation
Displaying ID in the standard type	The selected parameter will be displayed as indicated in the ID Properties .
Display ID decimally	The selected parameter will be displayed decimally.*
Display ID hexadecimally	The selected parameter will be displayed hexadecimally.*
Display ID binary	The selected parameter will be displayed binary.*
Structured list displays	If 'Show lists with structure' is activated in the context menu, the user can specify 'Name', 'Value', 'Unit length' (e.g. user lists).
Selection	Opens the window 'Parameter selection'.
Save ID in selection list	The number of the selected parameter is saved in the currently specified 'own list'. It is of no importance if the 'own list' is active.
Update display	Updating the online values. All values are discarded and read in anew from the connected drive.
Data compare	The content of the currently displayed parameter set can be compared with an already saved one.
Download/Copy project parameters set	With this menu item, the selected parameter is transferred between data set and online device. If the selection is in the column of the device data set, the parameter is copied from the offline data set to the online device. If the selection is in the column of the online data set, the parameter is saved from the online device to the offline data set. The name of the menu item changes, depending on the selected column.
Copy data	The contents of currently marked parts of the parameter table are copied to the Windows clipboard and thus are available as insertable text for many applications, such as text programs. If complete lines are marked in the table, an internal copy of this parameter is made as well.
Insert data	If there is an internal copy of parameters, then its content is inserted in the corresponding parameter. If no internal copy exists, then the content of the Windows clipboard is inserted unchecked as text as of the currently selected point in the table.
Export data set	Siehe 'Project' auf Seite 94.

Field	Explanation
Import data set	Siehe 'Project' auf Seite 94.
Properties	Display of the properties and attributes of the selected parameters.

* This excludes ASCII lists

3.5.2.1.4 ID Properties

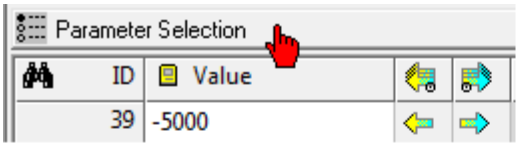


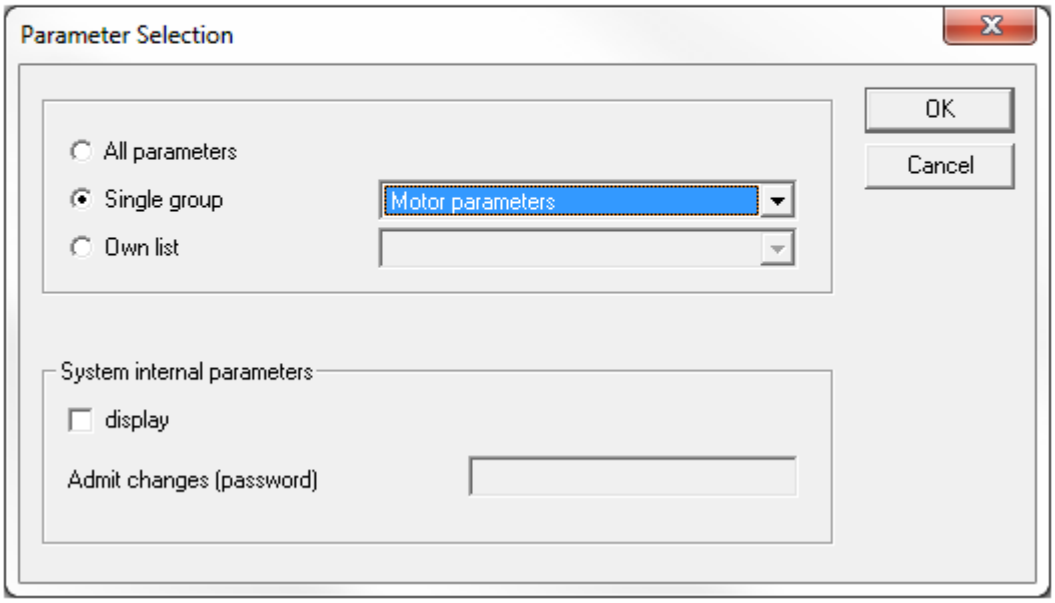
ID Properties	
ID	Parameter number
Name	Parameter name
Unit	Unit
Group	Parameter group
Data-len	Data length of the parameter in byte
Comma	Power of 10 scaling factor X (for 10 to the power of X)
N-Comma	Decimal places for parameter value display
Factor	Scaling factor in 10 to the power of X
MinMax	Minimum value / maximum value available
Min	Minimum input value
Max	Maximum input value
DrvSpec	Drive-specific parameter
InstSpec	Instance-specific parameter
	[Not DrvSpec and not InstSpec = Global parameter]
RO	Read Only
Hidden	System-internal parameter
Bin	Input format binary
Hex	Input format hexadecimal

ID Properties	
Ascii	Input format ASCII
Sign	Input format signed
Formal	Process date, entered value are not saved remanent.
Tempor.	Temporarily adjustable
List	Parameter is of the type list
ProgList	Current list length
MaxList	Maximum list length
Scaling	Quantification type
Complex	Type complex list
Data	Parameter value

3.5.2.2 Parameter - Selection

Open the dialog box by pressing the button **'Parameter selection'**.



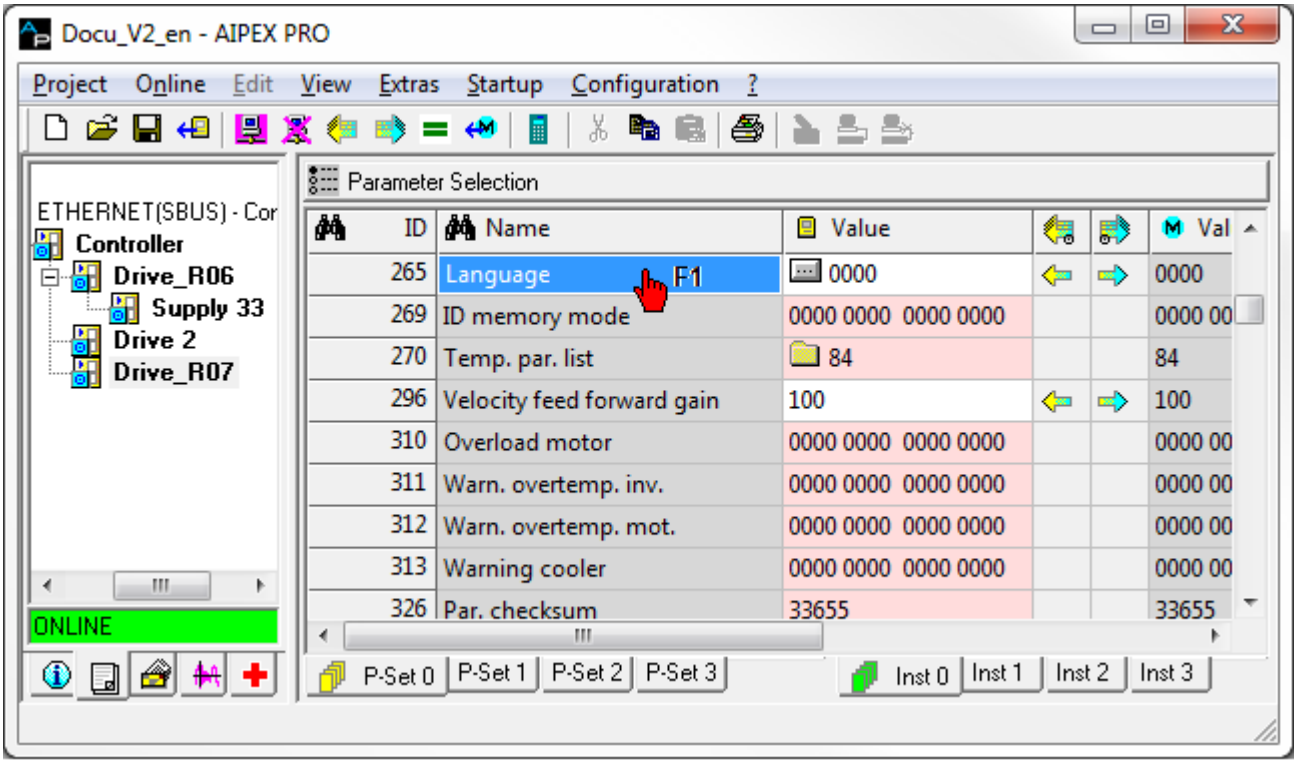


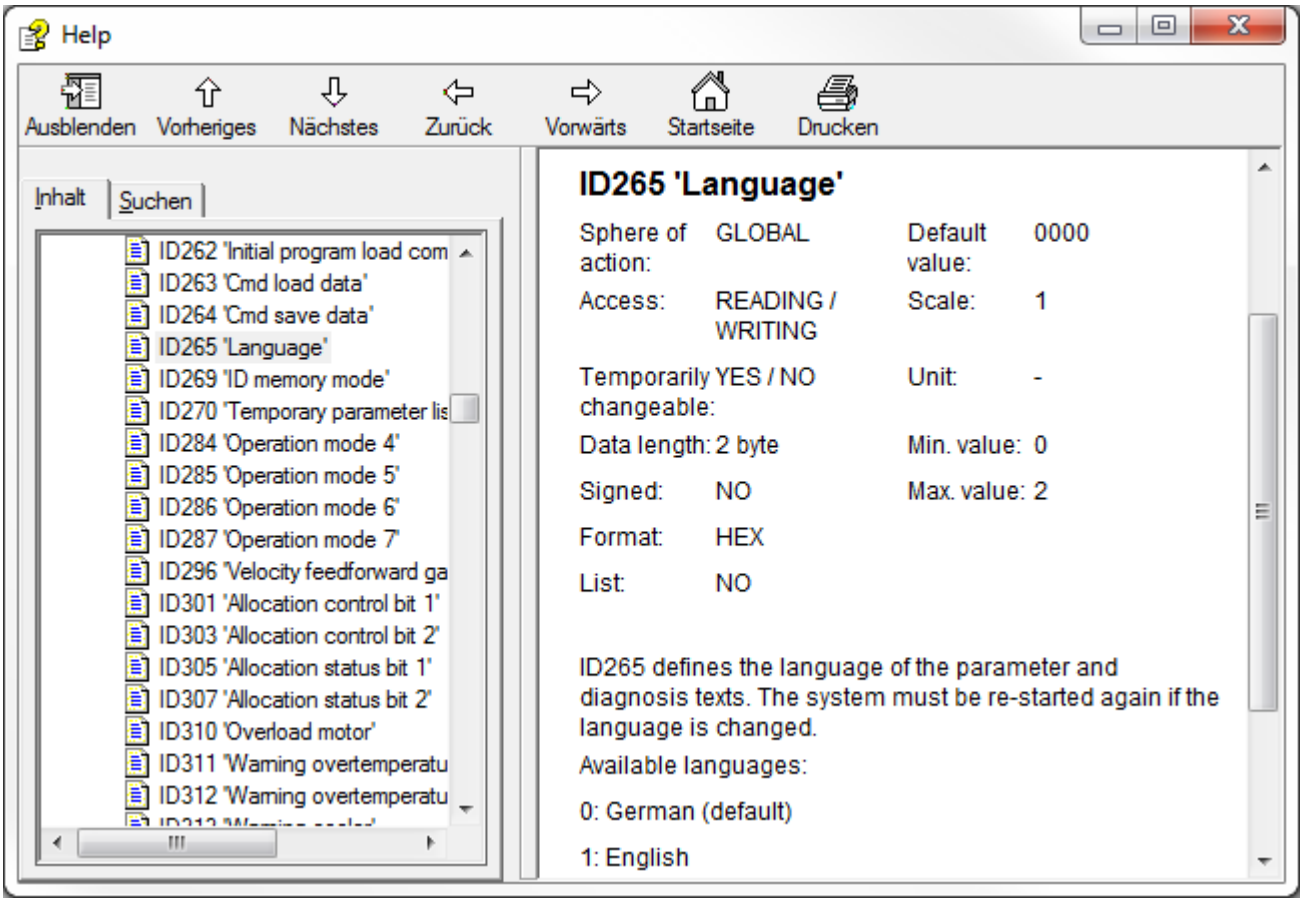
Selection	Description
All parameters	All parameters are displayed. (System-internal parameters need to be selected additionally)
Individual groups	All parameters of the selected parameter group are displayed.
Own list	All parameters of the selected own list are displayed. The own lists can be expanded as needed. Each list consists of a sequence of ID numbers or number ranges separated by commas and can be designated by a freely selectable name. Example: My list 1, 2, 5, 90, 32000-32100 Each list is deleted again by complete removal of the content.
System-internal Parameters	System-internal parameters are displayed in the parameter list (colour red)
Permit changes	After the release, system-internal parameters can be modified.

3.5.2.3 Parameter - Online help

Calling up the parameters online help

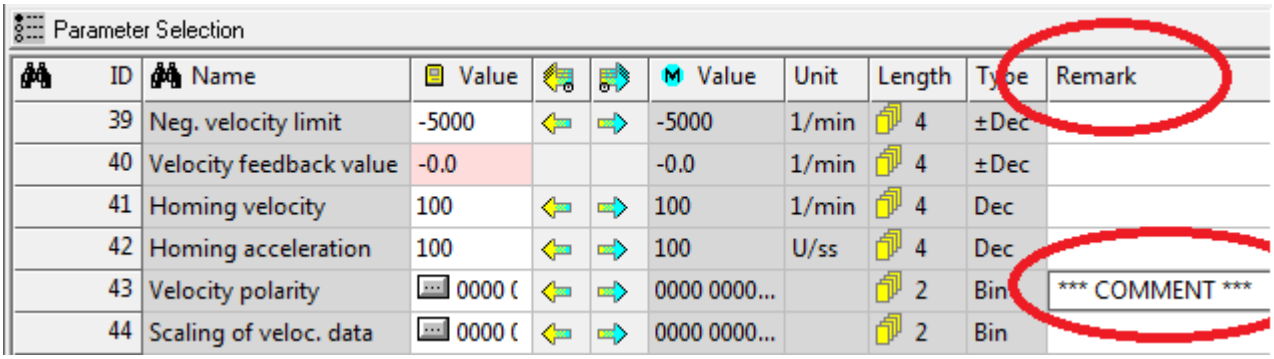
Mark a parameter. Press the **F1** button.





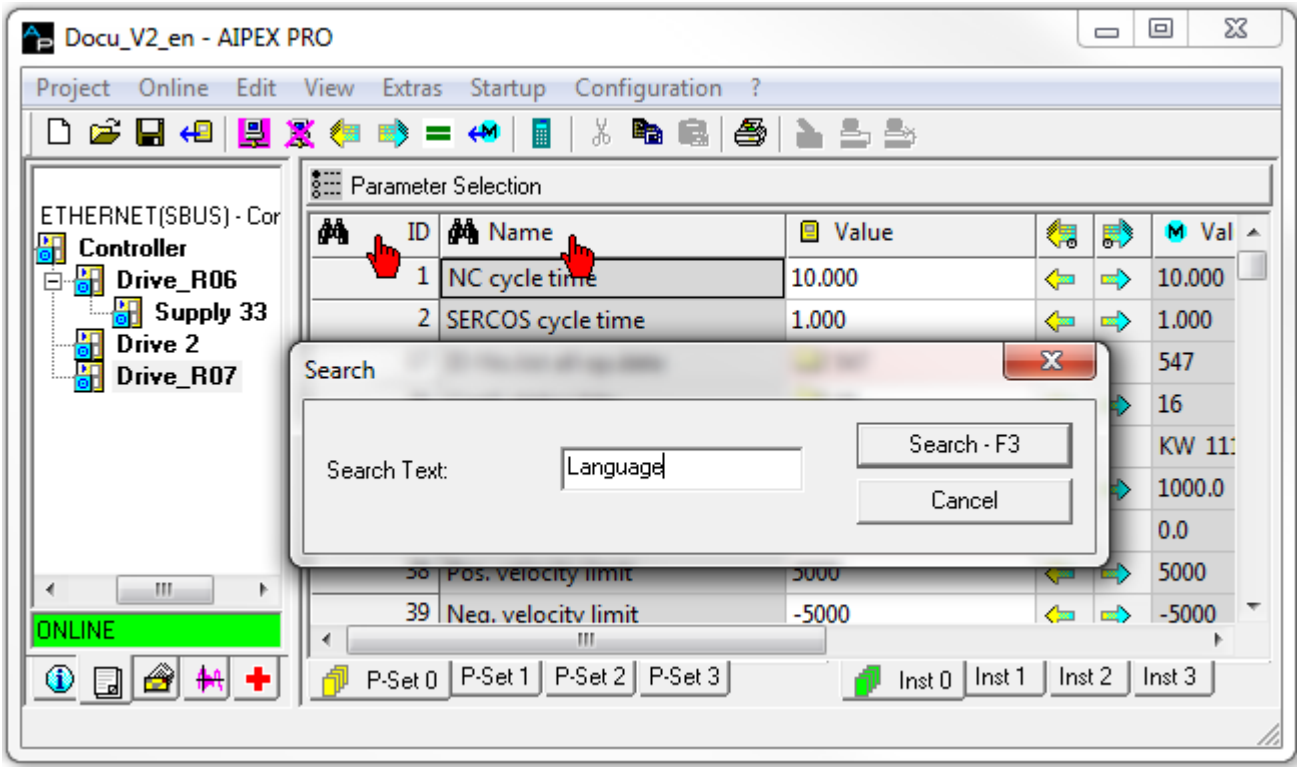
3.5.2.4 Parameter - Comments

In the last column of the table, any comments can be entered.
However, this is only possible if an offline data set is available; i.e. not for pure online work.
Comments are not just possible for each ID but also for each list element in an expanded list.

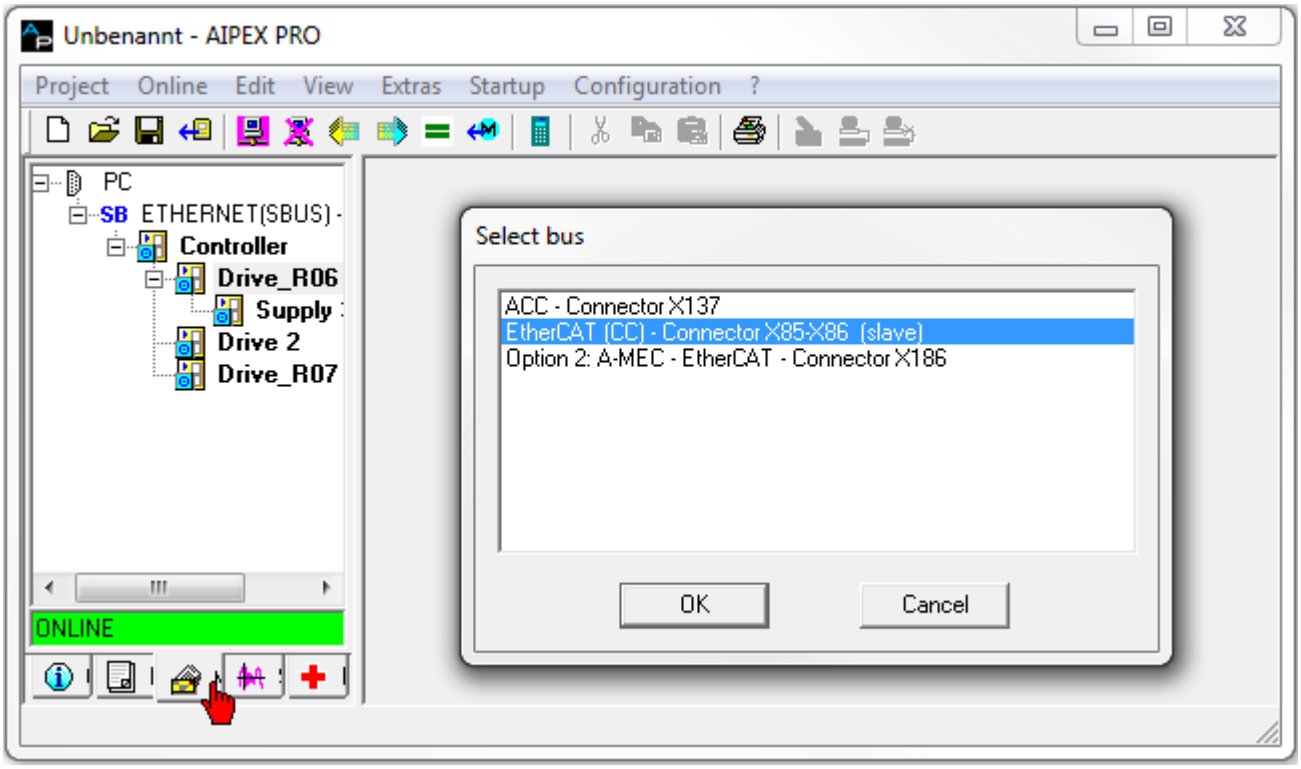


3.5.2.5 Parameter - Search function

Use the search function (binoculars symbol) to search in the columns for ID numbers or texts.



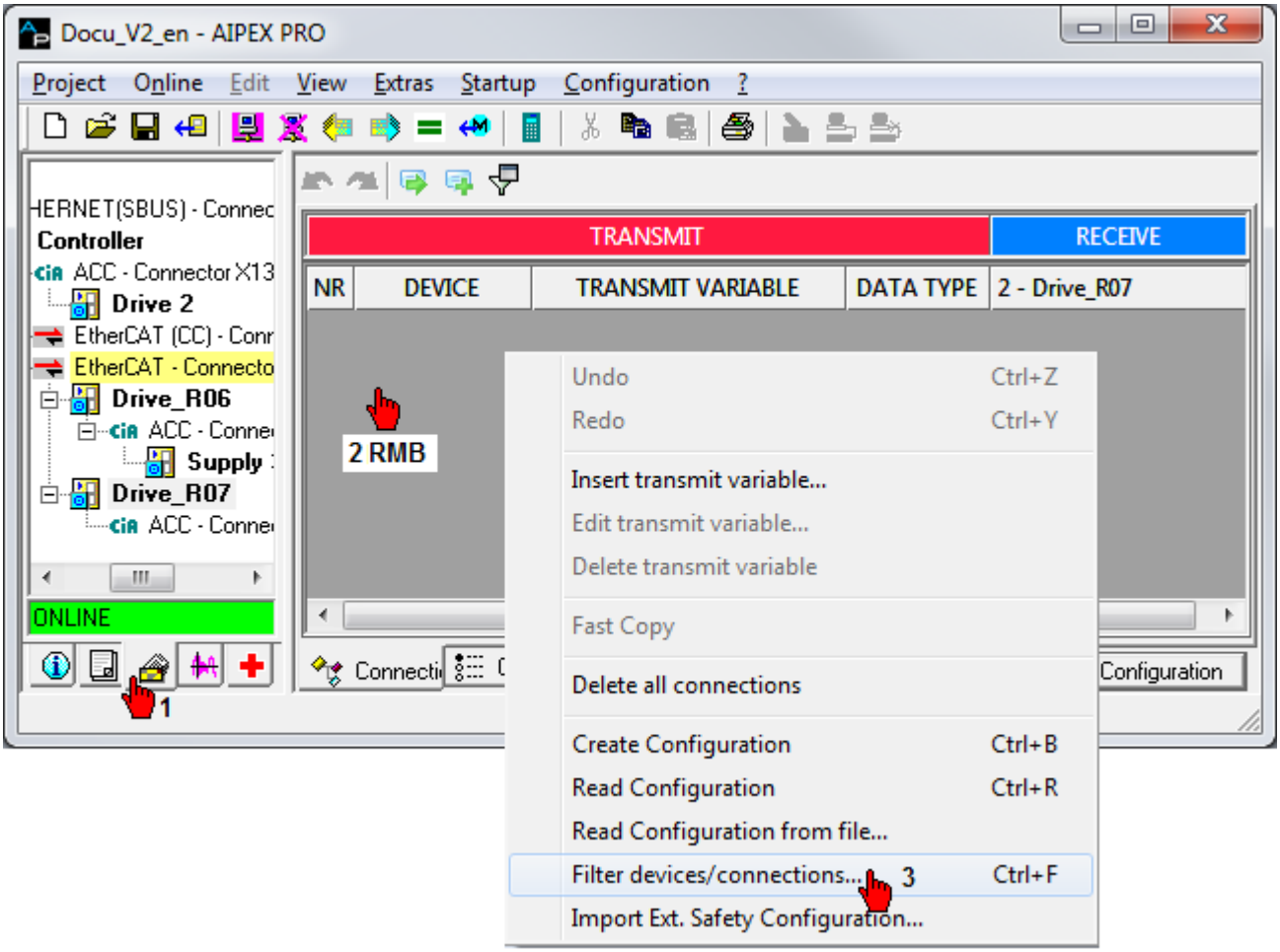
3.5.3 Messages

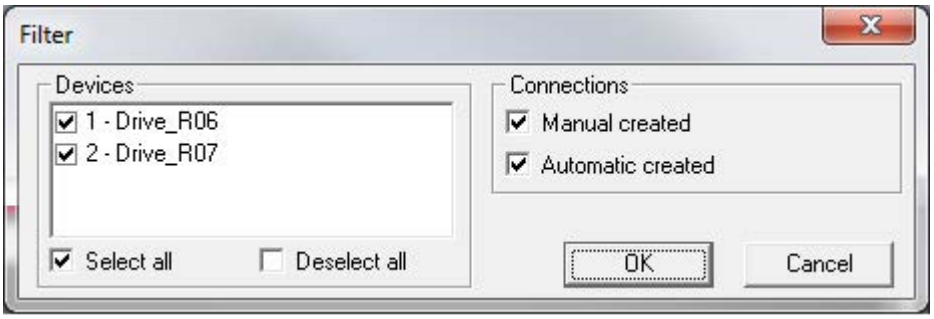


By the coupling of component configuration and programming environment, AIPEX PRO is capable of executing the automatic fieldbus configuration. Data is provided automatically thereby synchronously or asynchronously in the PLC program depending on their purpose. Drive data as well as I/O data is configured automatically.

3.5.3.1 Messages - Filter

Improve the design the message configurator's structure by adapting the display filter to your application. For this, click in the message configurator with the right mouse button.





Devices
Deactivate and activate various modules.

Select all
All modules are displayed in the message configurator.

Deselect all
All modules are deselected

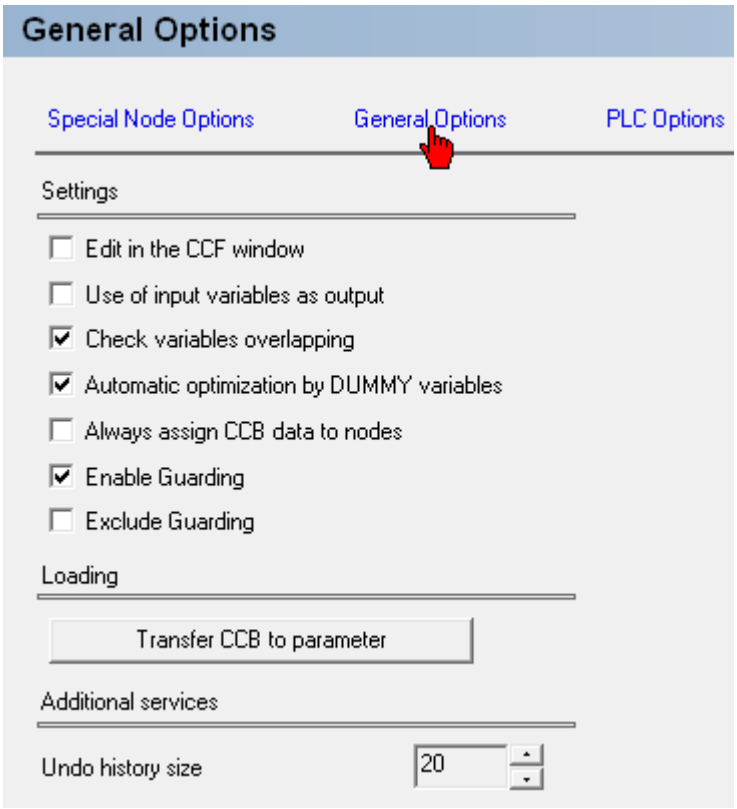
Connections

Manually generated
Only connections that are generated manually by the user are displayed in the message configurator.

Automatically generated
Only connections that are generated automatically by AIPEX PRO are displayed in the message configurator.
.

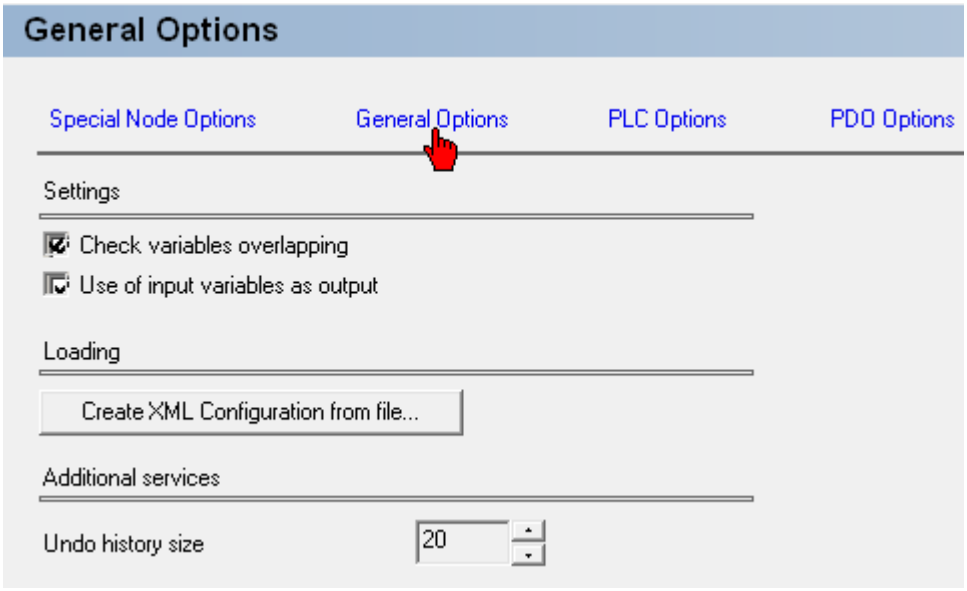
3.5.3.2 Messages - General options

3.5.3.2.1 ACC-Bus



- **Edit in the CCF window** enables a direct editing of the CCF file in the tab **CCF Output**.
- **Use of input variables as output** input variables that were described in the EDS file can be read back.
- **Check variables overlapping** in the tab **Connections** overlapping variables are not filtered.
- Automatic optimization by DUMMY variables
- **Always assign CCB data to nodes.** Activate this option if you want to implement a CANclient structure without master.
- **Enable guarding** (master) and life guarding (slave) monitoring. The presence of master and slave on the Bus is monitored reciprocative. If a node is missing (master or slave), an error message is generated. The necessary values "Guard Time" and "Life Time Factor" are automatically generated by AIPEX.
- **Exclude Guarding** The monitor functions node guarding (master) and life guarding (slave) are deactivated completely. (Useable for external terminals!)
- **Transfer CCB to Parameter** Existing *.eds files can be loaded directly in the project.

3.5.3.2.2 EtherCAT

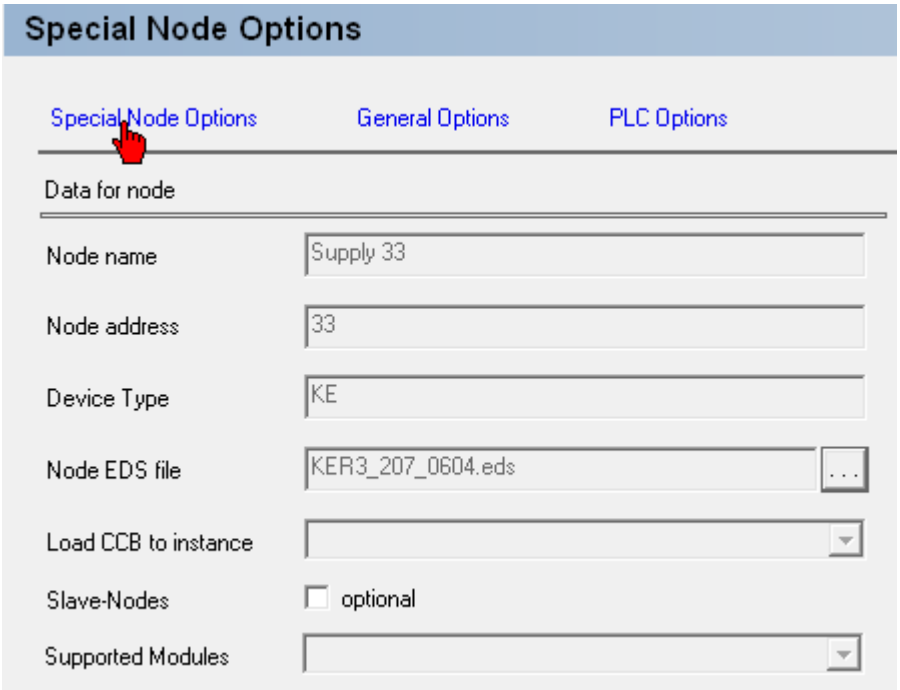


- **Check variables overlapping** in the tab **Connections** overlapping variables are not filtered.
- **Use of input variables as output** input variables that were described in the EDS file can be read back.

Create XML configuration from a file Existing *.XML files can be loaded directly in the project.

3.5.3.3 Messages - Special node options

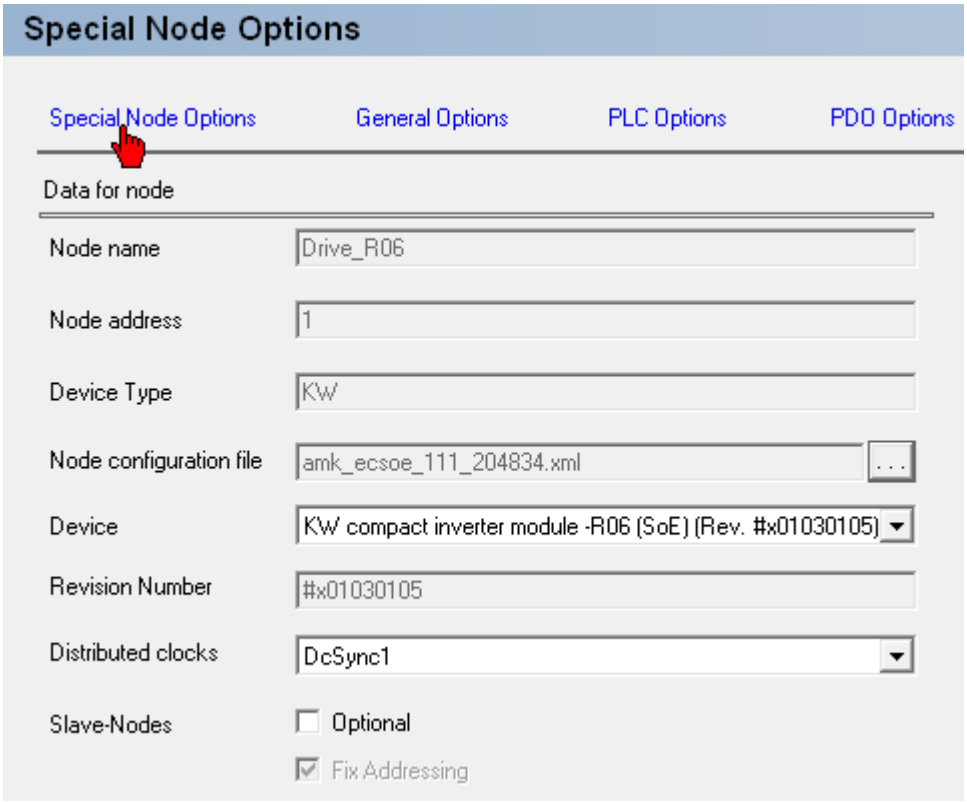
3.5.3.3.1 ACC-Bus



- **Node name** Name of the selected device (display only).
- **Node address** Bus address of the selected device (display only).
- **Device Type** Type of the selected device (only display).
- **Node EDS file** Name of the configuration file of the selected device. This entry can be changed by the selection of a different file.

- **Load CCB to instance** The "Loading at Instance" field specifies on which instance the created CCB file is loaded (in ID34036). This field is only active for the CAN main device since it loads the configuration for the entire network. The preset value for KW is 0 and for AMKAMAC AS-PL controllers it is 1.
- **Slave nodes - optional** Slaves that are marked as optional do not need to be present during the initialisation of the system. No error message is signalled.

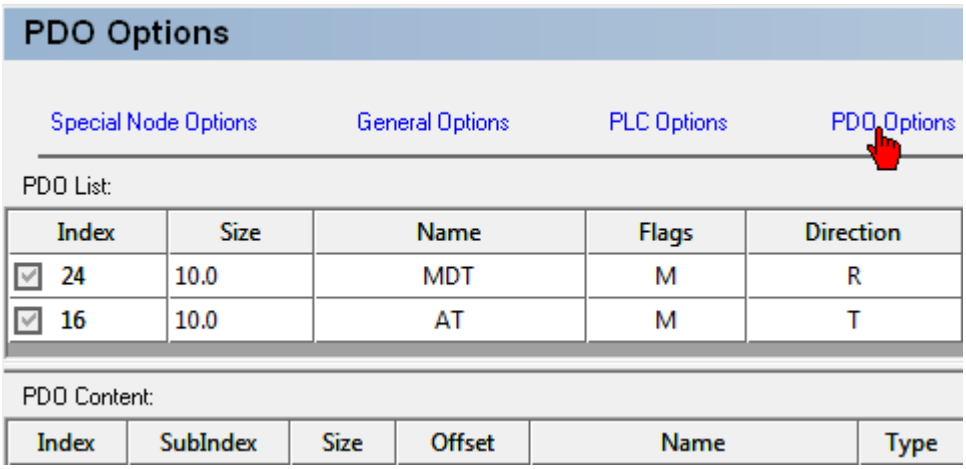
3.5.3.3.2 EtherCAT



- **Node name** Name of the selected device (display only).
- **Node address** Bus address of the selected device (display only).
- **Device Type** Type of the selected device (only display).
- **Node configuration file** Name of the configuration file of the selected device. This entry can be changed by the selection of a different file.
- **Device** The device needs to match the configuration file. If a new configuration file is selected, the corresponding device needs to be selected.
- **Slave nodes - optional** Slaves that are marked as optional do not need to be present during the initialisation of the system. No error message is signalled. Prerequisite is that the option "Fixed address" is set for this node.
- **Slave node - Fix address** Such a marked node basically receives the specified fixed address by the master. This option is always active for AMK devices.

3.5.3.4 Messages - PDO Options

EtherCAT



All PDOs contained in the description file of the device are listed.

- "Index" – unique numerical identification
- "Size" – Size of the PDO in bytes.bits
- "Name" – Designation of the PDO (free text)
- "Flags" – F:fixed, V:virtual, M:mandatory
- "Direction" – T:transmit, R:receive

The used PDOs can be selected/deselected with the following limitations:

- Mandatory PDOs cannot be deselected.
- Excluded PDOs can only be selected if the counterparts are deselected.

The PDO content is displayed in the lower table for the PDO selected in the PDO list.

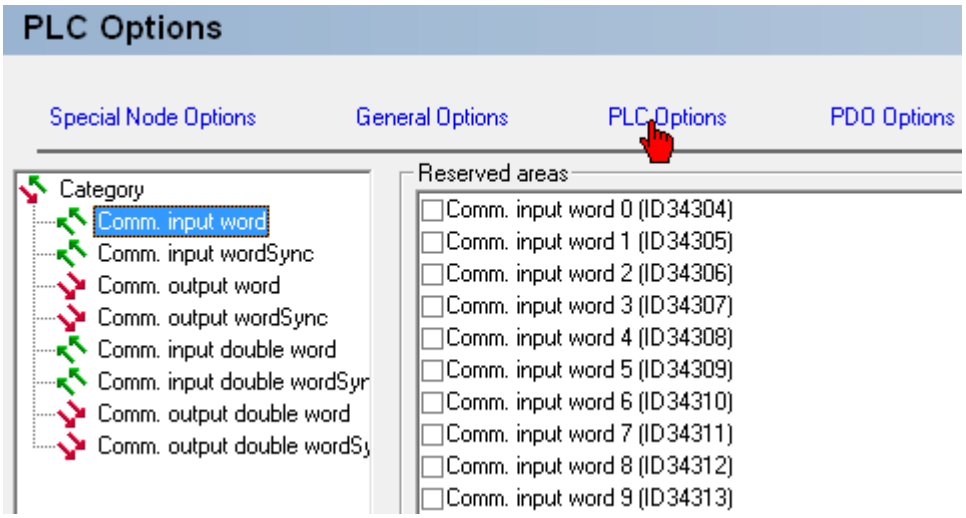
- "Index"/"SubIndex" – Unique numerical identification of the variables
- "Size" – Size of the variables in bytes.bits
- "Offset" – Position of the variables in the PDO
- "Name" – Designation of the variable (free text)
- "Type" – Data type of the variables

On non-fixed PDOs, the PDO content can be modified by the context menu. New elements can be added, for example.

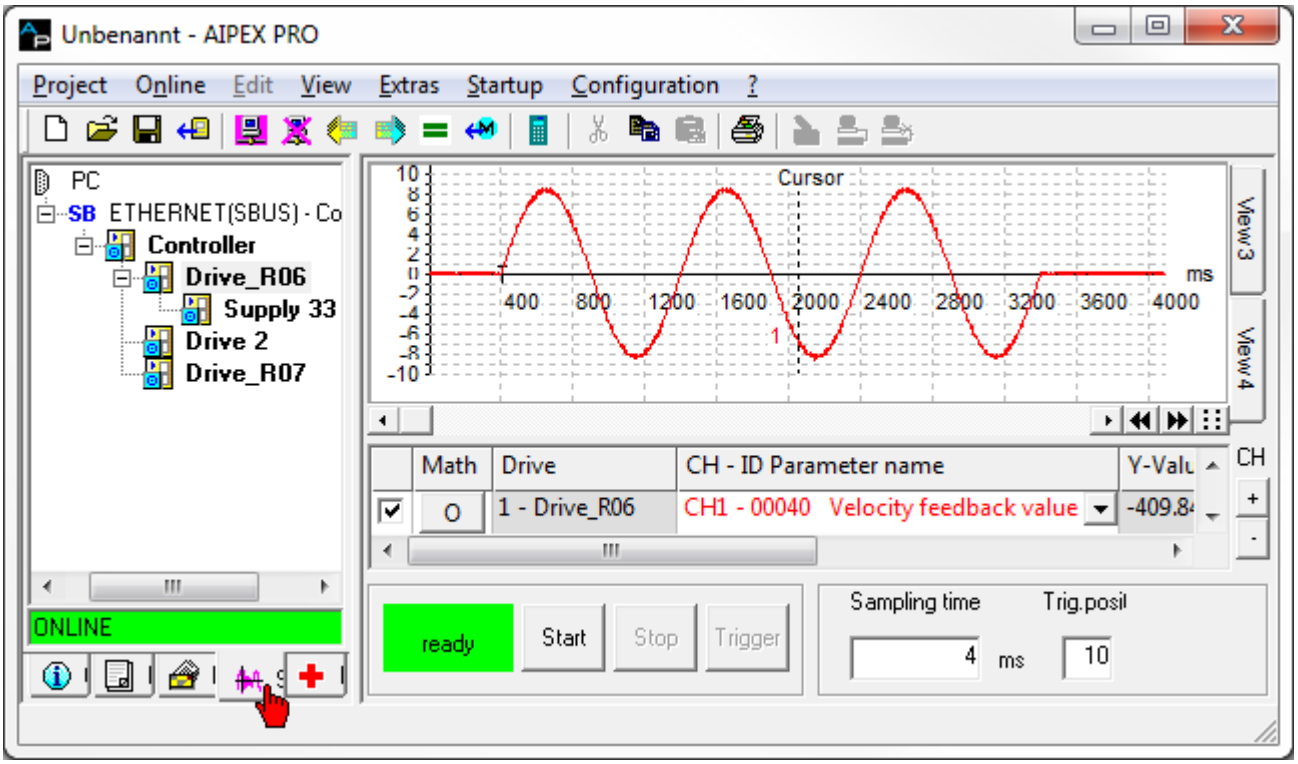
3.5.3.5 Messages - PLC Options

Selected (reserved) output variables will not be used by the automatically network configurator.
The adjustments are only valid for the device which is selected inside the device tree.

Example:
Profibus with an AMK PB option card uses a fix address area. It is not allowed for the automatic network configurator to use the same area. You have to reserve the variables.



3.5.4 Scope



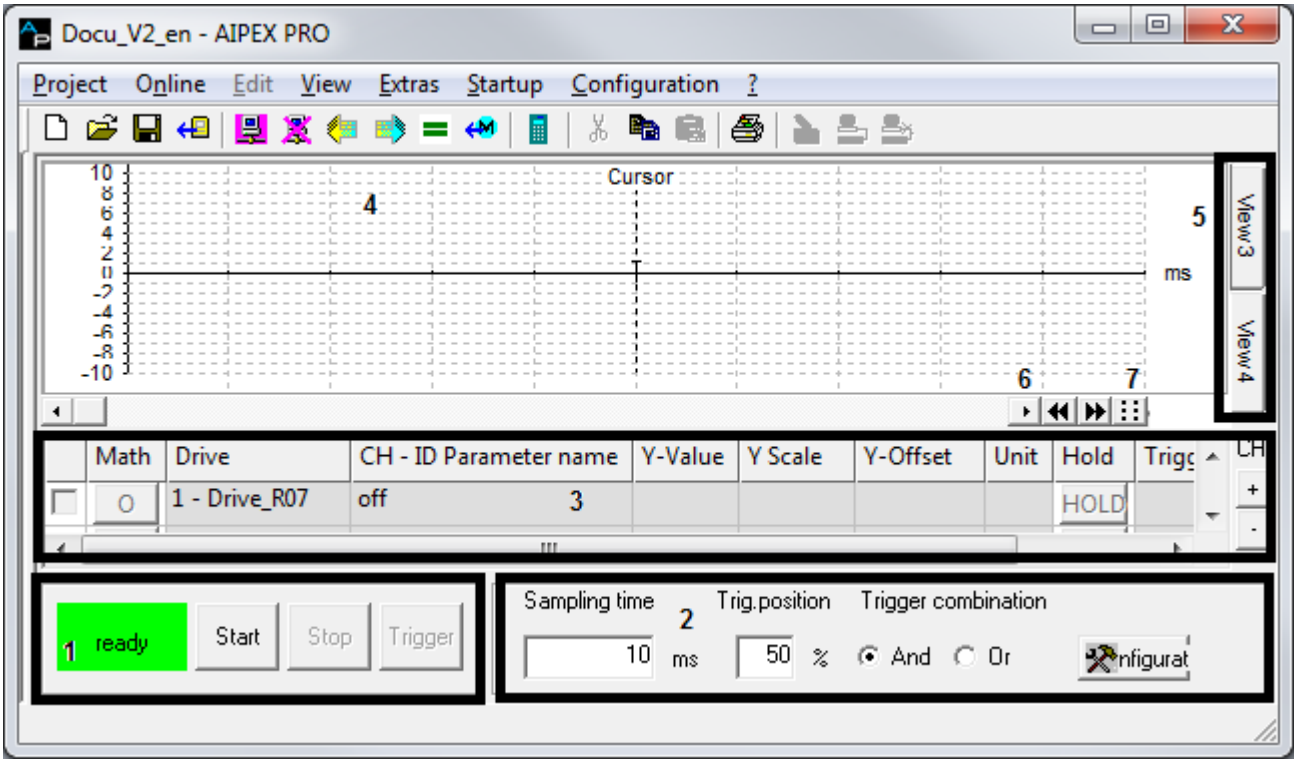
Each AMK drive features an internal oscilloscope function. Using AIPEX PRO, you can configure signals that can be recorded in the drive in real time afterwards. As soon as the internal memory in the drive is full, the measuring data is transferred and graphically displayed by AIPEX PRO.

3.5.4.1 Scope - Manual

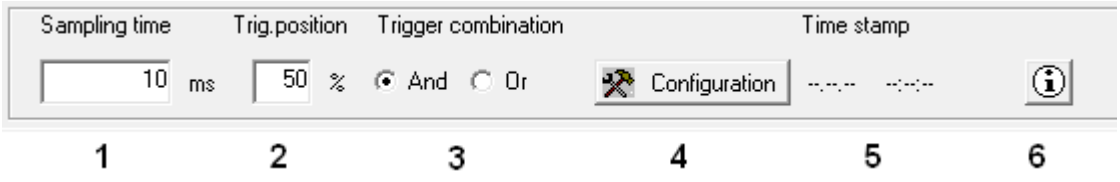
You can conduct drive-specific or device-spanning measurements (only ACC bus).

For drive-specific measurements, mark a drive in the device tree.

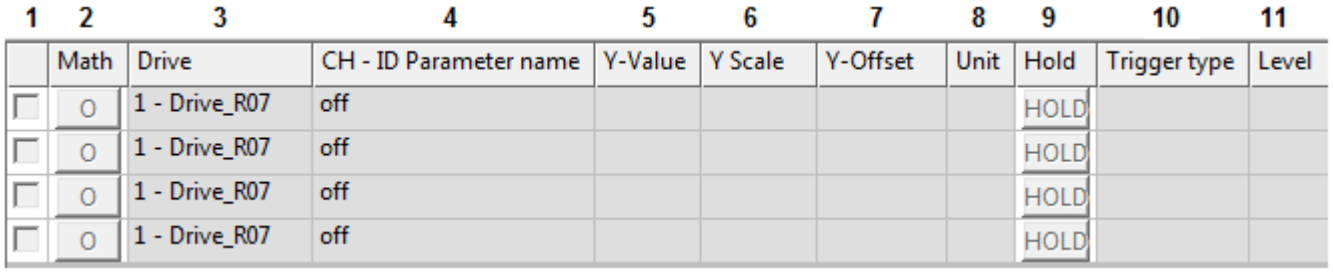
For device-spanning measurements, mark the fieldbus (ACC bus). You can then record and display the signals of several devices. The measurement is started on all drives by a common trigger signal.



Number	Function
1	Operating the oscilloscope (Start, Stop and Trigger button) Status display Offline (grey) Ready (green) Started (yellow) Triggered (yellow) Ready (green) Error (red)
2	Configuration field
3	Field for drive selection and signal parameters
4	Display
5	Each view has an own channel selection
6	Scaling the time axis
7	Grid in the display (on / off)



Number	Function
1	Input field for the scan period of the following measurements (device specific)
2	Trigger position. (The characteristic curve can be displayed before the trigger event ("pre trigger").
3	If multiple trigger signals are used, you can select between an "and" or an "or" link of the trigger signals.
4	Button for opening the configuration field
5	Display of the date and the time after a measurement
6	Button for opening and entering information about the measurement (a yellow button indicates that text information is available)



Number	Function
1	Control box for displaying the signal curve
2	Computation of signals (addition, subtraction...)
3	Drive name
4	Channel display: Signal (parameter number and name) of the selected channel
5	Display of actual value
6	Input box Y- Scale
7	Input box Y- Offset
8	Unit
9	A 'HOLD' curve is not overwritten by a subsequent measurement. 'LET' delete saved measurement
10	Configured trigger type
11	Configured level

Status indicators

Status display "grey" – no valid device is selected or work is being done offline.

Status "ready" – oscilloscope is ready for start.

Status "started" – configuration of the channels is valid, oscilloscope started. The current values are shown.

Status "triggered" – trigger fulfilled, or manually activated with the "Trigger" button. The data is retrieved from the drive.

Status "ready" – measurement is complete and the curves are recorded.

Status "INVALID" – configuration of the channels is invalid.
The oscilloscope does not support a signal on a configured channel, i.e. this signal is to be corrected or otherwise deactivated.

Status "ERROR" – can be an triggered by an error in the oscilloscope as well as a different error. The exact cause can be determined with the AIPEX PRO tab 'Diagnostics'.

3.5.4.2 Scope - Maximum recording time

The maximum recording time is affected by the variable data memory size, as well as the configured measuring signals and the sampling time

The data memory in the drive can be configured with ID34284 'OSC container length'.
The default value is 4096 bytes, the maximum value is 32600 bytes.

The maximum recording time is determined by the following formula:

Example for determining the maximum recording time:

Parameter: (AIPEX PRO 'Parameter')	ID34284 'OSC container length'	4096 Byte
Measuring signals: (AIPEX PRO 'Scope')	ID40 'Velocity feedback value', length 4 byte* ID84 'Torque feedback value', length 2 byte* ID380 'DC-bus voltage', length 2 byte*	(Summe) 8 Byte
	Sampling time	10 ms

* The length (in byte) of each parameter can be taken in the column 'length' (AIPEX PRO tab 'Parameters').

Maximum recording time =

The initialization time of the oscilloscope function in the drive, triggered by pushing the button 'Start' corresponds approximately to the maximum recording time

3.5.4.3 Interpretation of position setpoints and actual position values

Synchronize the oscilloscope to the EtherCAT bus

For synchronous recording, the sampling time of the oscilloscope must be set equal to ID2 'SERCOS cycle time'.

When using the interpolator (e.g. for homing), the sampling time must be 1 ms.

Interpretation of the position values displayed in the oscilloscope

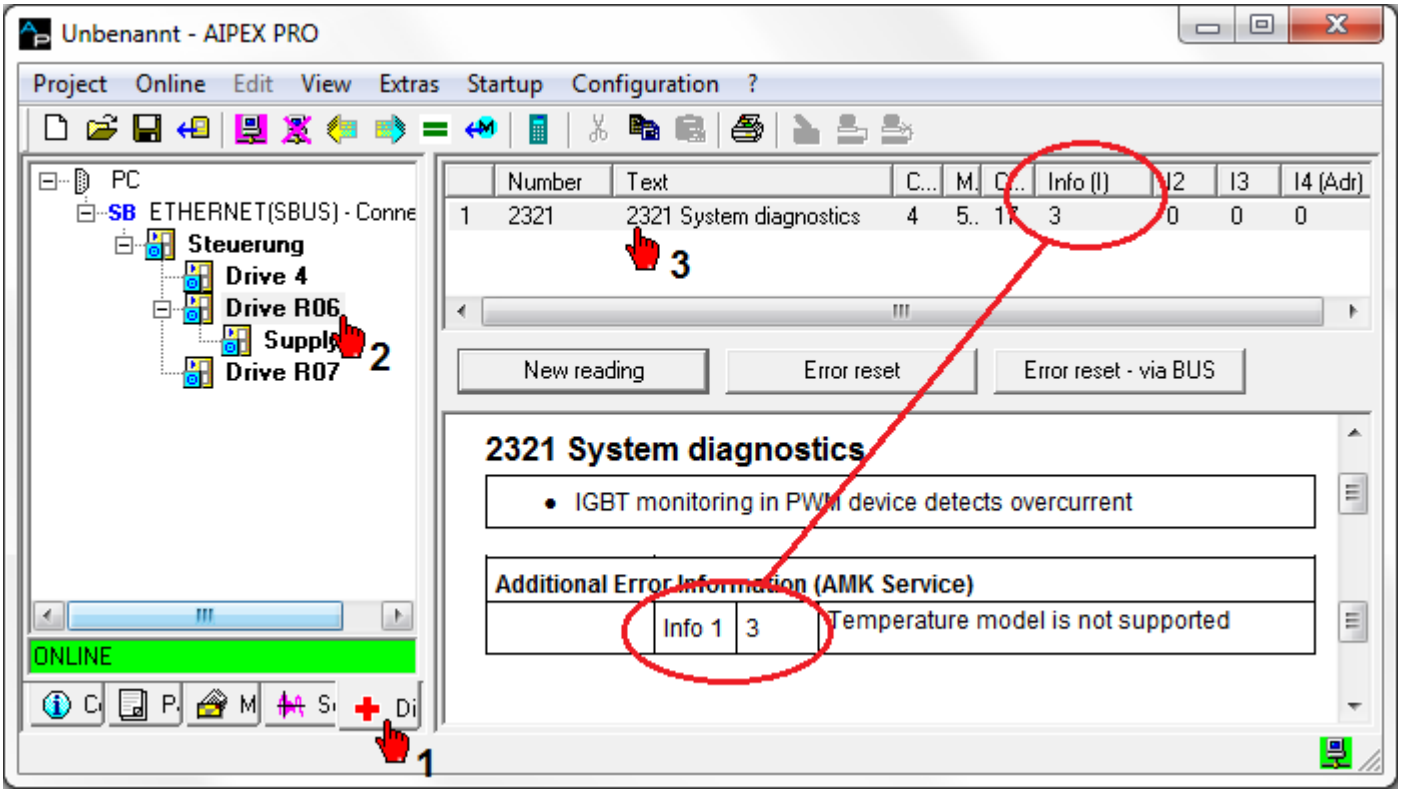
Due to the principle, the displayed position setpoint (ID47) precedes the displayed actual position value (ID51) by 2 measuring cycles.

A calculation of the following error in the same measuring cycle results in a corrupted value and differs from the displayed value ID189 'Following distance'.

Internal AMK calculation for the following error (ID189)

Following distance(t_x) = position setpoint(t_{x-2}) - position value(t_x)

3.5.5 Diagnose



With 'Diagnostics', the diagnostic messages can be read out from the selected device.

Click on each message to receive an explanation for it. You get further information if you analyse Info (I), I2 and I3

The first message of the list is the main activator of the fault; further displayed numbers might be resulting errors which will not appear any longer after rectifying the cause of the first diagnostic message.

Button **'New reading'**

Diagnostic messages will be read out from the selected device.

Button **'Error reset'**

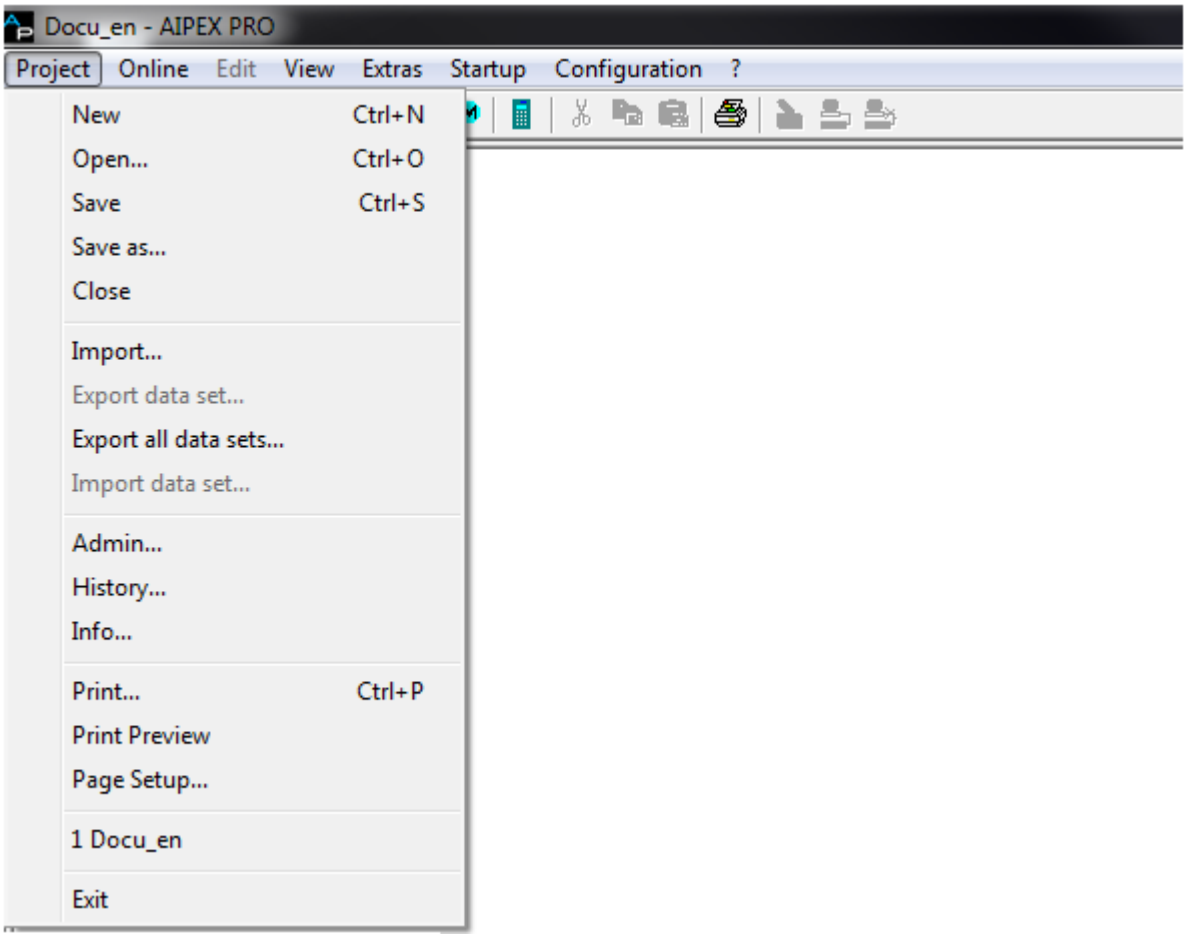
The errors will be deleted in the selected device .

Button **'Error reset - via BUS'**

The errors of all devices of a bus line will be deleted. To do this, select the bus in the device tree.

3.6 Menus

3.6.1 Project



New [Strg + N]

Icon

The opened project is closed and a new project is created.

Open... [Strg + O]

Icon

AIPEX PRO and AIPEX/AIPAR files can be opened.

Save [Strg + S]

Icon

When you save a file for the fist time, the 'Save as' dialog box is displayed automatically.

Save as...

Select a memory location. Enter your project name.

Close

The opened project is closed. Afterwards, you select the menu **'Project'** -> **'New'** to continue.

Import

Icon 

individual device data can be imported from a project data set.
Select your project data set.

Export data set...
With using the function 'Export data set', a XML file of the actual marked device will be generated and exported to the pc hard disk.

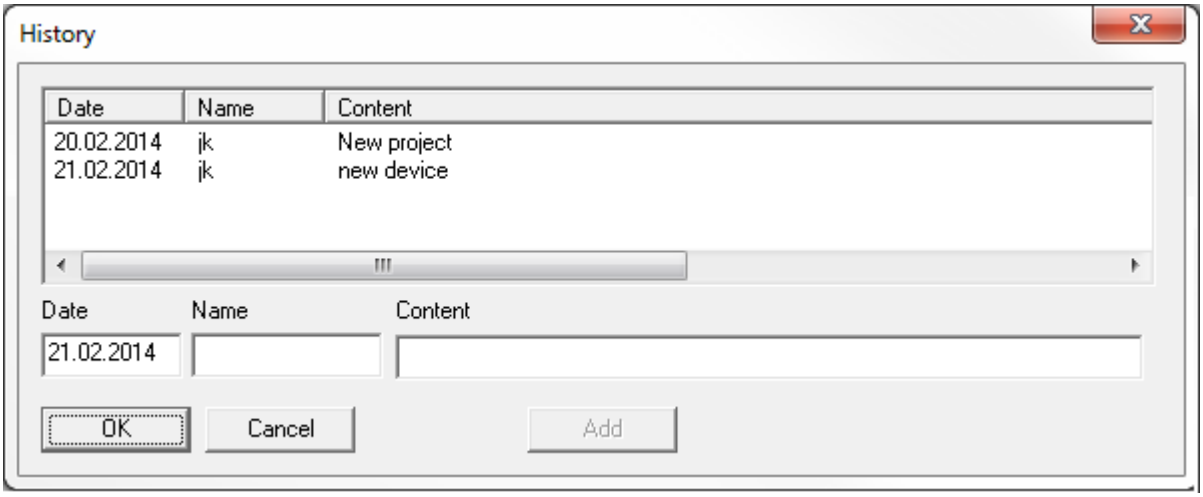
Export all data set...
With using the function 'Export all data set', a XML file of all device will be generated and exported to the pc hard disk.

Import data set...
By using the function 'Import data set' all existing and writeable data of an external XML file will be imported.

Admin...
Under 'Admin' you can delete complete AIPEX PRO project file.

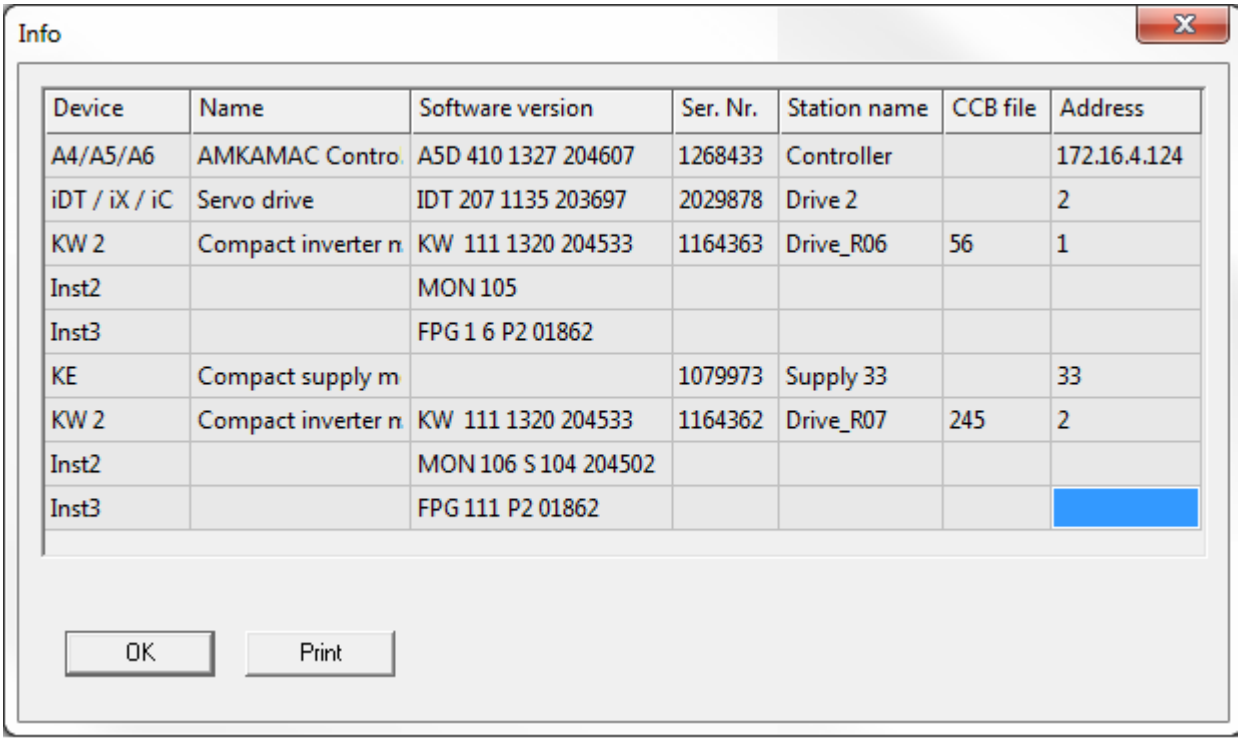
History...
Create a history for your project.

Adding entries:
Enter your text in the 'Content' field. To confirm, press the **'Add'** button.



Info...

The Info window provides detailed information about all devices that are in the project.



Print... [Strg + P]

Icon 

Use the function 'Print', the displayed data in the operating modi 'Parameter', 'Message' and 'Scope' can be printed out.

Print Preview

Use the function 'Print Preview' , the displayed data in the operating modi 'Parameter', 'Message' and 'Scope' will be displayed as print preview. Following the data can be printed.

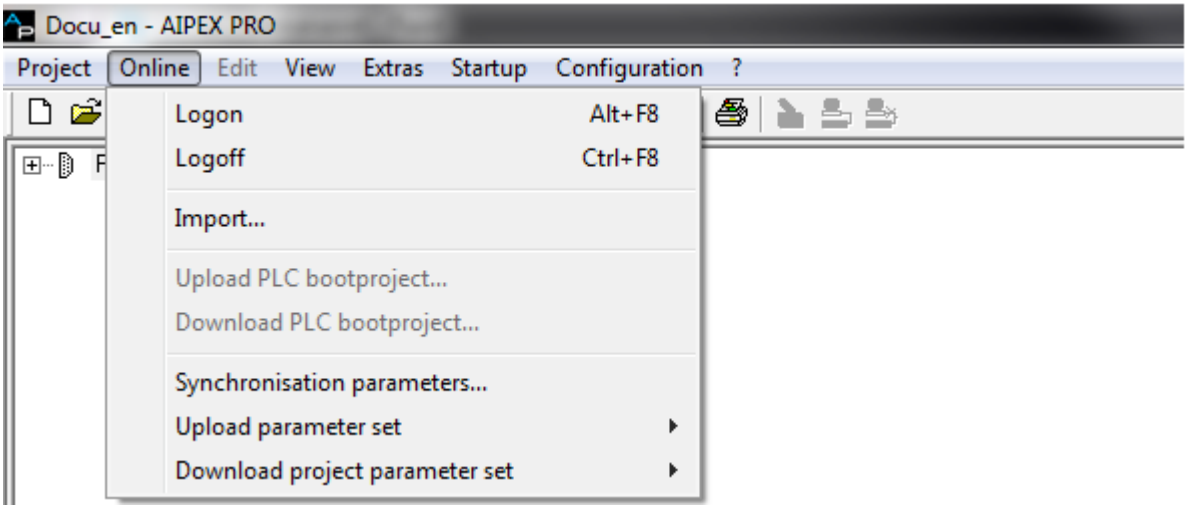
Page Setup...

Use the function 'Page Setup...' a printer can be adjusted.

Exit

The project and AIPEX PRO will be closed.

3.6.2 Online



Logon [Alt + F8]

Icon 

AIPEX PRO establishes a connection to the devices if there is an active interface.
All devices that are detected when the fieldbus is scanned appear in the window 'Import'.

Logoff [Strg + F8]

Icon 

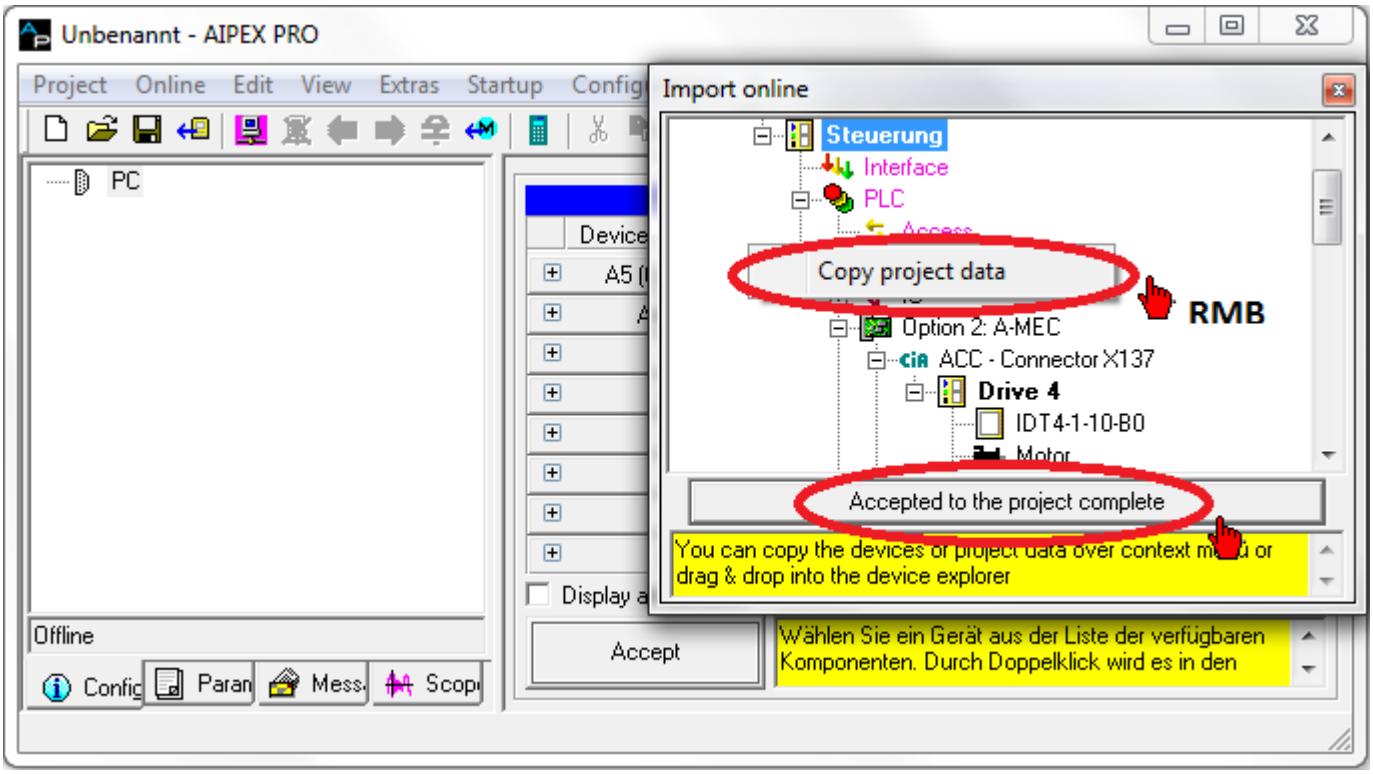
Closes the online connection to the devices

Import

Icon 

Right-click on the opened 'Import online' window. With the selection of **'Copy completely in the project'**, all device data is transferred.
Use the mouse (drag and drop) to import individual online devices from the import window into the current project.

The bus structure must always be maintained. (PC Connection – Bus master – Drive bus – Drives). Before you can import an individual drive for example into an empty project, you have to first insert the PC connection and a bus master manually.



Upload PLC bootproject...

The PLC program is copied from the selected controller into the AIPEX PRO project data set.
The PLC program is saved in ID34159 'PLC files'.

Download PLC bootproject...

The PLC program is copied from the AIPEX PRO project data set to the selected controller.
The PLC program is saved in ID34159 'PLC files'.

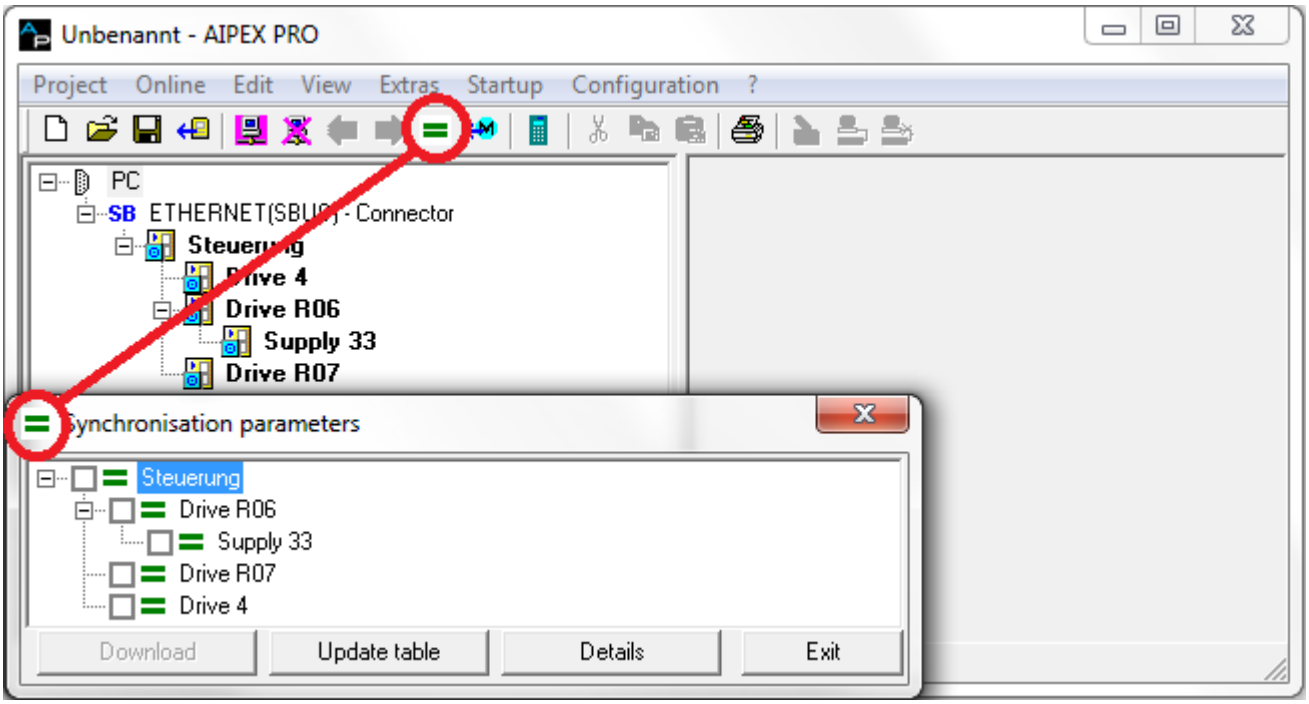
Synchronisation parameters...

Icon

The function **'Synchronisation parameters'** indicates whether the offline data set (PC) is identical to the online data set (device).

Pressing the button **'Download'** copies the offline data onto the selected devices.
(Function is identical to **'Online'** -> **'Download project parameters set'**)
Pressing the **'Finish read'** button copies the offline data of the selected devices to the PC (offline data).
(Function is identical to Online -> Upload parameter set)

The button of the Synchronisation parameter function indicates (And-linked) the device status.



Explanation of the symbols
Unequal (Control): Online and offline data set are not identical.
Question mark (Drive1): Online data are not yet available . (Press **Finish read** button.)
Equal (Drive): Online and offline data set are identical.

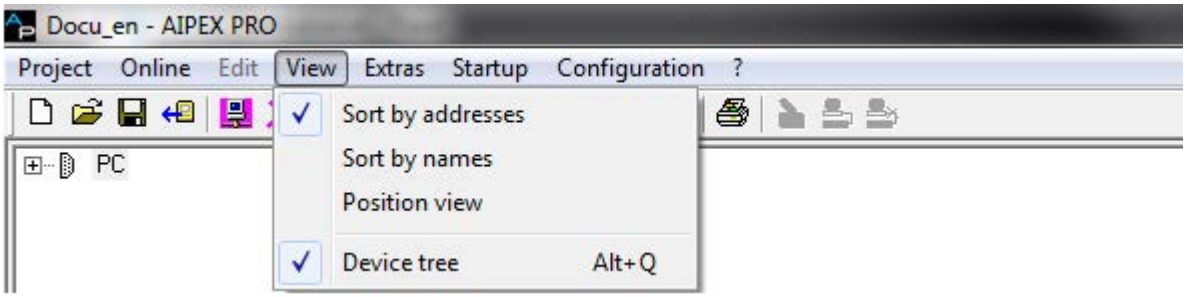
Upload parameter set
The function can only be invoked if **'Logon'** has been successfully completed first.
The current device data of a physically existing device (online device) is copied in the project data set (offline devices). Physically existing devices that do not exist in the project data set are not added as offline devices.

'Upload parameter set' -> **from selected device**: Only the device selected in the device tree is taken into account.
'Upload parameter set' -> **from all devices**: All devices existing in the device tree are taken into account

Download project parameter set
The function can only be invoked if **'Logon'** has been successfully completed first.
The data of the project data set (offline devices) are transferred to physically existing (online devices).

'Download project parameter set' -> **'to local device'**: Only the device selected in the device tree is taken into account.
'Download project parameter set' -> **'to all devices'**: All devices existing in the device tree are taken into account.

3.6.3 View



Sort by addresses
The devices in the device tree are sorted by ascending bus addresses.

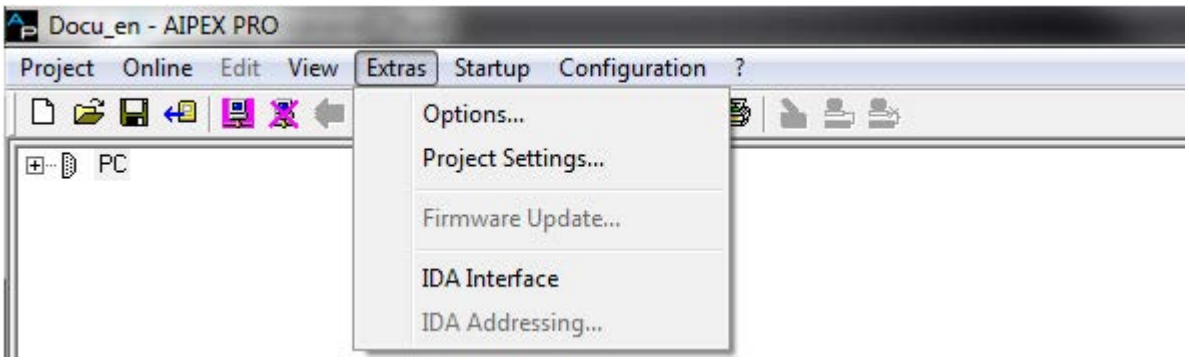
Sort by names
The devices in the device tree are sorted alphabetically according to their station names.

Position view
EtherCAT:
The devices are displayed in the same order, as the physically position in the bus system.
The devices can be moved inside the device tree.
Example: Siehe 'Moving device position in the EtherCAT bus' auf Seite 149.

ACC-Bus:
Unsorted

Device tree [Alt + Q]
The device tree can be display or hide.

3.6.4 Extras



Options...

Menu for 'Base Settings', 'PC Communication', 'Configuration create' and 'Data Update'.

Siehe 'Extras - Options' auf Seite 100.

Project Settings...

Menu for 'Base Settings' and 'Configuration create'. Valid for the actual open project.

Siehe 'Extras - Project Settings' auf Seite 108.

Firmware update...

Siehe 'Updating the device firmware' auf Seite 136.

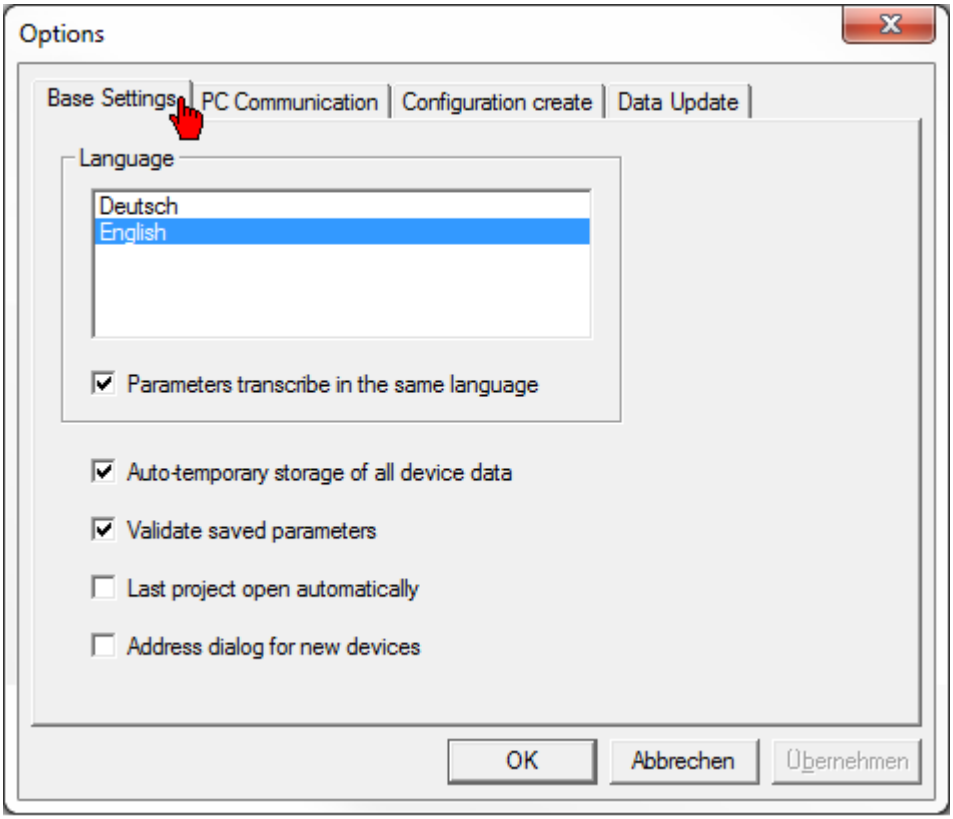
IDA Interface (AMK Service)

The IDA is a high-torque, external rotor, synchronous motor with integrated position encoder and servo controller as a ready to mount, compact unit.

IDA Addressing (AMK Service)

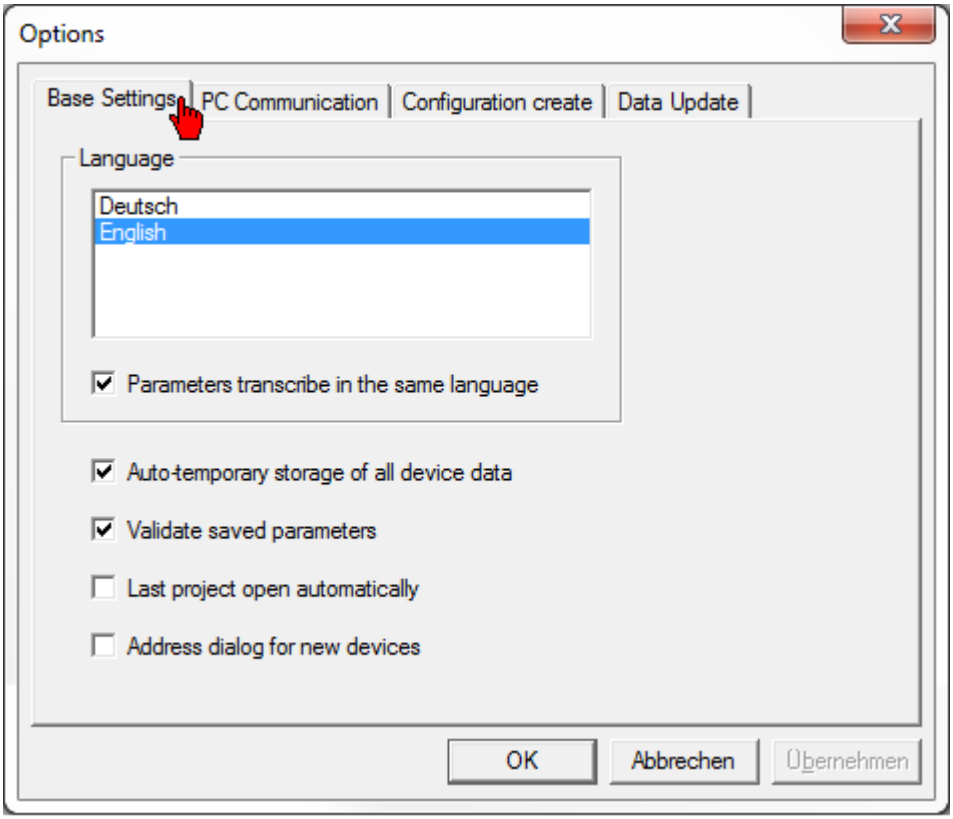
The IDA is a high-torque, external rotor, synchronous motor with integrated position encoder and servo controller as a ready to mount, compact unit.

3.6.4.1 Extras - Options



Siehe 'Base settings' auf Seite 101.
Siehe 'PC Communication' auf Seite 102.
Siehe 'Configuration create' auf Seite 103.
Siehe 'Data update' auf Seite 106.

3.6.4.1.1 Base settings



Language

Select the language in which the menu and the dialog texts as well as the online help should appear.

The device language is switched by the parameter ID265 'Language'.

Define here, in which language the menu and dialog texts should be displayed.

The device language will be changed with the parameter ID265 'Language'.

After the changing the language setting, AIPEX PRO needs to be rebooted.
Restart AIPEX PRO after changing the language.

Parameter transcribe in the same language

With deactivate option, the parameter names and units will be displayed with the language which is selected with ID265 'Language'

With active option, the parameter names and units will be displayed with the language which is selected in AIPEX PRO. This is only possible, if the ADB file with belongs to the firmware includes the responding text.

Auto-temporary storage of all device data

If this option is set, all parameters are read completely and buffered when a device is accessed the first time. (recommended).

If this option is not set, only currently needed parameters are read and buffered.

Validate saved parameters

The downloaded parameter values will be read back. Further the AIPEX PRO dataset will be compared with the online dataset.

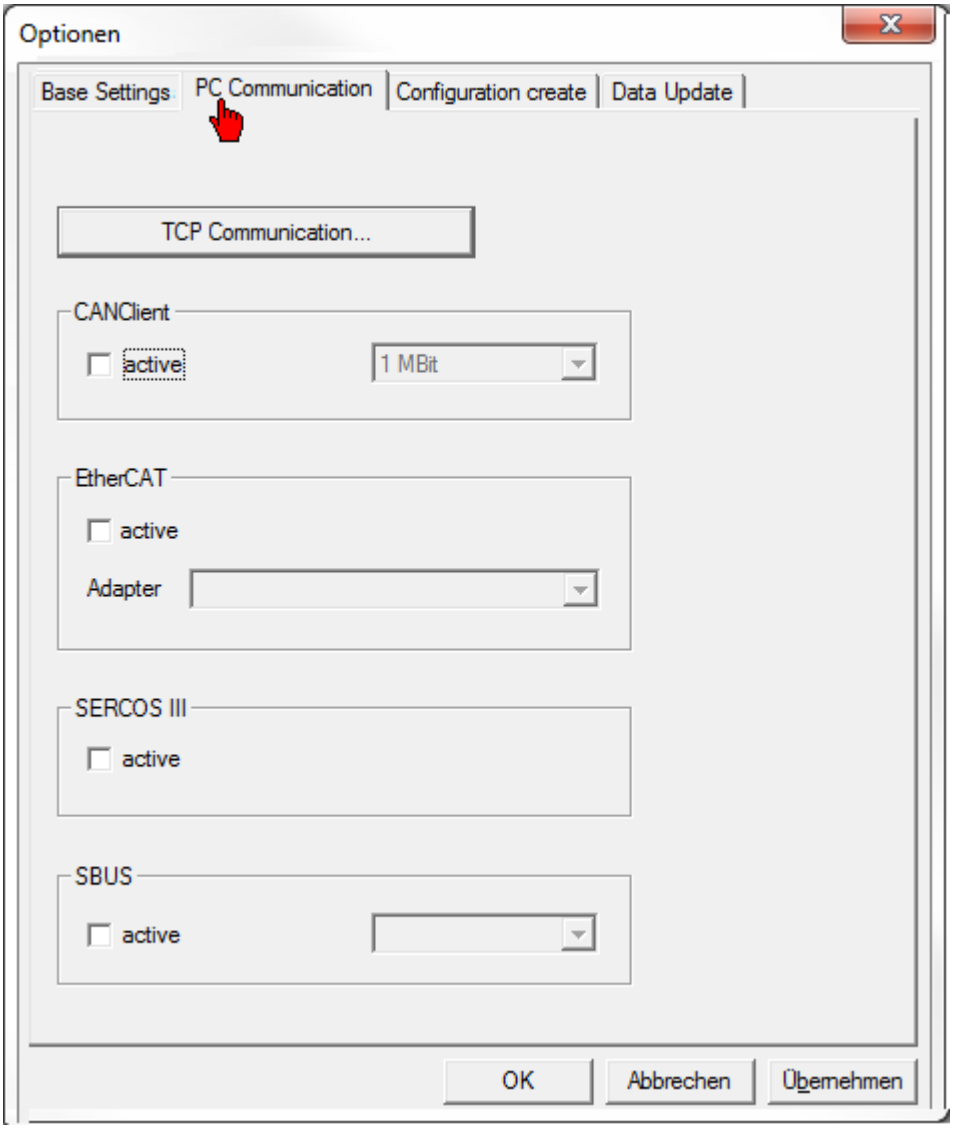
Last project open automatically

If this option is set, the last saved project is opened when AIPEX PRO is started.

Address dialog for new devices

If this option is set, a free device address can be entered when devices are inserted in the device configuration; otherwise, the next free address is issued automatically.

3.6.4.1.2 PC Communication

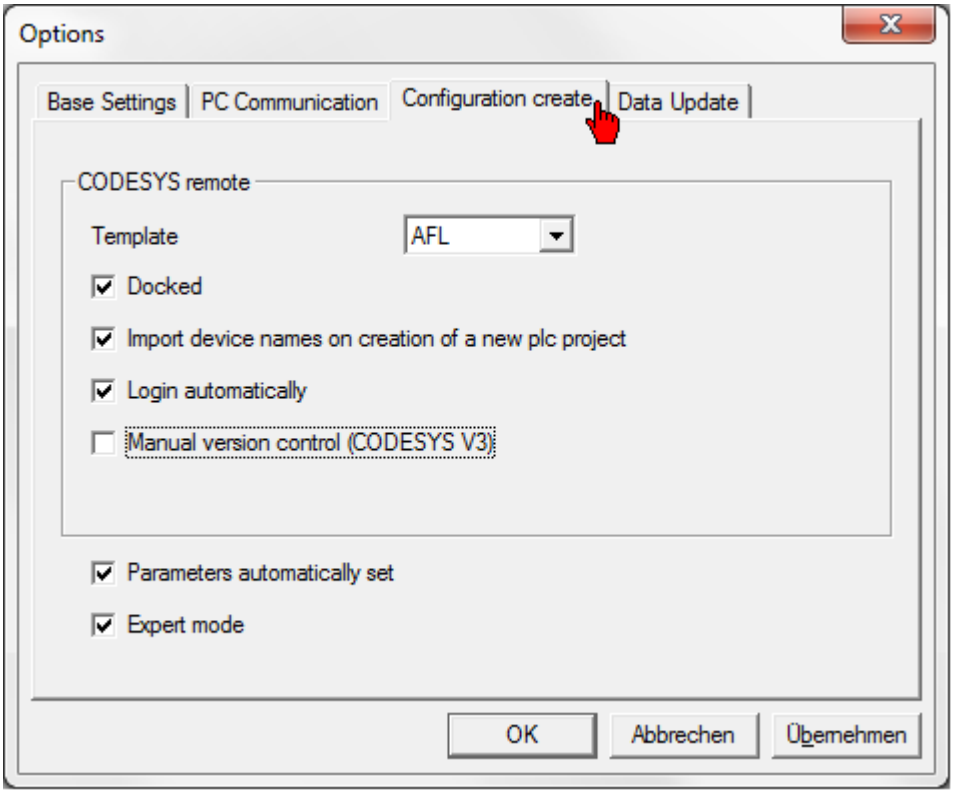


With the tap **'PC Communication'** you can choose the interface between PC and AMK device.
It depends on the target device, which interface is available.

- COM
- Ethernet (always active)
- CAN
- EtherCAT
- SERCOS III
- USB (always active)
- TwinCAT Gateway

Siehe 'Communication PC - AMK device' auf Seite 43.

3.6.4.1.3 Configuration create



Template

You define with 'Template' the start template of a new CODESYS project.

You can choose between 'ST structured text' 'FBD function block diagram' or if installed 'AFL library'.

ST is the default adjustment. The 'Template' must be set before the PLC project is generated.

Docked

The PLC editor CODESYS opens directly in the AIPEX PRO interface. If this option is not set, then the PLC editor CODESYS will be started as an independent application.

Import device names on creation of a new plc project

Devices that are physically available are automatically copied into the PLC project by the call up of 'Creating a PLC project'.

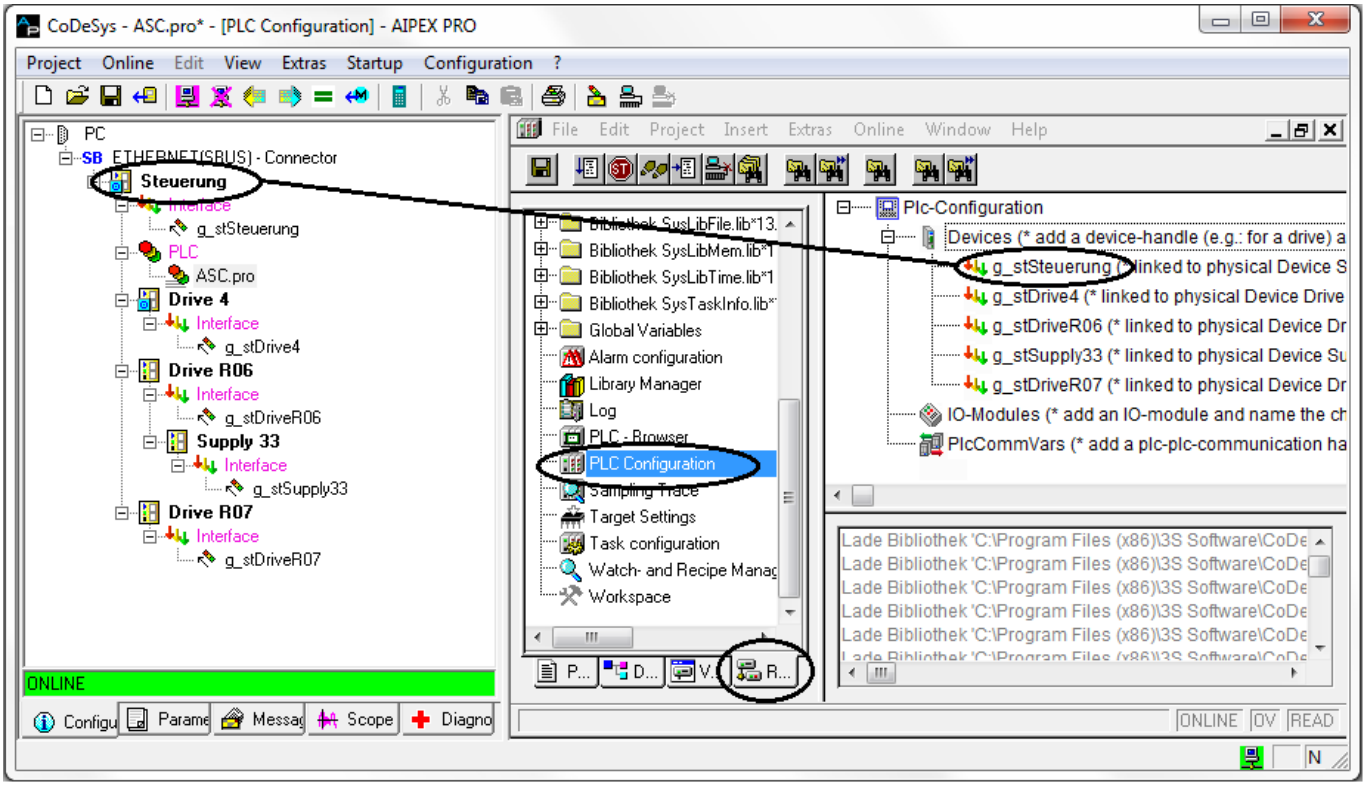
A later changing of the device name is not permitted.

Example CODESYS V2

You will find the symbolic device names at **'Resources'** -> **'Controller configuration'** -> **'PLC configuration'**.

In the device tree from AIPEX PRO is the PLC device handle name linked with the attendant device icon 'Interface' automatically.

The automatic message configuration use this information to create a message configuration file.

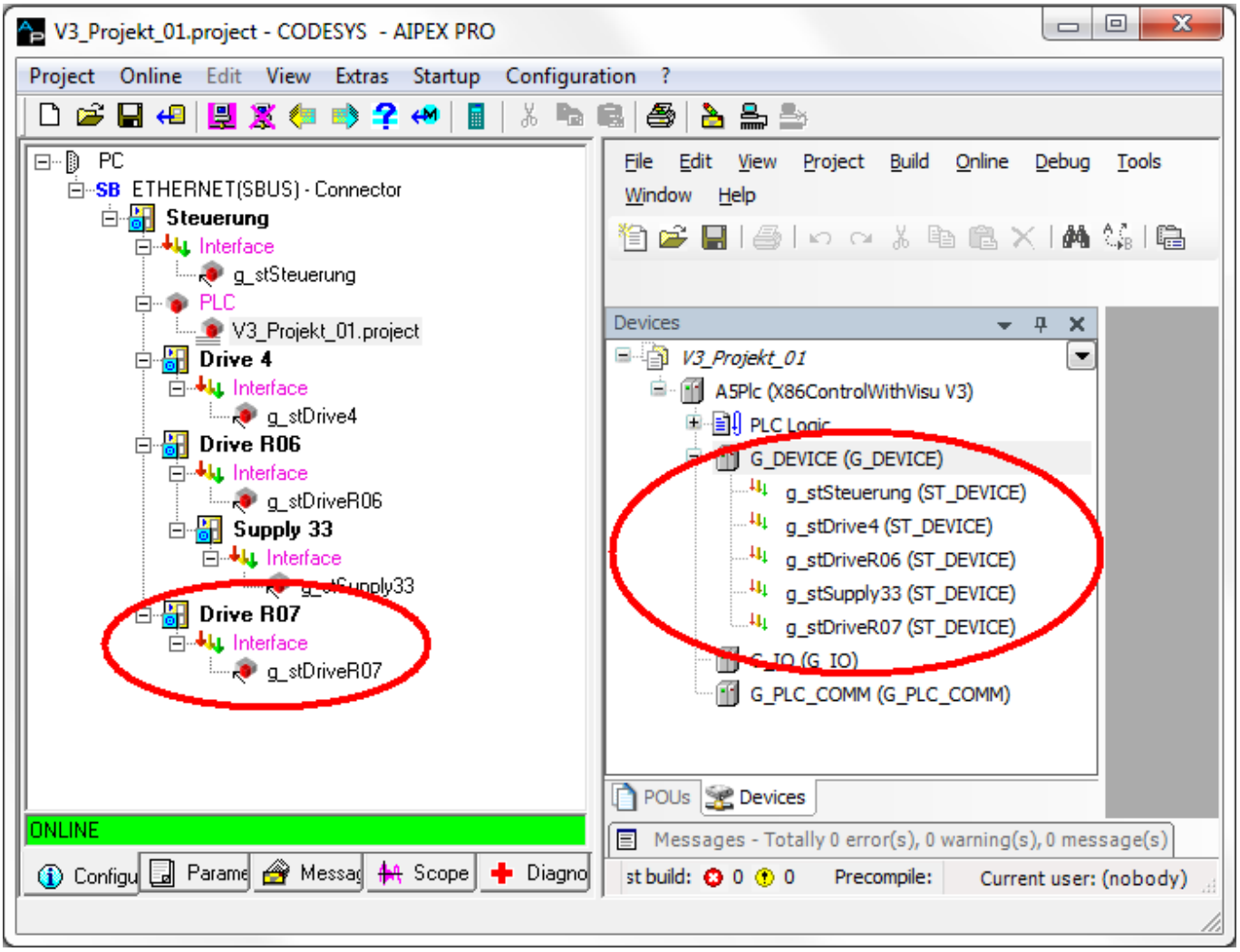


Example CODESYS V3

You will find the symbolic device names at **'Devices'** -> **'G_DEVICE (G_DEVICE)'** .

In the device tree from AIPEX PRO is the PLC device handle name linked with the attendant device icon 'Interface' automatically.

The automatic message configuration use this information to create a message configuration file.



Login automatically

If a configuration is created without faults, the CODESYS function 'Logon' is invoked.

From AIPEX PRO version 3.03 a manual or automatic CODESYS V3 version control is offered.

Manual version control (CODESYS V3) 'disabled' (Default setting)

- Create a new CODESYS V3 project → A new CODESYS V3 project will always be created with the latest installed CODESYS V3 version.
- Open a existing CODESYS V3 projec → AIPEX PRO automatically detects with which CODESYS V3 version an existing CODESYS V3 project was created. AIPEX PRO automatically opens the matching CODESYS V3 version. Is the matching CODESYS V3 version not installed, the project will open with the latest installed CODESYS V3 version.

Manual version control (CODESYS V3) 'enabled'

Before opening CODESYS V3, AIPEX PRO opens a window to select manually the CODESYS V3 version.

- Create a new CODESYS V3 project → Users must select an installed CODESYS V3 version, with the new project is to be created.
- Open a existing CODESYS V3 projec → Users must select an installed CODESYS V3 version, with the existing project is to be opened.

Siehe 'Version control CODESYS V3' auf Seite 709.

Parameters automatically set

For chosen AMK function blocks the required parameters are set automatically. This global setting can be individually adapted to the application.

Functional description and instructions for customizing:

Siehe 'Project Settings - Configuration create (project specific)' auf Seite 108.

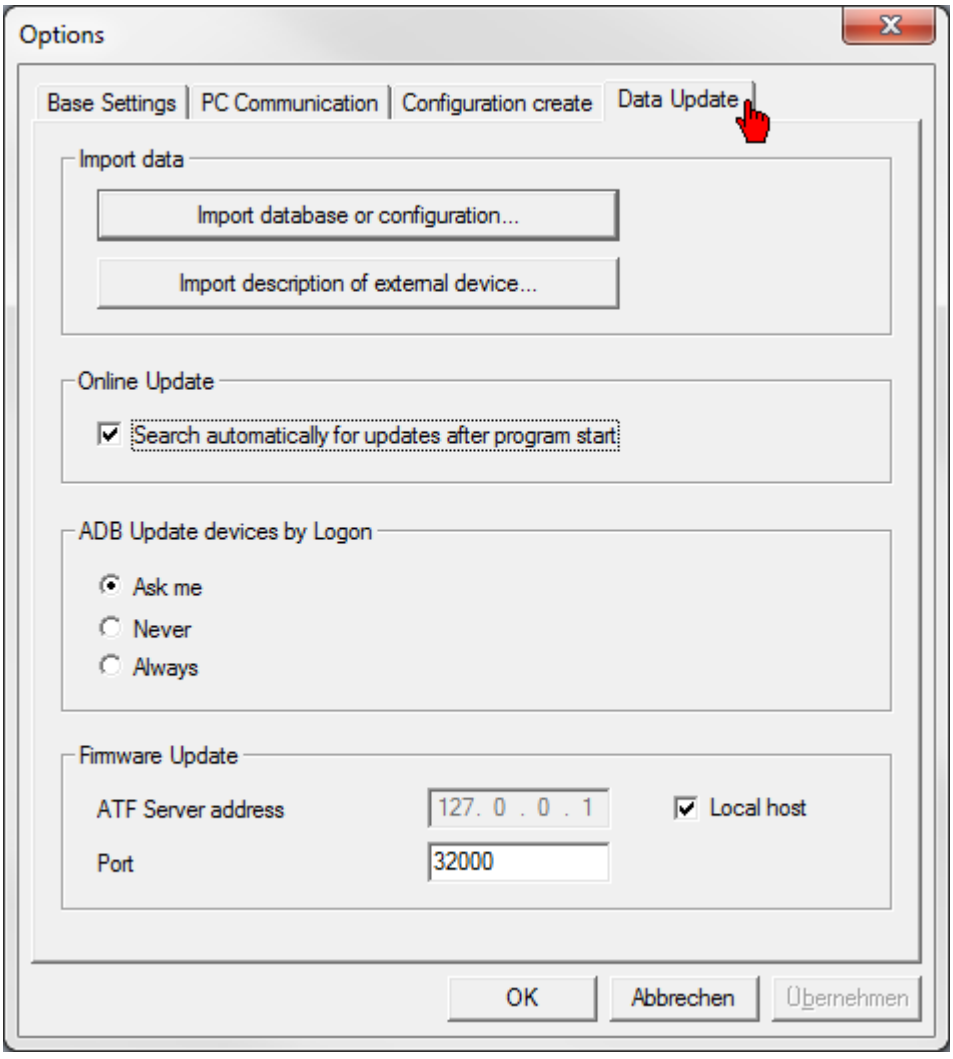
Expert mode

No new 'bus configuration' is created under **'CODESYS'** → **Logon**.

Changes in the PLC program are ignored that would have a new bus configuration as a consequence.

'Login' via CODESYS **'Online'** is available.

3.6.4.1.4 Data update



Import database or configuration

Any configuration file and complete updates can be imported by the function **'Import database or configuration'**.

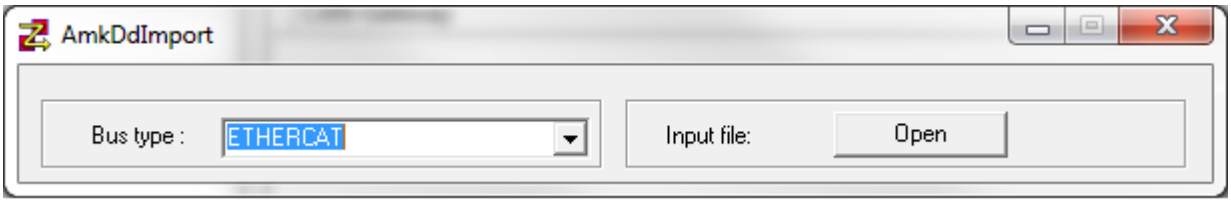
The selected file is automatically unpacked if necessary and copied into the correct AIPEX PRO directory.

Import external device configuration

The functionality of external devices, such as I/O terminals is included in device configuration files. For EtherCAT, it is XML files and for CAN (ACC bus), EDS files. Using the function **'Import external device'** configuration, the files can be imported.

No guarantee can be assumed for the functionality of additionally imported devices.

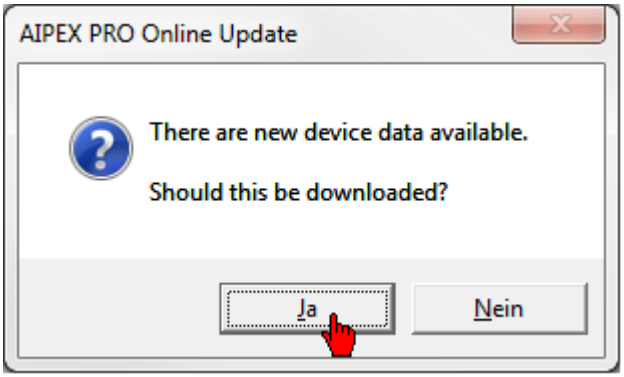
After the importing, AIPEX PRO needs to be rebooted.



Online Update

Starting the program there will be an automatic search for 'Online Updates' on AMK web site.

The download of an available Update must be confirmed.



Following files will be updated trough the function Online Update:

- ADB: AMK database, it contains information about all AMK parameters
- MDB: Motor database, it contains information about all AMK motor parameters
- Device file (e. g. for EtherCAT devices)
- Documentation

Requirements:

- Internet connection
- Administrator rights

For this update you must have administrator rights. If you start AIPEX PRO, the update files will be searched and found, but you are not able to install it.

You can also start the 'Online Update' manually. Siehe "?' - Documentations' auf Seite 122.

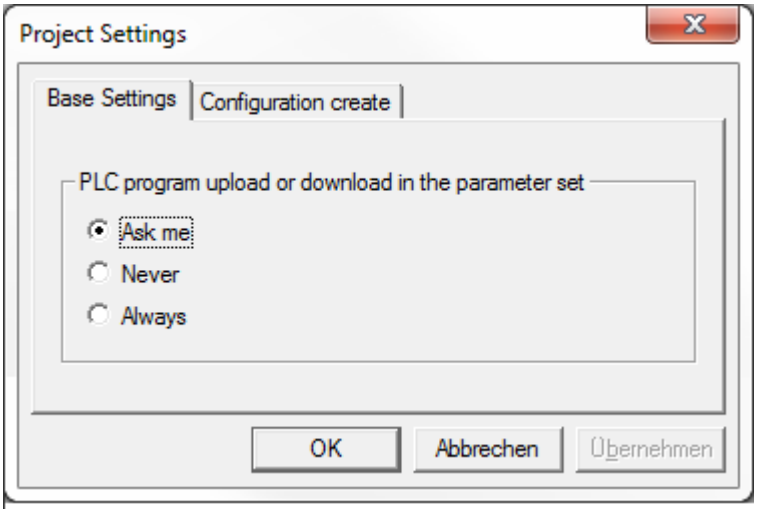
ADB update devices by logon

ADB (AMK database). The ADB file contains a modul specific parameter list, with parameter properties and default values.

When you logon, if required the ADB file will be updated into the existing controller. The ADB update is necessary, when the controller connected with updated (Firmware) devices as Bus-Slaves. The Controller can only accurate route to slaves, as is well known the firmware version on the internal ADB.

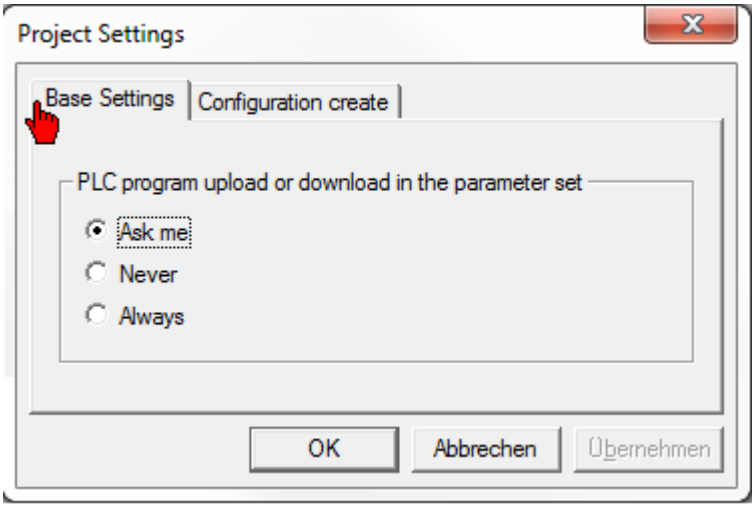
Firmware Update
PC address, where the software ATF (server software) (AMKASYN TOOL FLASHER) is installed.
Default adjustment ATF Port 32000.
If the port is used by another application, open the software AIPEX PRO and ATF and change the port.
Example: Siehe 'Updating the device firmware' auf Seite 136.

3.6.4.2 Extras - Project Settings



Siehe 'Project Settings - Base Settings' auf Seite 108.
Siehe 'Project Settings - Configuration create (project specific)' auf Seite 108.

3.6.4.2.1 Project Settings - Base Settings

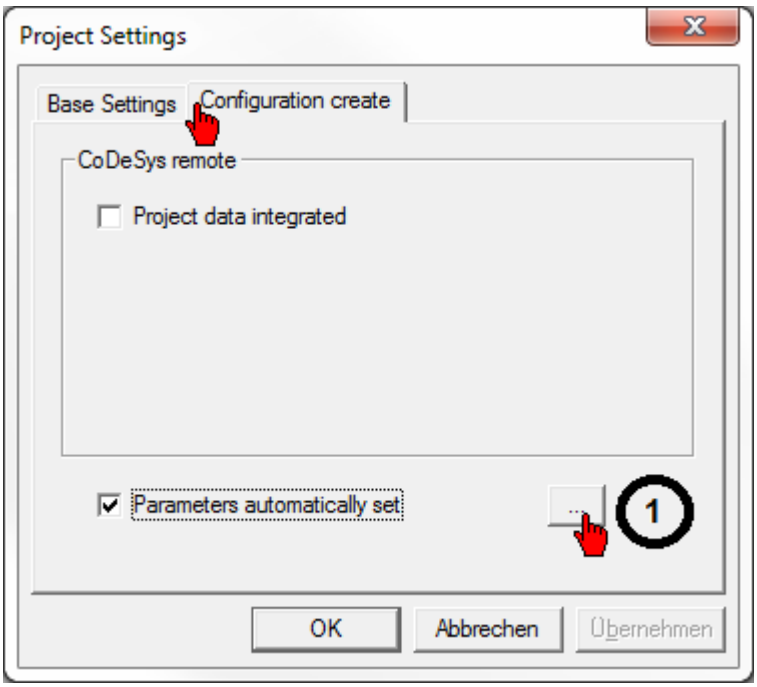


An existing PLC project will be stored inside ID34159 'PLC files'.

The 'Base Settings' have affect to the menu points:
'Online' -> **'Upload parameter set'**
'Online' -> **'Download project parameter set'**

3.6.4.2.2 Project Settings - Configuration create (project specific)

The project settings can be changed before each generation (AIPEX PRO menu 'Configuration' → 'Configuration create... ').



Project data is integrated

The CODESYS projects are integrated in the AIPEX project file.

If this option is not set, the CODESYS projects are saved in a separate directory next to the AIPEX project.

Parameter automatically set

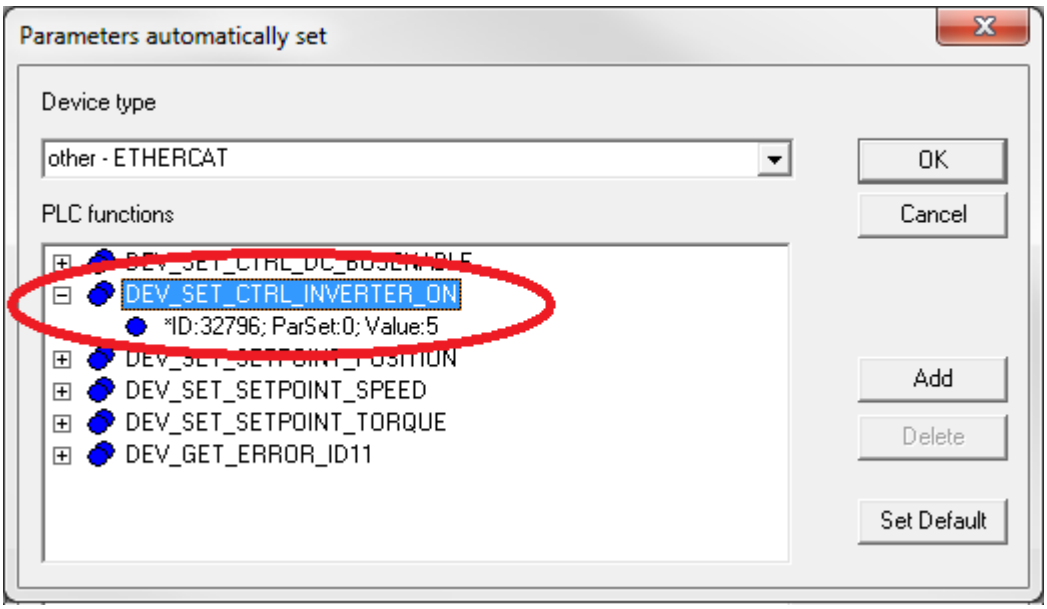
For chosen AMK function blocks the required parameters values are set automatically.

The function blocks which have a automatic parametrization can be found in the dialog box 'Parameters automatically set' (see button no 1).

Example:

If the AMK function block 'SET_CTRL_INVERTER_ON__RF' is used in CODESYS , AIPEX PRO sets the value of the ID32796 'Source RF' source controller enable to 5 (importance: Signal controller enable is set by the PLC).

The IDs and their significance are described in the parameter description.



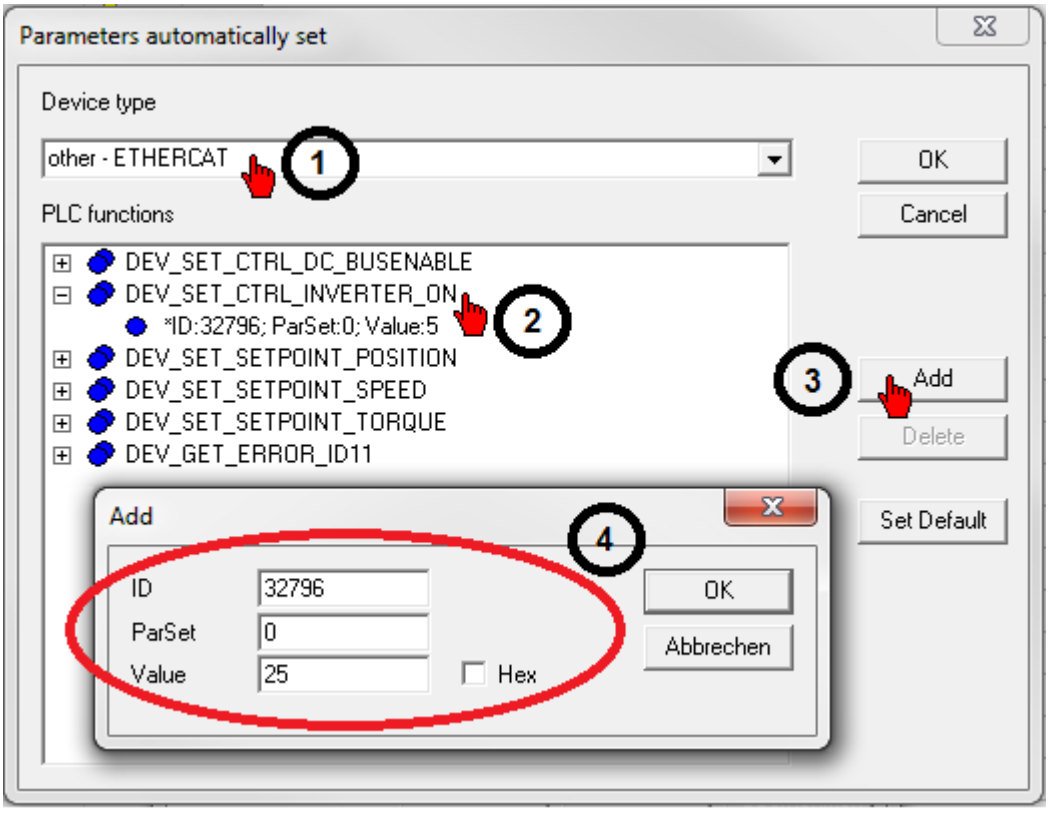
Selection - device type	meaning
EC1-100 - ETHERCAT	AMKASYN option KU/KW-EC1

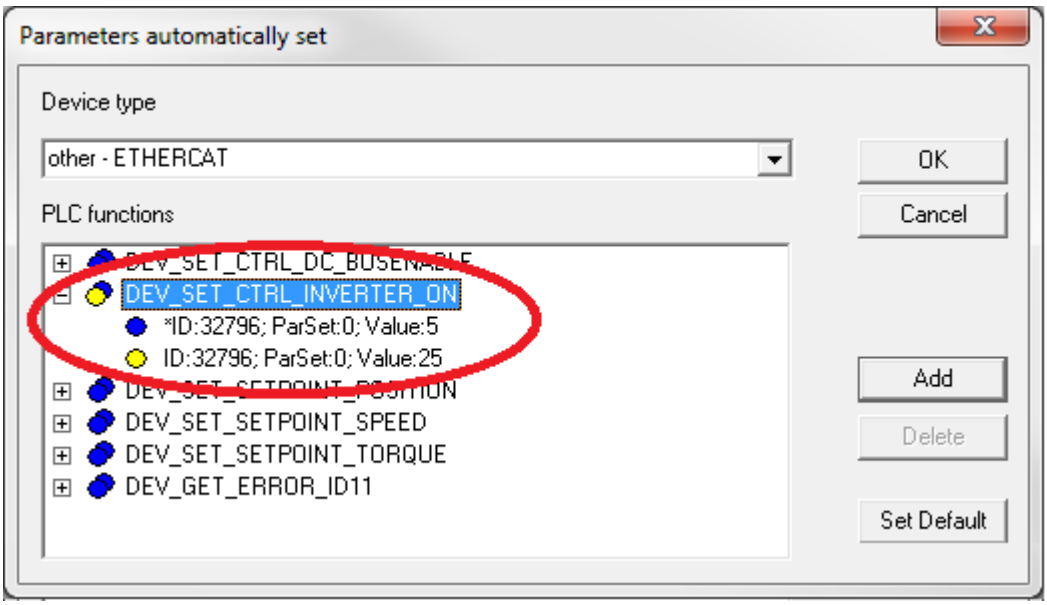
Selection - device type	meaning
IDT - 100 - CAN	AMKASMART iDT4
ihX - 100 - ETHERCAT	AMKASMART ihX
KE - 207 - CAN	AMKASYN KE compact power supply with ACC Bus
KE - 400 - ETHERCAT	AMKASYN KE compact power supply with EtherCAT
KWF - 103 - CAN	AMKASYN KWF compact frequency module with ACC Bus
KW-R2x - 200 - ETHERCAT	AMKASYN controller card KW-R24(-R), KW-R25, KW-R26, KW-R27
other - CAN	AMKASYN controller card KW-R03(P), KW-R04
other - ETHERCAT	AMKASYN controller card KW-R06, KW-R07, KW-R16, KW-R17 AMKASMART iC, IX, iDT5
other - Local	Local IOs onto option KU-/KW-PLC2

If there is no data set for a concrete device type, the general ('other') data set is used.

Using the button **'Add'** you can assign own device parameters to the PLC functions.
Using the button **'Delete'** you can delete own device parameters.
Pressing the **'Specification'** button resets to the original AMK conditions.

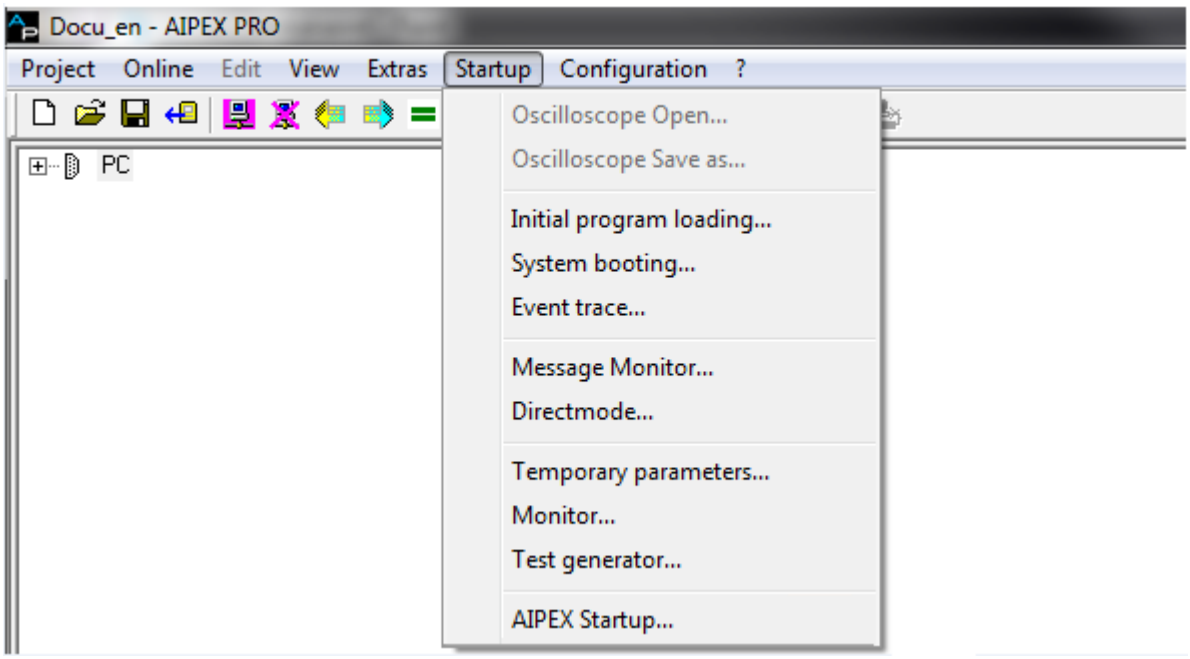
Example controller card KW-R06:
The default configuration ID32796 'Source RF' code 5 'Controller enable via EtherCAT' should be changed to code 25 'RF via EtherCAT AND-linked with the binary input RF'





To activate the changing, execute the AIPEX PRO menu 'Configuration' → 'Configuration create... '. Then 24 VDC OFF/ON.

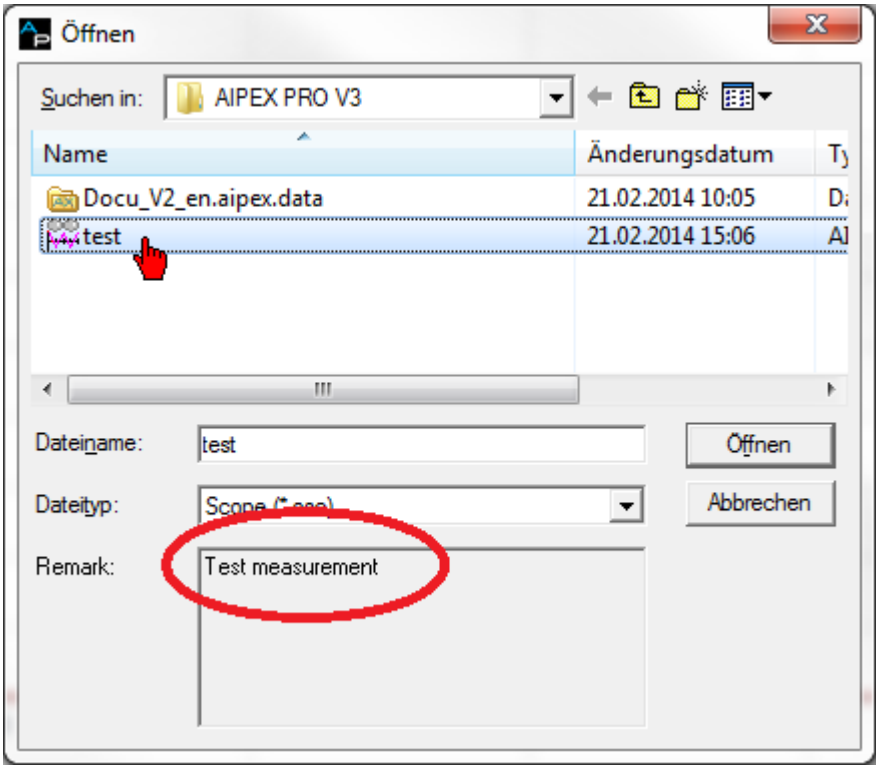
3.6.5 Start up



Oscilloscope open...

The menu item **'Oscilloscope open'** is only active in the tab **'Scope'** active.

A saved oscilloscope file (*.osc) can be opened.



Oscilloscope save as...

The menu item **'Oscilloscope Save as'** is active only in the tab **'Scope'**.

The current settings, measuring values and the corresponding description (comment field) are saved as oscilloscope files *.osc.

Initial program loading...

The 'Initial program loading' function resets AMK devices into their initial status (delivery status).

After a completed initial program loading, a system reset needs to be done.

Prerequisites for the initial program loading: A direct connection between PC and AMK device, additional password input.

System booting...

A 'System booting' is carried out on the selected device.

A 'System booting' causes a recalculation of the data management. (Actual values are maintained, drive bus continues running...)

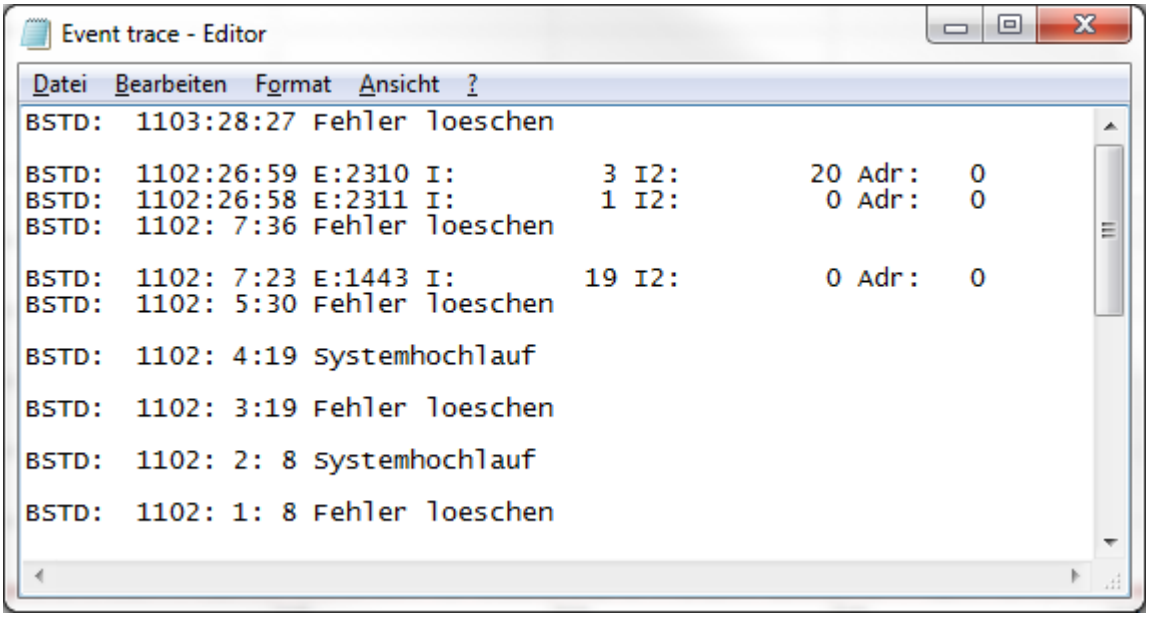
Event trace...

The 'Event trace' menu can be used to read the contents of ID34088 'Event trace' and view it as text in the text editor. This function can also be called up in offline mode if a project data set exists.

The ID34088 'Event trace' of the currently selected device will be displayed.

The list is created in LIFO mode (last in, first out) so that the last entry is seen first.

The number of saved errors depends on the type of device.



BSTD: Operating hours

E: Error message number

I: Additional Information

Adr: Device Address

Message Monitor...

The 'Message Monitor' is a dynamic display of mapped variables. A cyclic, but temporally undefined display of the values follows.

With **EtherCAT**, not the values are read out directly and displayed from the selected device, but rather their counterpart in the respective master.

Only the variables mapped to the master can be displayed (no slave – slave connection).

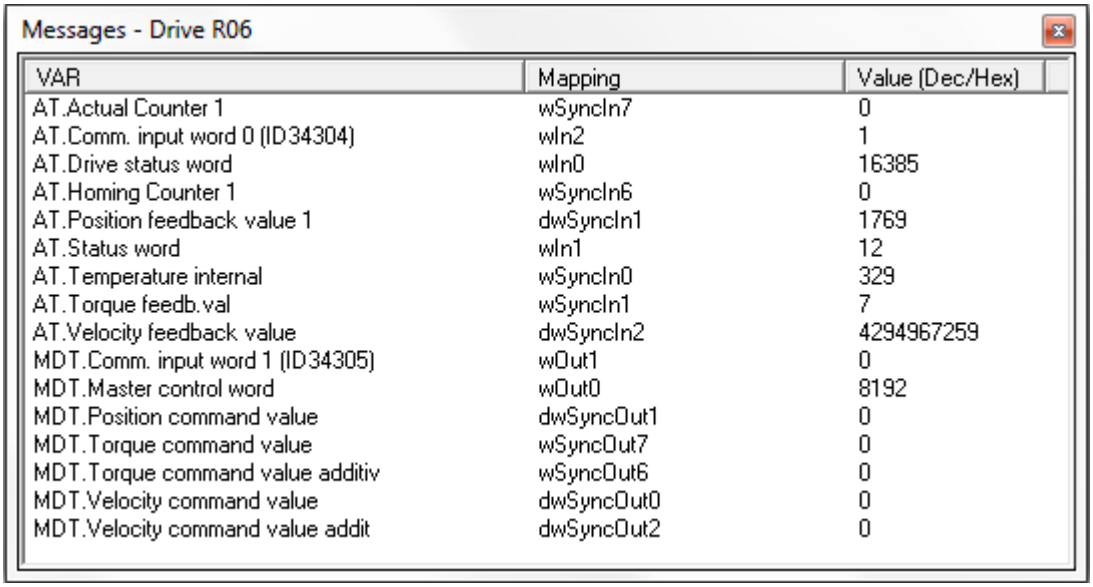
Only the values can be displayed that are mapped by an ID in the master.

The target device needs to be selected before calling up this menu item in the device tree.

For the **ACC bus**, an IXXAT CAN adapter needs to be integrated into the ACC bus. All messages in the selected ACC bus are displayed dynamically with sender and receiver as well as the current value.

The menu item can only be called up if the CAN adapter has not been activated for the online connection.

The ACC bus master needs to be selected before calling up this menu item in the device tree.



VAR	Mapping	Value (Dec/Hex)
AT.Actual Counter 1	wSyncIn7	0
AT.Comm. input word 0 (ID34304)	wIn2	1
AT.Drive status word	wIn0	16385
AT.Homing Counter 1	wSyncIn6	0
AT.Position feedback value 1	dwSyncIn1	1769
AT.Status word	wIn1	12
AT.Temperature internal	wSyncIn0	329
AT.Torque feedb.val	wSyncIn1	7
AT.Velocity feedback value	dwSyncIn2	4294967259
MDT.Comm. input word 1 (ID34305)	wOut1	0
MDT.Master control word	wOut0	8192
MDT.Position command value	dwSyncOut1	0
MDT.Torque command value	wSyncOut7	0
MDT.Torque command value additiv	wSyncOut6	0
MDT.Velocity command value	dwSyncOut0	0
MDT.Velocity command value addit	dwSyncOut2	0

Direct mode...

With 'Direct Mode' you get online access to the AMK devices.

The change will be immediately transferred to the device. In which time becomes active depends on the parameter attribute.

If you following want to transfer the online changing to the AIPEX PRO data set, you have to call the menu **'Online'** -> **'Upload parameter set'** . Siehe 'Online' auf Seite 96.

The 'Direct Mode' needs a active communication between PC and Device.
Siehe 'Communication PC - AMK device' auf Seite 43.

Siehe 'Direct mode' auf Seite 122.

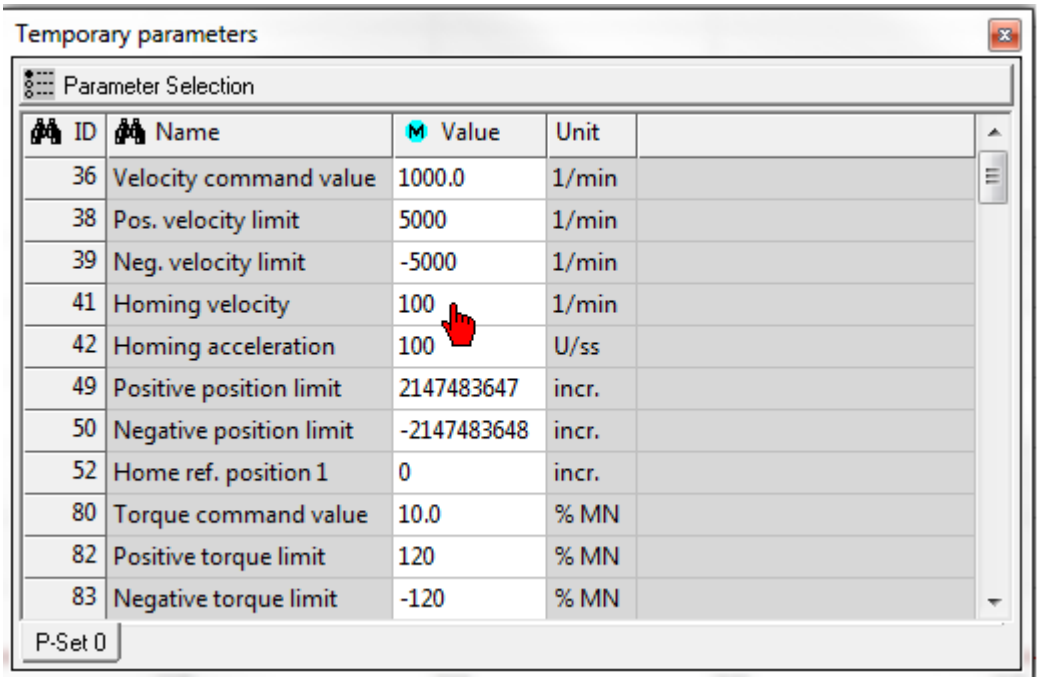
Temporary parameter...

Changes in the 'Temporary parameters' are effective immediately in the drive.

In the selection list, only parameters are offered that can be modified temporarily.

All modified values are displayed in green font.

When the 'Temporary parameters' window is closed, you can specify whether or not the changes should be saved permanently.



Monitor...

The **'Online Monitor'** is a dynamic display of device data and statuses. A cyclic, but temporally undefined display of the values follows.

Siehe 'Start up - Monitor' auf Seite 115.

Testgenerator...

The 'Test generator' function of the AIPEX PRO software gives support to the initial startup and controller tuning.
The startup function contains a setpoint generator which can set several curve forms (trapezoidal, square wave, triangular, sinusoidal) for different operation modes (torque, speed, position control).

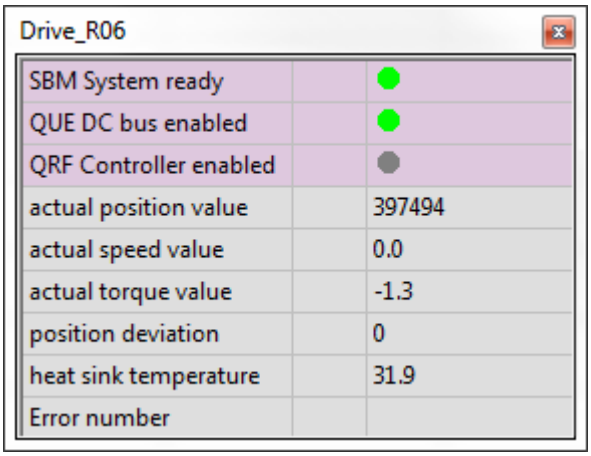
Siehe 'Test generator' auf Seite 160.

Startup application...
The 'AIPEX Startup' software supports the startup of AMK drive systems. AMK drive systems connected with an electronic nameplate are detected and identified automatically. At the same time, all the important variables for the motor, motor encoder and gear are displayed. The software can determine operational parameter values independently for the current controller, speed controller and position controller in a very short time and set them in the drive. The measured control response to a setpoint jump is recorded automatically and displayed graphically.
The user can influence and optimize the controller settings for his application.

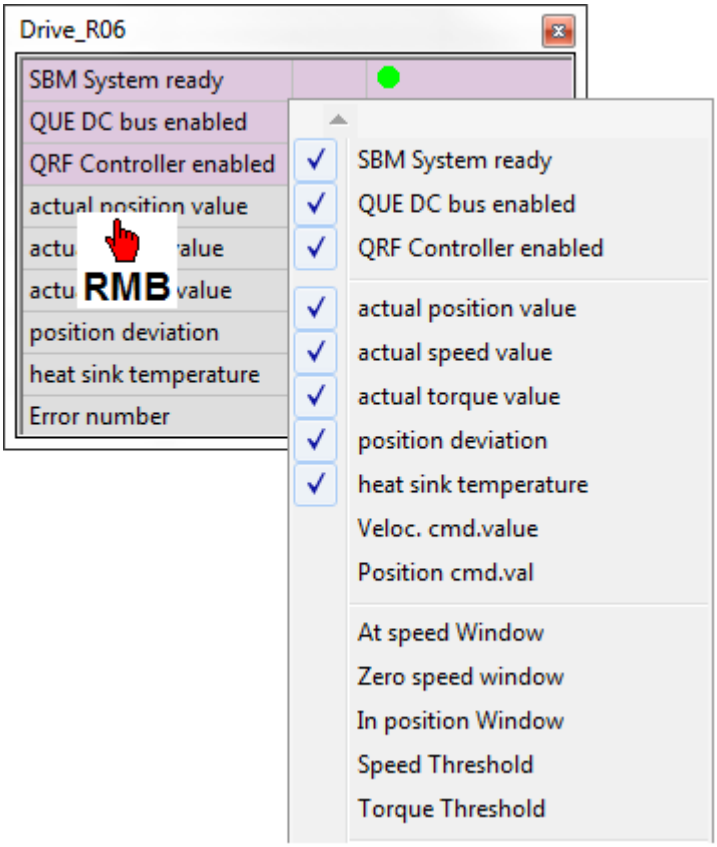
The 'AIPEX Startup' software is a own product and must be installed for using with the AIPEX PRO menu.
See document Software description AIPEX PRO Startup, (Teile-Nr. 205171).

3.6.5.1 Start up - Monitor

The **'Online Monitor'** is a dynamic display of device data and statuses. A cyclic, but temporally undefined display of the values follows.



By clicking the right mouse button on the **'Online Monitor'** a context menu opens that allows further display values to be taken into the 'Monitor'.



Parameter

With the field **'Parameter'** you can dynamic display any parameter.

Enter in the second column, the parameter number that should be displayed.

Dynamic display values, special for ACC bus

The content of the dynamic display of the 'Message 16' can be modified by ID32785 'Message 16'.

The content of the dynamic display of the 'Message 32' can be modified by ID32786 'Message 32'.

Example: ID32786 = 40 (actual speed); The actual speed is displayed dynamically by selecting 'Message 32'.

ByIn/Out, wIn/Out, dwIn/Out are PLC variables. Enter in the second column, the index of your PLC variable that should be displayed.

Example: In the PLC (controller configuration), the Word wOut4 is configured. Enter 4 as the index.

3.6.5.2 Test generator

DANGER	
	**Motor shaft movement!** Hazardous motor movement occurs when the motor shaft moves in an uncontrolled or unintentional manner. **Steps to prevent:** • Decouple the motor from the load so that the shaft can rotate freely • Specify the maximum permissible process speed and set ID113 accordingly. (ID113 = max. process speed/1.25)

The 'Test generator' function of the AIPEX PRO software gives support to the initial startup and controller tuning.
The startup function contains a setpoint generator which can set several curve forms (trapezoidal, square wave, triangular, sinusoidal) for different operation modes (torque, speed, position control).

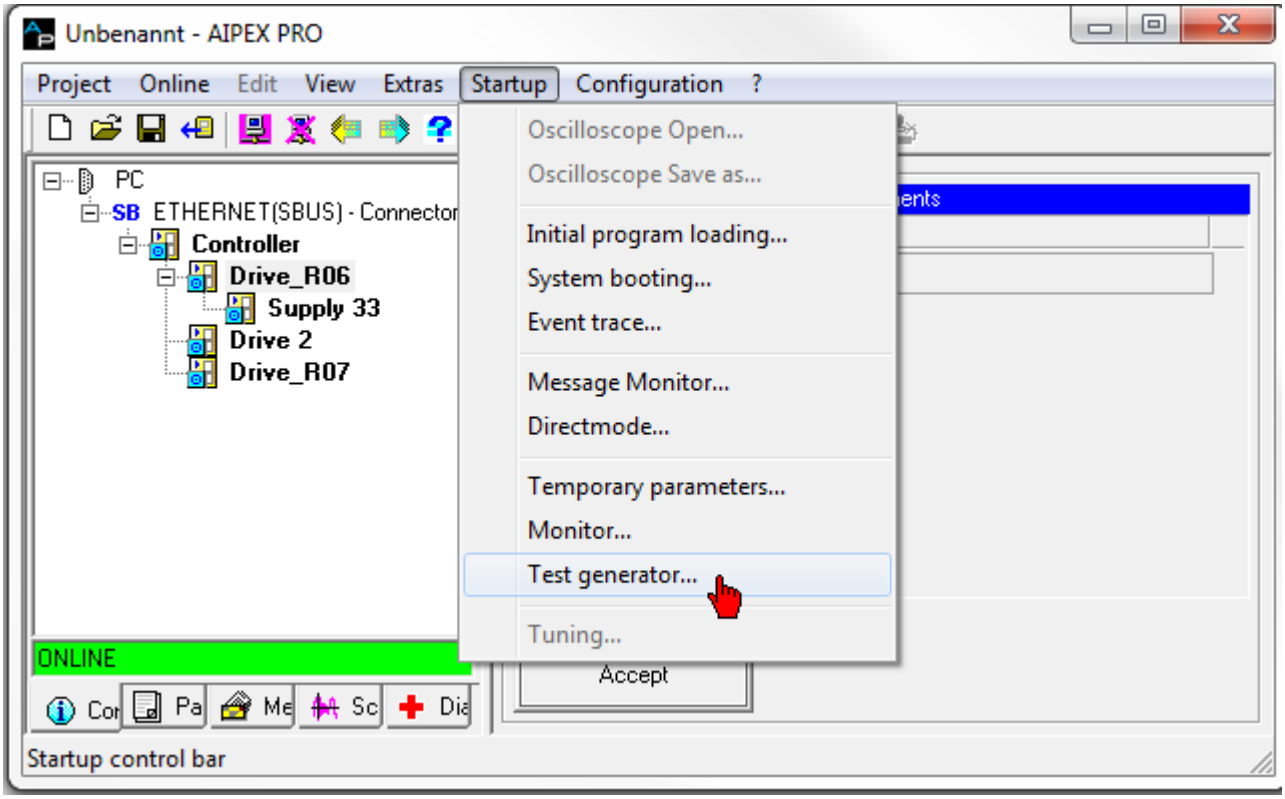
Before you first put the motor into operation, check the sense of rotation

The following example shows how to set a positive speed setpoint with acceleration and deceleration ramp.
By means of the 'Scope' function of AIPEX PRO, you can display and store the actual motor values

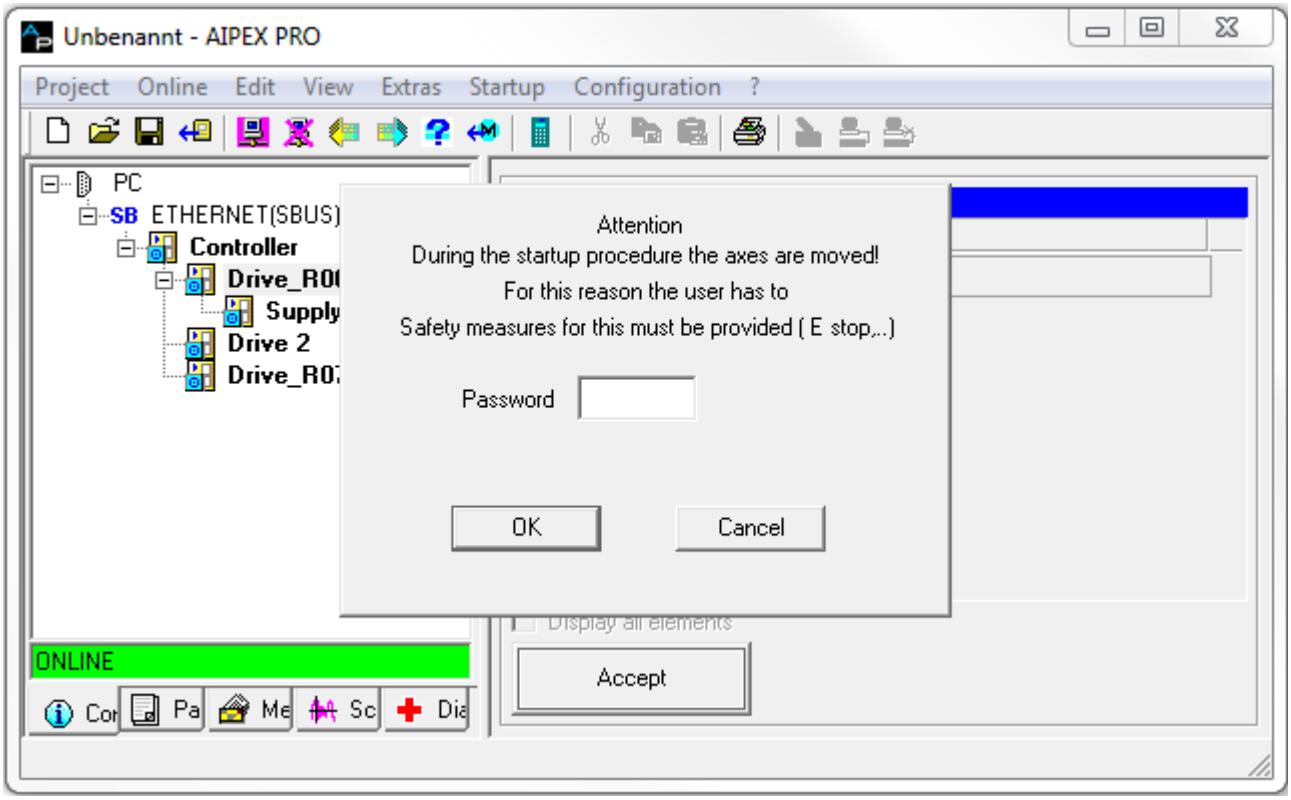
The sense of rotation can be inverted by ID32773, bit16 = 1

Starting the 'Test generator'

In the device tree click to that device you want to set a setpoint by the startup function. Click to the **'Startup'** menu and subsequent to **'Test generator...'**.

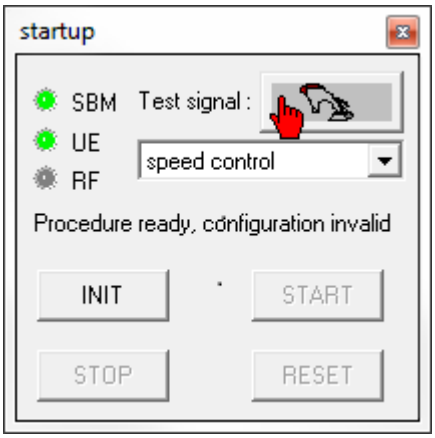


The 'Test generator' will be released by the AMK service password (500591).



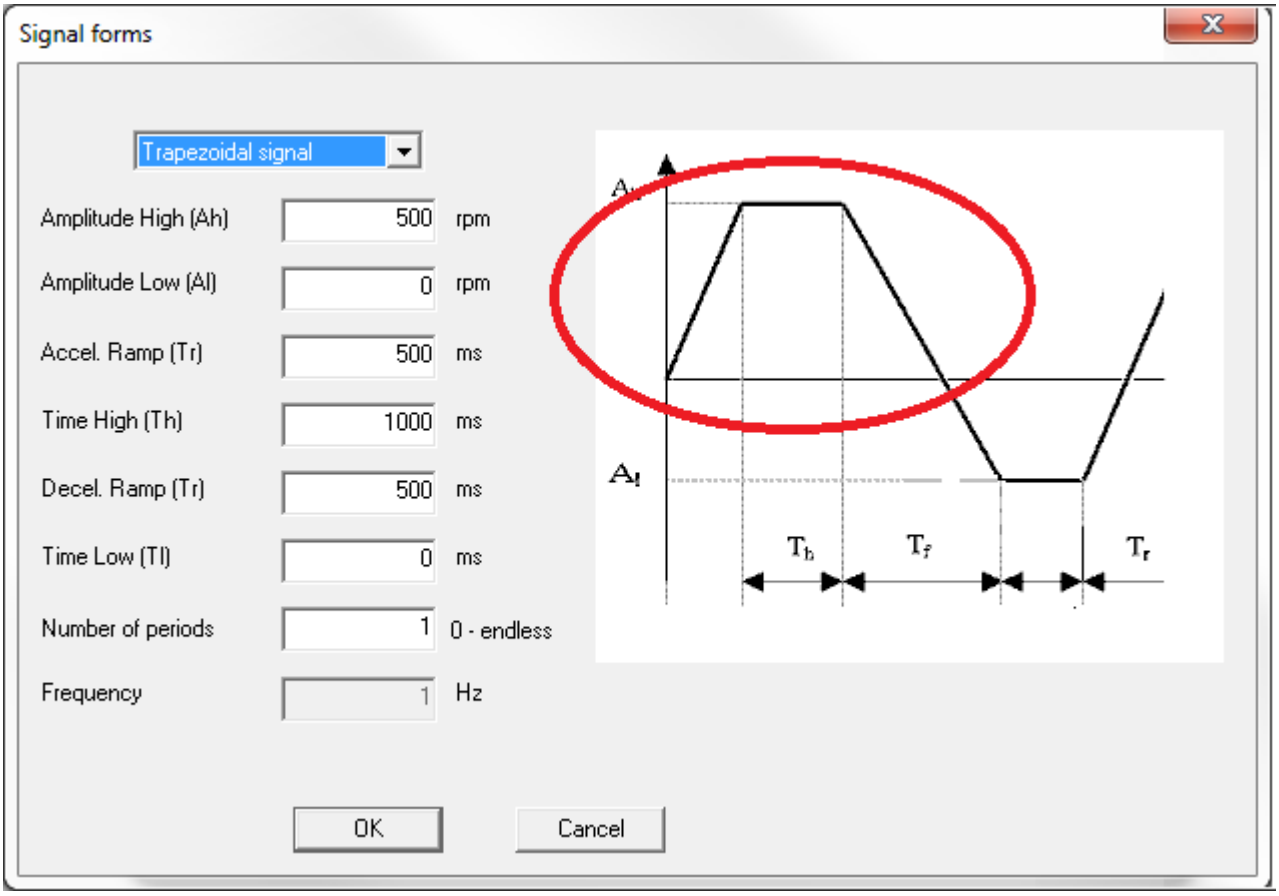
Open the configuration window 'Signal forms'

'Controller enable' RF must be withdrawn.



Example trapezoidal signal:

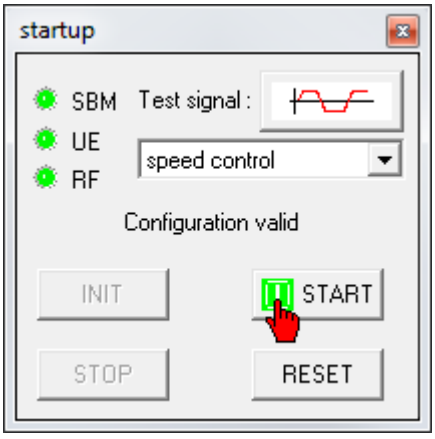
By means of the trapezoidal signal, you can set a positive and negative speed setpoint with acceleration and deceleration ramp. The example configures just a positive movement.



Now, activate the motor controller

The LEDs show the states of SBM, UE and RF (green = active)

The configured movement sequence is started by **'START'**.

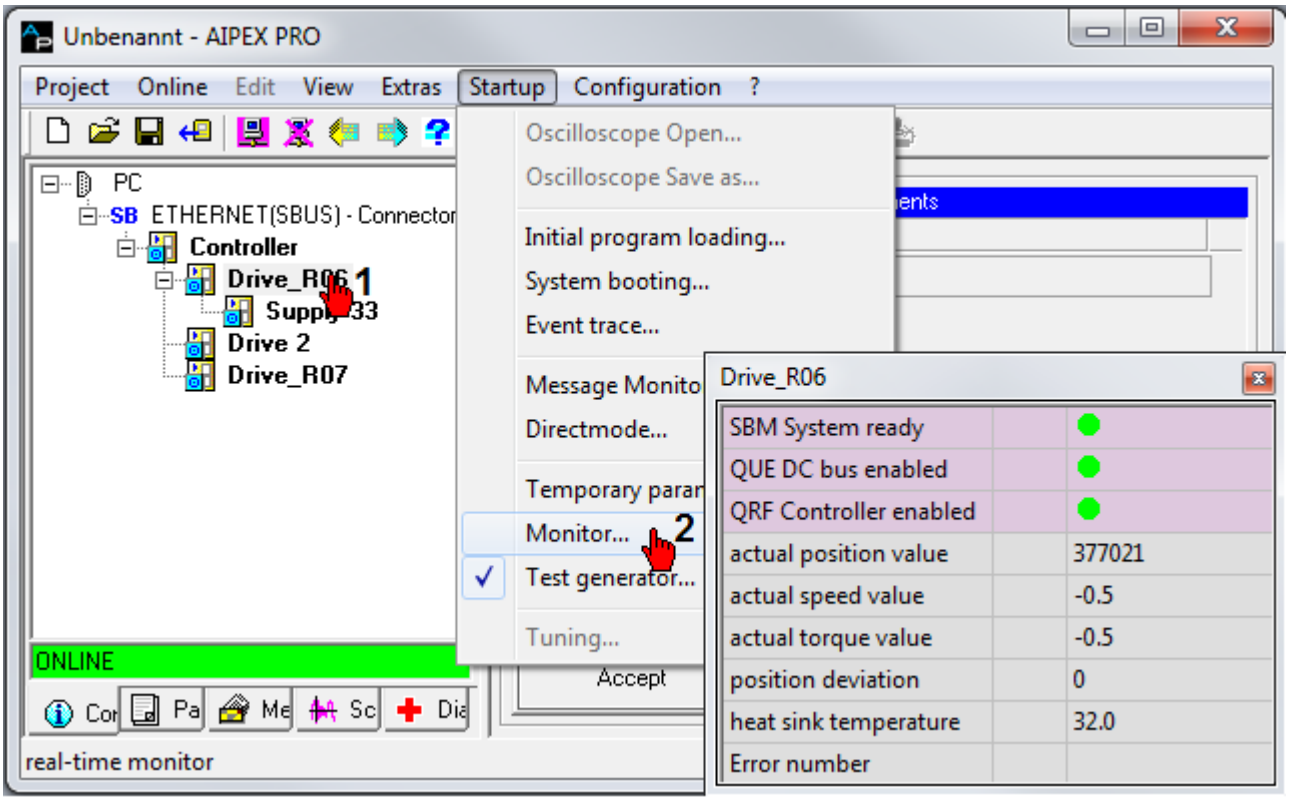


The movement can be interrupted by **'STOP'** and continued by **'START'**.

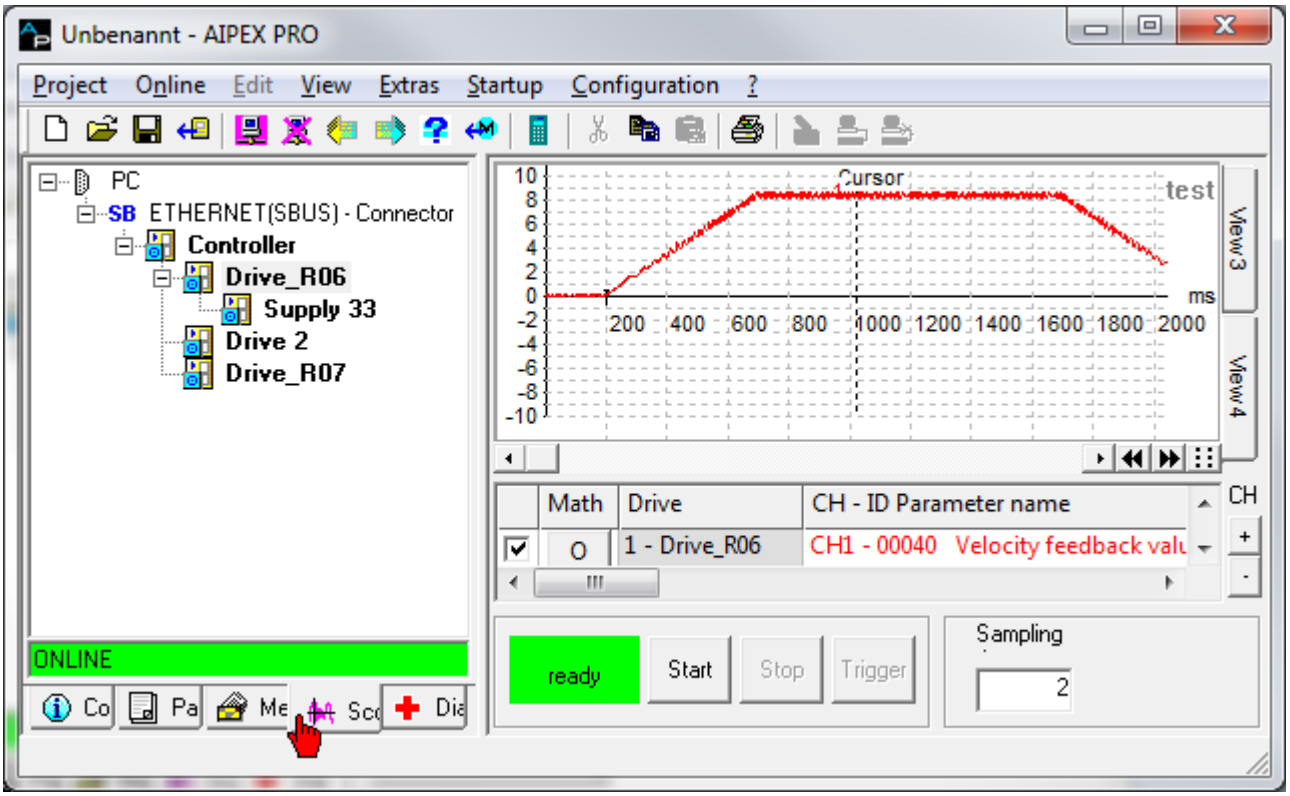
'STOP' and subsequent **'RESET'** terminates the sequence.

Display actual values

By means of **'Monitor...'**, you can display the controller state and actual values.
Check whether a positive setpoint causes a positive speed and an increasing position value. A positive rotation direction means clockwise rotation with view to the motor shaft



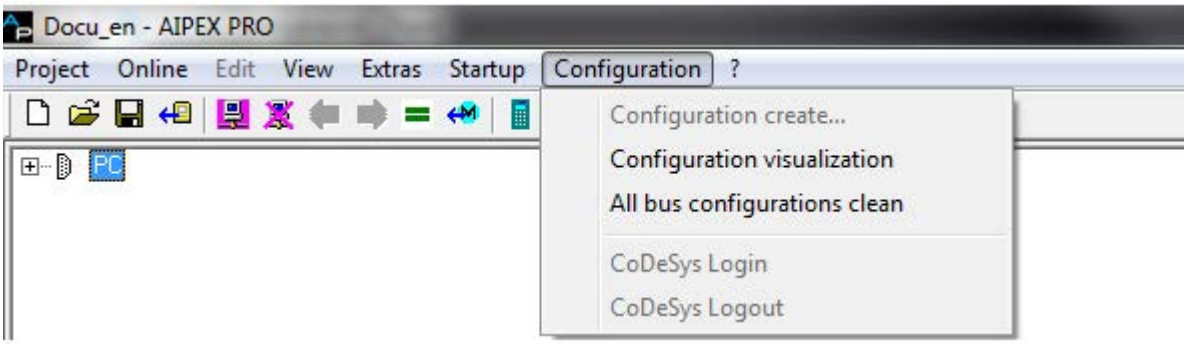
By means of the AIPEX PRO oscilloscope, you can store and interpret the movement.



Scope configuration help:

Siehe 'Scope - Manual' auf Seite 90.

3.6.6 Configuration



Configuration create...

Icon 

'Configuration create' automatically creates the message configuration between the devices and compiles the plc program. Additionally, device parameters are automatically adjusted to the function blocks used in the PLC editor. Siehe 'Project Settings - Configuration create (project specific)' auf Seite 108.

The link between the symbolic device names (variables) from the CODESYS programming system and the available physical devices must be manually created by 'drag and drop' (Siehe 'Automatic bus configuration' auf Seite 711.) or automatically with the AIPEX PRO function 'Import device name on creation of a new PLC project' (Siehe 'Configuration create' auf Seite 103.)

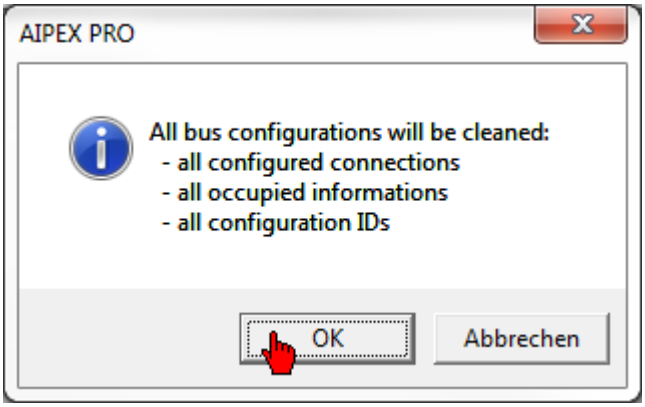
Configuration visualization

The function 'Configuration visualization' displays the network configuration with the PLC variables.

All bus configuration clean

The function 'All bus configuration clean' delete all links from all bus configurations, also the configuration IDs (34036 CCB-File, 1204 XML-File...) from all devices.

Restart after function call all devices.



CODESYS Login

Icon 

'CODESYS Login' connects AIPEX PRO with the PLC control and switches into the online mode.

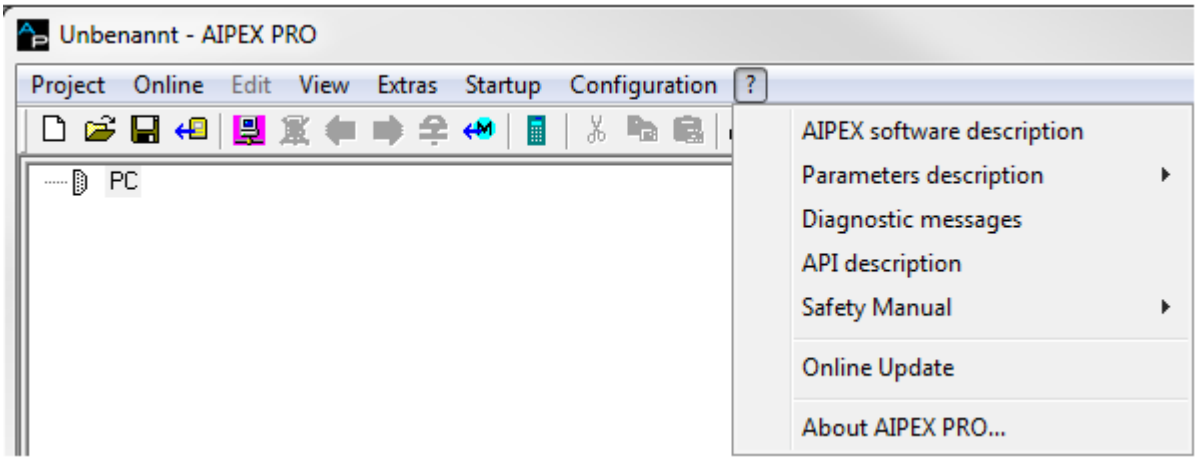
On obsolete configuration, the 'Configuration create' menu is opened automatically.

CODESYS Logout

Icon 

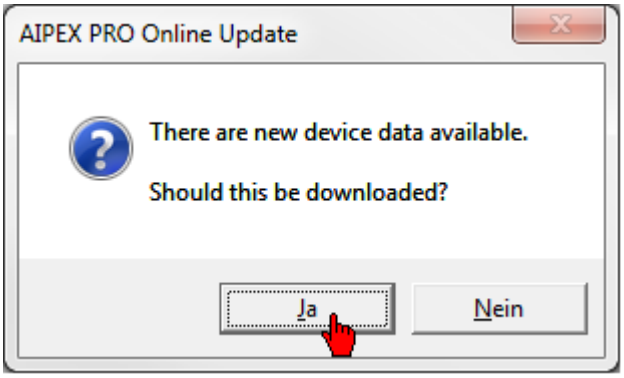
The connection to the controller is separated and switch is to the offline mode.

3.6.7 '?' - Documentations



If you install an 'AIPEX PRO service pack' or if you do a 'Online Update', the documentations will be automatically updated.
(The 'AIPEX PRO service pack can include several 'Online Updates')

Online Update
Starting the program there will be an automatic search for 'Online Updates' on AMK web site.
The download of an available Update must be confirmed.



Following files will be updated trough the function Online Update:

- ADB: AMK database, it contains information about all AMK parameters
- MDB: Motor database, it contains information about all AMK motor parameters
- Device file (e. g. for EtherCAT devices)
- Documentation

Requirements:

- Internet connection
- Administrator rights

For this update you must have administrator rights. If you start AIPEX PRO, the update files will be searched and found, but you are not able to install it.

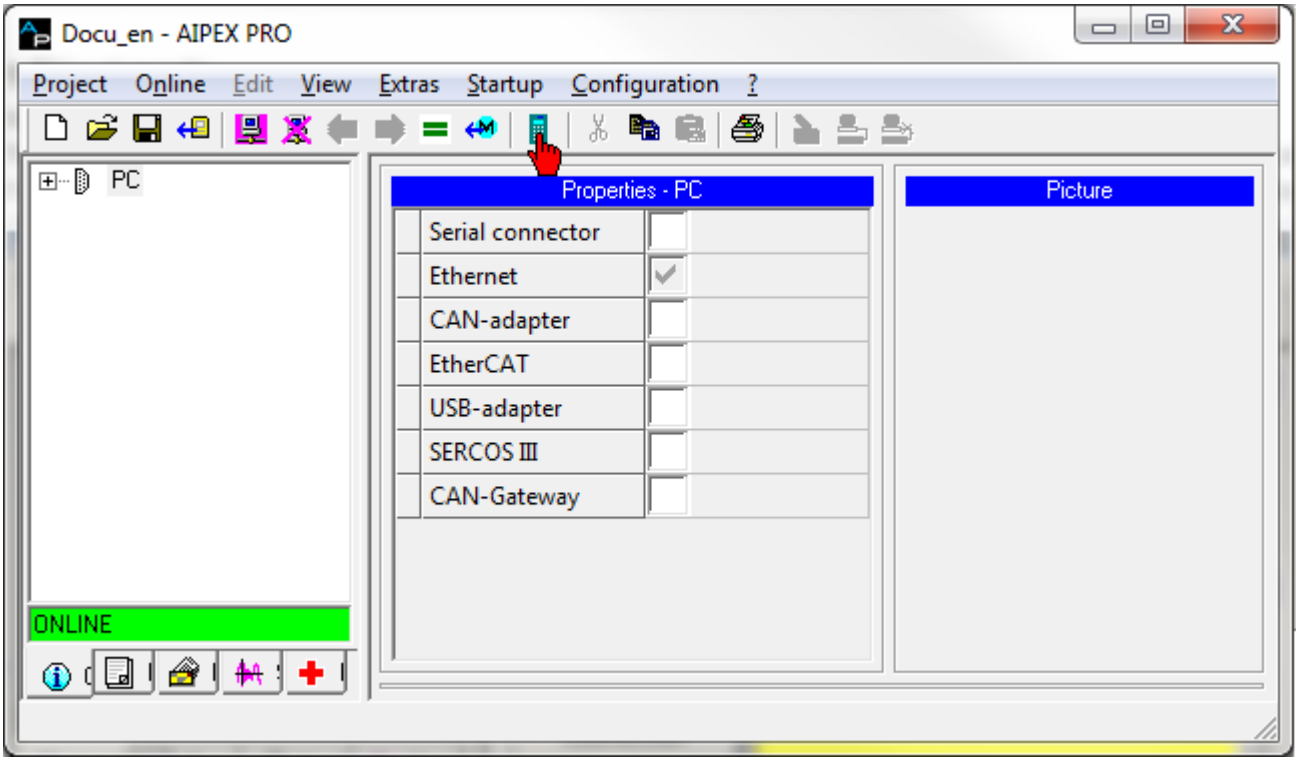
The ‘Online Update’ can be done automatically at program start. Siehe 'Data update' auf Seite 106.

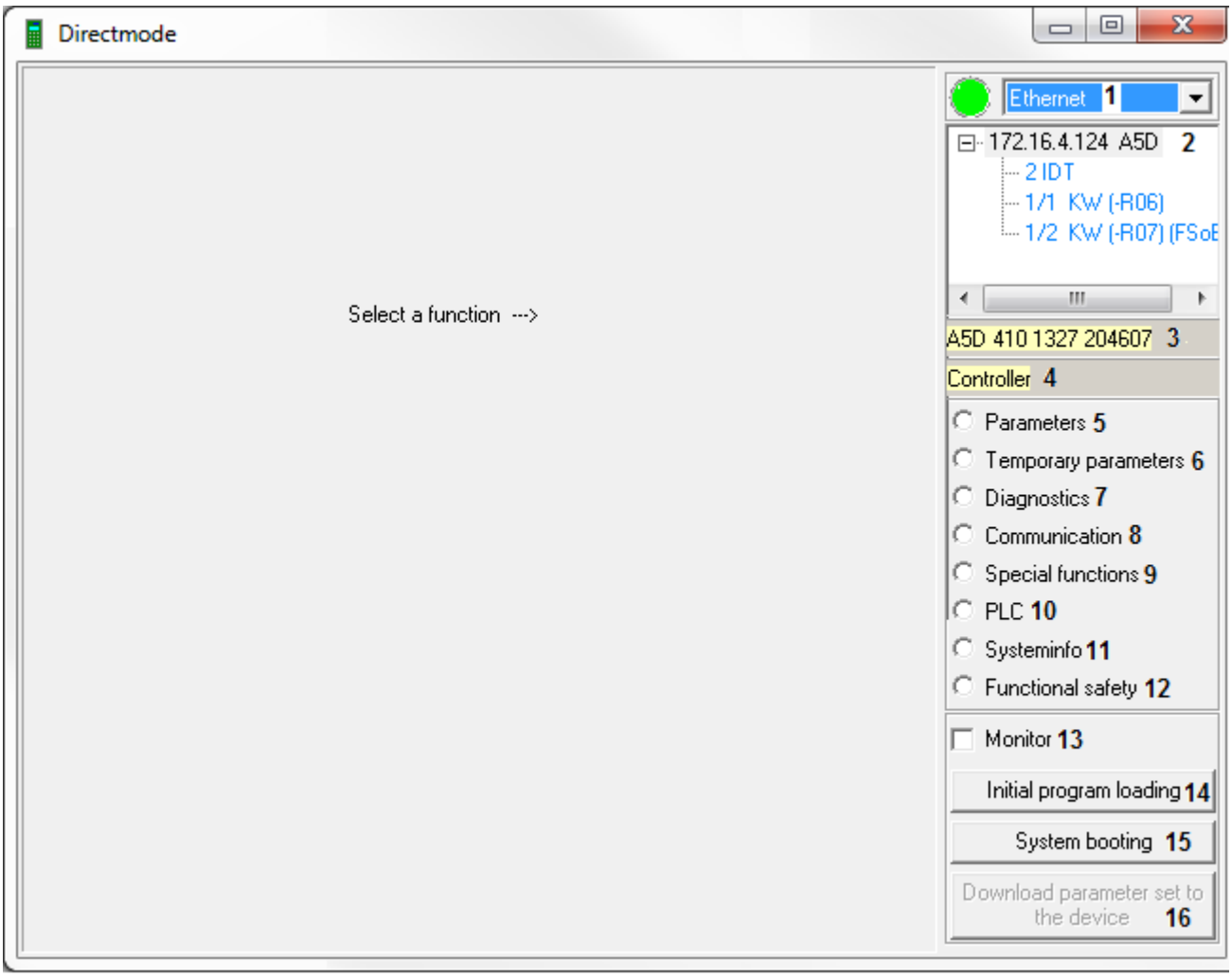
3.6.8 Direct mode

With 'Direct Mode' you get online access to the AMK devices.
The change will be immediately transferred to the device. In which time becomes active depends on the parameter attribute.

If you following want to transfer the online changing to the AIPEX PRO data set, you have to call the menu **'Online'** -> **'Upload parameter set'** . Siehe 'Online' auf Seite 96.

The 'Direct Mode' needs a active communication between PC and Device.
Siehe 'Communication PC - AMK device' auf Seite 43.



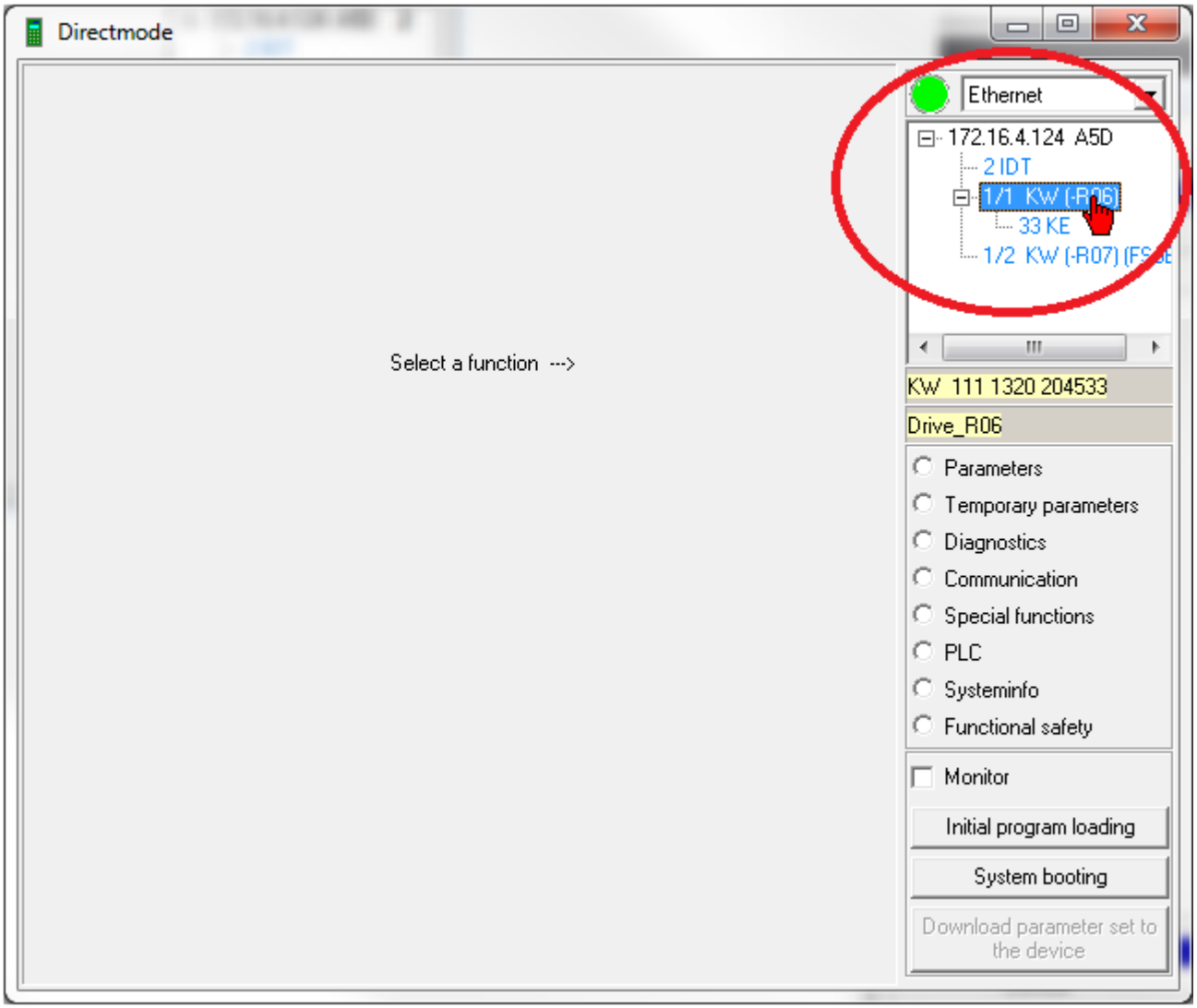


No.	Functionality
1	Activated interface between PC and device Green: Link to the device available Red: Link to the device not available
2	Display of online available devices
3	Firmware Version
4	System Name
5	Display and input possibility for parameter values
6	Display and input possibility for temporary parameter values
7	Display of diagnosis messages and the functions error clear
8	Display and input possibility for bus parameters
9	Set up functions
10	Display of PLC program information and the functions plc handling
11	Display of System information and additional system handling
12	Option 'Safety Functionality'
13	Cyclic display of actual online values
14	Function 'Initial program loading' The 'Initial program loading' function resets AMK devices into their initial status (delivery status). After a completed initial program loading, a system reset needs to be done. Prerequisites for the initial program loading: A direct connection between PC and AMK device, additional password input.

No.	Functionality
15	Function 'System booting' A 'System booting' is carried out on the selected device. A 'System booting' causes a recalculation of the data management. (Actual values are maintained, drive bus continues running...)
16	Function: Download Parameter set to the device

3.6.8.1 Activate the connection

In the following, read the explanation on how to access the connected AMK devices with 'Directmode'.



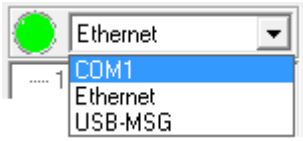
All active interfaces (connections between PC and AMK devices) are offered for selection.

Select the desired interface.

Status indicator:

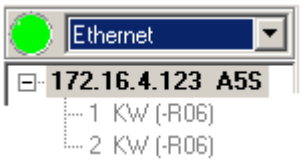
Green = Connection established

Red = Connection not established or interrupted



EtherCAT devices are displayed immediately.

For other drive buses, such as the ACC bus, you have to click on the corresponding Bus Master.

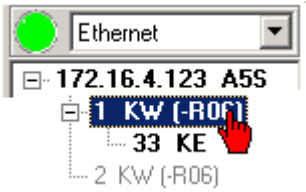


Status grey: Device data has not yet been read out.

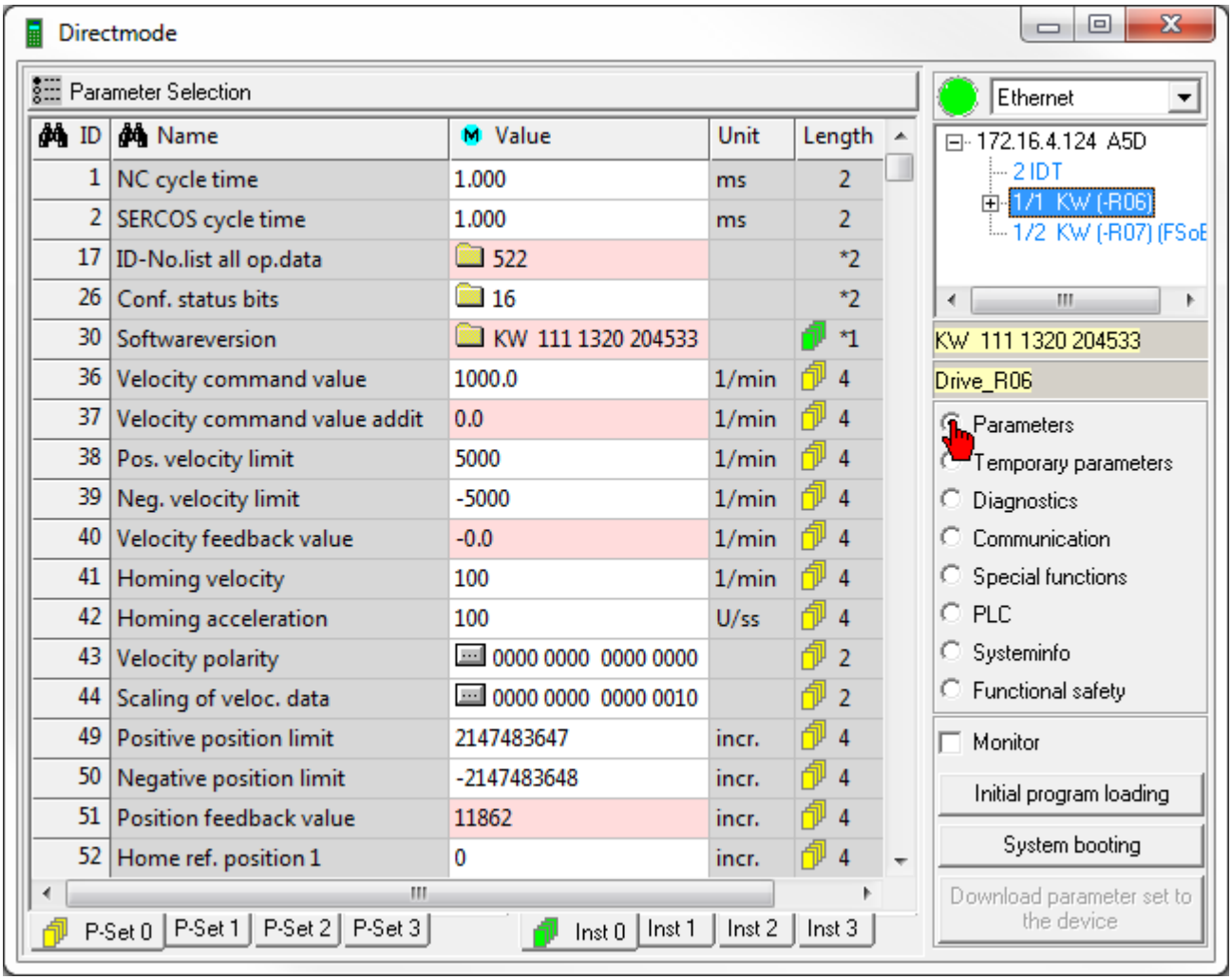
Status black: Device data has been read out and saved on the PC. By double-clicking a black device, the data is read out anew.

Example:

The supply module KE 33 is connected via ACC bus.

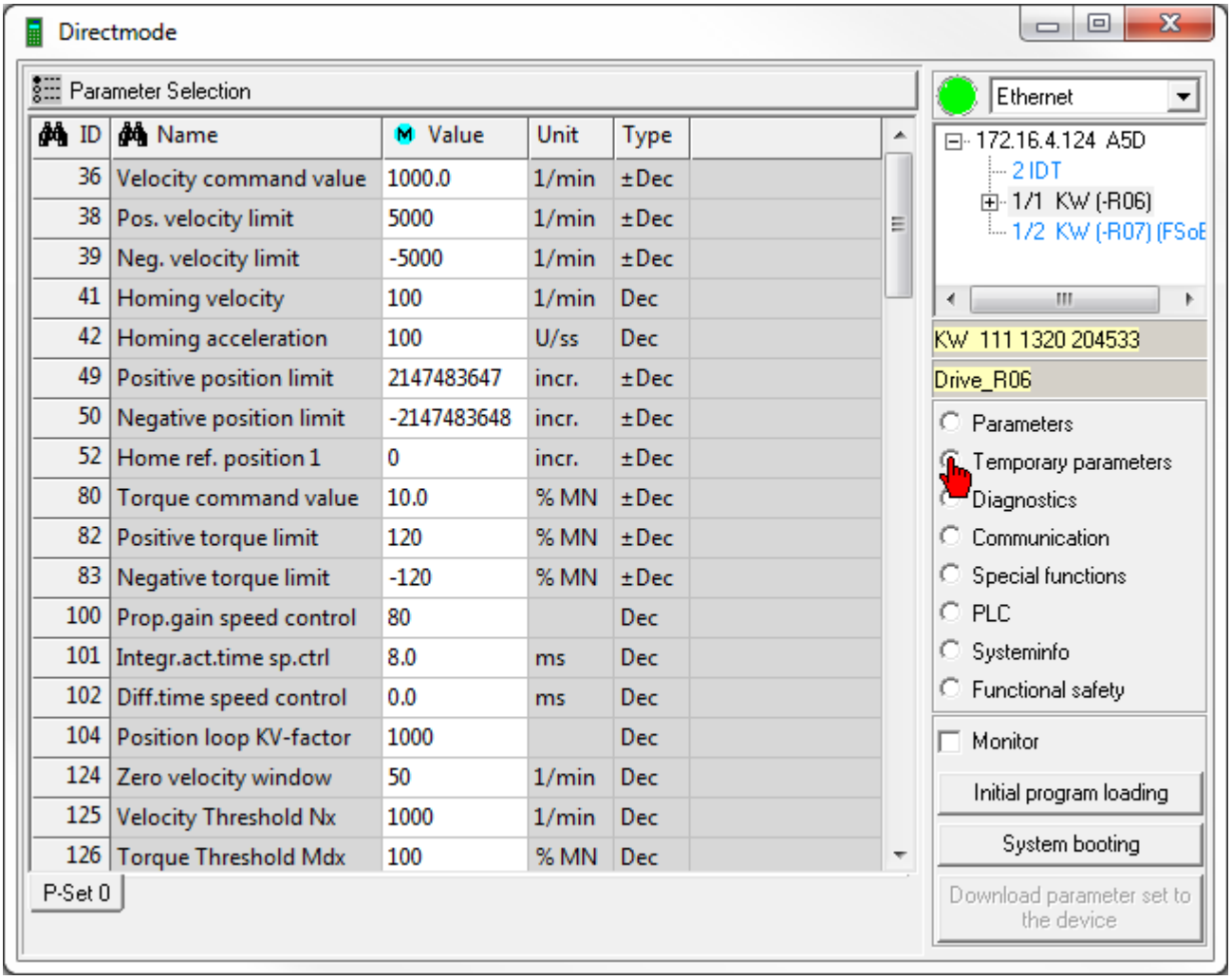


3.6.8.2 Parameter



'Parameter' displays available offline and online values of the device selected in the device tree.

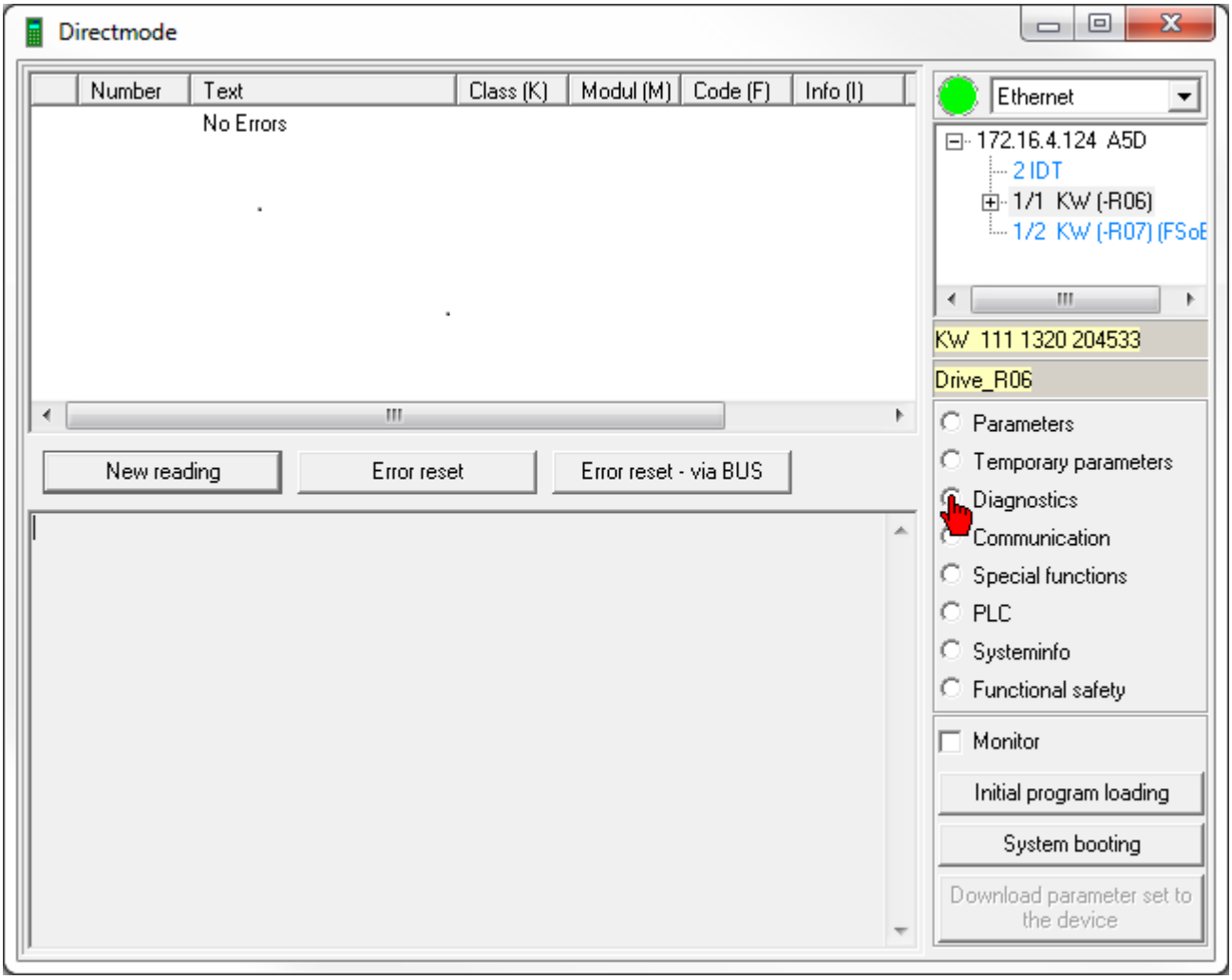
3.6.8.3 Temporary parameters



Changes in the 'Temporary parameters' are effective immediately in the drive.
In the selection list, only parameters are offered that can be modified temporarily.
All modified values are displayed in green font.

When the 'Temporary parameters' window is closed, you can specify whether or not the changes should be saved permanently.

3.6.8.4 Diagnostics



With 'Diagnostics', the diagnostic messages can be read out from the selected device.

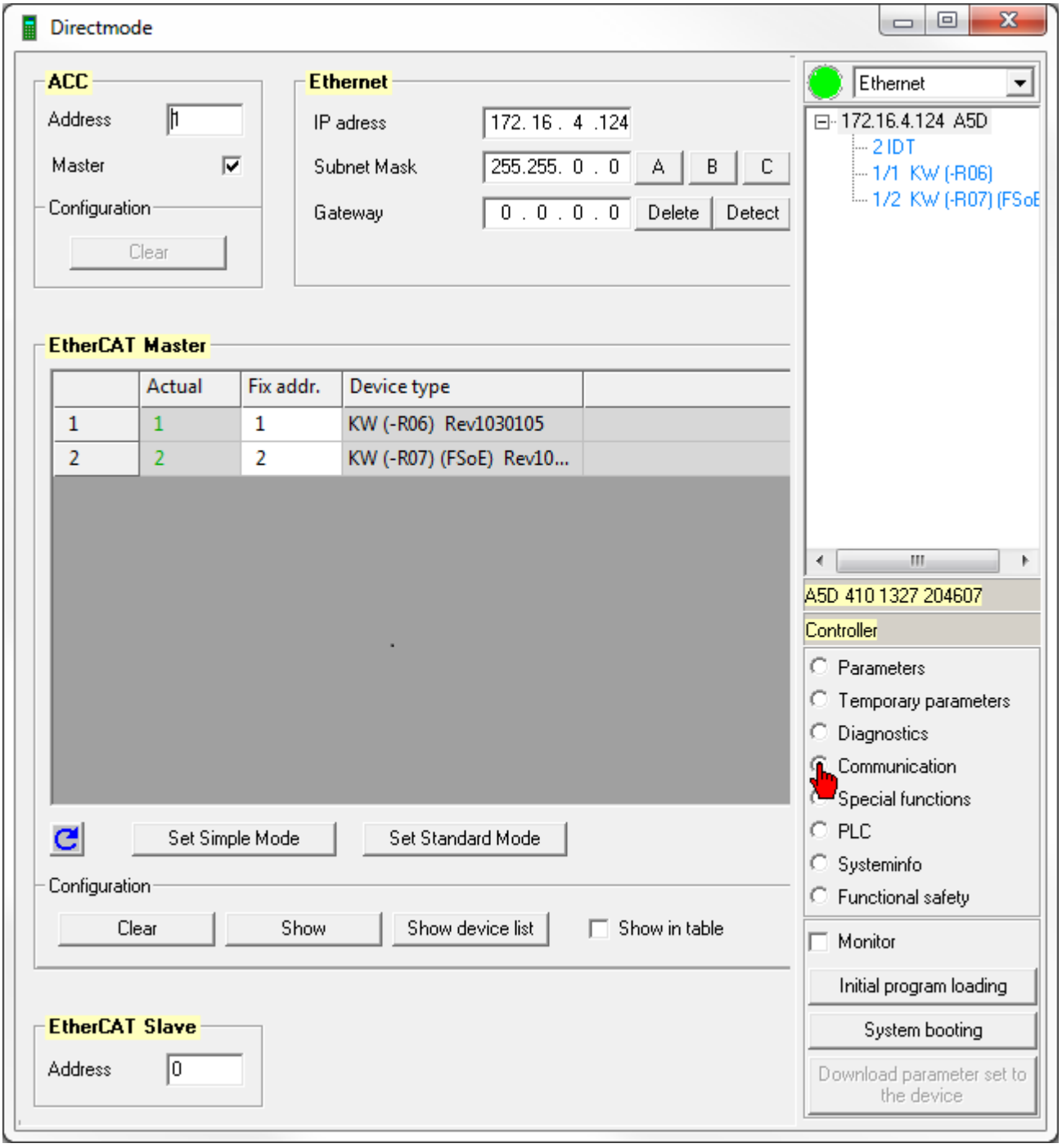
Click on each message to receive an explanation for it. You get further information if you analyse Info (I), I2 and I3
The first message of the list is the main activator of the fault; further displayed numbers might be resulting errors which will not appear any longer after rectifying the cause of the first diagnostic message.

Button **'New reading'**
Diagnostic messages will be read out from the selected device.

Button **'Error reset'**
The errors will be deleted in the selected device .

Button **'Error reset - via BUS'**
The errors of all devices of a bus line will be deleted. To do this, select the bus in the device tree.

3.6.8.5 Communication



ACC bus (instance 0)

Address: ACC bus device address (ID34023)

Master: Click in the checkbox to declare the device as ACC bus master. (ID34025)

Configuration: Button **'Clear'** delete the ACC bus configuration (ID34036)

Ethernet

IP address: Display and input box for the device IP address

Subnet mask:

Button **'A'** / Class A network / 255.0.0.0

Button **'B'** / Class B network / 255.255.0.0

Button **'C'** / Class C network / 255.255.255.0

Gateway:

Button **'Delete'**: Deactivates the current gateway setting.

Button **'Detect'**: Set the address of the current gateway of the PC in the Gateway setting of the selected device (this address is often, but not always, identical for both devices).

EtherCAT Master

Column 1 : Physical position after the EtherCAT master

Symbol  EtherCAT Bus status Operational (XML Configuration file valid)

Symbol  EtherCAT Bus status Operational (XML Configuration file not valid)

Column Actual: Current EtherCAT address

Column Fix addr.: Manual input of the EtherCAT address by the user (fixed address)

Column Given: Address information which are saved in the automatically generated XML network file

Column Device type: Device type and EtherCAT revision stand

Button **'Direction sign'**: Refresh the device list.

Button **Set Simple Mode**: Deletes the 'Fix' addresses in all slaves.

Button **Set Standard Mode** The 'Fix' slave addresses will be set to the actual Position value.

Configuration

Button **'Clear'**: Deletes the current EtherCAT configuration.

Button **'Show'** : Reads out and displays the current device configuration from the parameters (ID).

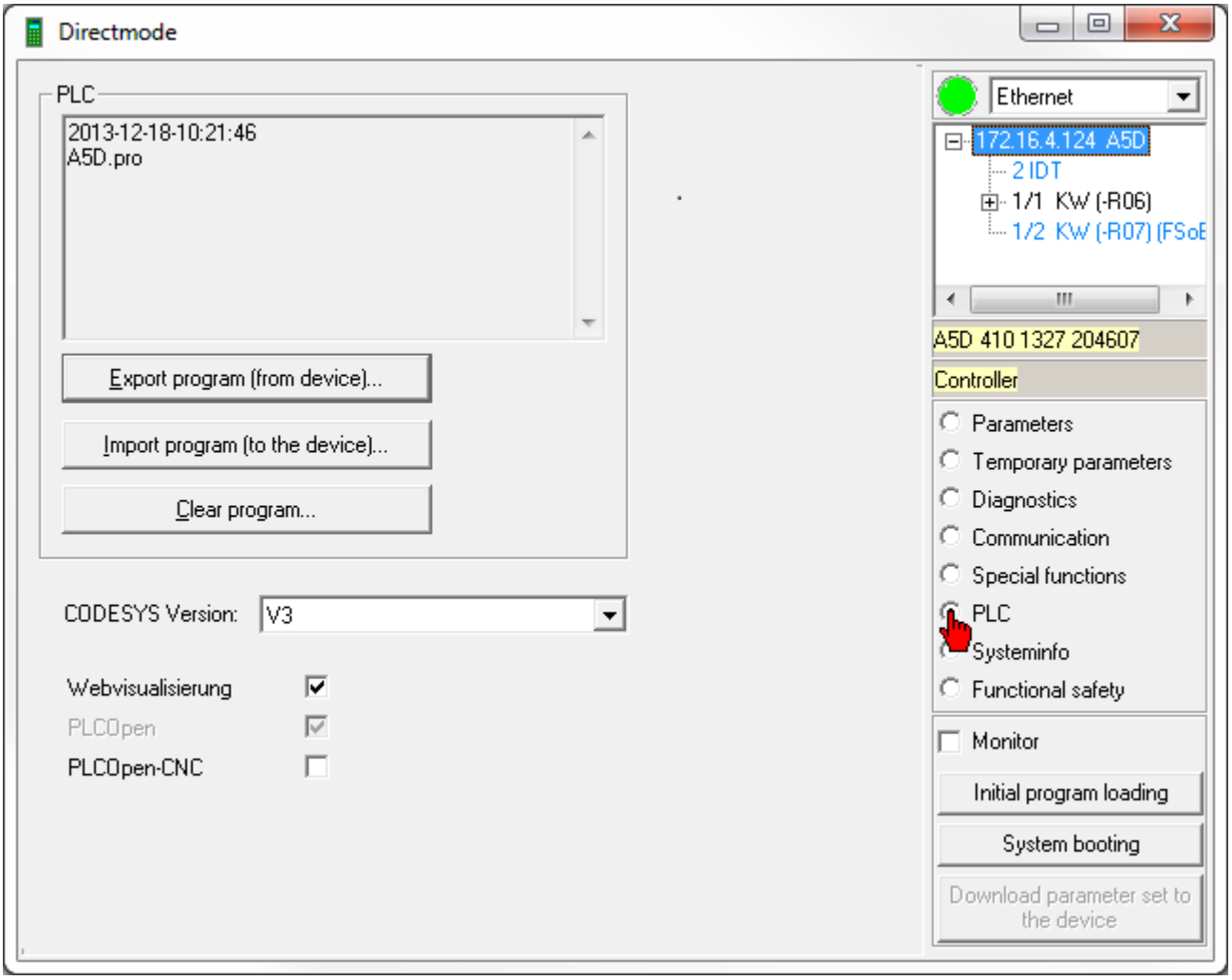
Button **'Show device list'**: Extracts the device list from the current device configuration and displays it in the editor.

Check box **'Show in table'**: Additional column 'Given' and 'Device type'.

EtherCAT Slave

Display and input box for the EtherCAT slave address.

3.6.8.6 PLC



PLC

Display of the ID34172 'PLC project info'. The following data can be entered and displayed using the CODESYS PLC editor (menu item **Project** -> **Project information**).

- Date
- Project name
- Title
- Version
- Author
- Comment

The function **'Import program (to the device)'**and **'Clear program'** are protected by AMK service password.
The PLC program is saved in ID34159 'PLC files'.

Button **'Export program (from device)'**
The PLC program is read by the selected device and saved in a freely selectable directory on the PC hard disk.
(A file with the suffix *.bin is created)

Button **'Import program (to the device)'**
The PLC program is read from a file (PC hard disk) and written into the selected device.
Only *.bin PLC files exported by AIPEX PRO can be imported.

Button **'Clear program'**

The PLC program will be deleted in the selected device.

CODESYS Version

Choose the CODESYS Version which should be used later at the controller.

The A4 and A5 controller default setting is CODESYS V2.
The A6 and iSA controller default setting is CODESYS V3.

Controller options

Valid for CODESYS V3.

The installed controller options at factory setting can be disabled and enabled by using the checkboxes.

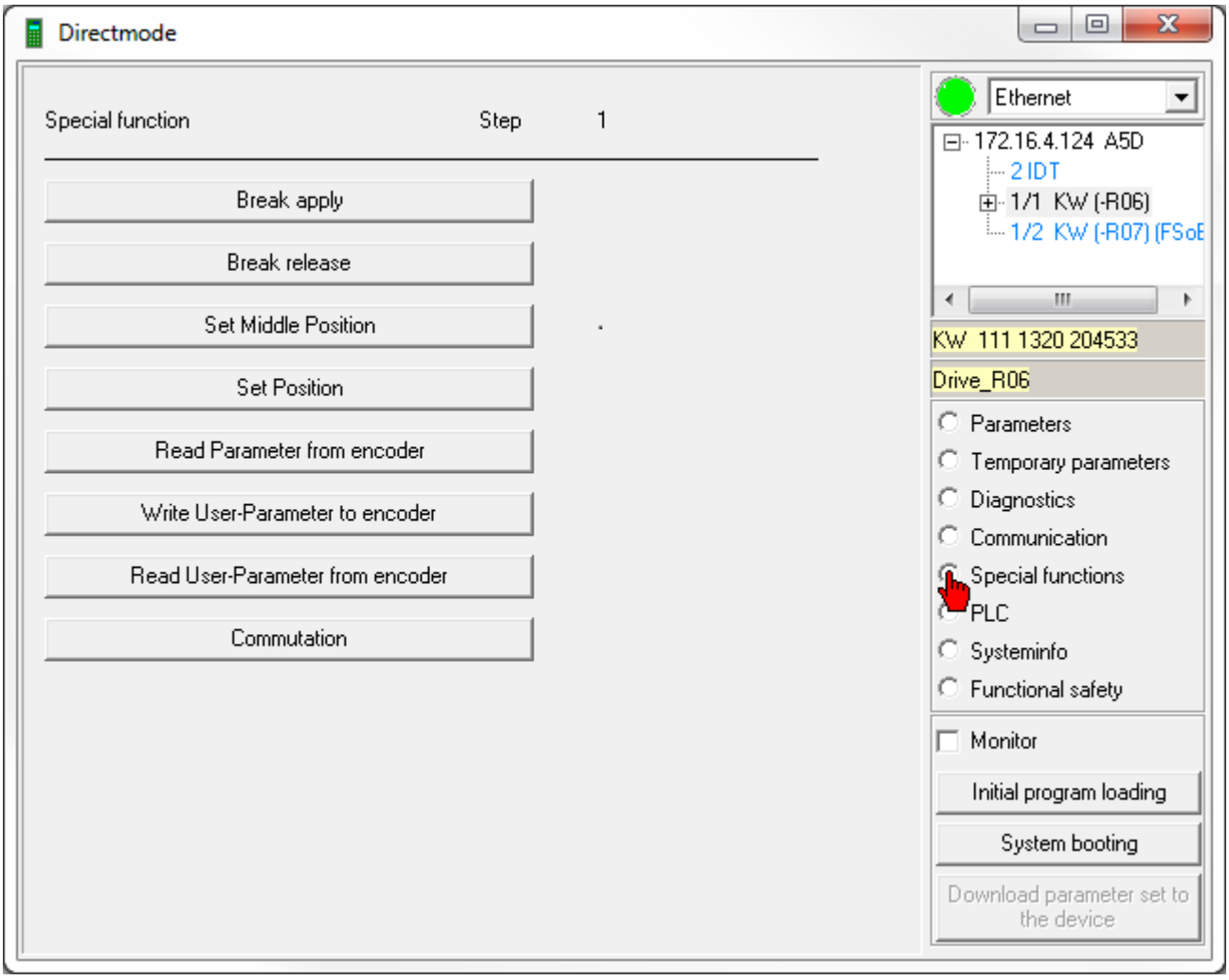
After dis- or enabling the options, the controller must be restarted (24 VDC OFF / ON).

Example:

For an ISA controller with additional option PCO (PLCopen) and option VIS (visualization) uses the target system: ArmPLCopenControlWithVisu V3.

After disabling the checkbox 'PLCopen' the device will be a iSA controller with additional option VIS (visualization) with target system: ArmControlWithVisu V3.

3.6.8.7 Special functions



Button **'Brake apply'**

Motor holding brake will be closed

For converters with the option "power output stage enable EF" or "STO" the brake can only be controlled if the power output stage enable is set.

Button **'Brake release'**

⚠ DANGER	
	**Risk of injury from hanging axes** The optional motor brake is a holding brake and does NOT provide sufficient protection for persons. Hanging axes can fall and lead to severe injury. **Steps to prevent:** • All hanging axes must be mechanically secured against falling with a fall arrester or a supplementary external brake, for instance. • People must not stand under hanging loads

Motor holding brake will be opened.

For converters with the option "power output stage enable EF" or "STO" the brake can only be controlled if the power output stage enable is set.

Button **'Set Middle Position'**

Reserved for AMK internal use!

Button **'Set Position'**

Reserved for AMK internal use!

Button **'Read parameter from encoder'**

Drive must not be energized!
The service command must not be executed if the drive is in control mode. Controller enable must be off (RF = 0).

The parameter values set via ID32841 'Encoder list motor' are read from the absolute encoder memory and stored in the current parameter set.

Button **'Write User-Parameter to encoder'**

Drive must not be energized!
The service command must not be executed if the drive is in control mode. Controller enable must be off (RF = 0).

Drive must stop!
The service command may only be executed if the drive is at a standstill. Caution in the case of motors are on external forces, eg hanging axes. If external forces affect an axis, the user must ensure that the axis can not move while the service command is active.

The current values of the parameters entered in ID32842 'Encoder list customer' are written in the absolute encoder memory.

After the parameters have been written in the encoder, mains off / on must be carried out.

Button **'Read User-Parameter from encoder'**

Drive must stop!

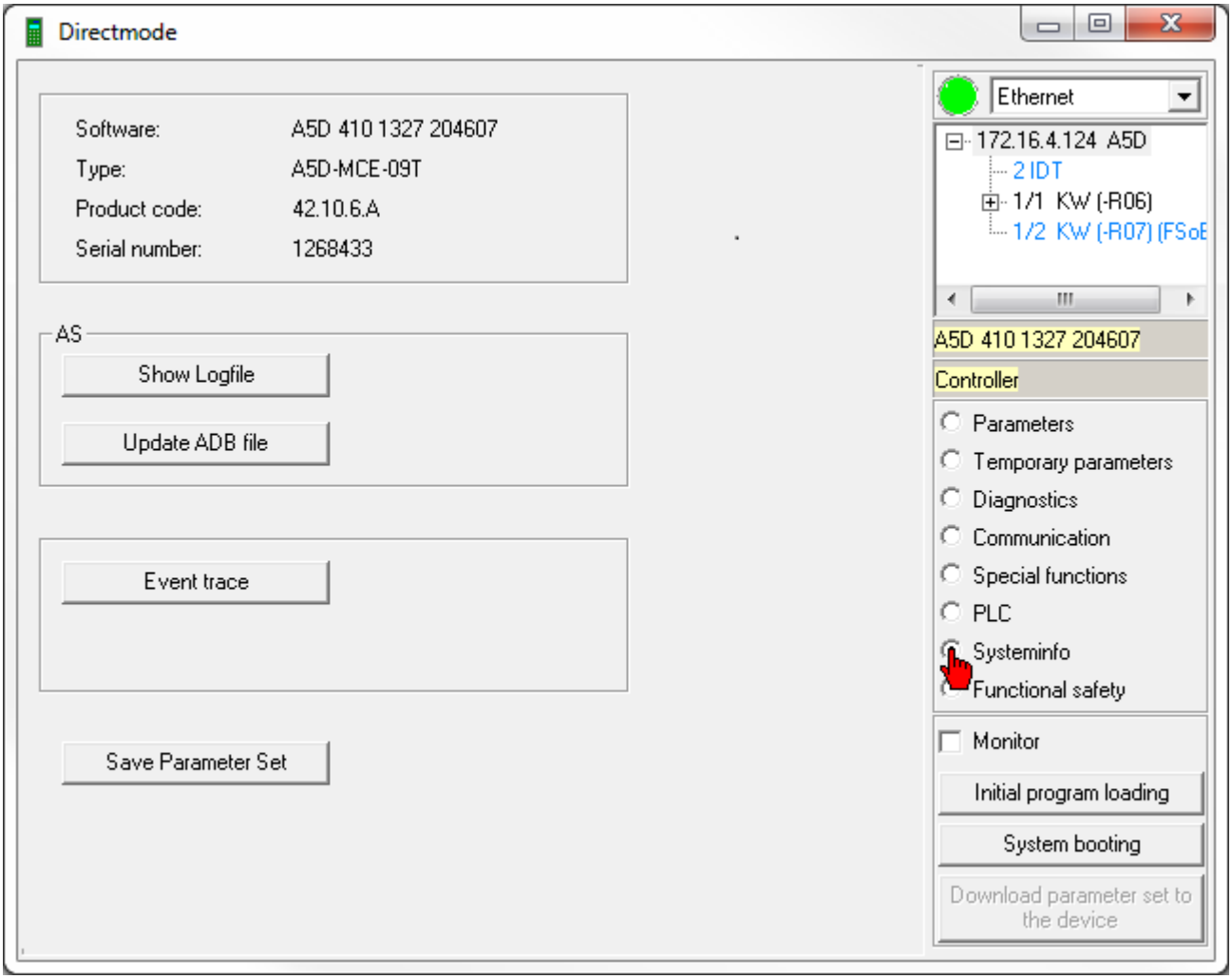
The service command may only be executed if the drive is at a standstill. Caution in the case of motors are on external forces, eg hanging axes. If external forces affect an axis, the user must ensure that the axis can not move while the service command is active.

The parameter values set via ID32842 'Encoder list customer'are read from the absolute encoder memory and stored in the current parameter set.

Button **'Commutation'**

Reserved for AMK internal use!

3.6.8.8 System info



Systeminfo

Following information will be displayed for the selected device:

- Software
- Type
- Product code
- Serial number

AS - AMKAMAC controller

Button **'Show Logfile'**

The Logfile of the currently selected controller will be displayed.

Button **'Update ADB file'**

The ADB file of the currently selected controller will be updated.

Button **'Event trace'**

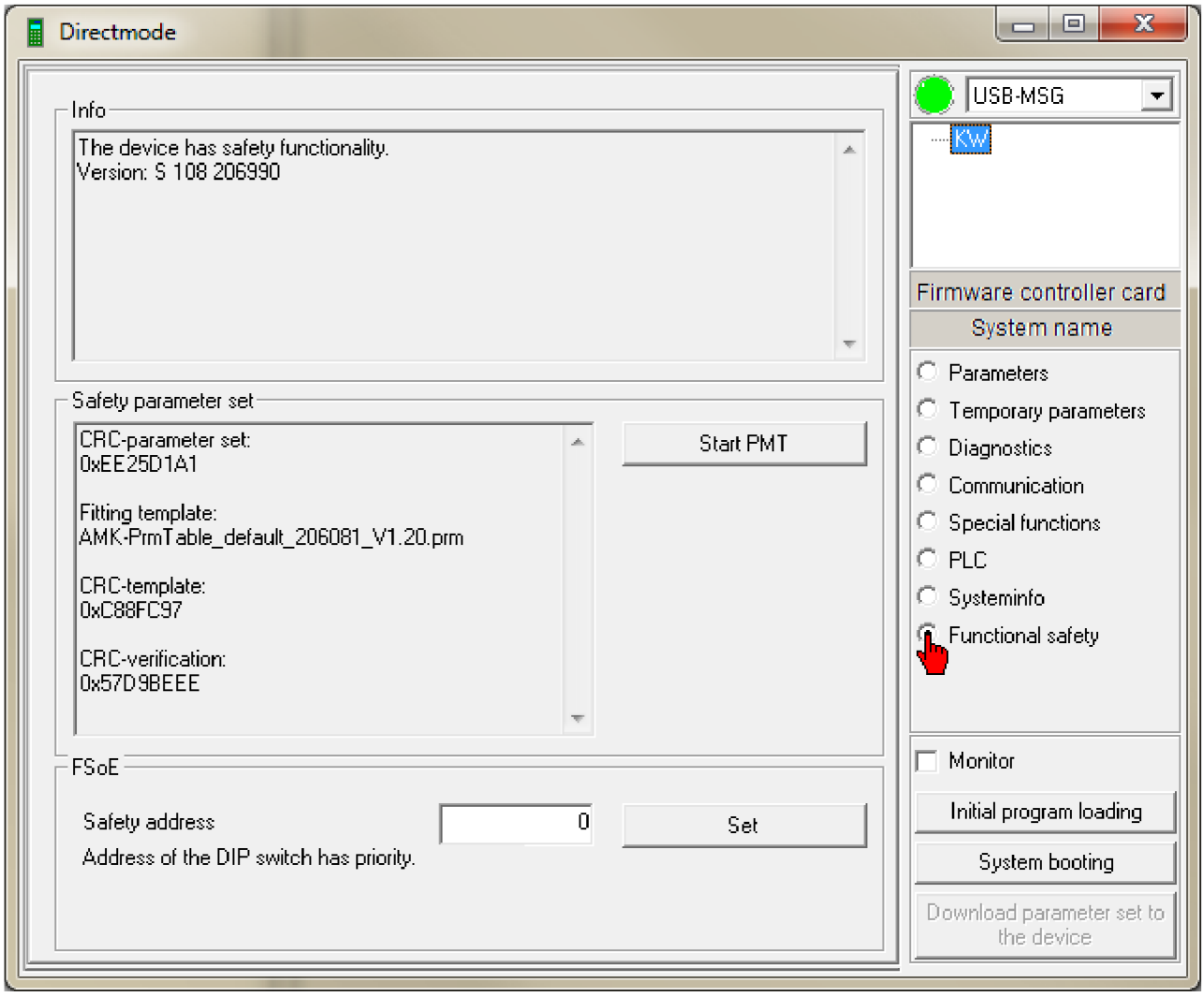
The ID34088 'Event trace' of the currently selected device will be displayed.

Button **'Save Parameter Set'**

The parameter set of the currently selected device will be saved to the PC.

You can reopen the created file with the Software AIPEX PRO to display the parameter set.
It is not possible to download the saved *.apu parameter set to a device.

3.6.8.9 Functional safety



Window content 'Info'

The field displays the status of the connection, and also the information if the linked device has a safety board or not.
Version: Firmware safety board
Matching template device version (firmware safety board)

Window content 'Safety parameter set'
Displays the safe parameter set check sum (CRC) and the used template.

CRC-parameter set: Check sum (CRC) customer specific safe parameter set
Fitting template: Matching template device version (firmware safety board)
CRC-template: Check sum (CRC) template
CRC-verification: Check sum (CRC) customer specific safe parameter set + serial number device

The button **'Start PMT'** calls the software SafePMT (Safe parameter editor).

FSoE
The entered FSoE adress transfer with the button **'Set'** in die ID33201 'Safety address'.

The address setting by DIP switch is prior to addresses via parameter and can not be over written.

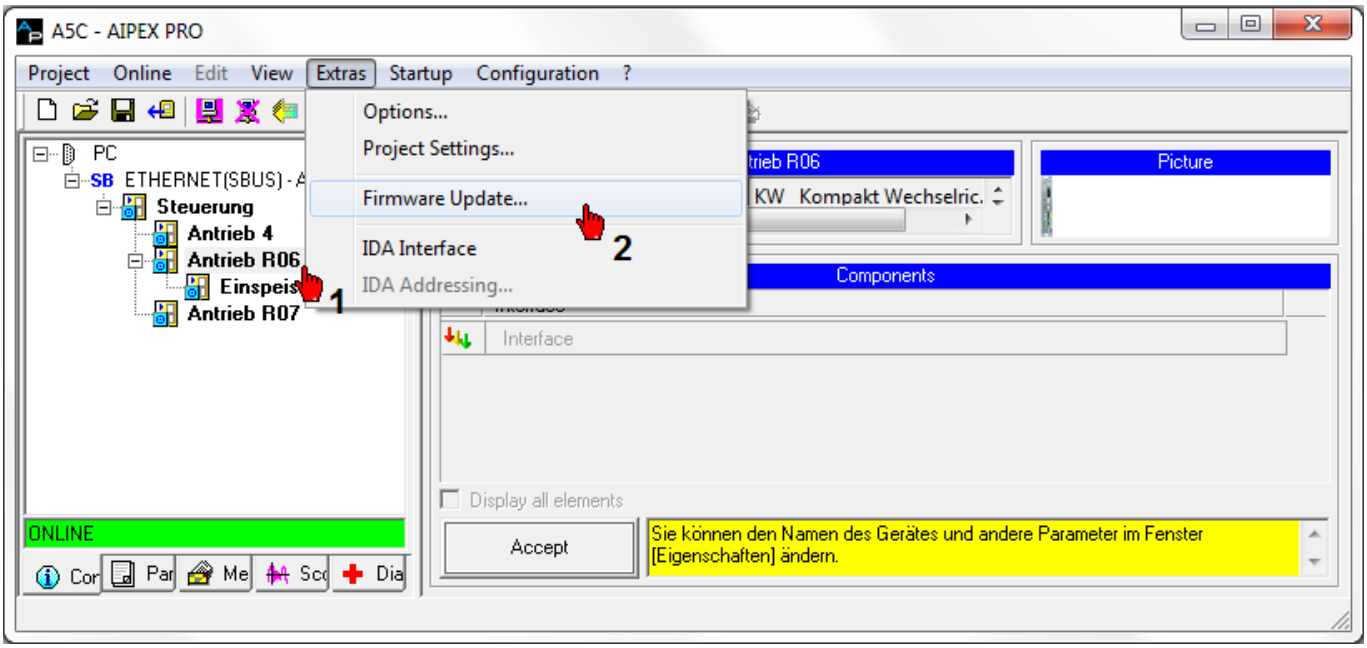
During firmware update or initial program loading, the address parameter ID332201 will be reset to 0.

3.7 Functions

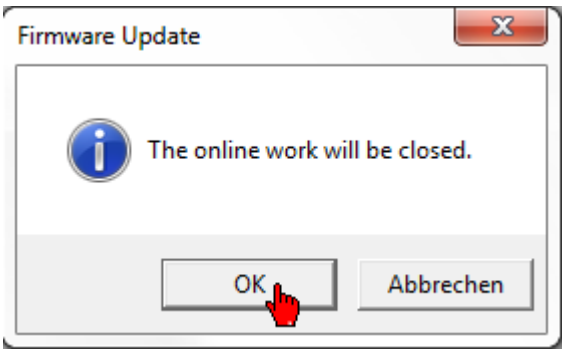
3.7.1 Updating the device firmware

The firmware serves as the operating system for the controller and option cards as well as for the PLC controllers.
The current firmware version can be read, for example, from the ID30 'Software version'.
The function **'Firmware Update'** is primarily intended for updating devices with new firmware from an existing project.
The function cannot be used without a project. In this case, generate a new AIPEX PRO project by logging in and copying the device data.
The firmware files are not integrated into AIPEX PRO. These need to be imported in a one-off process. They receive the firmware files from AMK.

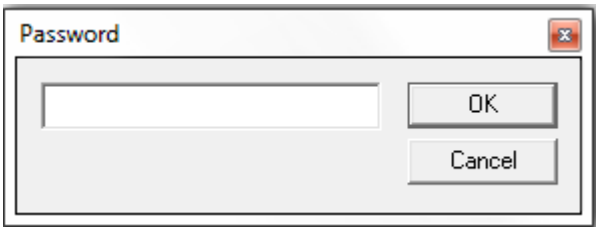
Select the device that you wish to update from the device tree.
Start the update procedure by selecting **'Extras'** → **'Firmware Update'**



If you are online, confirm with **'OK'**.

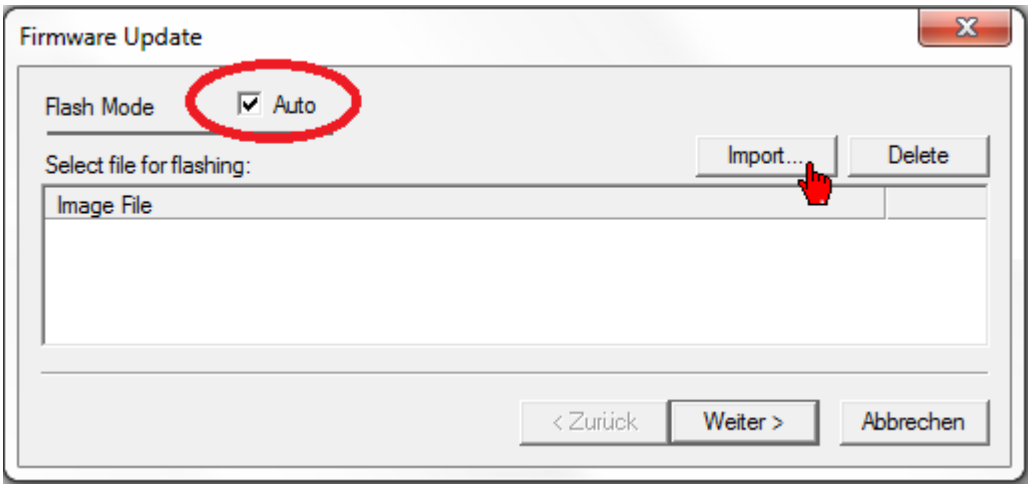


This function is password-protected.
AMK provides the password to authorized persons.

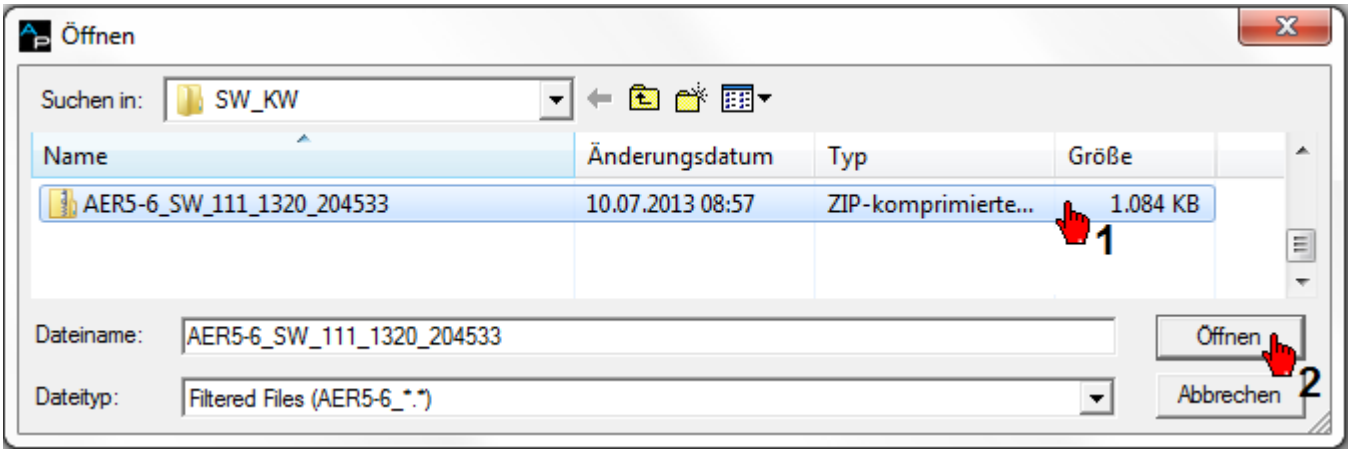


'Flash Mode Auto' enabled. The device automatically switches to Flash Mode.
'Flash Mode Auto' disabled. The device must be manually switched to Flash Mode.

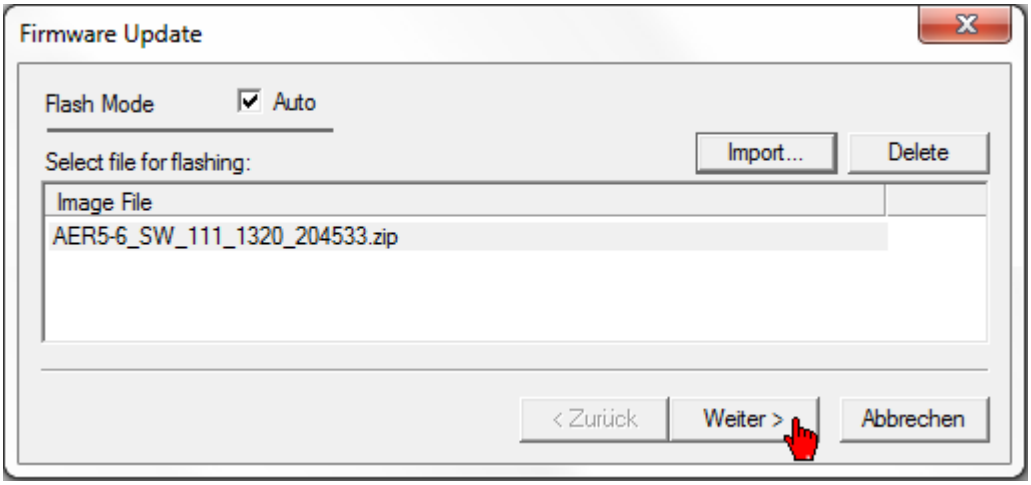
You can import the new firmware files by pressing the **'Import'** .



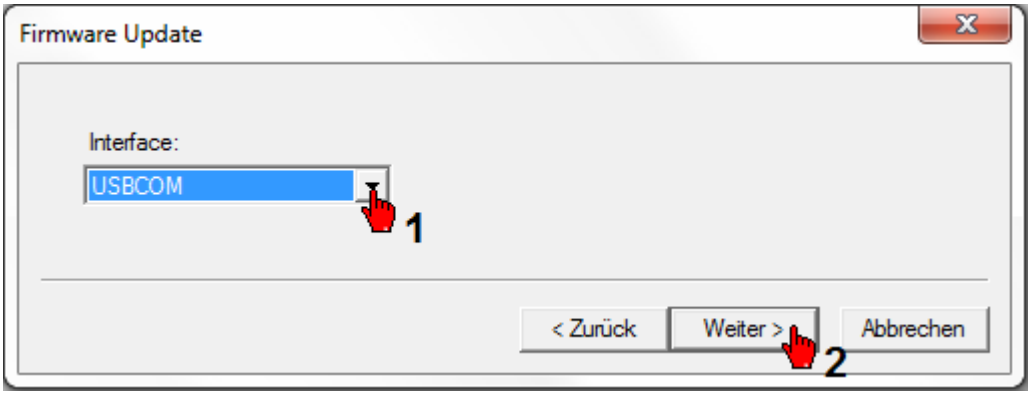
Once a file has been imported, it is saved under 'Image File' and can be selected immediately.
Select the firmware file. Confirm by pressing **'Open'**.



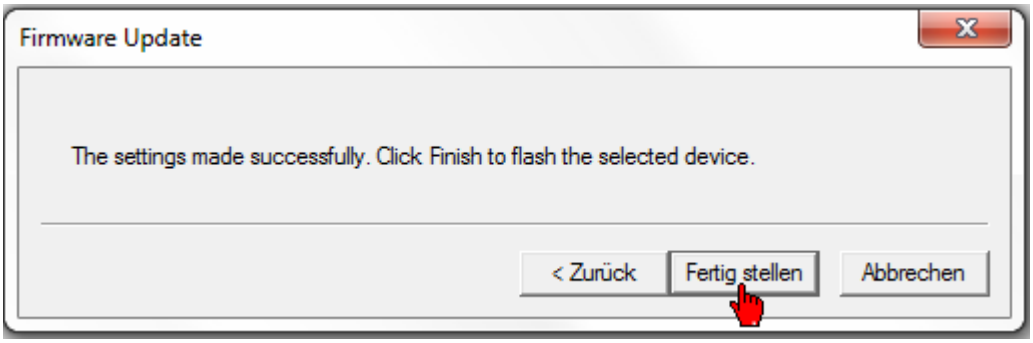
Confirm by pressing **'Next'**.



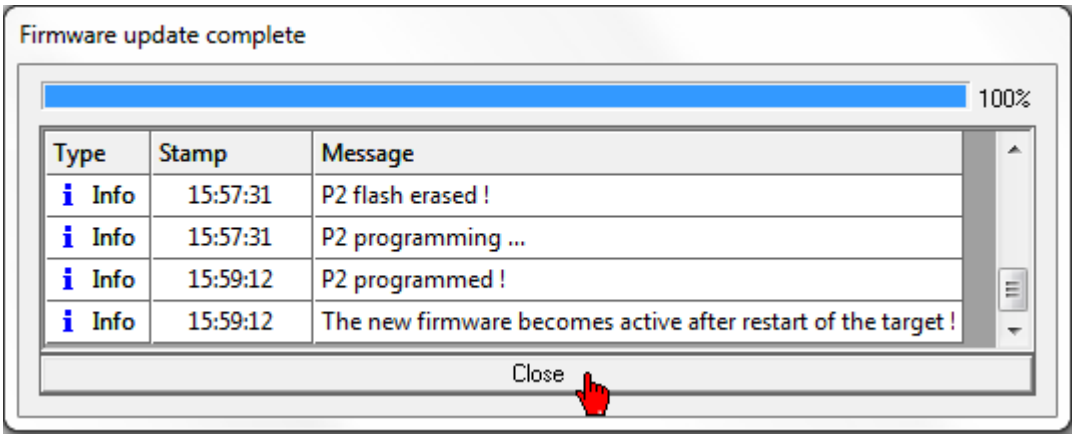
Establish a direct connection between the PC and device. E.g. via a USB or EtherCAT connection.
Select the interface used.



Press **'Finish'** to start the firmware update.



Progress bar



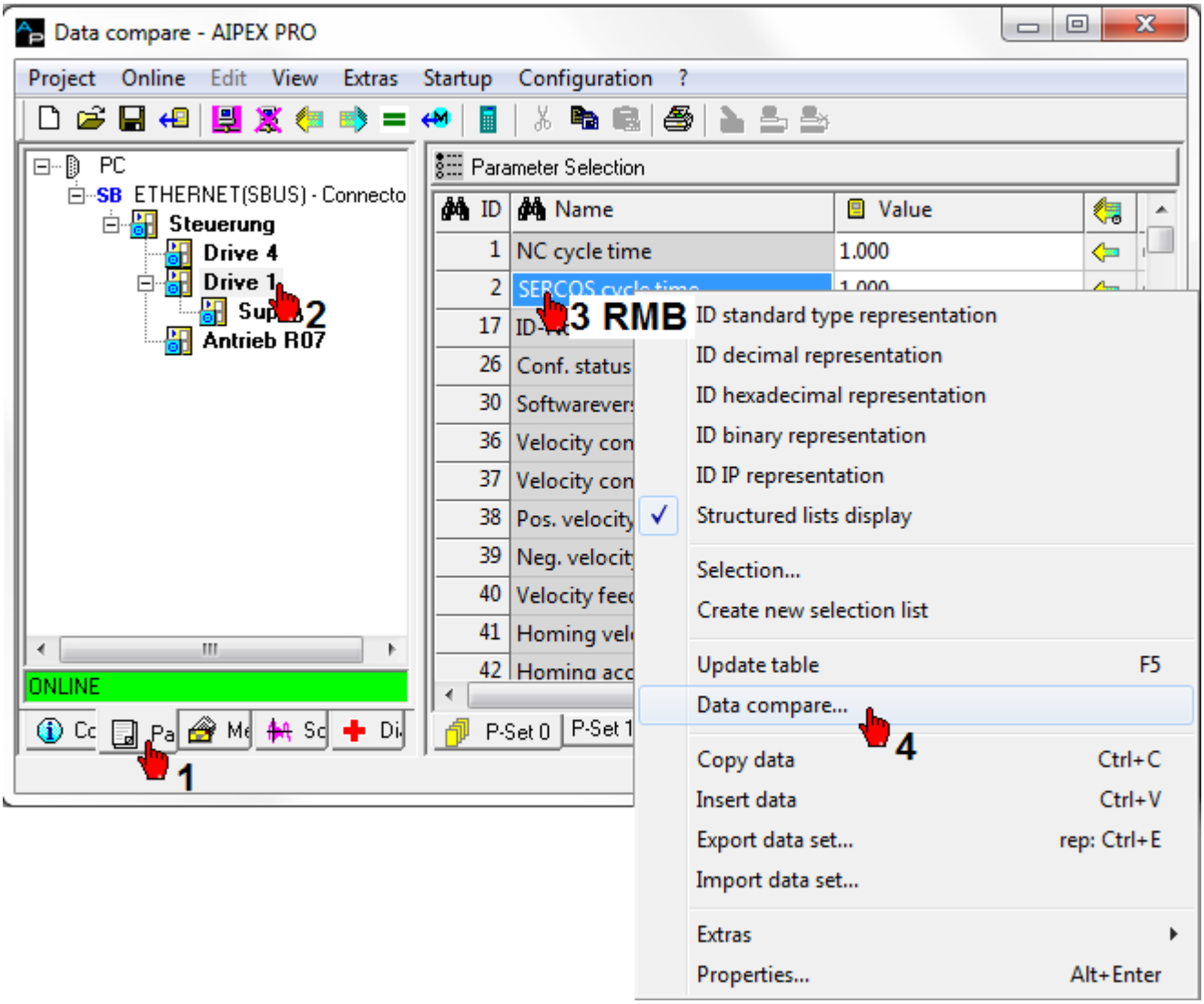
The new firmware update is first activated after restarting the device!

Depending on the new firmware version, 'initial program loading' is then necessary.

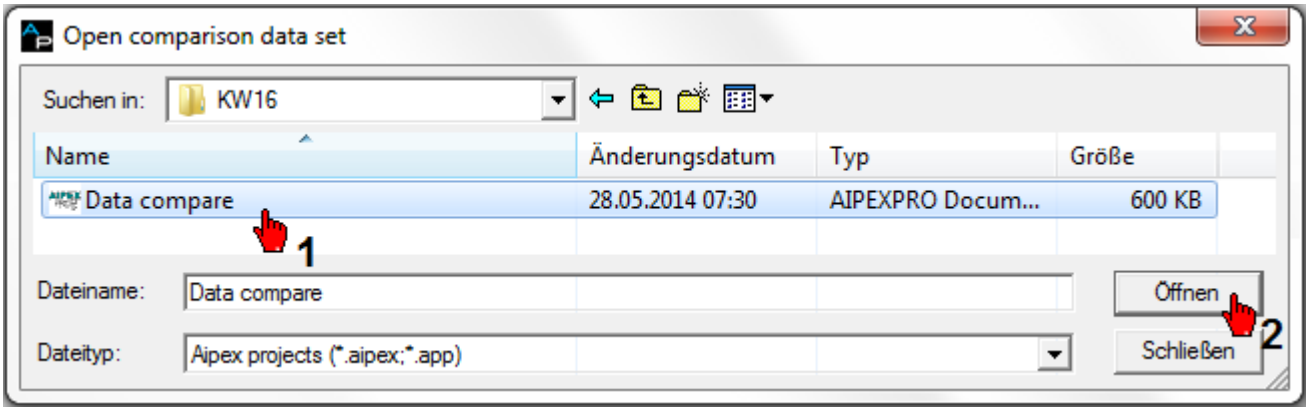
3.7.2 Data compare

With the Data compare function, you can compare the content of the currently displayed parameter set with the content of a data set already saved.

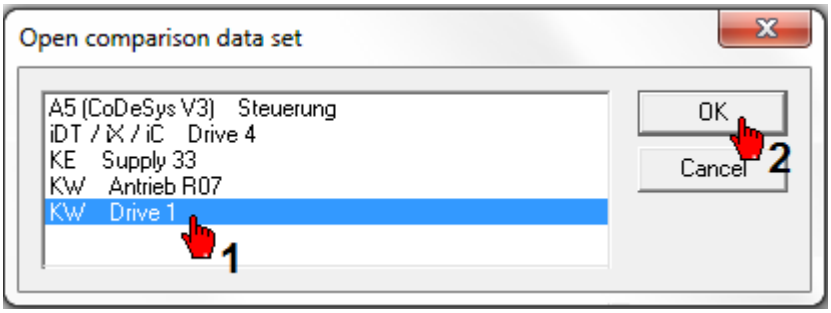
Select the **'Parameter'**tab. By selecting any parameter with the right mouse button, the context menu opens. Select **'Data compare'**.



Select the folder and comparative data set. Confirm by pressing **'Open'**.

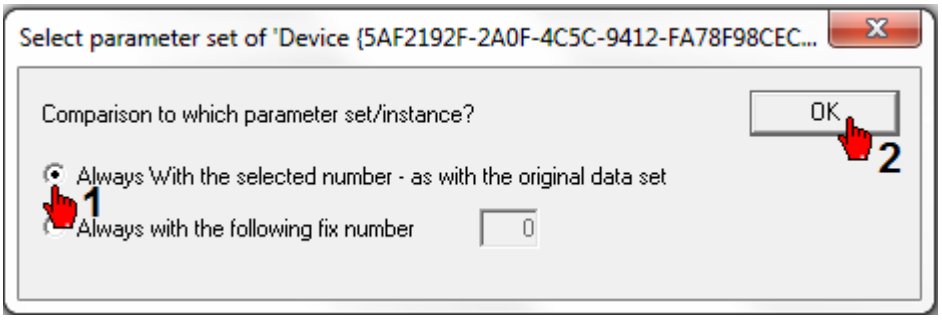


If the comparative data set contains several devices, you have to select the comparative device based on the station name.



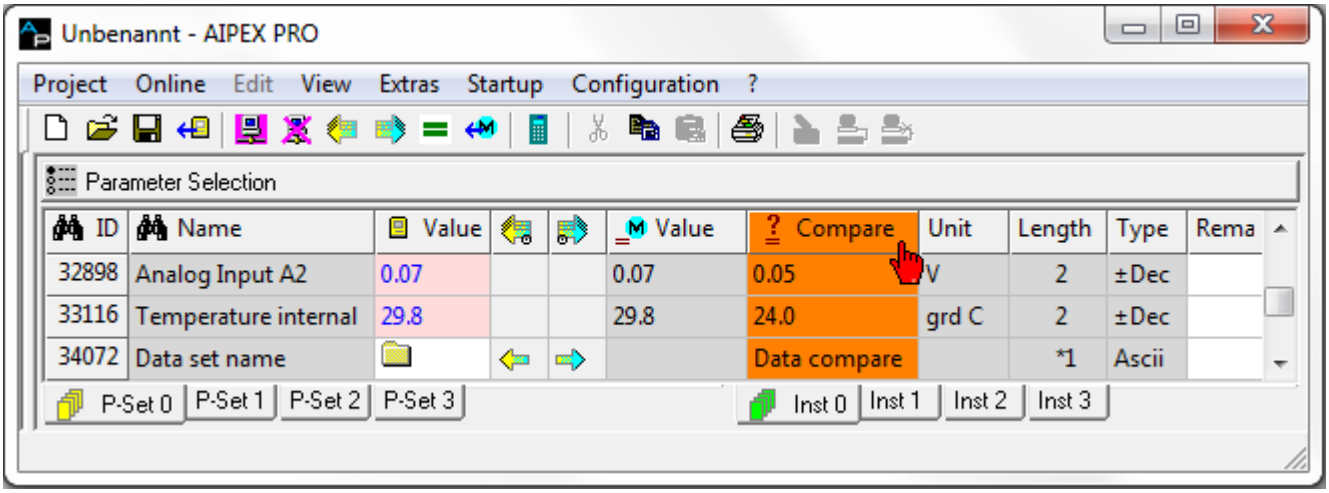
If the upper option is selected, the same parameter set or same instance of the comparative file is always used as with the original data set; i.e. the tabs ('P-set 0', 'P-set 1'... 'Inst 0', 'Inst 2'..)) below the table are also valid for the comparative data set.

With the lower option, a set value for the parameter set and the instance of the reference file is selected. The ('P-set 0', 'P-set 1'... 'Inst 0', 'Inst 2'...) tabs below the table have no effect on the reference file in this case.

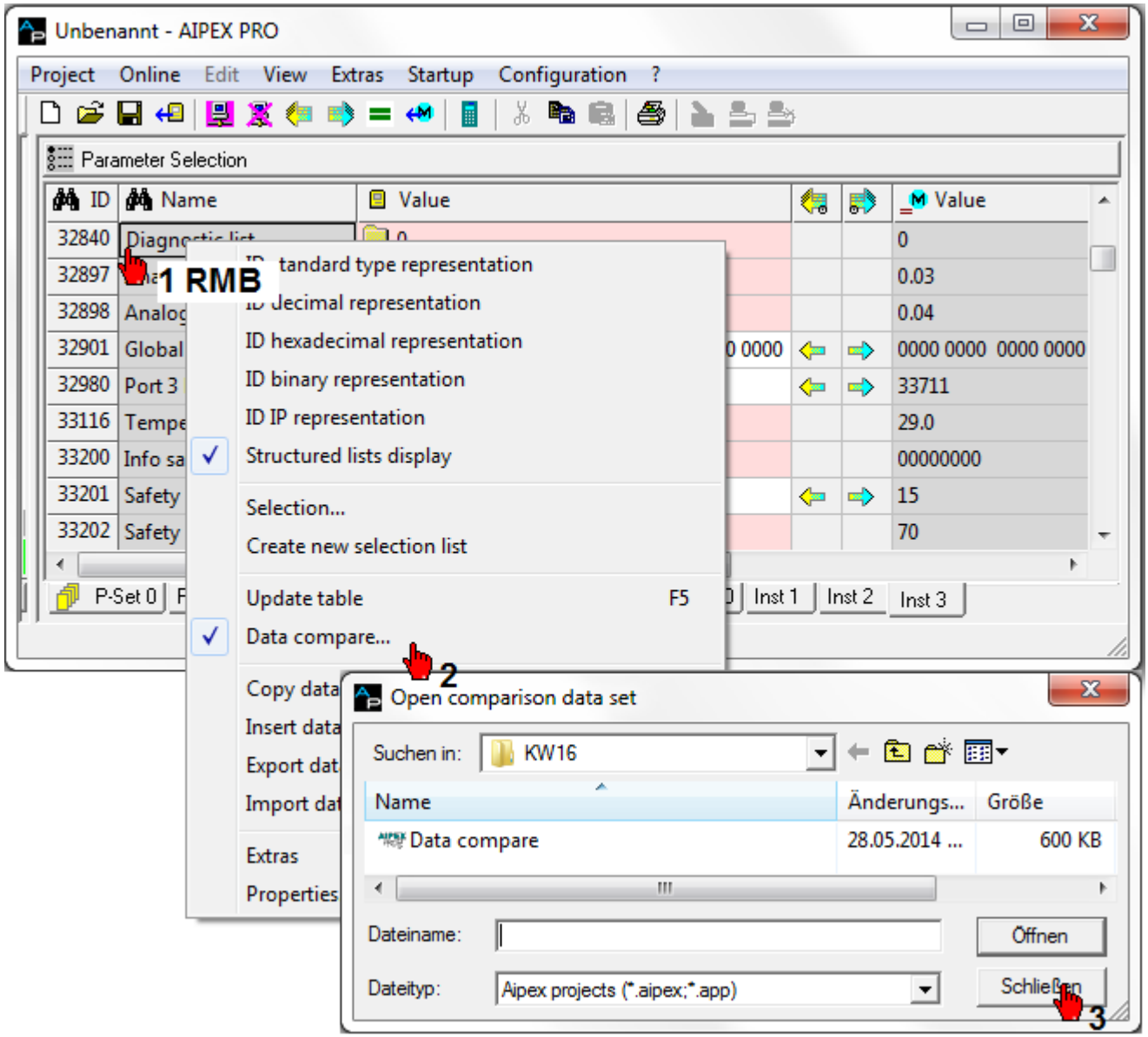


The data of the reference file are displayed in an additional column but cannot be modified. All data of the reference file differing from the original data set are highlighted in orange.

By clicking in the title field of the reference file, you can change between the display of all parameters and a display of the differentiating parameters.



Open the context menu again with the right mouse button. By clicking on the **'Data compare'** menu item, the comparison with **'Close'** can then be terminated in the subsequent dialog window.



3.7.3 Diagnosis with AIPEX PRO

With 'Diagnostics', the diagnostic messages can be read out from the selected device.

Click on each message to receive an explanation for it. You get further information if you analyse Info (I), I2 and I3

The first message of the list is the main activator of the fault; further displayed numbers might be resulting errors which will not appear any longer after rectifying the cause of the first diagnostic message.

Button **'New reading'**

Diagnostic messages will be read out from the selected device.

Button **'Error reset'**

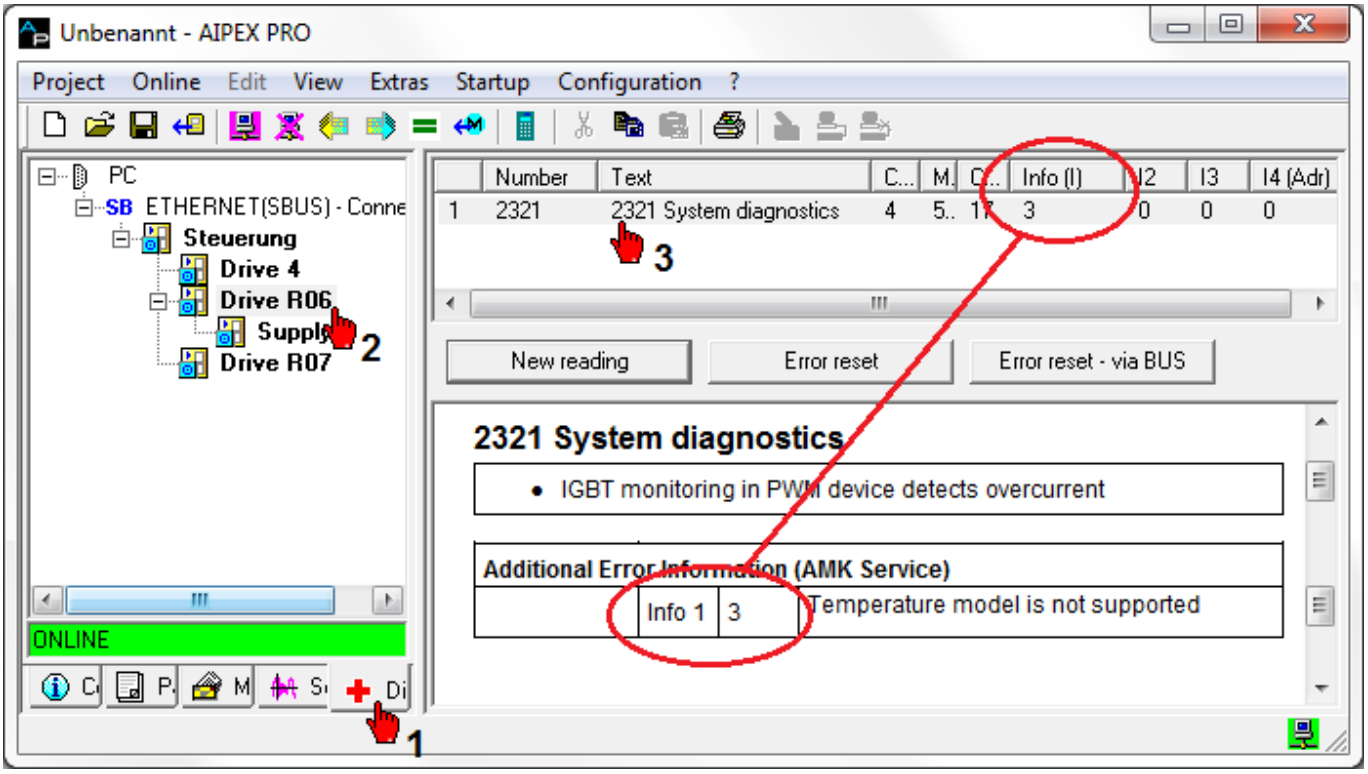
The errors will be deleted in the selected device .

Button **'Error reset - via BUS'**

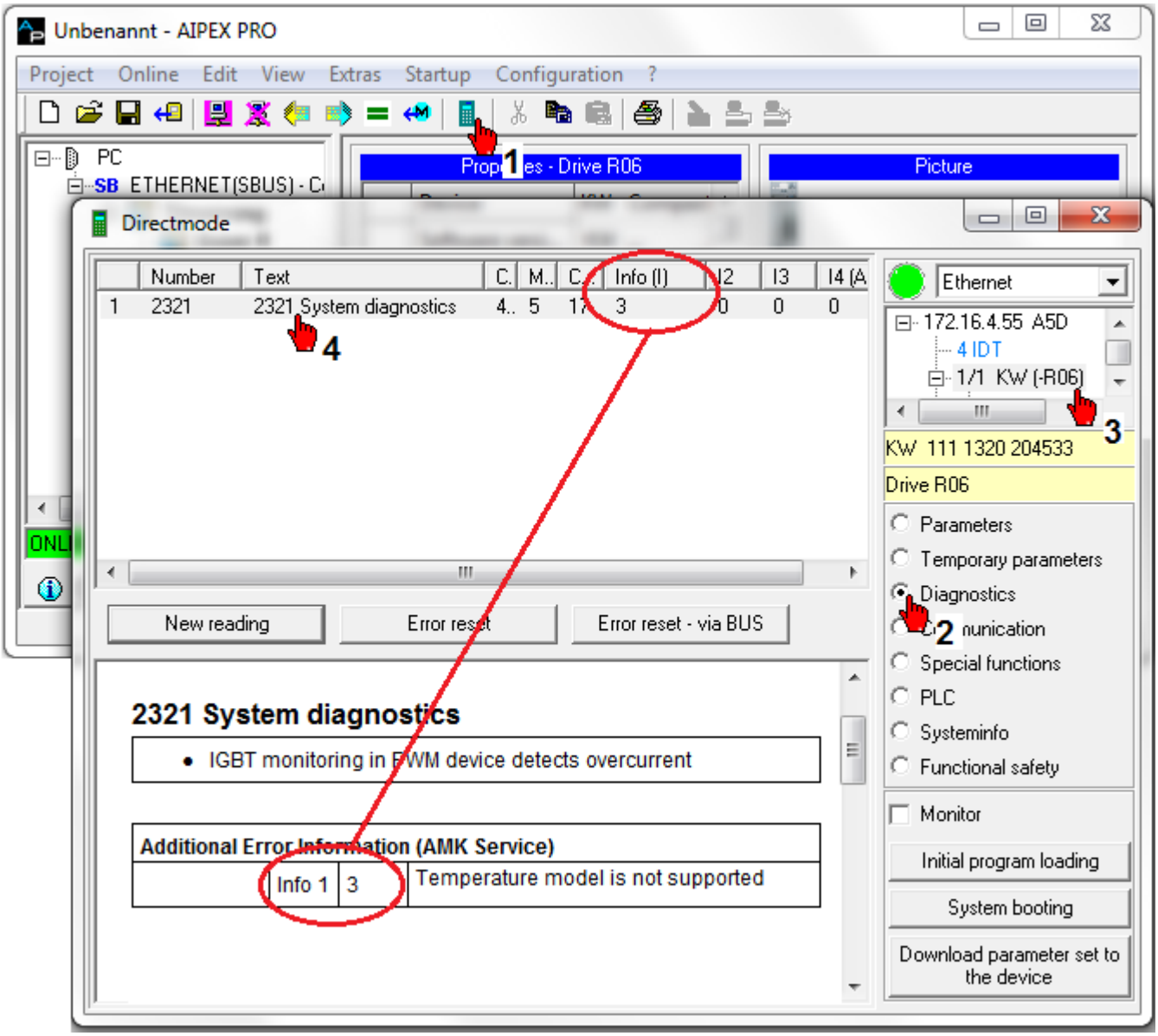
The errors of all devices of a bus line will be deleted. To do this, select the bus in the device tree.

After successful removal of errors, 'clear error' causes a system reboot and SBM is set.

Diagnosis in AIPEX PRO project ('Diagnosis' tab)



Diagnosis in 'Direct mode'

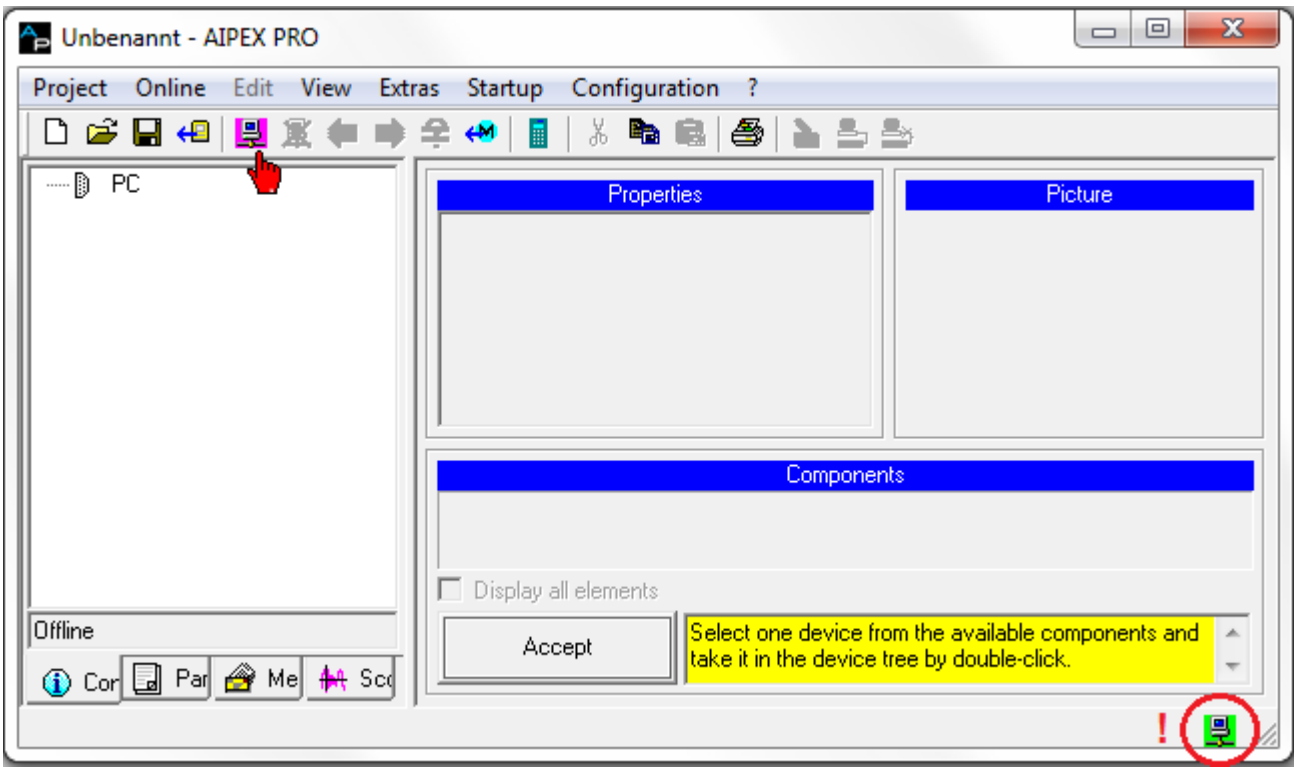


3.7.4 Log on

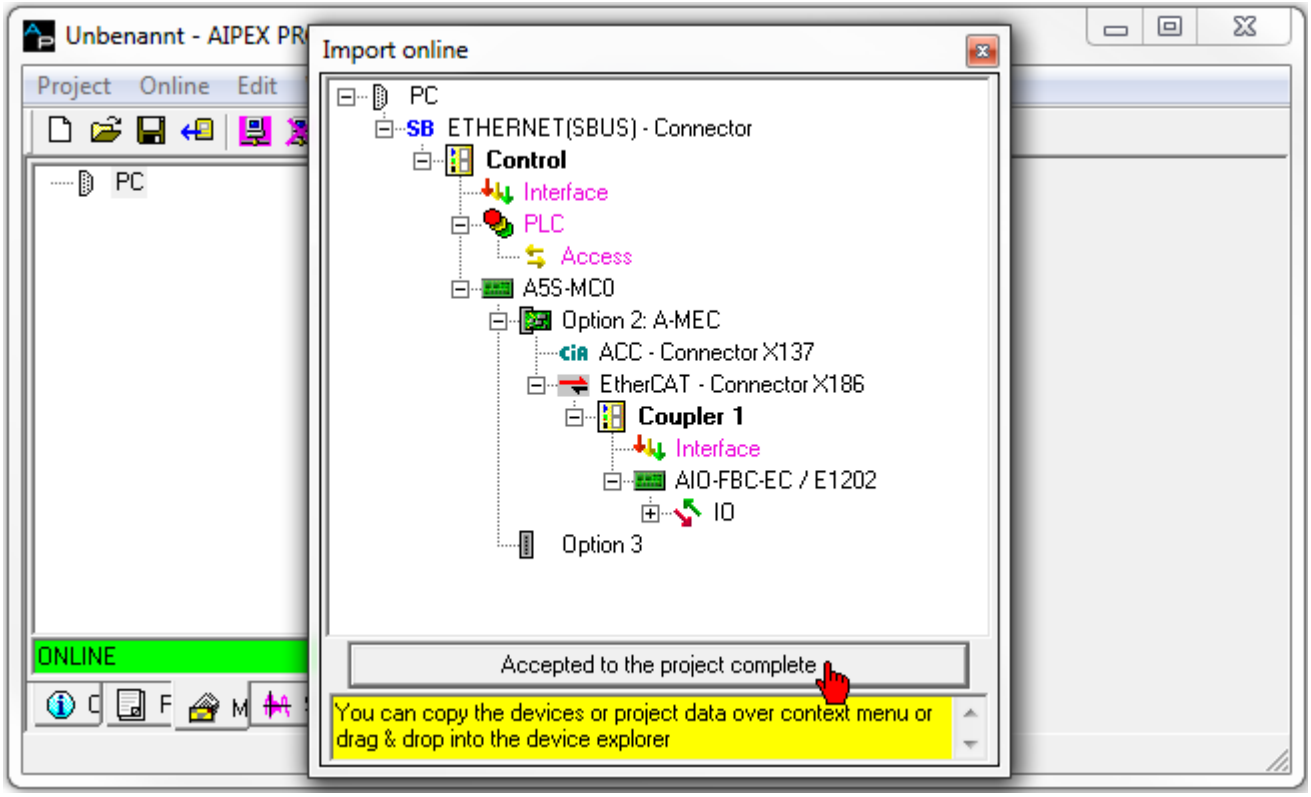
Log on and create project

Use this variant if you then want to save the device data as an AIPEX PRO project on your PC.

Press **'Logon'**

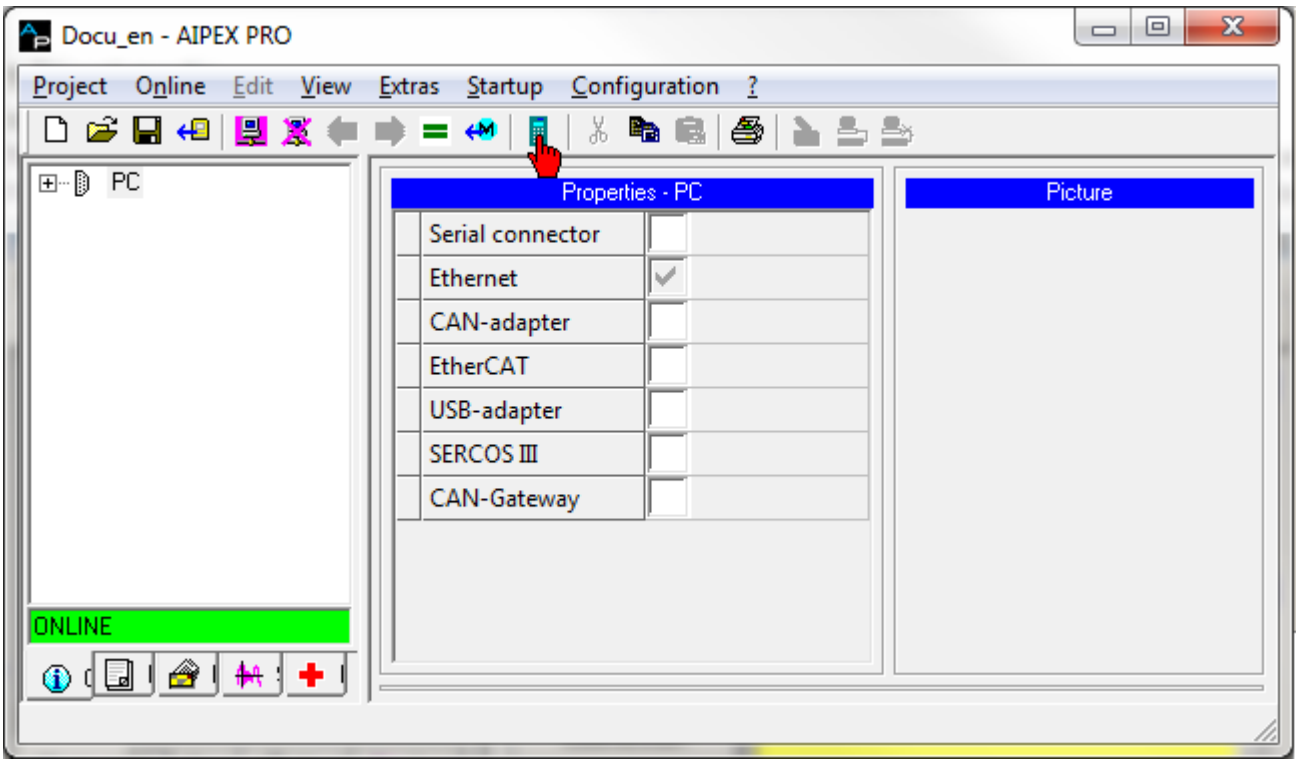


Add the device data to your project. Press **'Accepted to the project complete'**



Log on via direct mode

In **'Direct mode'** you can access AMK devices online.
Changes are transferred to the device immediately. The parameter characteristics determine when the change becomes active.

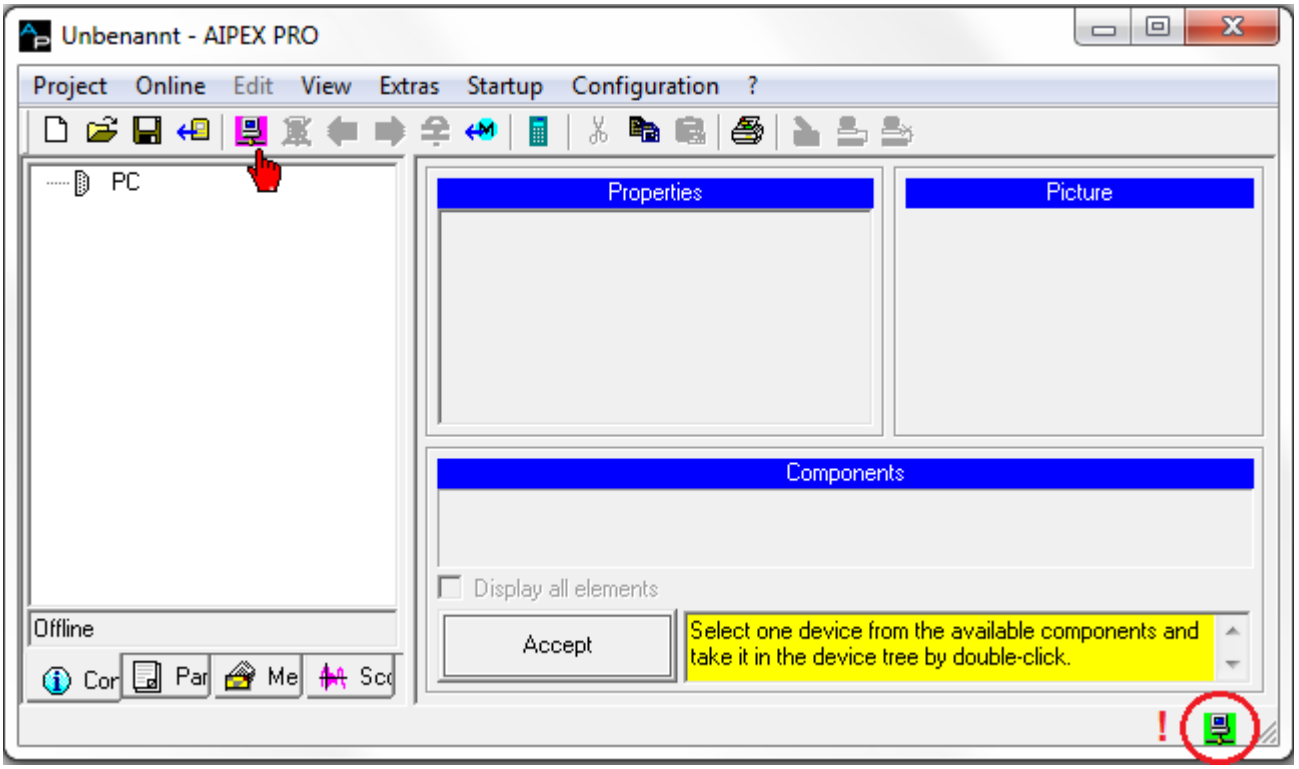


Siehe 'Direct mode' auf Seite 122.

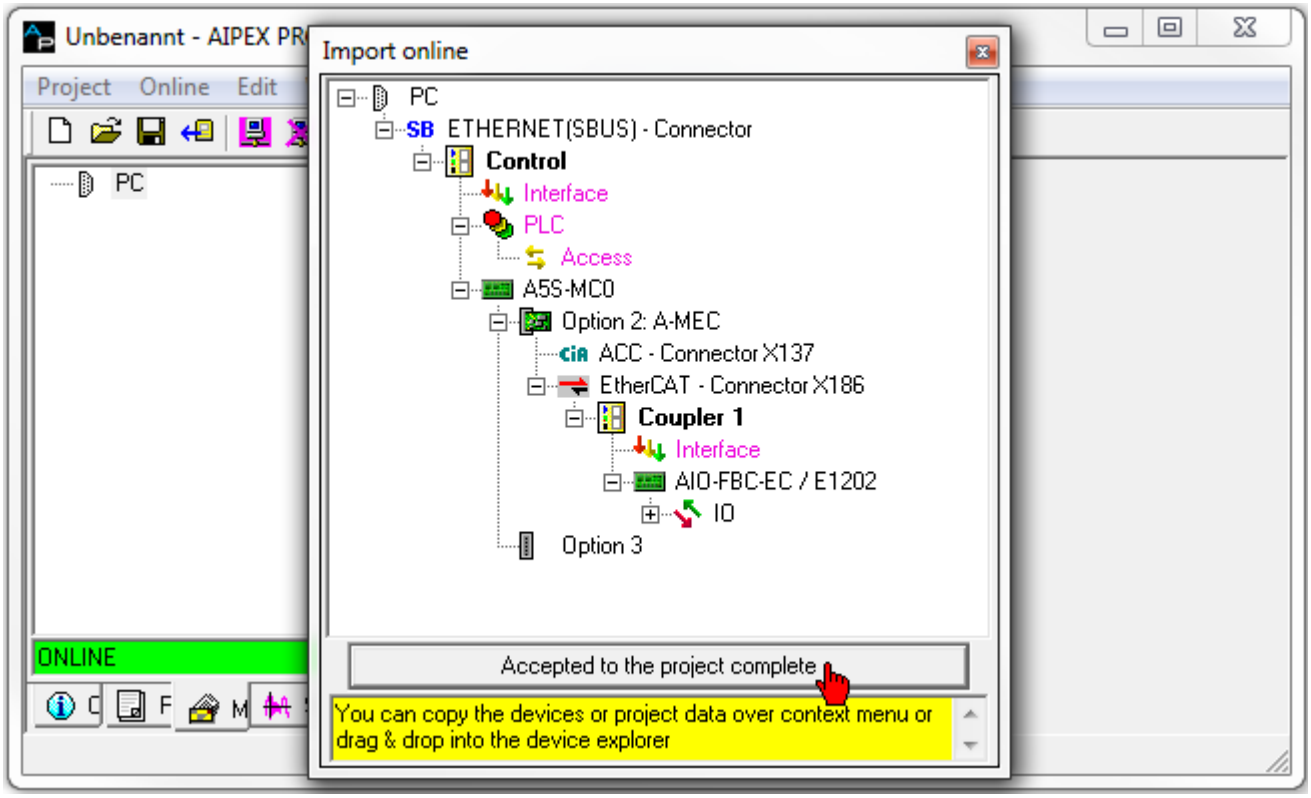
3.7.5 Reading out and saving device data

The following example explains how to read out device data and save it on the PC.

Press **'Logon'**

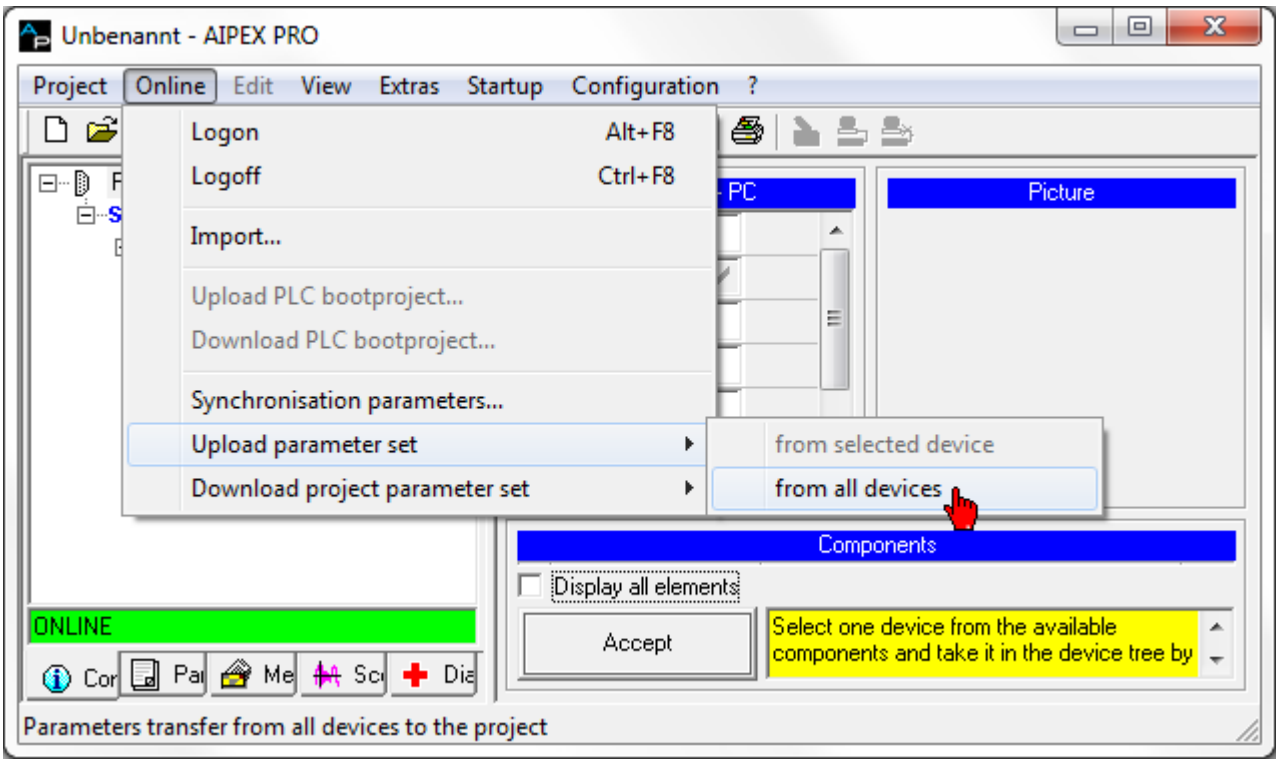


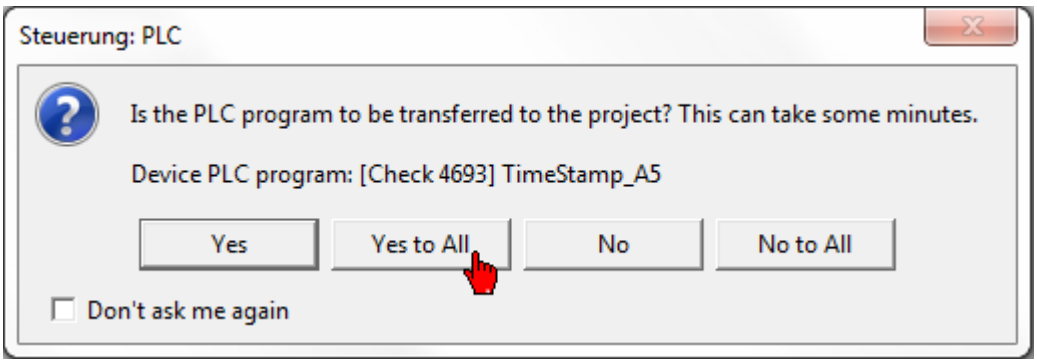
Add the device data to your project. Press **'Accepted to the project complete'**



The loading time depends on the connection used (Ethernet, SBUS, CAN ...) and the number of devices connected.

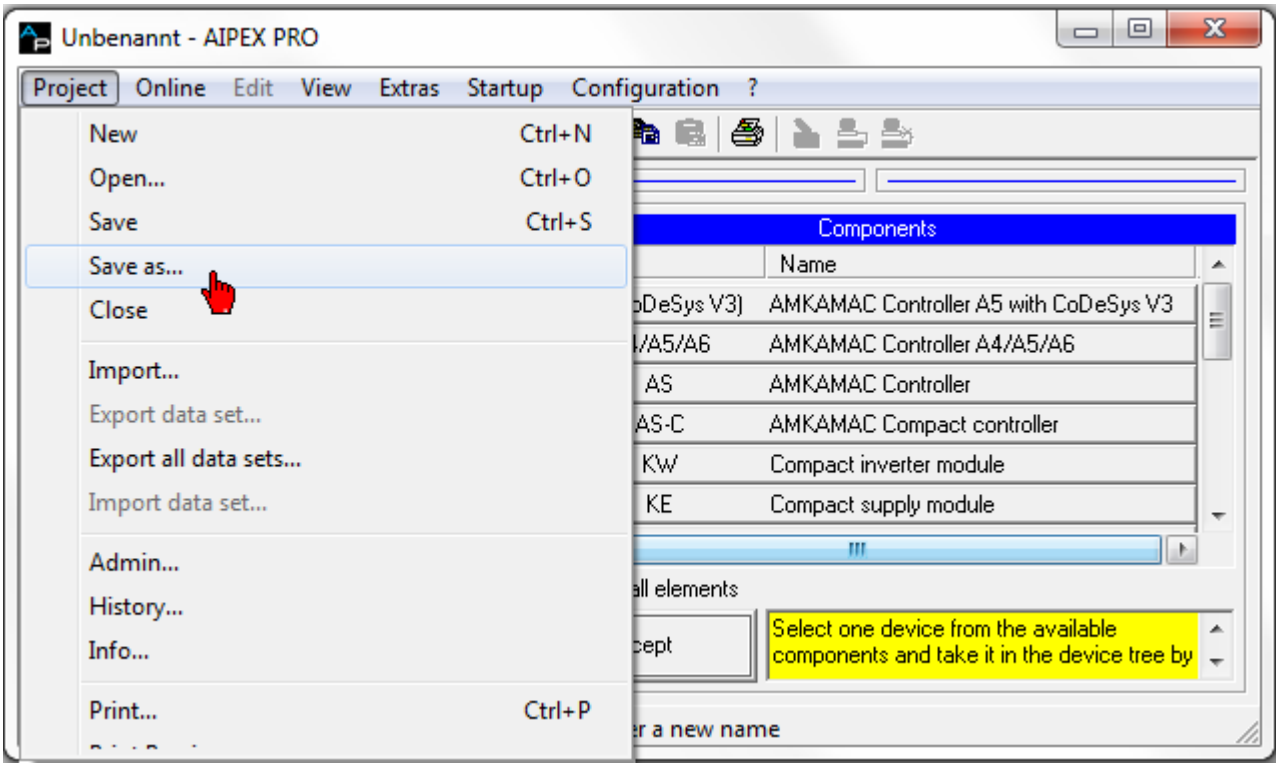
The PLC program is only added to the AIPEX PRO project if you execute and confirm the function **'Online'** → **'Upload parameter set'** → **'from all devices'**.





Setting options: Siehe 'Project Settings - Base Settings' auf Seite 108.

Save the project. In the dialog box, you can then specify the directory and the file name.



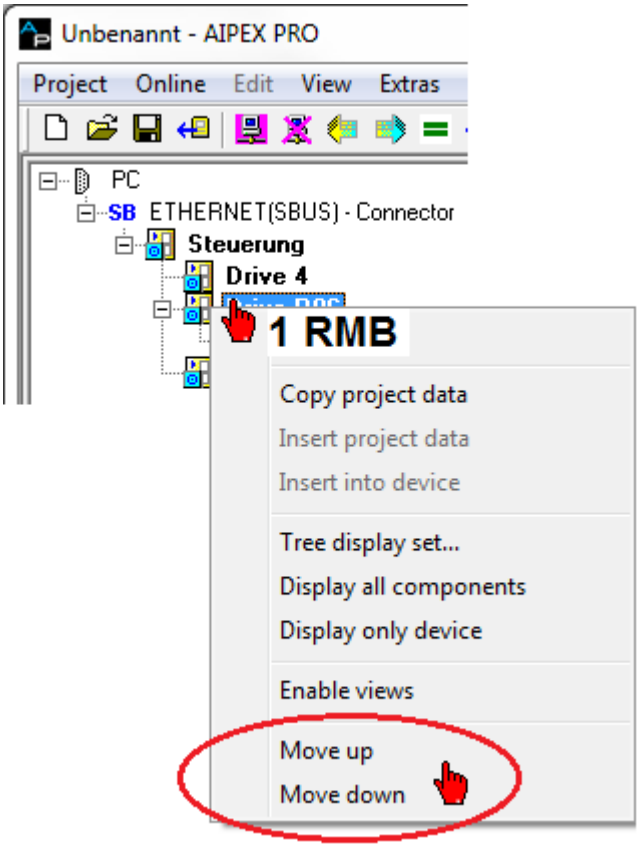
3.7.6 Moving device position in the EtherCAT bus

Prerequisite:

Activate the position view (Menu 'View') in the device tree.

Siehe 'View' auf Seite 99.

Click with the 'RMB' on the device you want to move. You can move the position step-by-step using the menu items **'Move up'** or **'Move down'**.



3.7.7 Importing an external device description

The functionality of external devices, such as I/O terminals, is included in the device configuration files. For EtherCAT, these are XML files, and for CAN (ACC bus) EDS files. Using the **Import description of external device**function, device description files can be imported, which are not integrated in AIPEX PRO by default.

No guarantee can be assumed for the functionality of additionally imported devices.

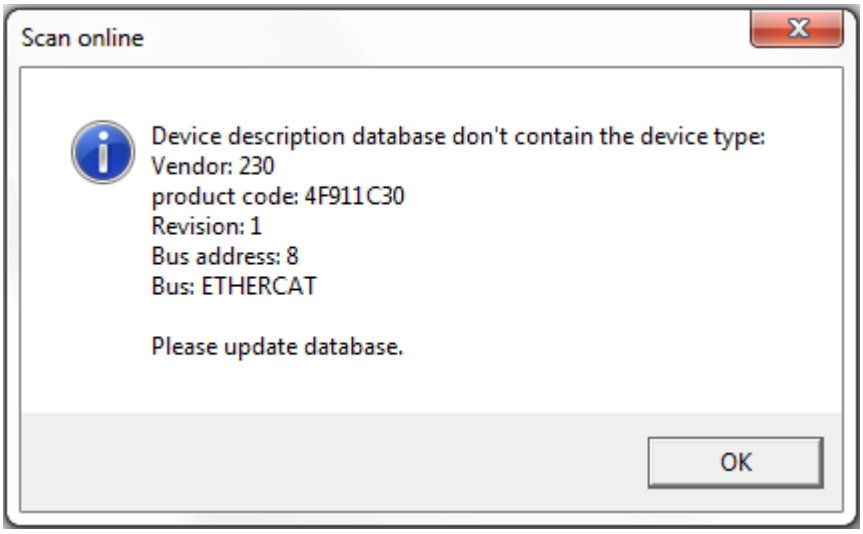
After the importing, AIPEX PRO needs to be rebooted.

If, during **'Logon'**, AIPEX PRO detects an external device without a device description file, the following diagnostic message is displayed:

The device description does not contain any information about:

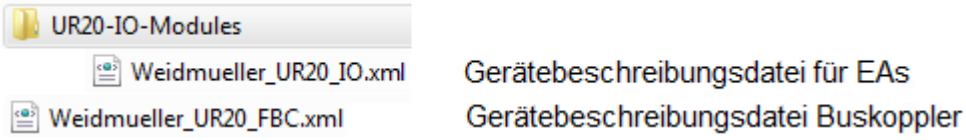
.....

Please update database

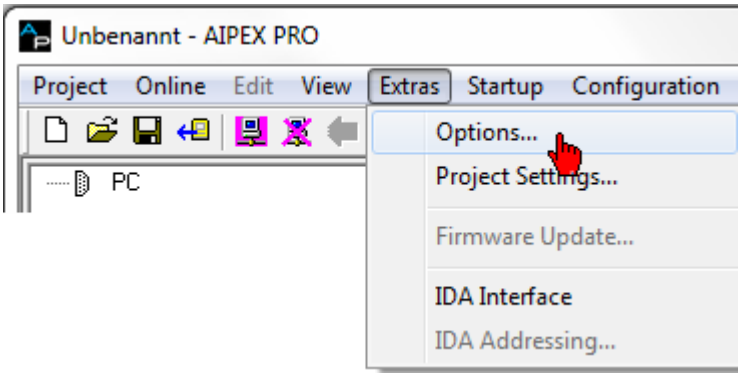


Device description files are imported as follows:
Save the required device descriptions of the terminal manufacturer in the AIPEX PRO folder:
Windows 7: C:\Program Files (x86)\Common Files\AMK\EtherCAT
Windows XP: C:\Programs\Shared files\AMK\EtherCAT

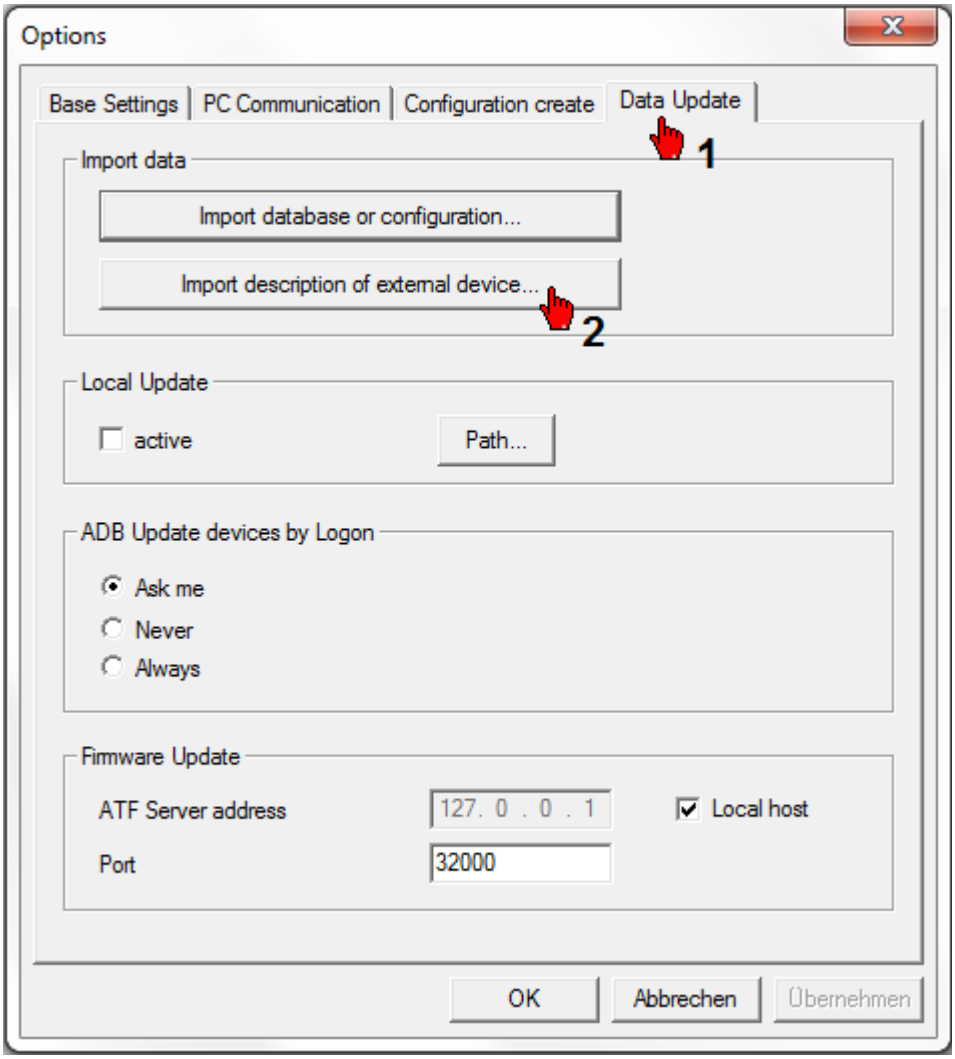
Example of Weidmüller terminal:



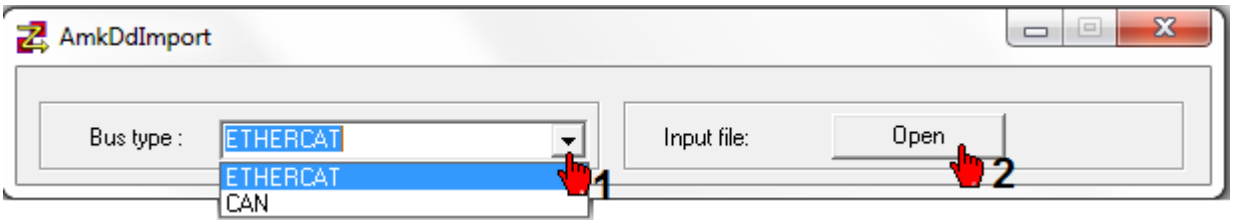
Open the **'Options'** window.



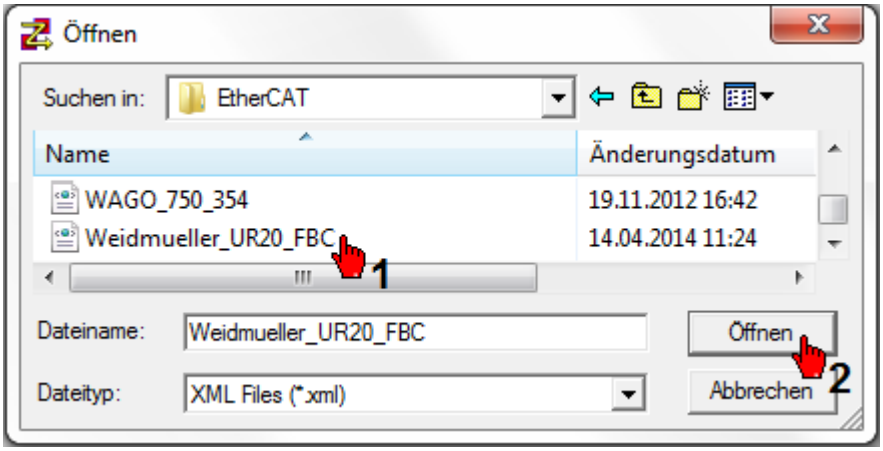
Select the **'Data update'** tab. Press the **'Import description of external device'**button.



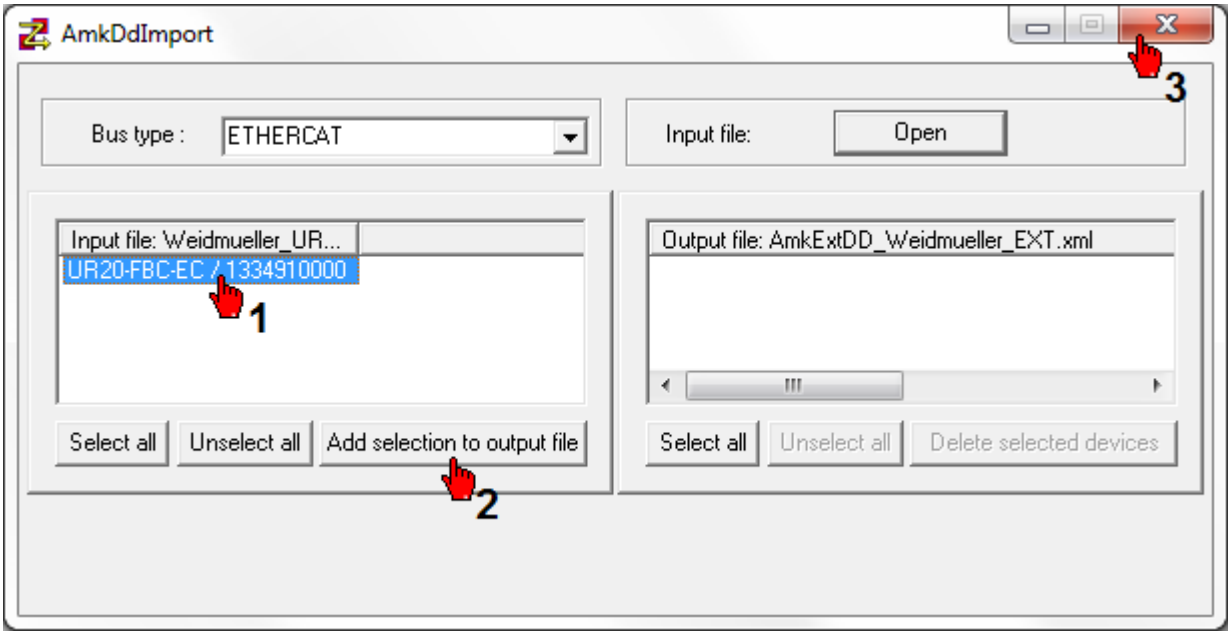
Select the fieldbus.



Select the device description file of the bus coupler to be imported.



Press the **'Add selection to output device'** button to import the required file.



Restart AIPEX PRO.

3.7.8 Configuring modular device

A Modular Device is a device that occupies a address on the EtherCAT bus.

Internally, the modular device from more participants, which communicate with each other via an internal bus. The EtherCAT bus sees only 1 address for a Modular Device.

The following example describes how to configure a 'modular device' (a AIO terminal in the example).

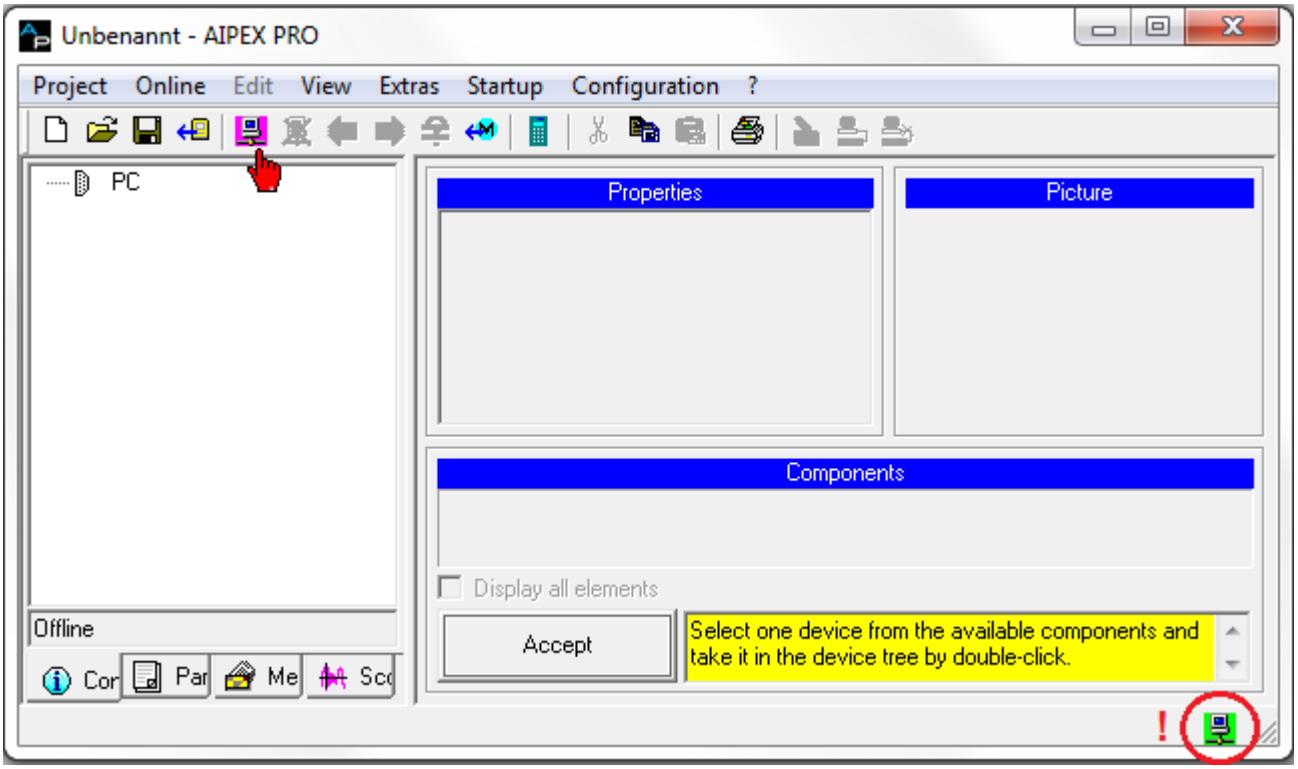
Prerequisite:

The device description file of the bus coupler is imported.

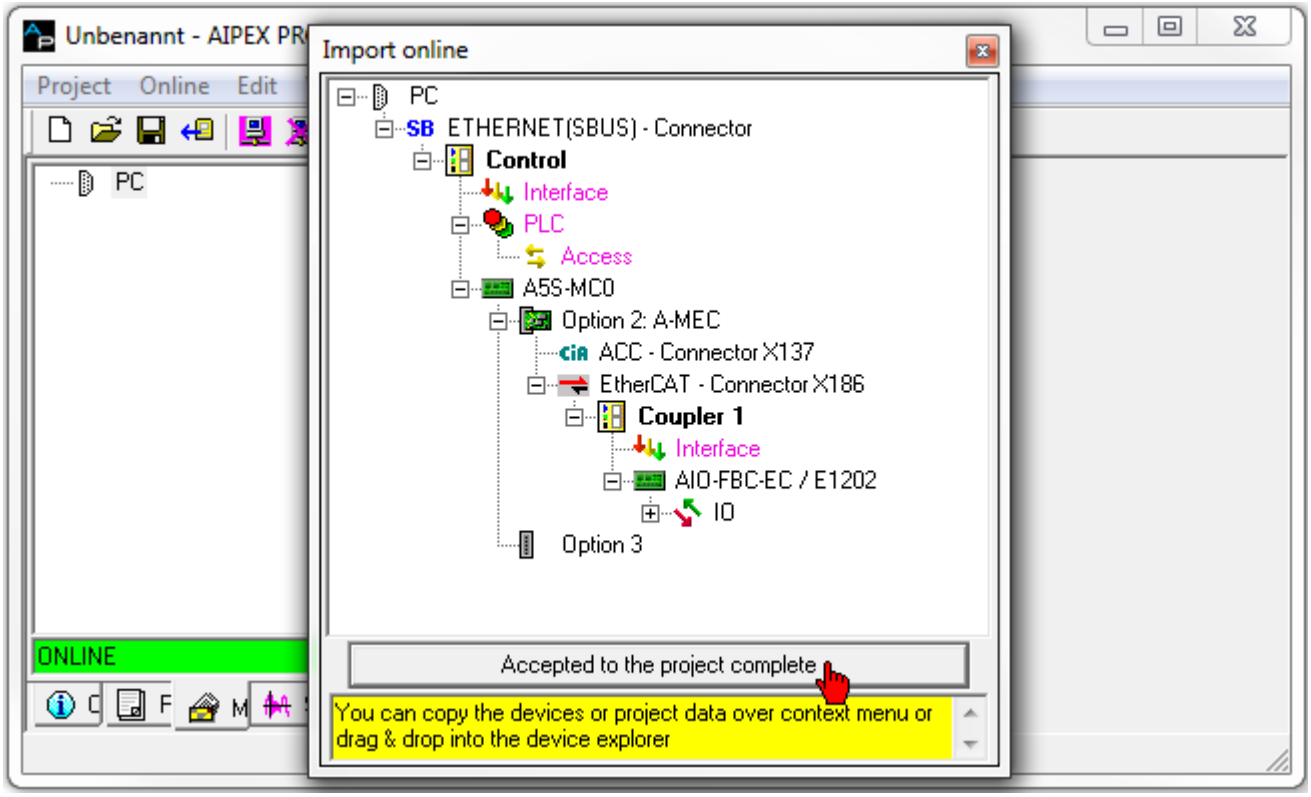
Siehe 'Data update' auf Seite 106. 'Import description of external device' function

Start AIPEX PRO.

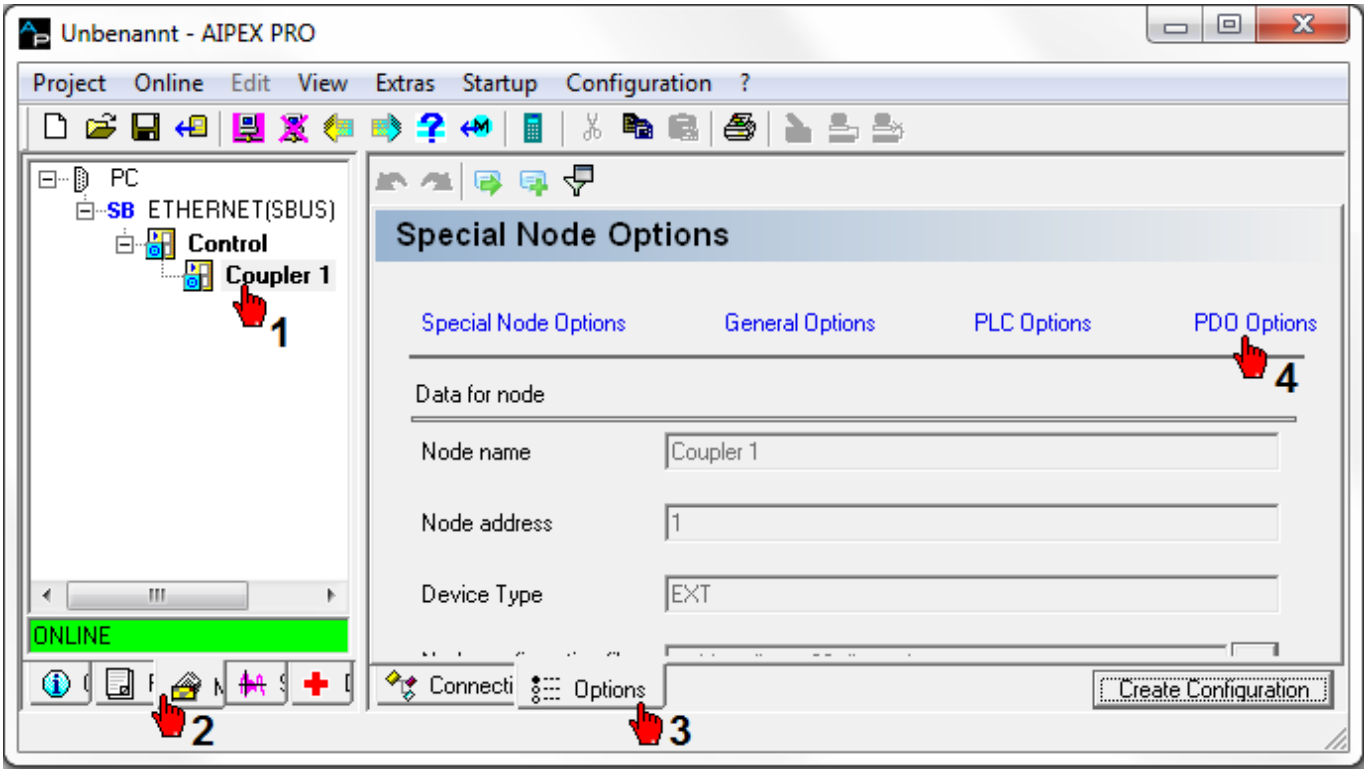
Press **'Logon'**



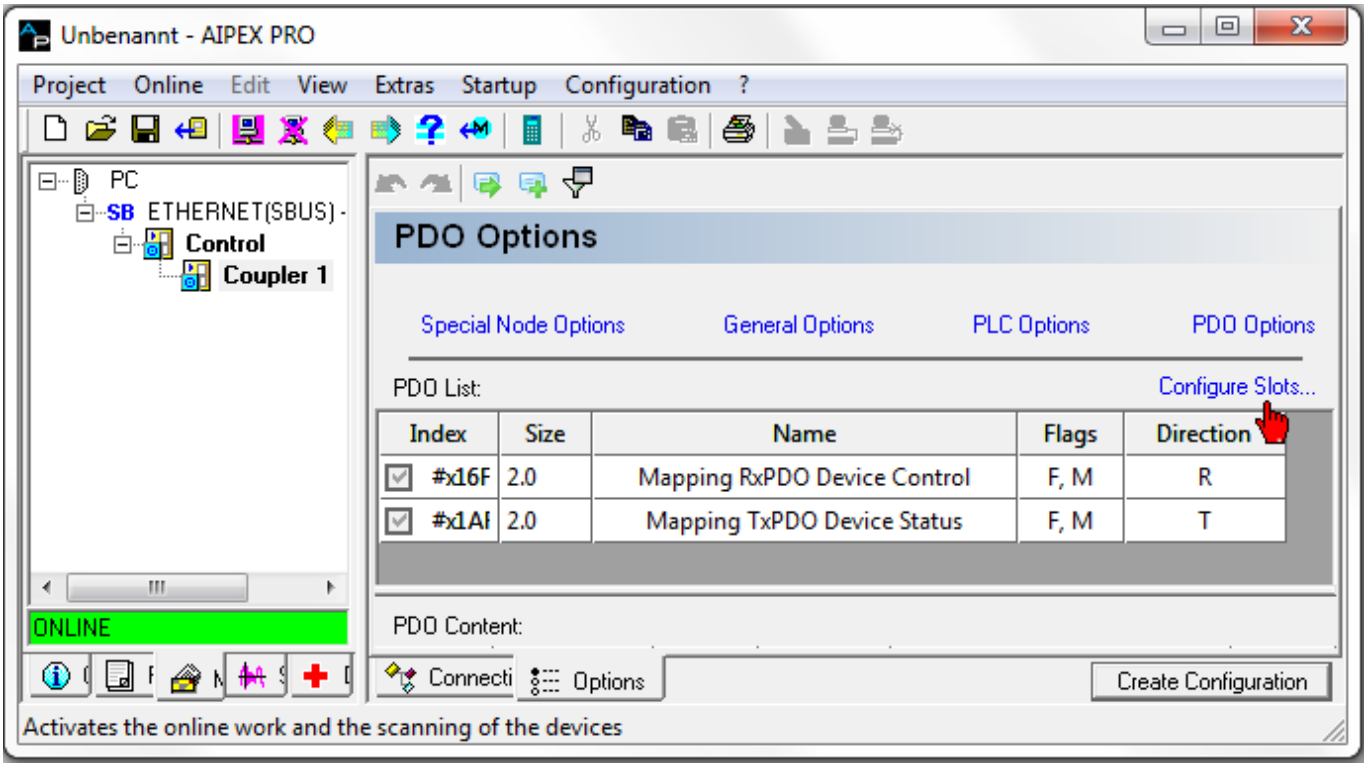
Add the device data to your project. Press **'Accepted to the project complete'**



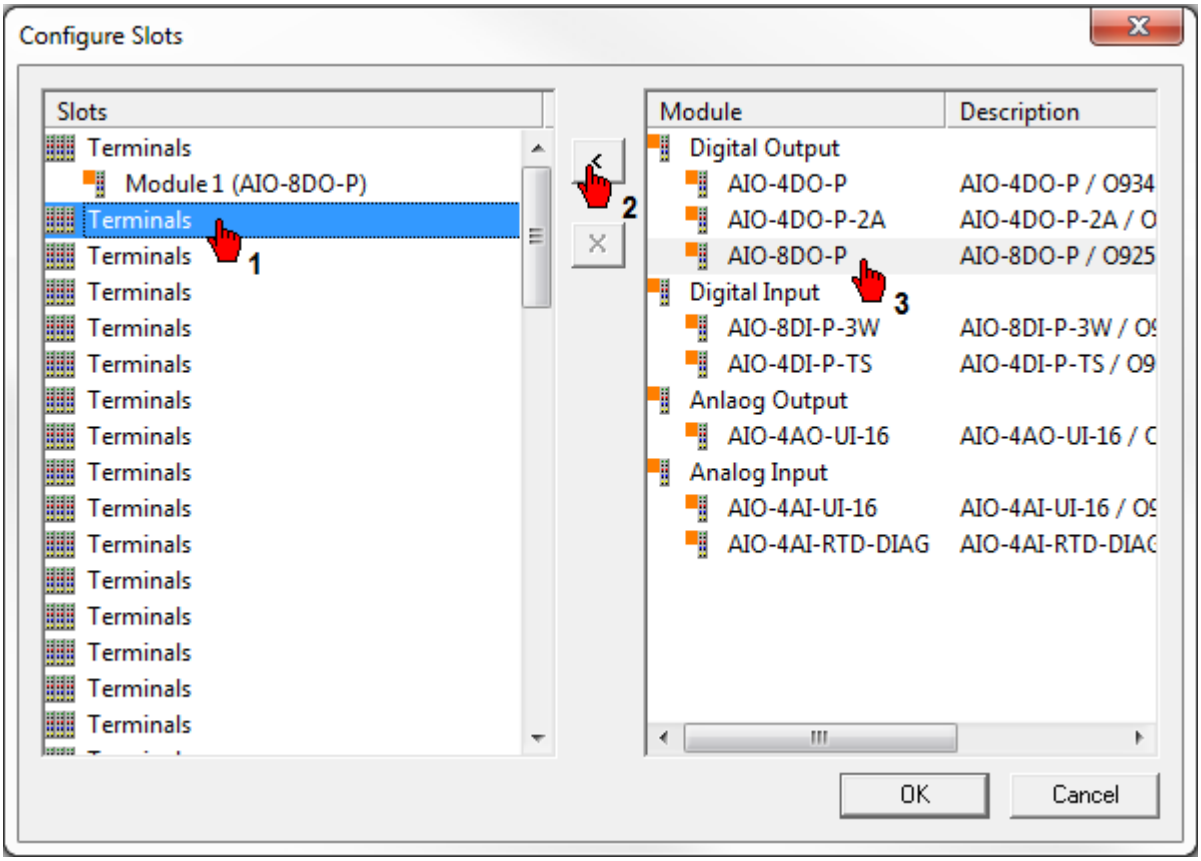
Select the**'PDO options'** in the **'Messages'** → **'Options'** tab.



Select the **'Configure Slots...'**.



The first terminal corresponds physically to the first terminal block after the bus coupler (etc.)
Assign the physically existing terminal blocks to the terminals.

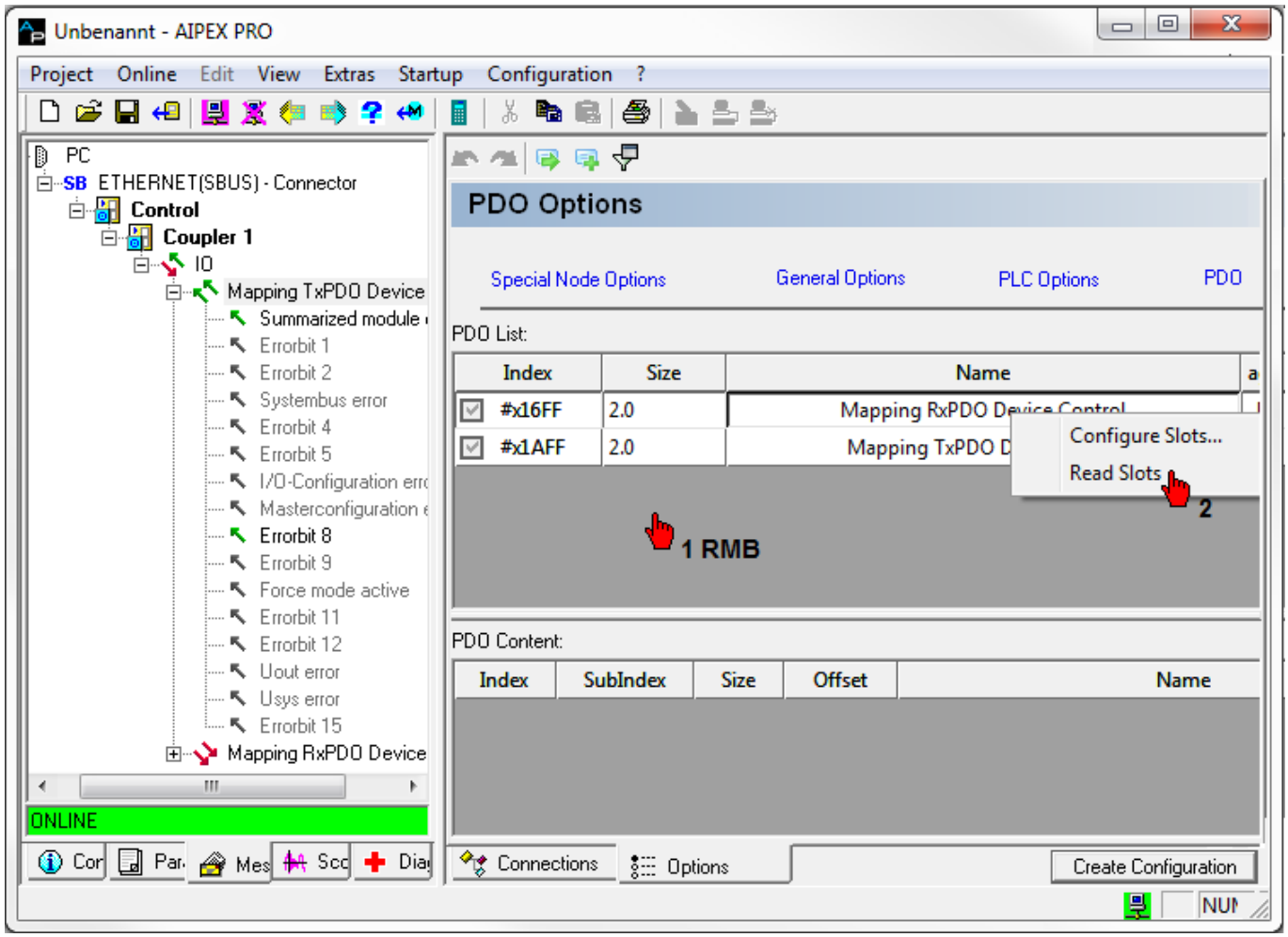


Example of read and write I/O terminal with CODESYS.

Siehe 'External I/O terminal (asynchronous)' auf Seite 674.
Siehe 'External I/O terminal (cyclical)' auf Seite 678.

As an alternative to manual input of occupied slots in 'modular device', the current configuration can be reed directly from the connected device.

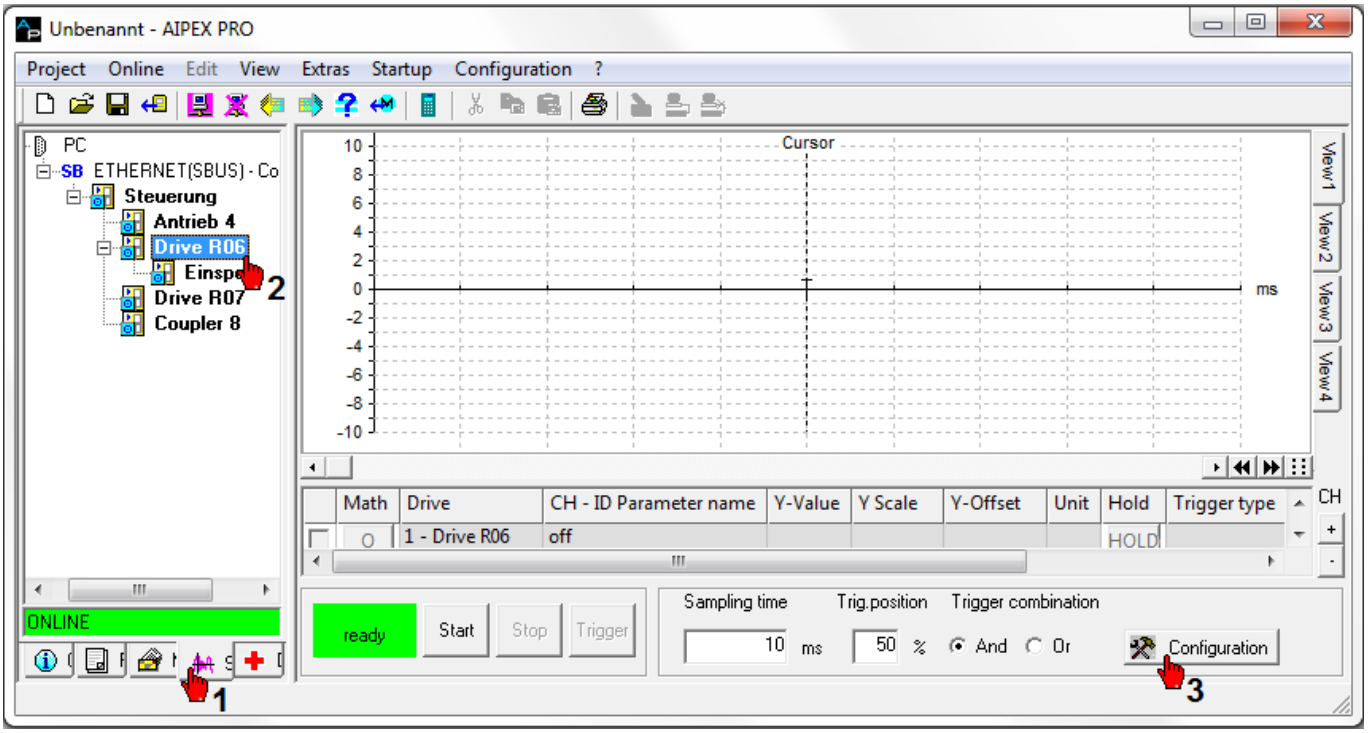
Prerequisite:
Online mode active.



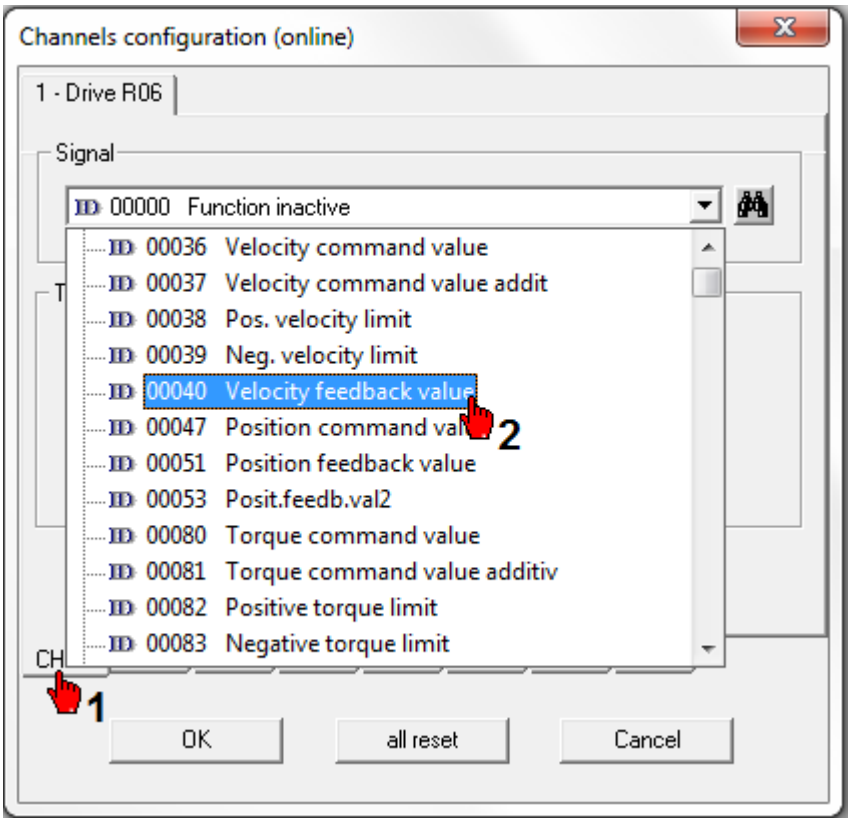
3.7.9 Configuring oscilloscope

The example shows how to record a speed actual value with the AIPEX PRO oscilloscope.

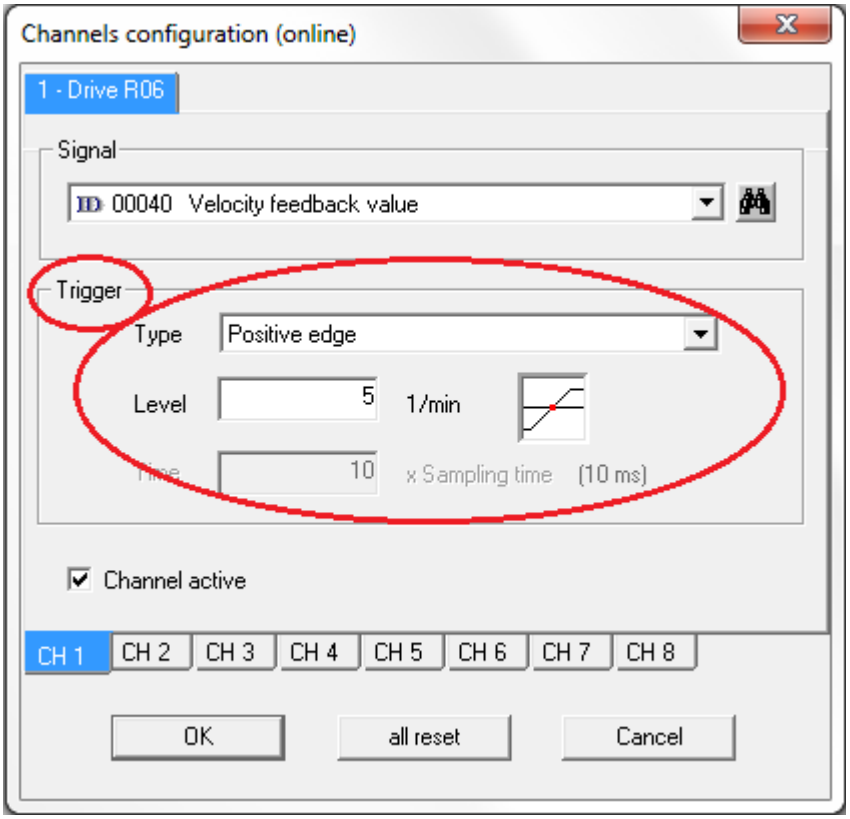
Open the **'Configuration'** window.



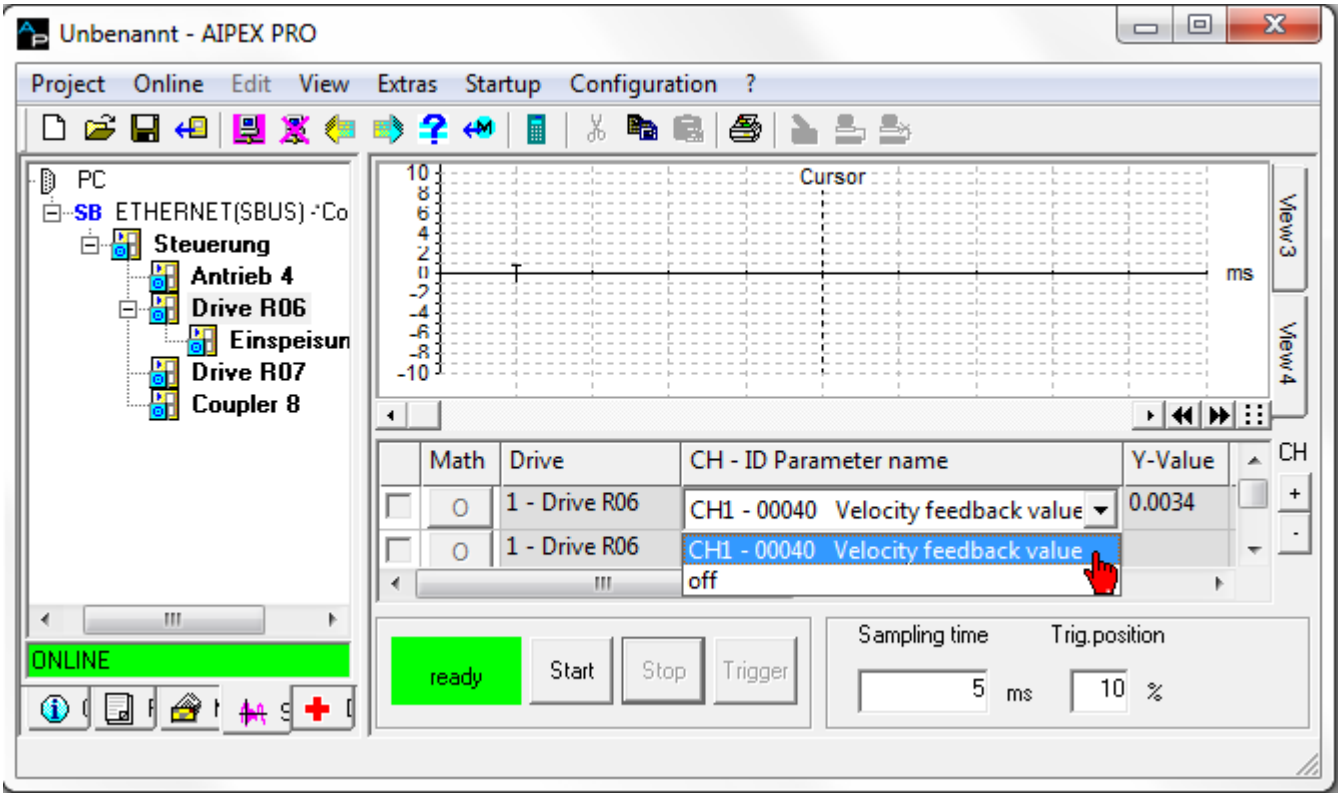
Enter ID40 'Velocity feedback value' in 1 (CH1) channel.



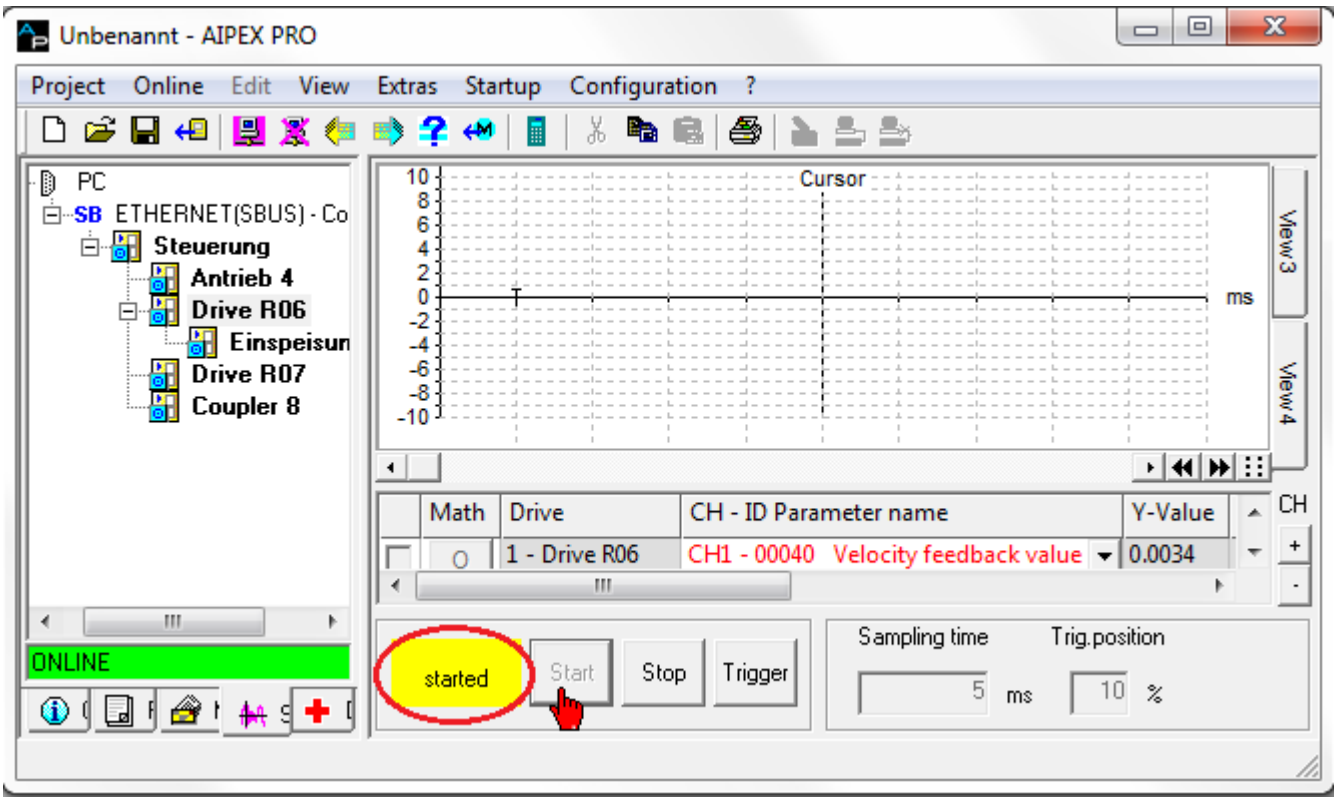
Trigger settings:
Trigger type: Positive edge, level 5 1/min. The measurement is started as soon as the speed actual value 5 1/min is exceeded.



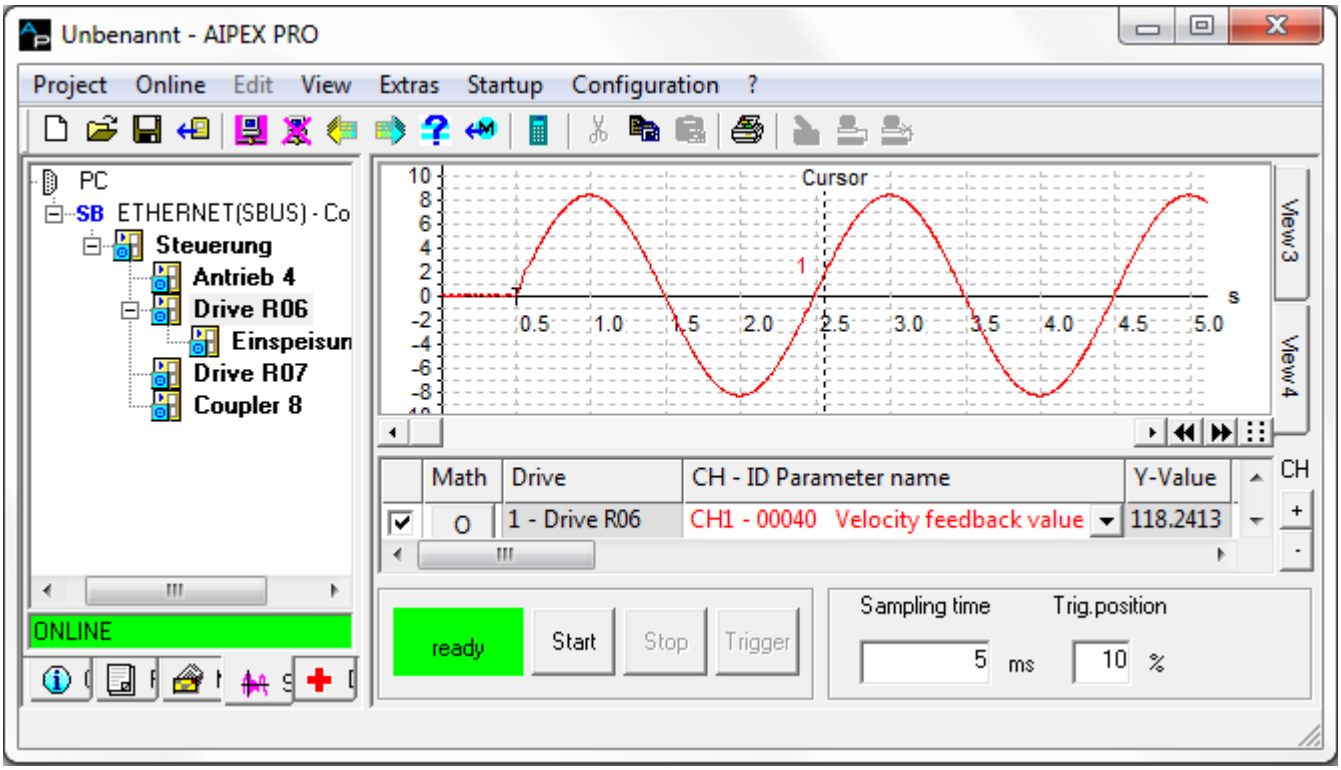
The configured signals are shown and hidden using the 'CH – ID Parameter names' field.
Adjust the 'Sampling time' and 'Trigger position' to your measurement.



Click on **'Start'**. The status must change to 'started'.



The scope waits for the configured trigger signal. The recorded measuring data is then transferred and displayed graphically. (In the example, the setpoint is specified by the test generator).



3.7.10 Test generator

DANGER	
⚠	**Motor shaft movement!** Hazardous motor movement occurs when the motor shaft moves in an uncontrolled or unintentional manner. **Steps to prevent:** • Decouple the motor from the load so that the shaft can rotate freely • Specify the maximum permissible process speed and set ID113 accordingly. (ID113 = max. process speed/1.25)

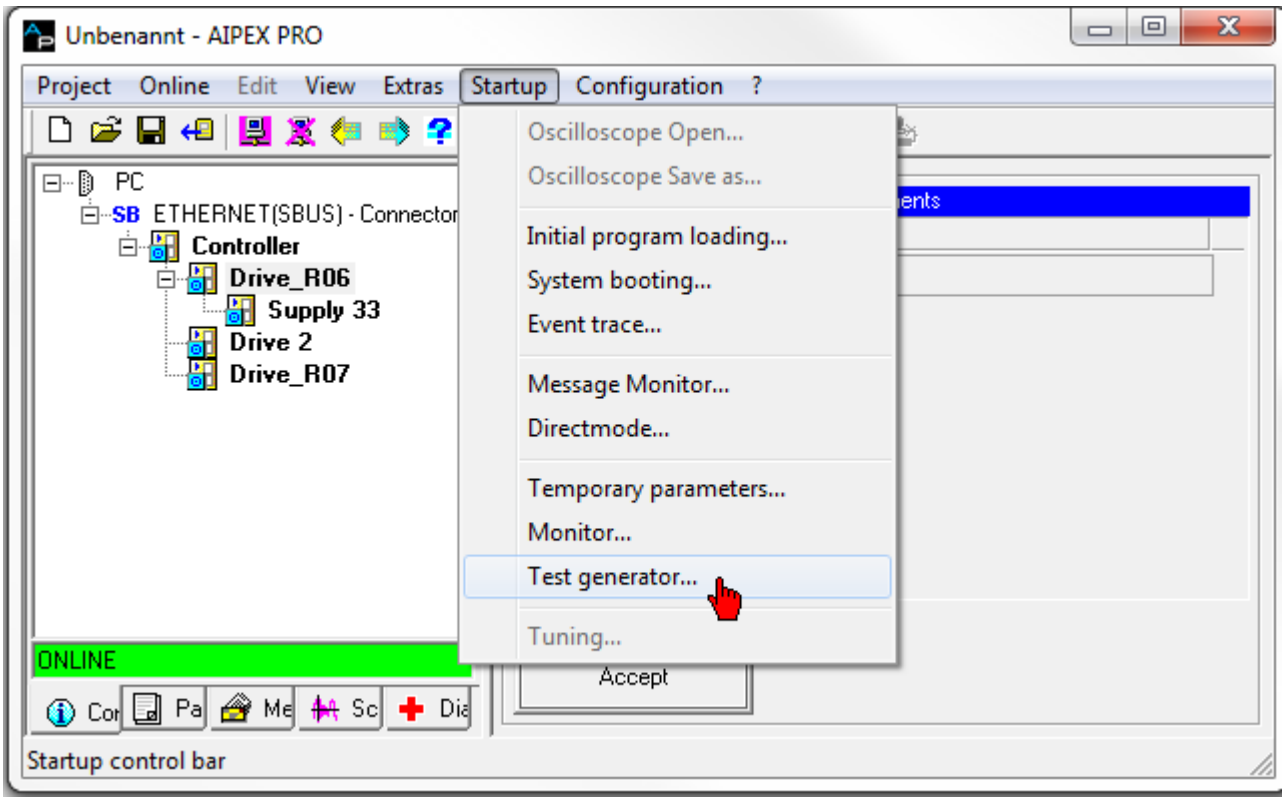
The 'Test generator' function of the AIPEX PRO software gives support to the initial startup and controller tuning.
The startup function contains a setpoint generator which can set several curve forms (trapezoidal, square wave, triangular, sinusoidal) for different operation modes (torque, speed, position control).

Before you first put the motor into operation, check the sense of rotation
The following example shows how to set a positive speed setpoint with acceleration and deceleration ramp.
By means of the 'Scope' function of AIPEX PRO, you can display and store the actual motor values

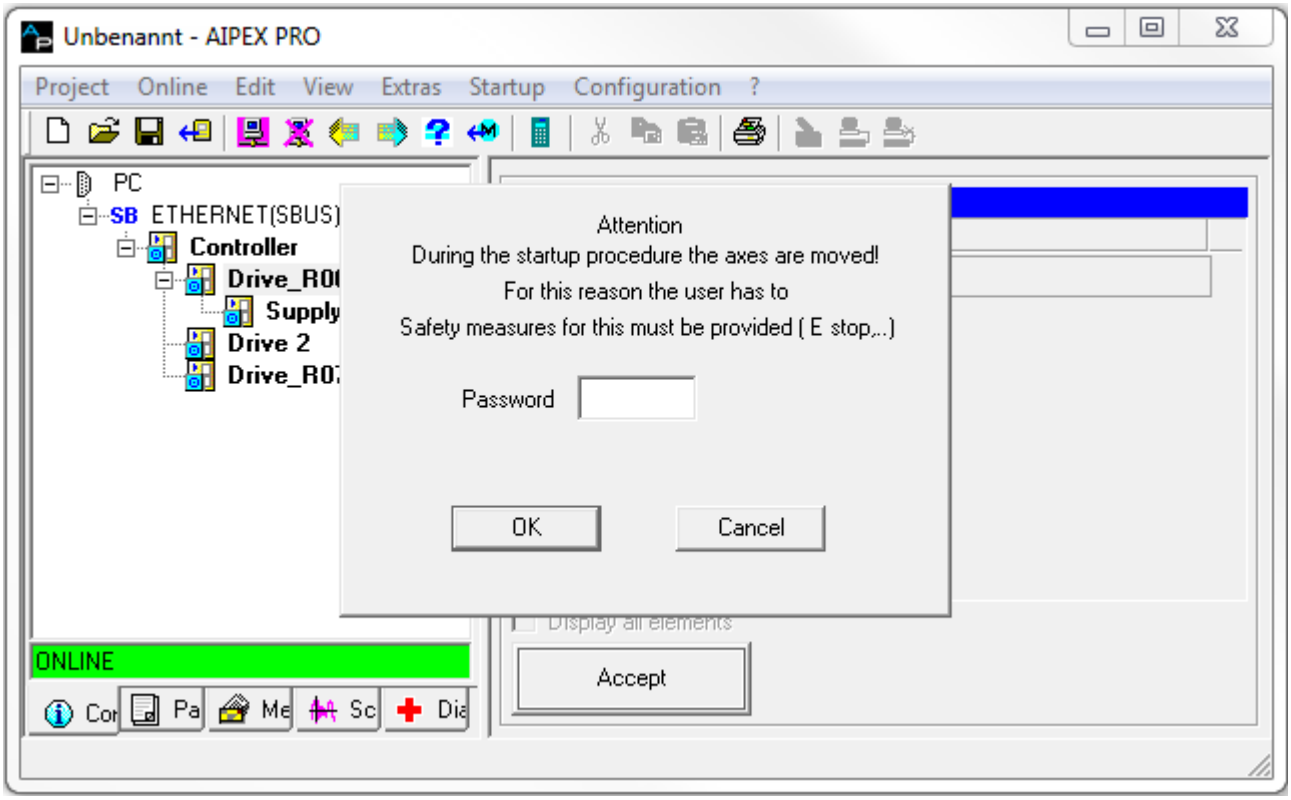
The sense of rotation can be inverted by ID32773, bit16 = 1

Starting the 'Test generator'

In the device tree click to that device you want to set a setpoint by the startup function. Click to the **'Startup'** menu and subsequent to **'Test generator...'**.

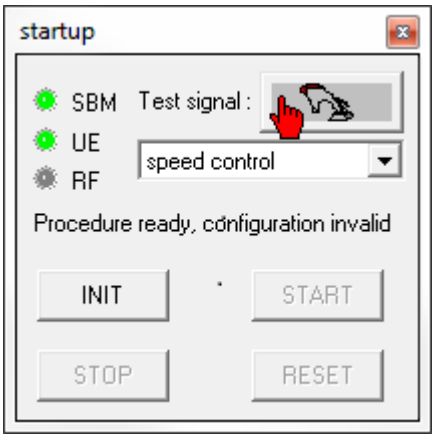


The 'Test generator' will be released by the AMK service password (500591).



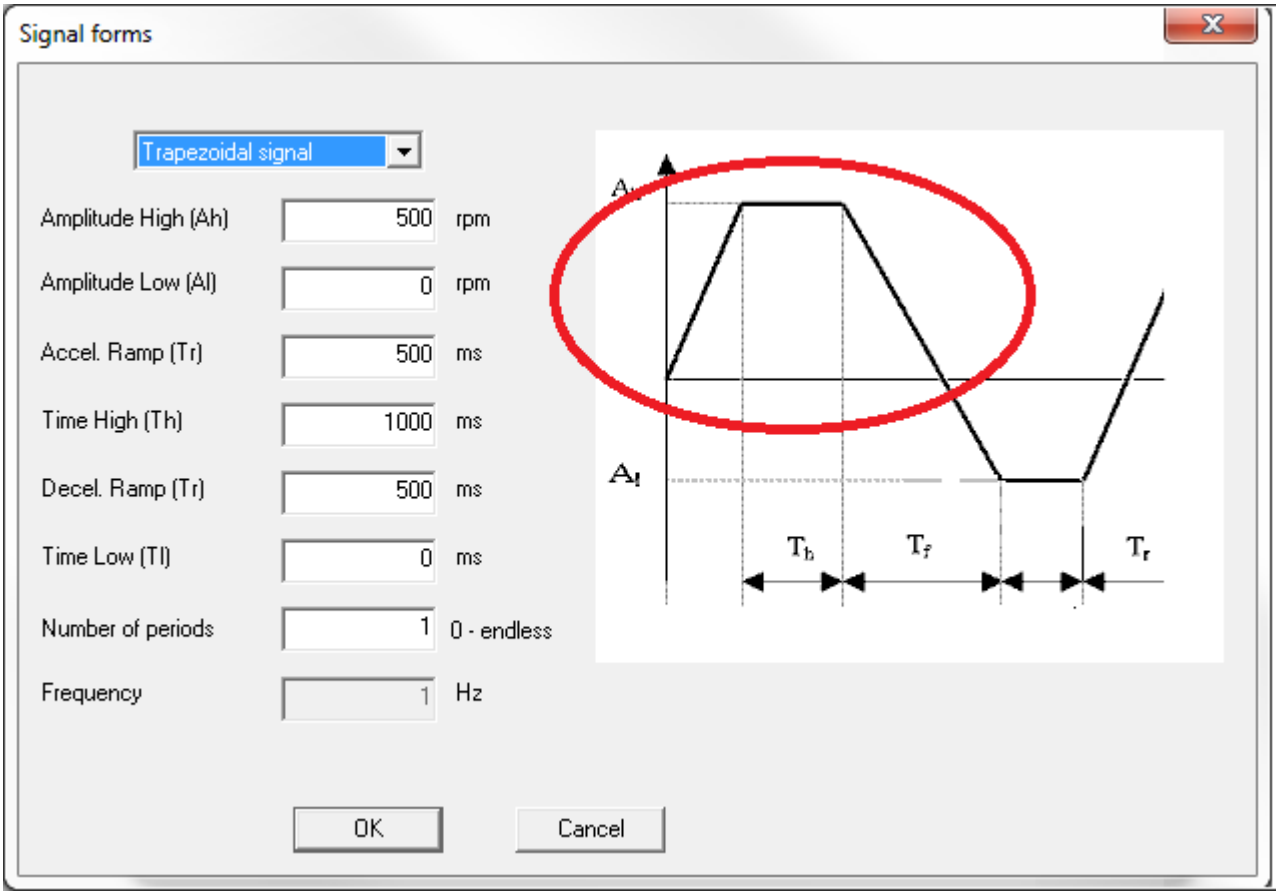
Open the configuration window 'Signal forms'

'Controller enable' RF must be withdrawn.



Example trapezoidal signal:

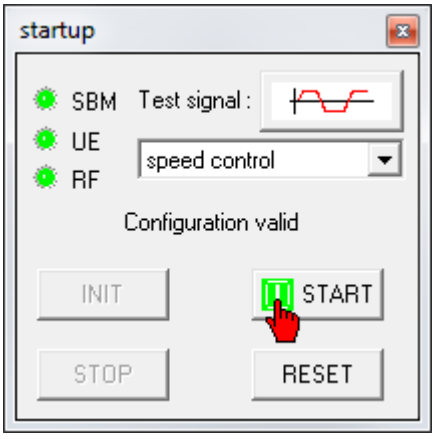
By means of the trapezoidal signal, you can set a positive and negative speed setpoint with acceleration and deceleration ramp. The example configures just a positive movement.



Now, activate the motor controller

The LEDs show the states of SBM, UE and RF (green = active)

The configured movement sequence is started by **'START'**.

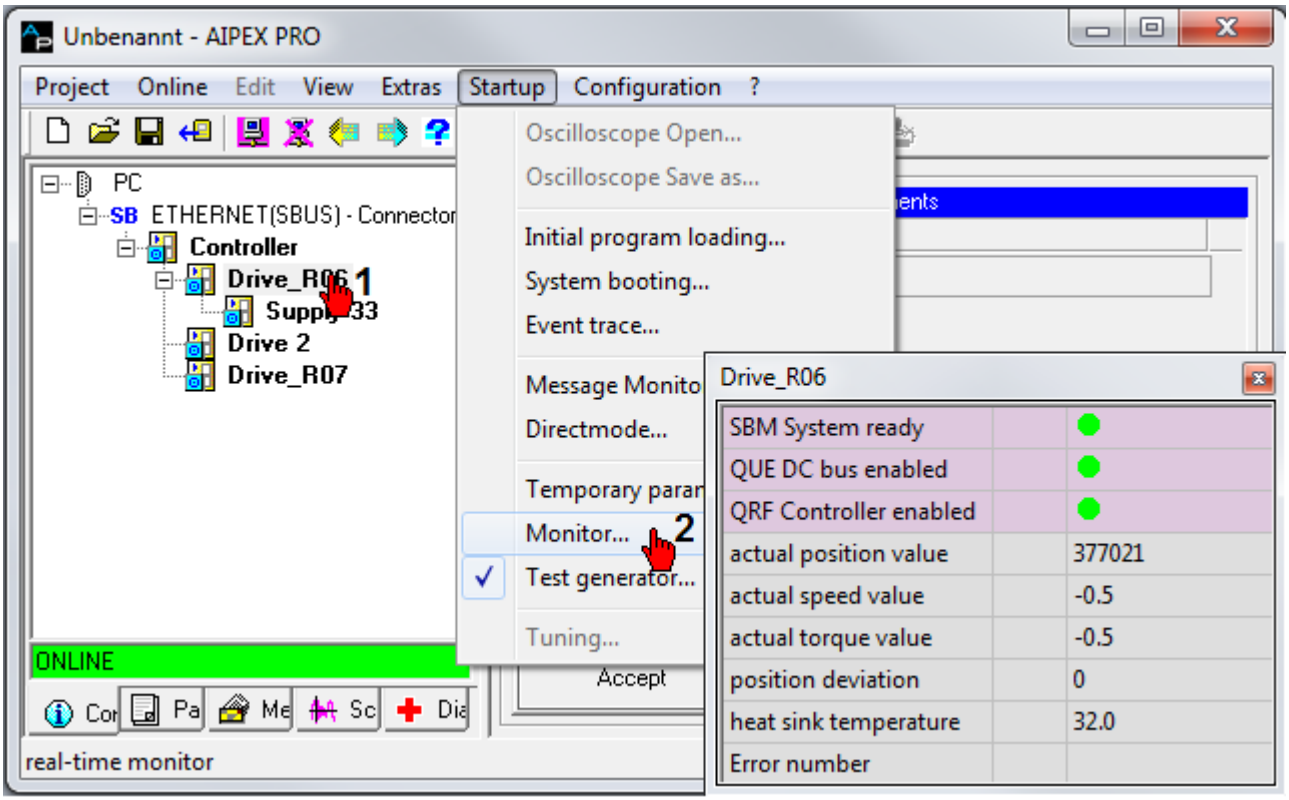


The movement can be interrupted by **'STOP'** and continued by **'START'**.

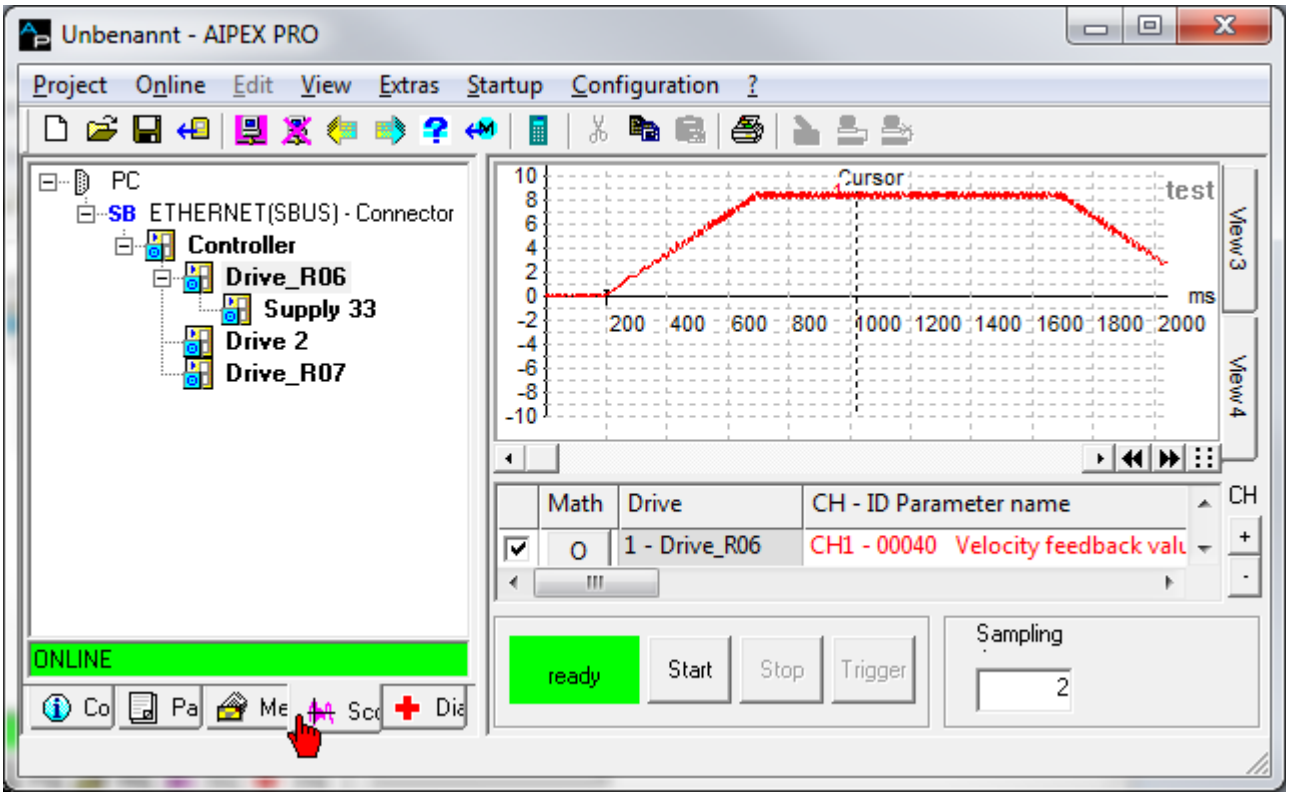
'STOP' and subsequent **'RESET'** terminates the sequence.

Display actual values

By means of **'Monitor...'**, you can display the controller state and actual values.
Check whether a positive setpoint causes a positive speed and an increasing position value. A positive rotation direction means clockwise rotation with view to the motor shaft



By means of the AIPEX PRO oscilloscope, you can store and interpret the movement.



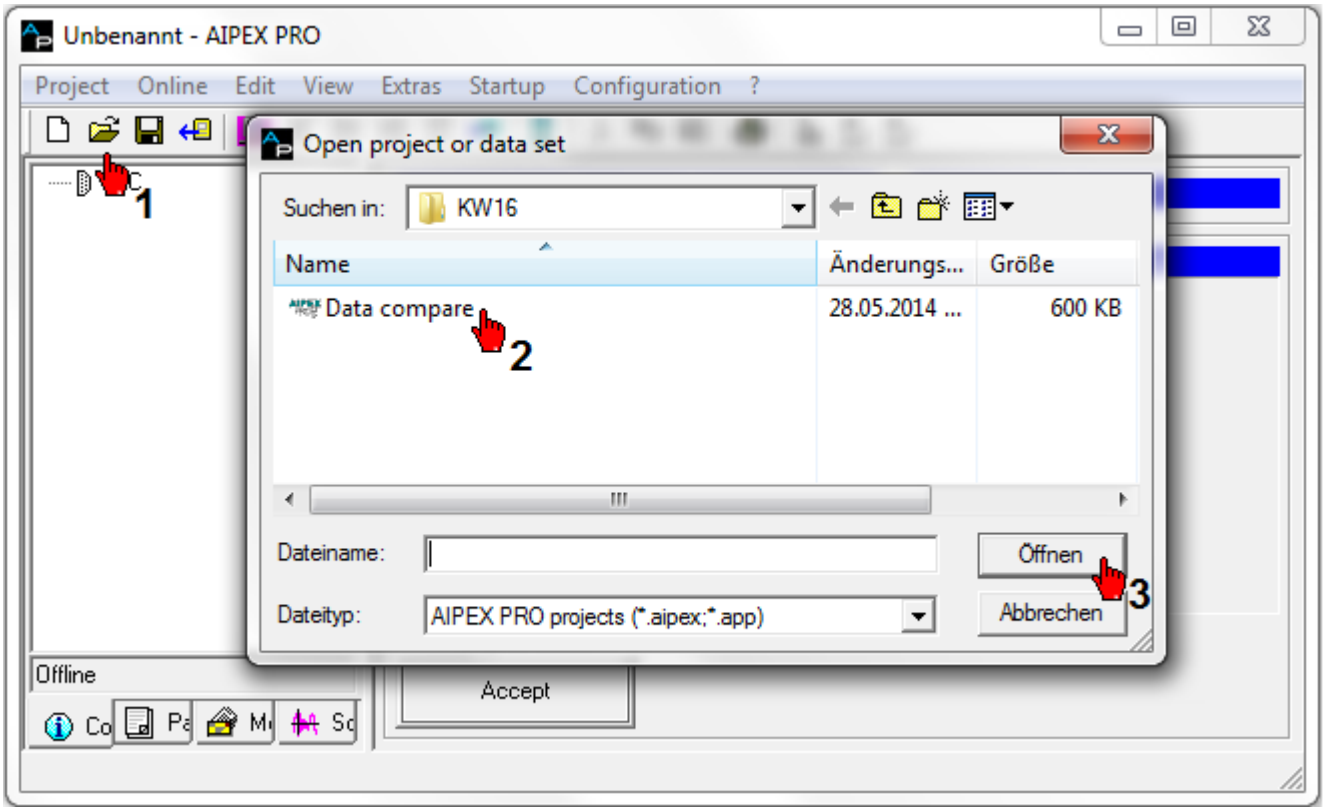
Scope configuration help:

Siehe 'Scope - Manual' auf Seite 90.

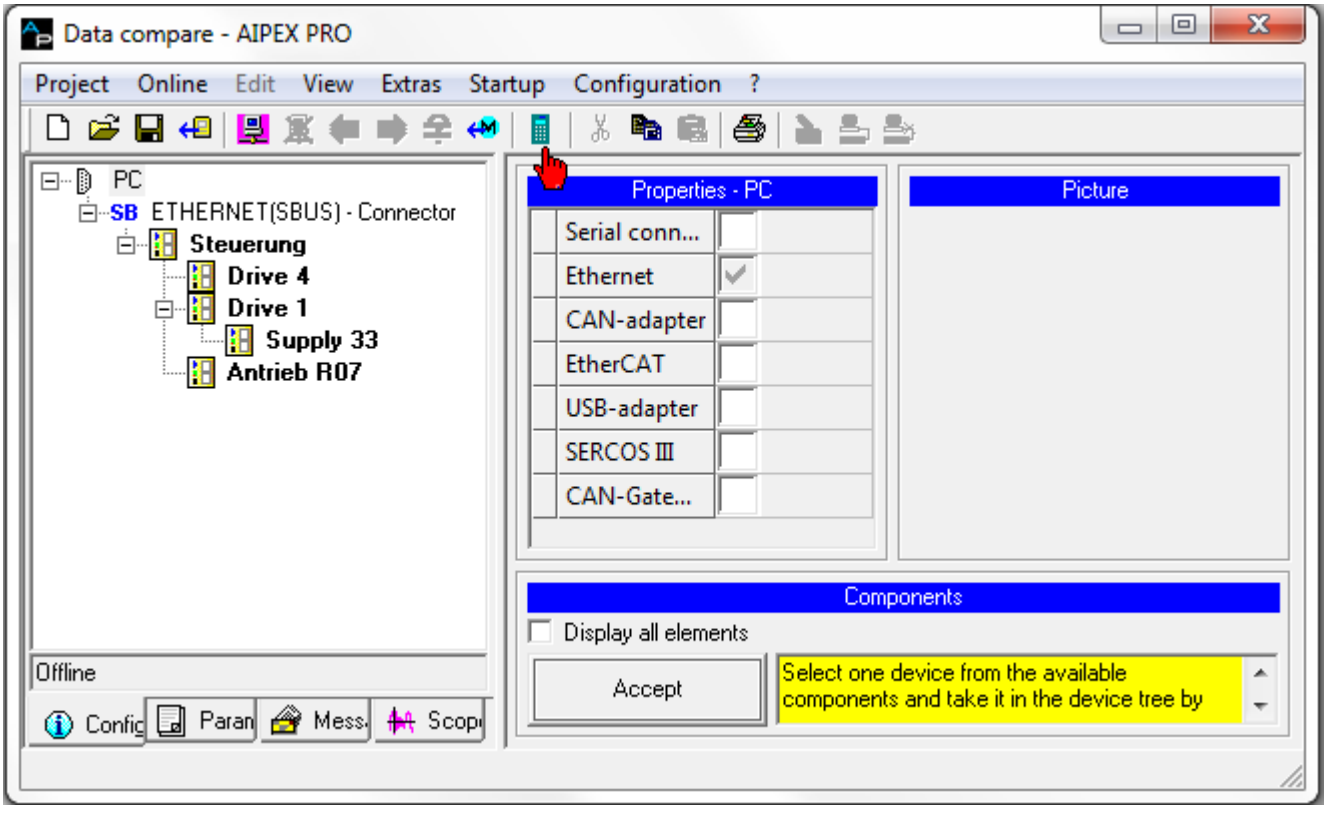
3.7.11 Transferring an offline project to a device

The following example describes how to open a parameter set and transfer the dataset to an AMK device.

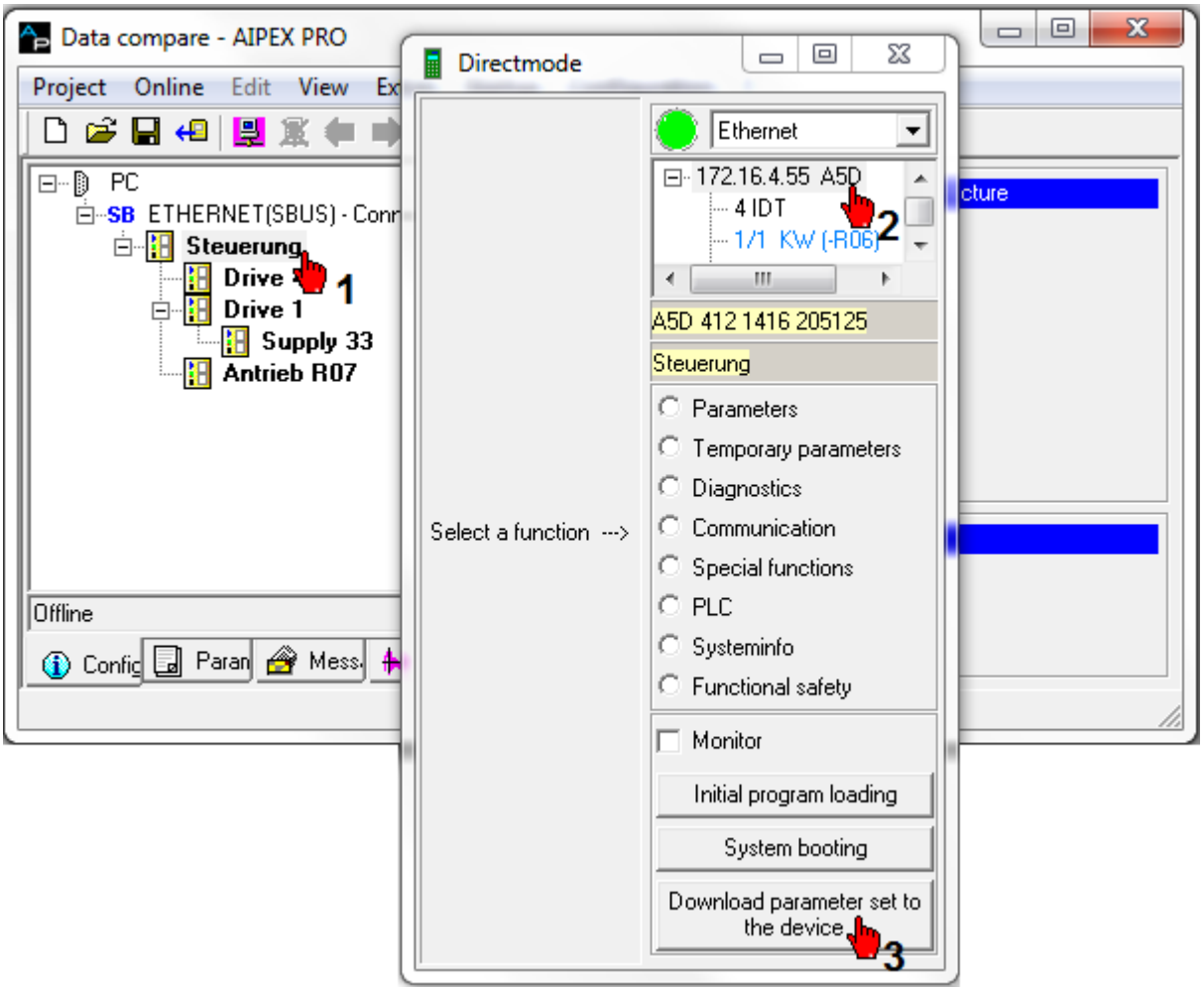
Click on the **'Open'** button to open the 'Open project or dataset' dialog box. Select the dataset you require.



Open the 'Direct mode'.



The dataset of the device selected in the device tree is transferred to the device selected in the direct mode.
For this purpose, a physical connection must exist between the PC and the device (direct connection or via fieldbus).
Select the interface used. An interface is active if its status is green.
The transfer is started with **'Download parameter set to the device'**.



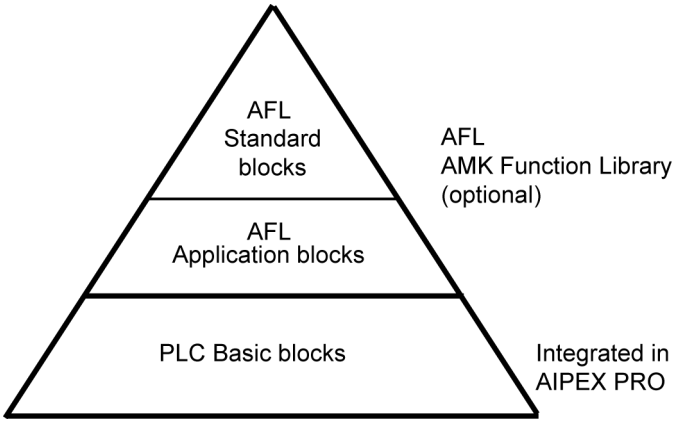
4 AFL (AMK Function Libraries)

4.1 AmkAfl Standard blocks

4.1.1 Introduction AFL Standard blocks for CODESYS V3

With AFL standard blocks the functionality of the AMK devices can be integrated into a PLC project. Programming time by using the AFL standard modules is significantly reduced. The AFL standard modules consist of a plurality of function blocks from different AMK libraries.

4.1.1.1 Overview of the AMK libraries



The AFL Standard blocks consist of AFL Application blocks and they are a part of the AFL Function Library.

Siehe 'AMK AFL Application blocks' auf Seite 348.

The AFL Application blocks consist of PLC Basic blocks (See document Software description AmkLibraries (Part no. 205210))

Example based on standard block 'STANDARD AXIS':

AFL standard blocks

The standard block 'STANDARD_AXIS' (FB) consists of following AFL application blocks 'MANUAL_JOG_VAJ', 'POSITION_ABSOLUT_VAJ', 'MANUAL_VELOCITY', 'POSITION_HOMING_FIXED_STOP' and some more from AMK AFL Library (AmkAfl.lib). The source code of the AFL Standard block is user-editable.

AFL application blocks

AMK support with AFL Application blocks complex functions. They consists of PLC Basic blocks. The functionality of AFL Application blocks is predefined and can not be changed.

Example: MANUAL_JOG_VAJ (FB) from AMK AFL Library (AmkAfl.lib)

The function block 'MANUAL_JOG_VAJ' realises the jog operation (plus/minus) in position control. In addition to position, speed and acceleration, the user can also specify the jerk.

PLC basic function blocks

The function block 'MANUAL_JOG_VAJ' consists of following PLC Basic function blocks 'VGEN_AJ' und 'RATIO_INC' from library AmkBase.lib. The functionality of PLC Basic blocks is predefined and can not be changed.

Overview AFL Function library's and documentation:

The AFL Function Library must be installed separately to AIPEX PRO. The version of the AFL Function Library depends on the used CODESYS version.

Version overview

AIPEX PRO + integrated CODESYS V3 version + compatible AFL Version

AIPEX PRO version	CODESYS V3 version	CODESYS profile	compatible AFL version
3.04	3.5.10.4	CODESYS V3.5 **SP10** Patch 4 AIPEX PRO	AFL V4 Version 3.5.5.0 2015/41 (part-no. 206004)
	3.5.5.5	CODESYS V3.5 **SP5** Patch 5 AIPEX PRO	
	3.5.3.6	CODESYS V3.5 **SP3** Patch 6 AIPEX PRO	AFL V4 Version 3.5.3.0 2014/06 (part-no. 204786)
3.03	3.5.5.5	CODESYS V3.5 **SP5** Patch 5 AIPEX PRO	AFL V4 version 3.5.5.0 2015/41 (part-no. 206004)
3.02	3.5.3.6	CODESYS V3.5 **SP3** Patch 6 AIPEX PRO	AFL V4 version 3.5.3.0 2014/06 (part-no. 204786)
3.01			
3.00			

Steuerung	Firmware Version	CODESYS V3 Profil
iSA / A4	≥ 4.22	CODESYS V3.5 SP10 Patch 4 AIPEX PRO
iSA / A4	≤ 4.21	CODESYS V3.5 SP5 Patch 5 AIPEX PRO
A5 / A6	alle	CODESYS V3.5 SP10 Patch 4 AIPEX PRO (recommended) (with restrictions[1)])
		CODESYS V3.5 SP5 Patch 5 AIPEX PRO (with restrictions[1)])
		CODESYS V3.5 SP3 Patch 6 AIPEX PRO

1) New features that affect the runtime system are not supported.

Corresponding CODESYS version must also be installed when installing AIPEX PRO.

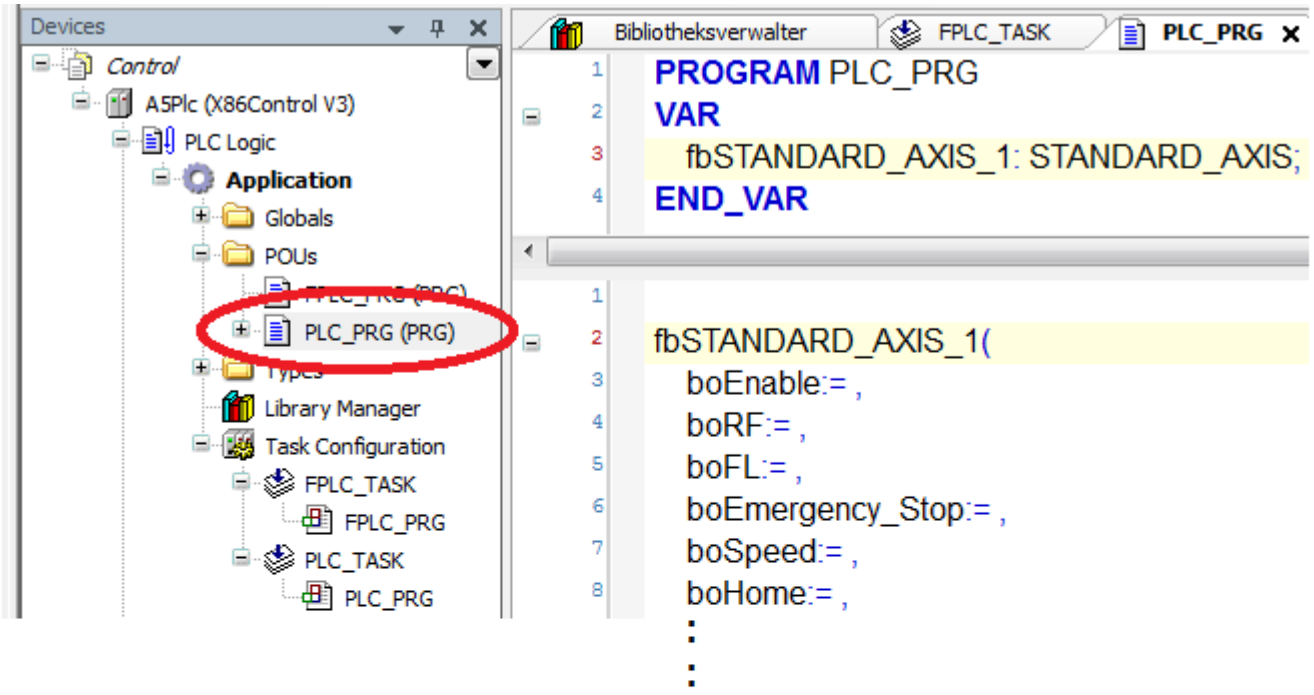
4.1.1.2 Using the AFL Standard blocks

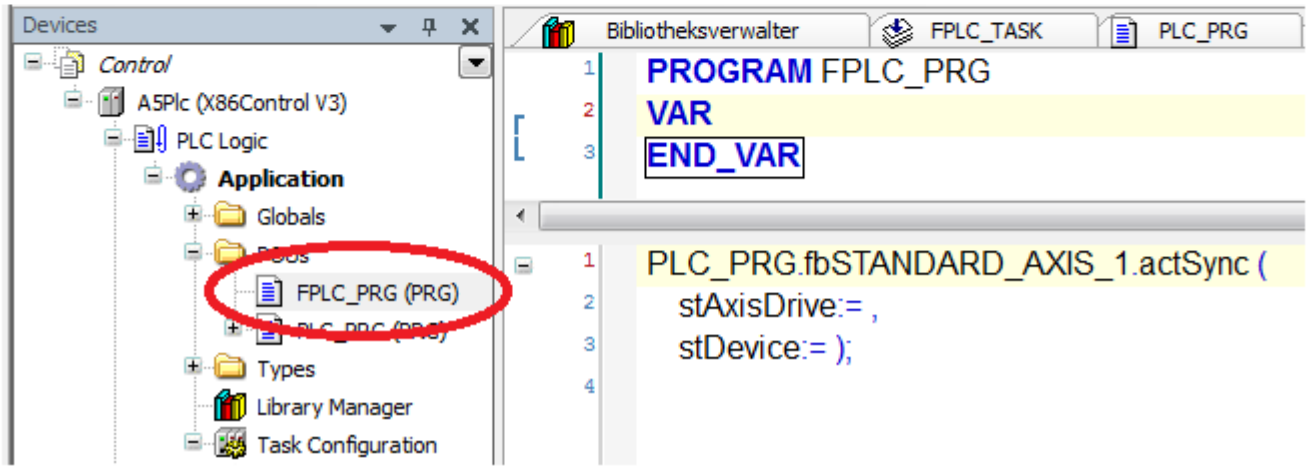
The AFL Standard blocks allow the user to easily access to all standard functions of the controllers, power supply's and axes. The predefined structures make it possible to access the setpoint/actual values, control/status bits, parameters and diagnostic messages. Advanced functions such as follow-up axis, gear ratios, winders, print mark control and table editing (cams) are available. The functionality will be continuously expanded.

The AFL Standard blocks will be imported inside the PLC project. The source code is open and can be edited by the user for customer specific applications. The AFL Standard blocks are called in the CODESYS program block 'PLC_PRG'. Axis blocks also include a synchronous action (actSync). The synchronous action is called in 'FPLC_PRG'.

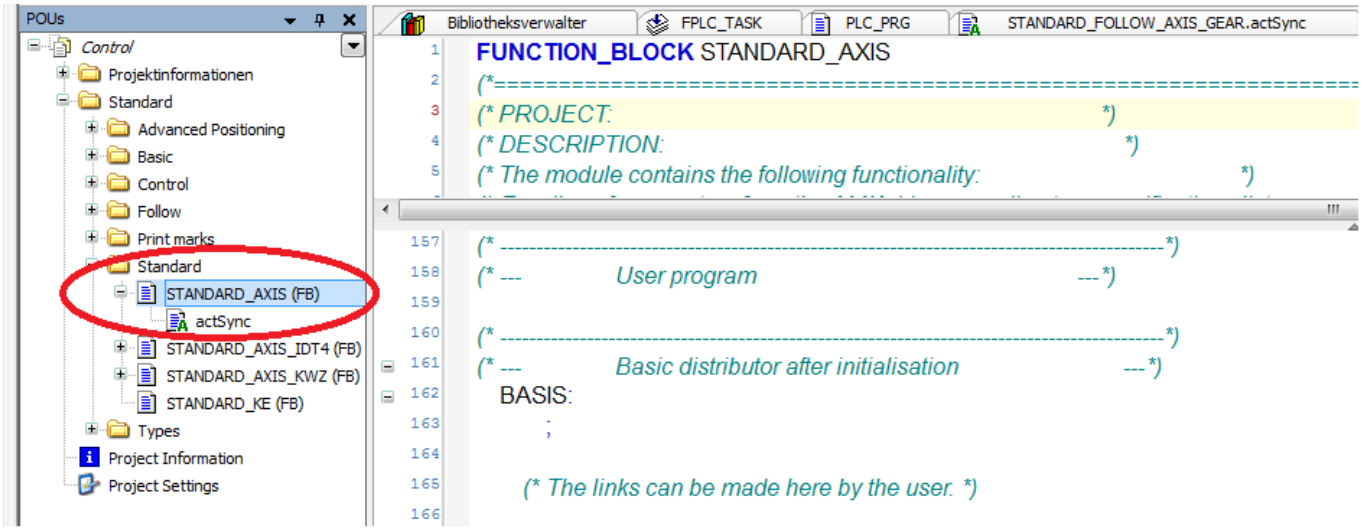
Example: Standard instanced block type 'STANDARD_AXIS'

Program block 'PLC_PRG':	Freewheeling task for asynchronous functionality / call and declaration 'STANDARD_AXIS'
Program block 'FPLC_PRG':	External event task, synchronized to ID2 'SERCOS cycle time' for synchronous functionality / call synchronous action (actSync)





Editable source 'STANDARD_AXIS'



A PLC project can be enlarged AFL Application blocks and PLC Basic function blocks.

Make sure that the functionality of used function blocks do not contradict each other. Eg. several blocks in a project that can set or reset the control signal 'RF Controller enable'.

4.1.1.3 Program structure with CODESYS V3 and AFL Standard blocks

Insert for each device (supply, control, drives ...), on that you want to have access with the PLC, an AFL Standard block.

The AFL Standard blocks are called in the CODESYS program block 'PLC_PRG'. Axis blocks also has a synchronous action (act Sync). The synchronous action must be called in 'FPLC_PRG'.

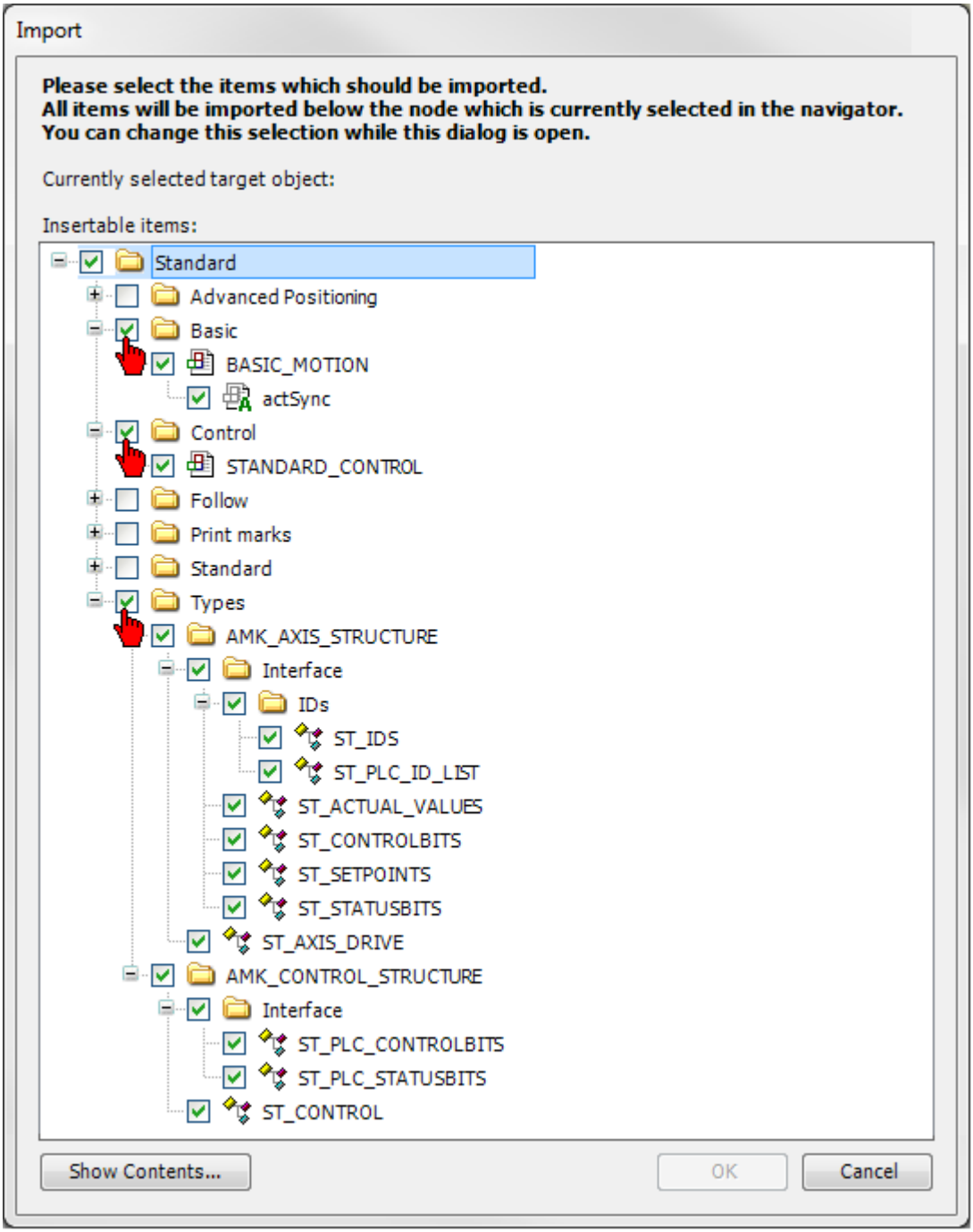
The FB 'BASIC_MOTION' is integrated into each standard axis block. That means FB 'BASIC_MOTION' must not be called separately in the PLC program. The interface between the device and AFL Standard block is realized with the Variable 'ST_DEVICE'. 'SET_CONTROL' summarizes the control-specific and 'ST_AXIS_DRIVE' the axis-specific data together.

The AMK file 'V3_Import.export' contains the AFL Standard blocks and associated structures. The contents of the folder 'Basic', 'Control' and 'Type' must necessarily be imported for each CODESYS V3 project. Using CODESYS V3 the AFL Standard blocks are imported in the current PLC project with the menu 'Project' → 'Import'.

The visualization elements of AFL Standard blocks are contained in an additional library.

Requirements:
AMK software AFL Library for CODESYS V3 AMK part no. O913 installed.
AFL template selected. Siehe 'Basic AIPEX PRO Settings' auf Seite 639.

CODESYS V3 Menü 'Projekt' → 'Import'
(C:\Program Files (x86)\3S CODESYS\AmkAddition\Imports\Application\V3_import\V3_Import.export)



Overview of standard blocks and the associated macros:

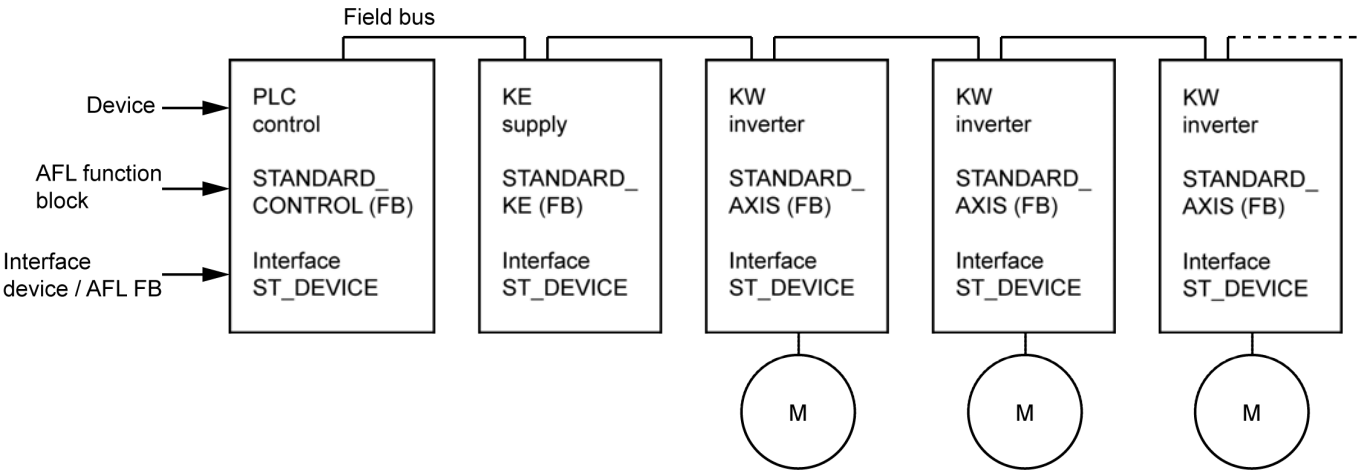
Standard blocks	Macro folder	Macro	For device
STANDARD_CONTROL (FB) [1]	AflImportStandard	AflImportStandardControl	AMKAMAC A4, A5, A6
STANDARD_CONTROL _iSA (FB) [1]	AflImportStandard	AflImportStandardControl_iSA	AMKASMART iSA
STANDARD_KE (FB)	AflImportStandard	AflImportStandardKE	AMKASYN KE/KEN/KES
STANDARD_AXIS_IDT4 (FB) [2]	AflImportStandard	AflImportStandardAxisIDT4	AMKASMART iDT4
STANDARD_AXIS_KWZ (FB) [2]	AflImportStandard	AflImportStandardAxisKWZ	AMKASYN KWZ
STANDARD_AXIS_ihX (FB) [2]	AflImportStandard	AflImportStandardAxis_ihX	AMKASMART ihX
STANDARD_AXIS (FB) [2]	AflImportStandard	AflImportStandardAxis	AMKASYN KW/KWD (KW-R06, KW-R07, KW-R16, KW-R17, KW-R24-R, KW-R25, KW-R26, KW-R27) AMKASMART iDT5 AMKASMART ihX [3]
STANDARD_AXIS_ POSITION_INTERPOSED (FB) [2]	AflImportAdvanced-Positioning	AflImportAdvancedPositioning	
STANDARD_AXIS_ POSITION_TO_THE_MARK (FB) [2]	AflImportPrintMarkAxis	AflImportStandardAxis-PrintMarkPositionToTheMark	
STANDARD_AXIS_ PRINTMARK_INSETTER_ INTERVAL (FB) [2]	AflImportPrintMarkAxis	AflImportStandardAxis-PrintMarkInsetterInterval	
STANDARD_AXIS_ PRINTMARK_INSETTER_ CONT (FB) [2]	AflImportPrintMarkAxis	AflImportStandardAxis-PrintMarkInsetterCont	
STANDARD_AXIS_ PRINTMARK_REGISTER_ CONTROL_CONT (FB) [2]	AflImportPrintMarkAxis	AflImportStandardAxis-PrintMarkRegisterControlCont	
STANDARD_AXIS_ PRINTMARK_REGISTER_ CONTROL_DISCONT (FB) [2]	AflImportPrintMarkAxis	AflImportStandardAxis-PrintMarkRegisterControlDiscont	
STANDARD_FOLLOW_AXIS_GEAR (FB) [2]	AflImportFollowAxis	AflImportFollowAxisGear	
STANDARD_FOLLOW_AXIS_ CONTINUOUS_TABLE1_TABLE2 (FB) [2]	AflImportFollowAxis	AflImportFollowAxis-ContinuousTable1Table2	
STANDARD_FOLLOW_AXIS_ SWITCHING_TABLE_WITH_ OUTGEAR (FB) [2]	AflImportFollowAxis	AflImportFollowAxis-SwitchingTableWithOutgear	
STANDARD_FOLLOW_AXIS_ SWITCHING_TABLE1_TABLE2 (FB) [2]	AflImportFollowAxis	AflImportFollowAxis-SwitchingTable1Table2	
STANDARD_FOLLOW_AXIS_ TABLE_CONTINUOUS (FB) [2]	AflImportFollowAxis	AflImportFollowAxis-ContinuousTable	
STANDARD_FOLLOW_TABLE_ ENGAGE_DISENG_AXIS (FB) [2]	AflImportFollowAxis	AflImportFollowAxis-TableEngageDiseng	
STANDARD_FOLLOW_TABLE_ START_AUTOSTOP_AXIS (FB) [2]	AflImportFollowAxis	AflImportFollowAxis-TableStartAutoStop	
STANDARD_WINDER_DANCER_ AXIS (FB) [2]	AflImportStandardWinder	AflImportStandardWinderDancer	
STANDARD_WINDER_TORQUE_ AXIS (FB) [2]	AflImportStandardWinder	AflImportStandardWinderTorque	

[1] All standard blocks of the type CONTROL require the 'AMK CONTROL STRUCTURE' 'ST_CONTROL (STRUCT)'. 'ST_CONTROL (STRUCT)' is imported with the macro AflImportBasicControl from the folder AflImportBasic. 'ST_CONTROL (STRUCT)' needs imported 1 x per PLC project.

[2] All standard blocks of the type AXIS call the FB 'BASIC_MOTION'. Der FB 'BASIC_MOTION' s imported with the macro AflImportBasicDrive from the folder AflImportBasic. FB 'BASIC_MOTION' needs imported 1 x per PLC project.

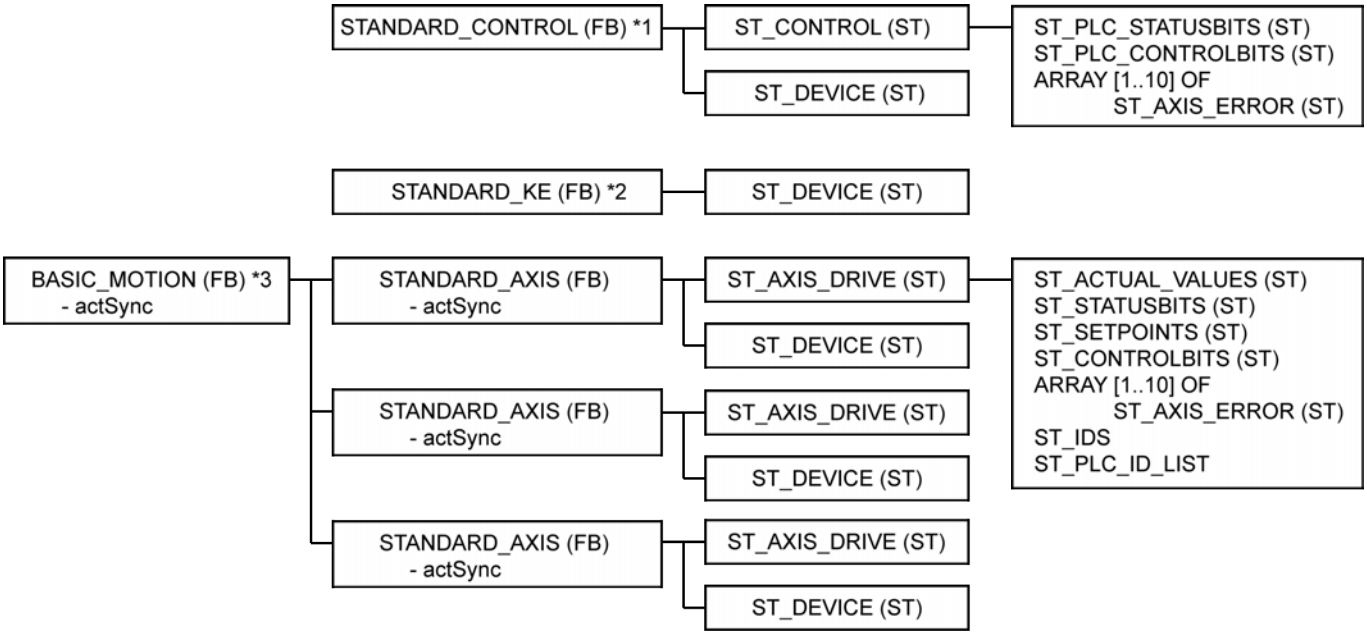
[3] All standard blocks with expanded functionality can be used with a modification for the ihX. Siehe 'Modified 'Standard axes' for ihX drives' auf Seite 208.

Application example:



Structure PLC program:

The macros 'AflImport ...' Importing the AFL Standard blocks and structures in the current PLC project.



*1 AflImportBasicControl import the structure 'ST_CONTROL'. It has to call one time for each project.

*2 Only if you use power supply module KE/KEN/KES (exception KEN 05-xx).
The devices AMKASMART iC and AMKASYN KEN 05-xx charge automatically the DC-Bus to supply the connected inverters.

*3 AflImportBasicDrive import 'BASIC_MOTION' (FB). It has to call one time for each project. 'BASIC_MOTION' must not be called by the user. It will be called automatically by the function block 'STANDARD_AXIS'.

You have to call for each project an instance of 'STANDARD_CONTROL' (for A-series) or 'STANDARD_CONTROL_iSA (for iSA) must be necessarily used in each PLC project.

The combination between AFL Standard block 'STANDARD_AXIS' and ID32796 'Source RF' Code 0: 'Controller enable (RF) via binary input' is not allowed.

The control signal RF 'inverter on' must be set via a function block.

Allowed:

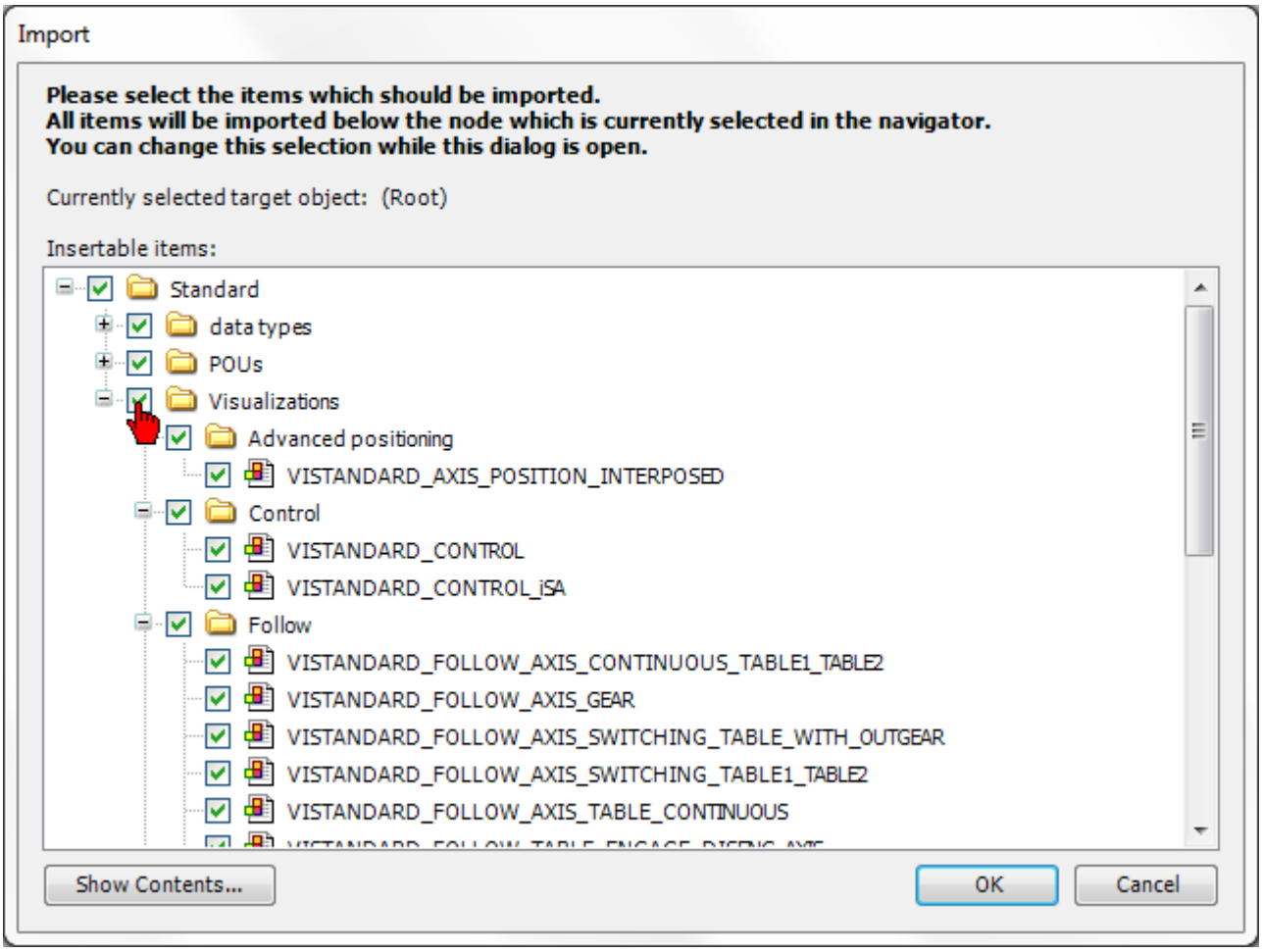
Code 5: 'Controller enable via EtherCAT'

Code 25: 'RF via EtherCAT AND-linked with the binary input RF'

4.1.1.4 Visualization Standard blocks

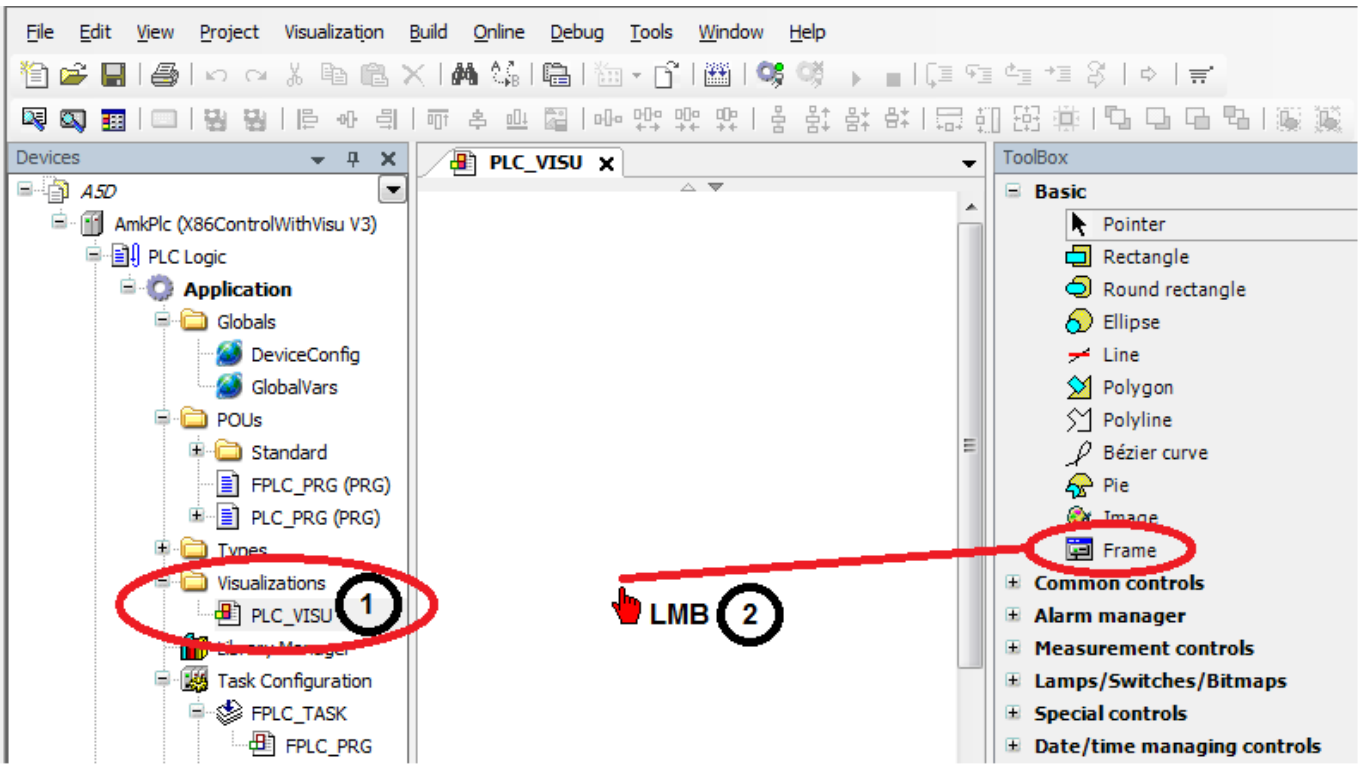
The Standard block visualizations will be integrated (with CODESYS V3) inside the actual project by using the 'Import' function. CODESYS menu 'Project' → 'Import'.

C:\Programs (x86)\3S CODESYS\AmkAddition\Imports\Application\V3_import.

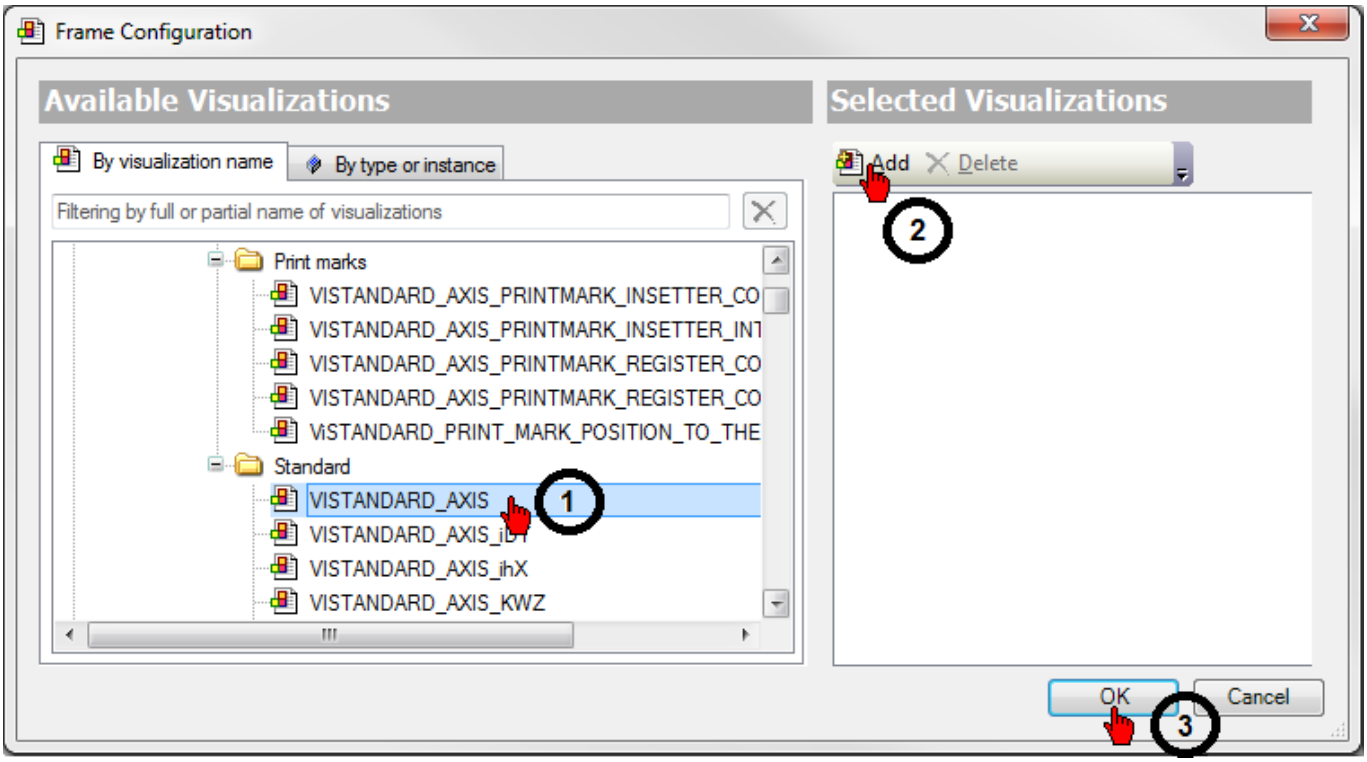


(The AFL Application block visualizations contained in an additional library. Siehe 'Visualization of AFL blocks' auf Seite 806.)

The visualizations are integrated with Vis-tool 'Frame'.

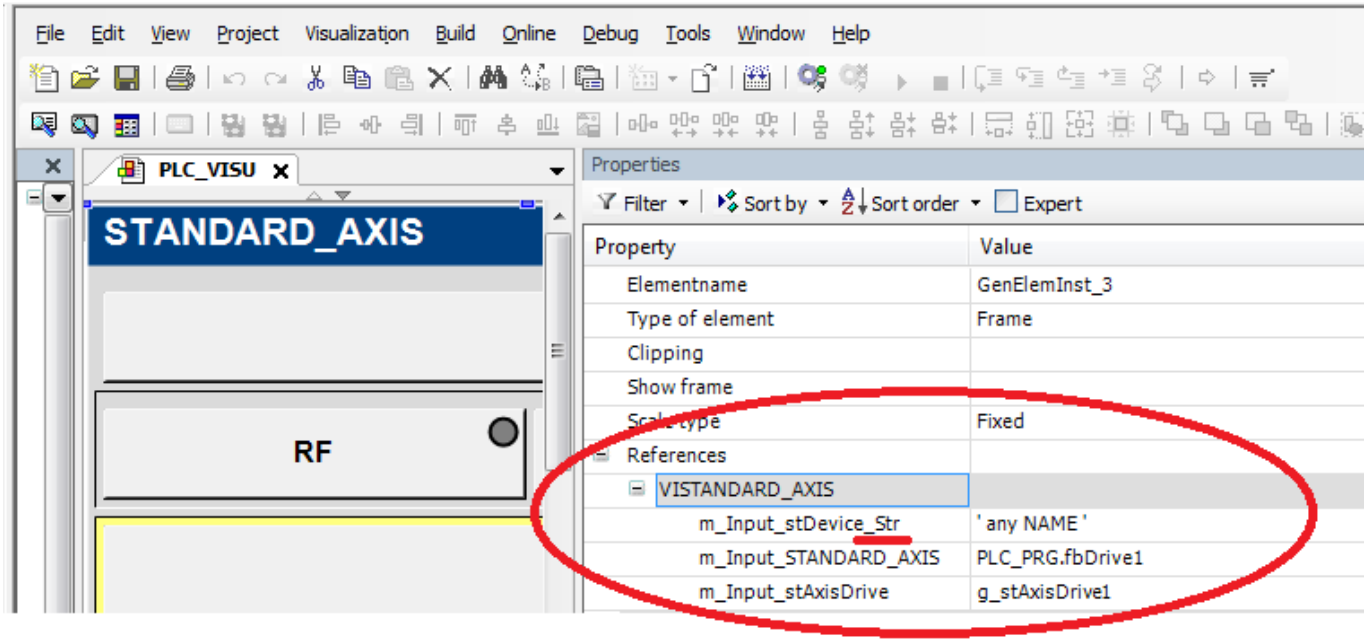


Chose the desired visualization with the button 'add'.

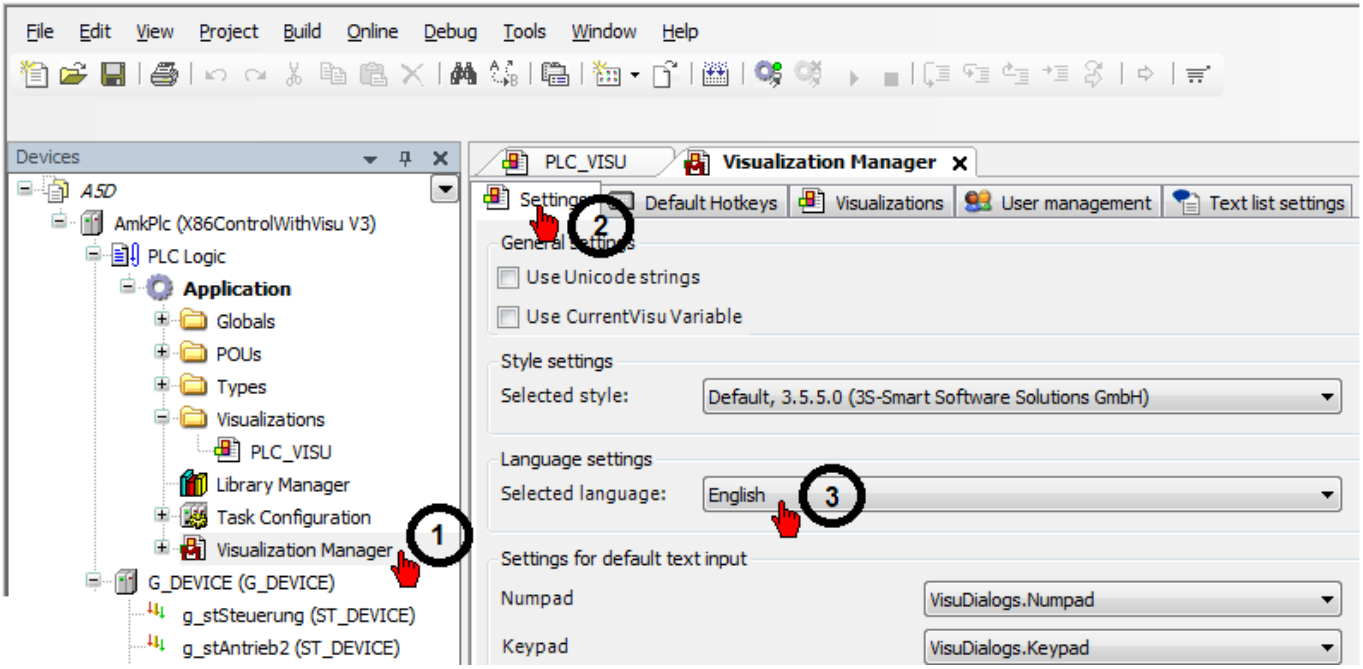


The inserted frame in the visualization must be linked to the instanced AFL Standard block and the associated structure ST_AXIS_DRIVE.

With m_Input_stDevice_Str specified a name (STRING) which will be displayed in the visualization.



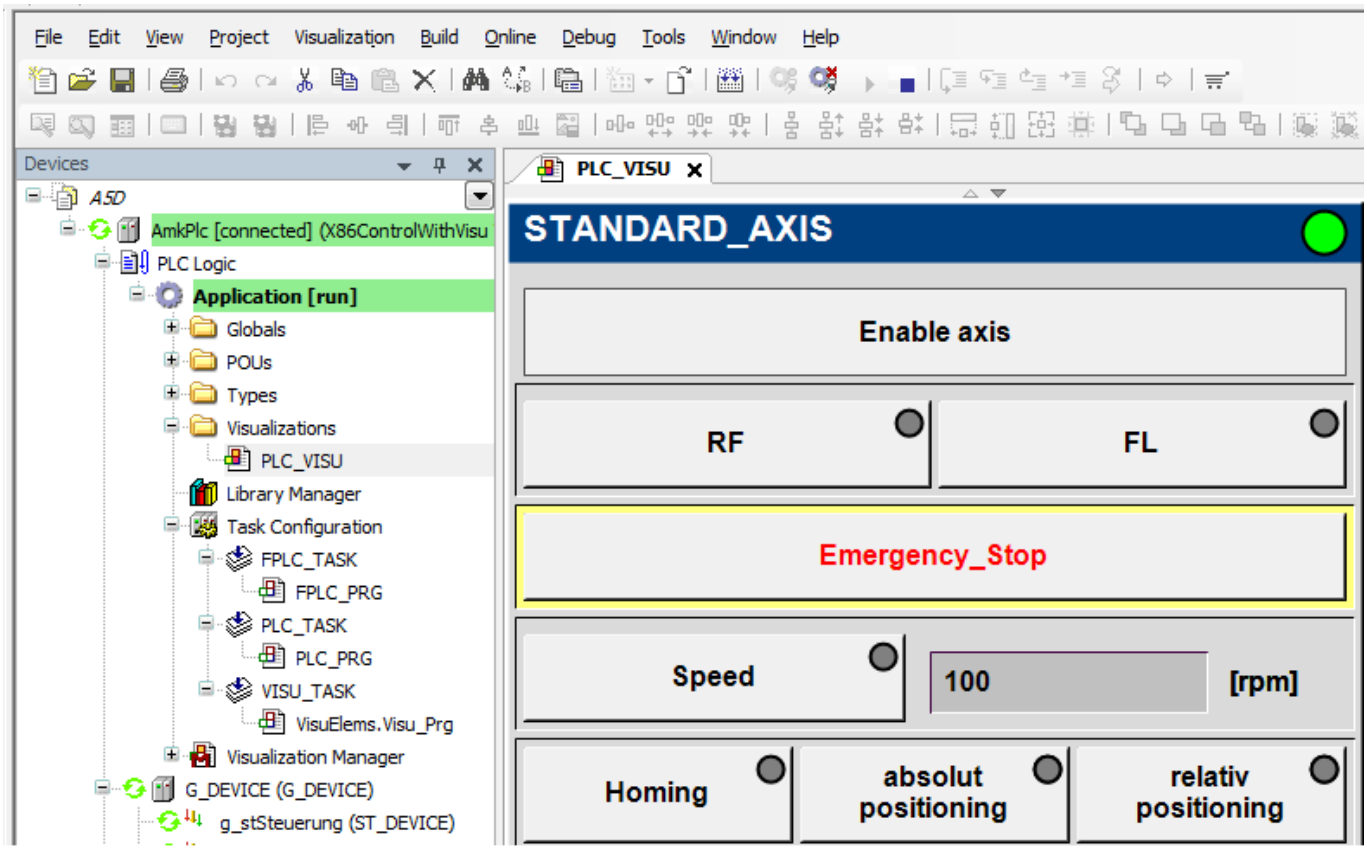
Selection visualization language.



Language-dependent texts are displayed only in the online mode.

The displayed texts (German and English) are stored in a text list: folder 'POUs' → 'Standard' → 'Visualizations' → 'Standard' → 'TextListStandard'

Display in 'RUN mode'



4.1.1.5 Overview - AFL Standard blocks and axis structures

The AFL Standard blocks will be integrated (with CODESYS V3) inside the actual project by using the 'Import' function. See CODESYS menu 'Project' → 'Import'.

STANDARD_CONTROL (FB)

(Import Ordner: POUs → Control)

The following basic functionalities for the controller are available:

- Status bits (SBM System ready message, QFL Acknowl. error cleared, network and bus information)
- The controlling of FL
- The reading out of diagnostic messages

Siehe 'STANDARD_CONTROL (FB)' auf Seite 195.

STANDARD_CONTROL_iSA (FB)

(Import Ordner: POUs → Control)

The following basic functionalities for the controller with supply are available:

- The importing of status bit information (network ready, bus ready, bus warning, bus error, SBM, QUE and QFL)
- The controlling of FL
- The reading out of diagnostic messages

Siehe 'STANDARD_CONTROL_iSA (FB)' auf Seite 197.

STANDARD_KE (FB)
(Import Ordner: POUs → Standard)
The following basic functionalities for the input are available:

- Status bits (QUE Acknowl. DC bus ON, SBM System ready message)
- Control bits (UE DC bus ON, FL Clear error)
- Output of diagnostics number

Siehe 'STANDARD_KE (FB)' auf Seite 199.

STANDARD_AXIS (FB)
STANDARD_IDT4 (FB)
STANDARD_ihX (FB)
STANDARD_KWZ (FB)
(Import Ordner: POUs → Standard)
The following basic functionalities for the axis are available:

Inputs/outputs on the function block

- Control bits (RF Controller enable, FL Clear error)
- Emergency stop
- Speed control
- Homing cycle
- Positioning (absolute/relative)
- Jog mode (minus/plus)
- Status signals of axis functions

Access via program code

- Configuration of status messages (ID26 'Configuration status bits')
- Reading of parameters according to a specifications list
- Actual position cached
- Control with feedforward (torque and speed)

Siehe 'STANDARD_AXIS (FB) (STANDARD_AXIS_KWZ, STANDARD_AXIS_IDT4, STANDARD_AXIS_ihX)' auf Seite 201.

4.1.1.5.1 Advanced standard axes

The advanced axes feature the same basic functionalities as function block 'STANDARD_AXIS'.

It is possible to implement a dancer winder or sensorless rewinder and unwinder with 'Winder' type axes.
It is possible to implement a printing mark control or interposed positioning with 'Position' type axes.
It is possible to implement a printing mark control with 'Printmark' type axes.
'Follow' type axes are available. These follow a setpoint value (master) synchronously.
In addition tables can also be calculated for cam functions, for example.

STANDARD_AXIS_POSITION_INTERPOSED (FB)
(Import Ordner: POUs → Advanced Positioning)
Expanded functionality:

- Interposed positioning

Siehe 'STANDARD_AXIS_POSITION_INTERPOSED (FB)' auf Seite 208.

STANDARD_AXIS_POSITION_TO_THE_MARK (FB)
(Import Ordner: POUs → Print marks)
Expanded functionality:

- Printing mark control

Siehe 'STANDARD_AXIS_POSITION_TO_THE_MARK (FB)' auf Seite 217.

STANDARD_AXIS_PRINTMARK_INSETTER_CONT (FB)

(Import Ordner: POUs → Print marks)

Expanded functionality:

- Print mark control: The position of the web to be treated is controlled in relation to a master format (insetting)

Siehe 'STANDARD_AXIS_PRINTMARK_INSETTER_CONT (FB)' auf Seite 226.

STANDARD_AXIS_PRINTMARK_INSETTER_INTERVAL (FB)

(Import Ordner: POUs → Print marks)

Expanded functionality:

- Print mark control: The position of the web to be treated is controlled in relation to a master format (insetting)

Siehe 'STANDARD_AXIS_PRINTMARK_INSETTER_INTERVAL (FB)' auf Seite 237.

STANDARD_AXIS_PRINTMARK_REGISTER_CONTROL_CONT (FB)

(Import Ordner: POUs → Print marks)

Expanded functionality:

- Print mark control: The position of the treating tool (e.g. cutter, punch) is controlled in relation to a master format (register control)

Siehe 'STANDARD_AXIS_PRINTMARK_REGISTER_CONTROL_CONT (FB)' auf Seite 249.

STANDARD_AXIS_PRINTMARK_REGISTER_CONTROL_DISCONT (FB)

(Import Ordner: POUs → Print marks)

Expanded functionality:

- Print mark control: The position of the treating tool (e.g. cutter, punch) is controlled in relation to a master format (register control)

Siehe 'STANDARD_AXIS_PRINTMARK_REGISTER_CONTROL_DISCONT (FB)' auf Seite 260.

STANDARD_FOLLOW_AXIS_GEAR (FB)

(Import Ordner: POUs → Follow)

Expanded functionality:

- The slave axis continuously follows a master.
- A gear ratio can be defined.

Siehe 'STANDARD_FOLLOW_AXIS_GEAR (FB)' auf Seite 271.

STANDARD_FOLLOW_AXIS_CONTINUOUS_TABLE1_TABLE2 (FB)

(Import Ordner: POUs → Follow)

Expanded functionality:

- The slave axis continuously follows a freely selectable table.
- At the end of the table, a table switchover can be carried out.

Siehe 'STANDARD_FOLLOW_AXIS_CONTINUOUS_TABLE1_TABLE2 (FB)' auf Seite 279.

STANDARD_FOLLOW_AXIS_SWITCHING_TABLE_WITH_OUTGEAR (FB)

(Import Ordner: POUs → Follow)

Expanded functionality:

- The slave axis continuously follows a freely selectable table.
- At the end of the table, a table switchover can be carried out.
- The table can be modified via a gear ratio (compress and extend).

Siehe 'STANDARD_FOLLOW_AXIS_SWITCHING_TABLE_WITH_OUTGEAR (FB)' auf Seite 288.

STANDARD_FOLLOW_AXIS_SWITCHING_TABLE1_TABLE2 (FB)

(Import Ordner: POUs → Follow)

Expanded functionality:

- The slave axis continuously follows a freely selectable table with a phasing-in and phasing out table.
- At the end of the table, a table switchover can be carried out.

Siehe 'STANDARD_FOLLOW_AXIS_SWITCHING_TABLE1_TABLE2 (FB)' auf Seite 296.

STANDARD_FOLLOW_AXIS_TABLE_CONTINUOUS (FB)

(Import Ordner: POUs → Follow)

Expanded functionality:

- The slave axis continuously follows a freely selectable table.

Siehe 'STANDARD_FOLLOW_AXIS_TABLE_CONTINUOUS (FB)' auf Seite 305.

STANDARD_FOLLOW_TABLE_ENGAGE_DISENG_AXIS (FB)

(Import Ordner: POUs → Follow)

Expanded functionality:

- The slave axis continuously follows a freely selectable table with a phasing-in and phasing out table.

Siehe 'STANDARD_FOLLOW_TABLE_ENGAGE_DISENG_AXIS (FB)' auf Seite 314.

STANDARD_FOLLOW_TABLE_START_AUTOSTOP_AXIS (FB)

(Import Ordner: POUs → Follow)

Expanded functionality:

- The slave axis continuously follows a freely selectable table with a phasing-in and phasing out table.

Siehe 'STANDARD_FOLLOW_TABLE_START_AUTOSTOP_AXIS (FB)' auf Seite 322.

STANDARD_WINDER_DANCER_AXIS (FB)

(Import Ordner: POUs → Winder)

Expanded functionality:

The function block 'STANDARD_WINDER_DANCER_AXIS' realizes a dancer winder with the following characteristics with following features:

- PID regulated dancer winder
- Operation as unwinder and rewinder
- Diameter calculation
- Search diameter at the start of the winder

Siehe 'STANDARD_WINDER_DANCER_AXIS (FB)' auf Seite 330.

STANDARD_WINDER_TORQUE_AXIS (FB)

(Import Ordner: POUs → Winder)

Expanded functionality:

- The function block 'STANDARD_WINDER_TORQUE_AXIS' realizes a sensorless rewinder and unwinder with following features:
- Torque controlled center winder
- Operation as rewinder and unwinder
- Diameter calculation
- Search diameter at the start of the winder
- Determine the reel of inertia of unknown diameter
- Synchronize onto a running web

Siehe 'STANDARD_WINDER_TORQUE_AXIS (FB)' auf Seite 339.

4.1.1.5.2 Structures

The following access options are available:

PLC controller (ST_CONTROL):

(Import Ordner: POUs → Data typs → AMK_CONTROL_STRUCTURE)

- Status bits (SBM System ready message, network status)
- Control bits (FL Clear error, Read diagnostic message)
- Diagnostic array (memory for ten diagnostic messages)

Siehe 'ST_CONTROL (ST)' auf Seite 189.

Axis (ST_AXIS_DRIVE):

(Import Ordner: POUs → Data typs → AMK_AXIS_STRUCTURE)

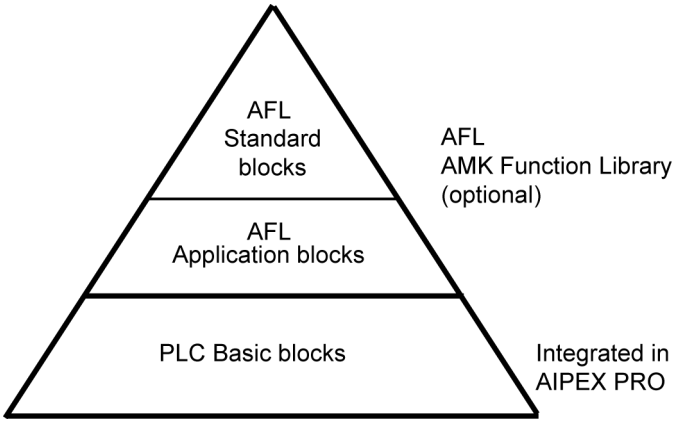
- Setpoint values (speed, position, torque, pilot control values)
- Actual values (speed, position, actual torque, probe values, etc.)
- Control bits (RF controller enable, FL Clear error, Read diagnostic message)
- Status bits (Speed window, Standstill window, etc.)
- Diagnostic array (memory for ten diagnostic messages)
- Read parameter list
- Configuration (PLC functions)

Siehe 'ST_AXIS_DRIVE (ST)' auf Seite 191.

4.1.2 Introduction AFL Standard blocks for CODESYS V2

With AFL standard blocks the functionality of the AMK devices can be integrated into a PLC project. Programming time by using the AFL standard modules is significantly reduced. The AFL standard modules consist of a plurality of function blocks from different AMK libraries.

4.1.2.1 Overview of the AMK libraries



The AFL Standard blocks consist of AFL Application blocks and they are a part of the AFL Function Library.

Siehe 'AMK AFL Application blocks' auf Seite 348.

The AFL Application blocks consist of PLC Basic blocks (See documentSoftware description AmkLibraries (Part no. 205210))

Example based on standard block 'STANDARD AXIS':

AFL standard blocks

The standard block 'STANDARD_AXIS' (FB) consists of following AFL application blocks 'MANUAL_JOG_VAJ', 'POSITION_ABSOLUT_VAJ', 'MANUAL_VELOCITY', 'POSITION_HOMING_FIXED_STOP' and some more from AMK AFL Library (AmkAfl.lib). The source code of the AFL Standard block is user-editable.

AFL application blocks

AMK support with AFL Application blocks complex functions. They consists of PLC Basic blocks. The functionality of AFL Application blocks is predefined and can not be changed.

Example: MANUAL_JOG_VAJ (FB) from AMK AFL Library (AmkAfl.lib)

The function block 'MANUAL_JOG_VAJ' realises the jog operation (plus/minus) in position control. In addition to position, speed and acceleration, the user can also specify the jerk.

PLC basic function blocks

The function block 'MANUAL_JOG_VAJ' consists of following PLC Basic function blocks 'VGEN_AJ' und 'RATIO_INC' from library AmkBase.lib. The functionality of PLC Basic blocks is predefined and can not be changed.

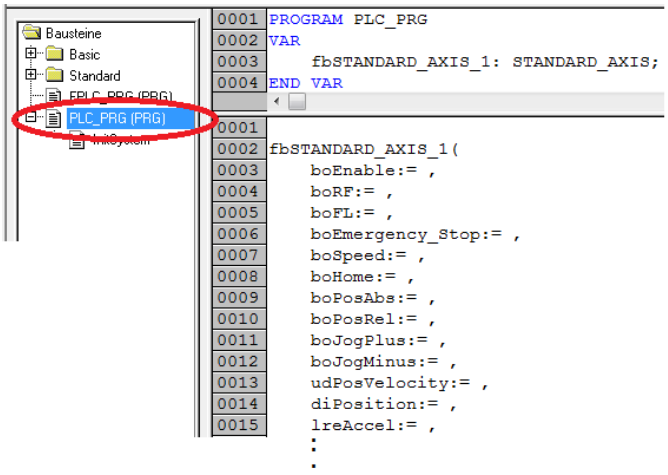
4.1.2.2 Using the AFL Standard blocks

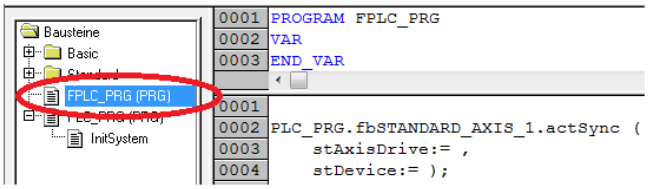
The AFL Standard blocks allow the user to easily access to all standard functions of the controllers, power supply's and axes. The predefined structures make it possible to access the setpoint/actual values, control/status bits, parameters and diagnostic messages. Advanced functions such as follow-up axis, gear ratios, winders, print mark control and table editing (cams) are available. The functionality will be continuously expanded.

The AFL Standard blocks will be imported inside the PLC project. The source code is open and can be edited by the user for customer specific applications. The AFL Standard blocks are called in the CODESYS program block 'PLC_PRG'. Axis blocks also include a synchronous action (actSync). The synchronous action is called in 'FPLC_PRG'.

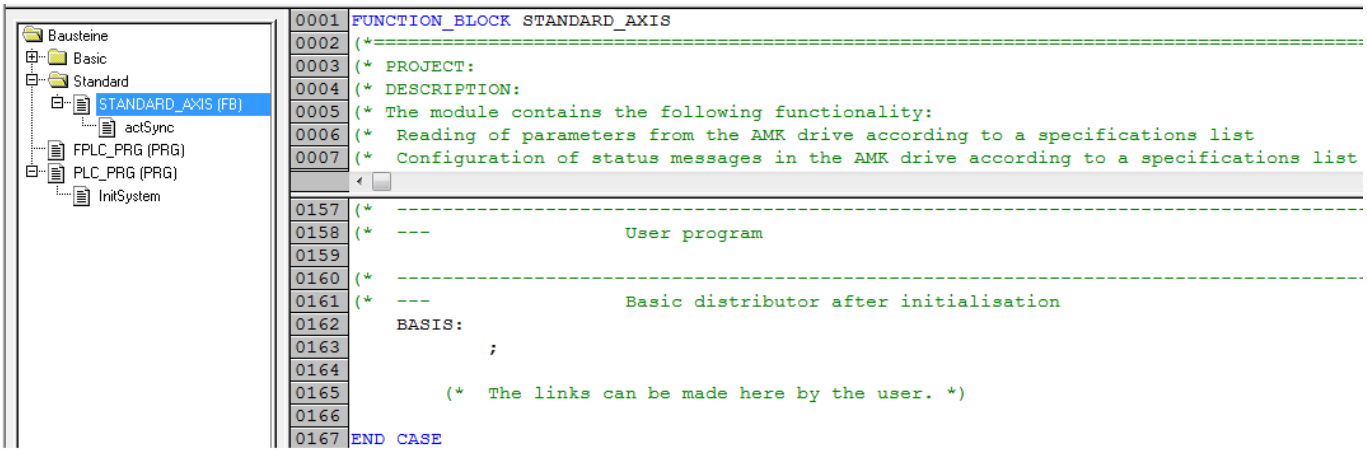
Example: Standard instanced block type 'STANDARD_AXIS'

Program block 'PLC_PRG':	Freewheeling task for asynchronous functionality / call and declaration 'STANDARD_AXIS'
Program block 'FPLC_PRG':	External event task, synchronized to ID2 'SERCOS cycle time' for synchronous functionality / call synchronous action (actSync)





Editable source 'STANDARD_AXIS'



A PLC project can be enlarged AFL Application blocks and PLC Basic function blocks.

Make sure that the functionality of used function blocks do not contradict each other. Eg. several blocks in a project that can set or reset the control signal 'RF Controller enable'.

4.1.2.3 Program structure with CODESYS V2 and AFL standard blocks

Insert for each device (supply, control, drives ...), on that you want to have access with the PLC, an AFL Standard block.

The AFL Standard blocks are called in the CODESYS program block 'PLC_PRG'. Axis blocks also has a synchronous action (act Sync). The synchronous action must be called in 'FPLC_PRG'.

The FB 'BASIC_MOTION' is integrated into each standard axis block. That means FB 'BASIC_MOTION' must not be called separately in the PLC program. The interface between the device and AFL Standard block is realized with the Variable 'ST_DEVICE'. 'SET_CONTROL' summarizes the control-specific and 'ST_AXIS_DRIVE' the axis-specific data together.

For each standard block a matching visualization is imported.

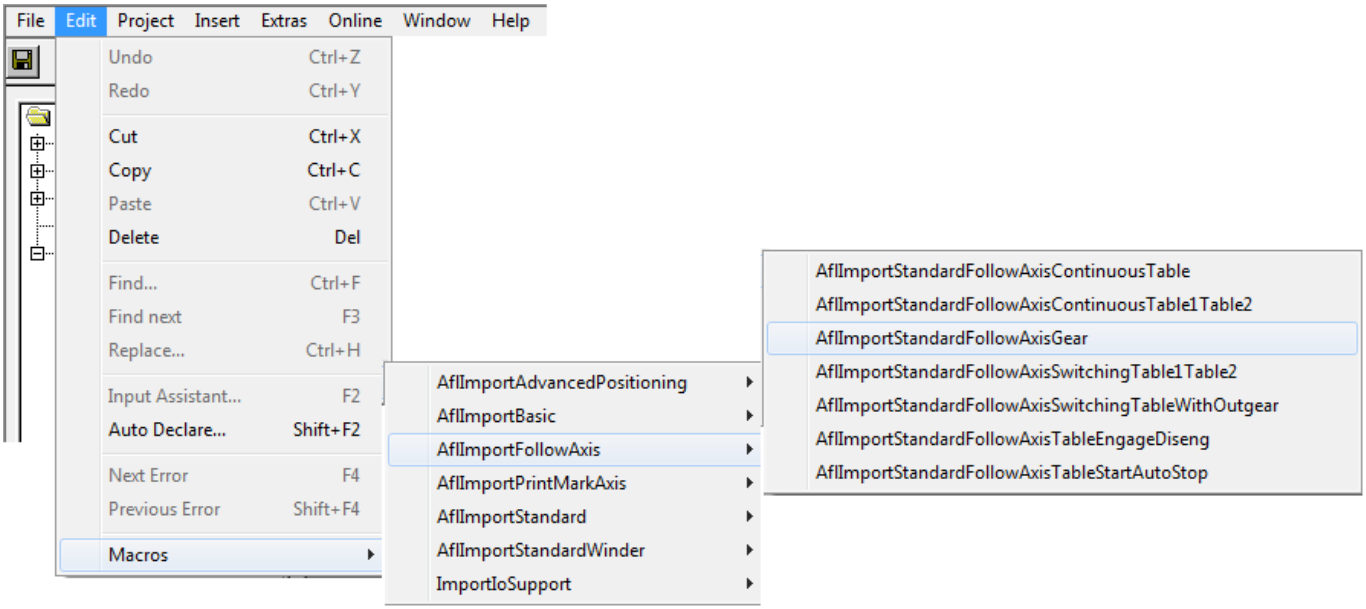
The standard blocks will imported in the current PLC project by using an macros. The macro contains all the instructions to import an imported standard block in the current project.

Requirements:

AMK software AFL Library for CODESYS V2 AMK part no. O877 installed.

AFL template selected. Siehe 'Configuration create' auf Seite 734.

By using CODESYS V2 the macros are execute with the menu 'Edit' → 'Macros'.



Overview of standard blocks and the associated macros:

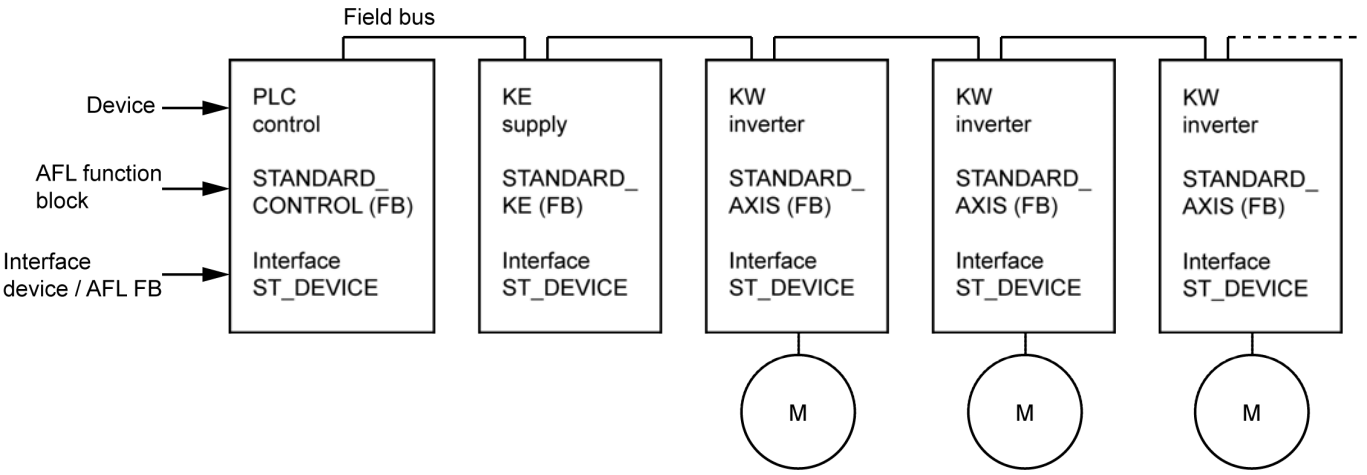
Standard blocks	Macro folder	Macro	For device
STANDARD_CONTROL (FB) [1]	AflImportStandard	AflImportStandardControl	AMKAMAC A4, A5, A6
STANDARD_CONTROL _iSA (FB) [1]	AflImportStandard	AflImportStandardControl_iSA	AMKASMART iSA
STANDARD_KE (FB)	AflImportStandard	AflImportStandardKE	AMKASYN KE/KEN/KES
STANDARD_AXIS_IDT4 (FB) [2]	AflImportStandard	AflImportStandardAxisIDT4	AMKASMART iDT4
STANDARD_AXIS_KWZ (FB) [2]	AflImportStandard	AflImportStandardAxisKWZ	AMKASYN KWZ
STANDARD_AXIS_ihX (FB) [2]	AflImportStandard	AflImportStandardAxis_ihX	AMKASMART ihX
STANDARD_AXIS (FB) [2]	AflImportStandard	AflImportStandardAxis	AMKASYN KW/KWD (KW-R06, KW-R07, KW-R16, KW-R17, KW-R24-R, KW-R25, KW-R26, KW-R27) AMKASMART iDT5 AMKASMART ihX [3]
STANDARD_AXIS_ POSITION_INTERPOSED (FB) [2]	AflImportAdvanced-Positioning	AflImportAdvancedPositioning	
STANDARD_AXIS_ POSITION_TO_THE_MARK (FB) [2]	AflImportPrintMarkAxis	AflImportStandardAxis-PrintMarkPositionToTheMark	
STANDARD_AXIS_ PRINTMARK_INSETTER_ INTERVAL (FB) [2]	AflImportPrintMarkAxis	AflImportStandardAxis-PrintMarkInsetterInterval	
STANDARD_AXIS_ PRINTMARK_INSETTER_ CONT (FB) [2]	AflImportPrintMarkAxis	AflImportStandardAxis-PrintMarkInsetterCont	
STANDARD_AXIS_ PRINTMARK_REGISTER_ CONTROL_CONT (FB) [2]	AflImportPrintMarkAxis	AflImportStandardAxis-PrintMarkRegisterControlCont	
STANDARD_AXIS_ PRINTMARK_REGISTER_ CONTROL_DISCONT (FB) [2]	AflImportPrintMarkAxis	AflImportStandardAxis-PrintMarkRegisterControlDiscont	
STANDARD_FOLLOW_AXIS_GEAR (FB) [2]	AflImportFollowAxis	AflImportFollowAxisGear	
STANDARD_FOLLOW_AXIS_ CONTINUOUS_TABLE1_TABLE2 (FB) [2]	AflImportFollowAxis	AflImportFollowAxis-ContinuousTable1Table2	
STANDARD_FOLLOW_AXIS_ SWITCHING_TABLE_WITH_ OUTGEAR (FB) [2]	AflImportFollowAxis	AflImportFollowAxis-SwitchingTableWithOutgear	
STANDARD_FOLLOW_AXIS_ SWITCHING_TABLE1_TABLE2 (FB) [2]	AflImportFollowAxis	AflImportFollowAxis-SwitchingTable1Table2	
STANDARD_FOLLOW_AXIS_ TABLE_CONTINUOUS (FB) [2]	AflImportFollowAxis	AflImportFollowAxis-ContinuousTable	
STANDARD_FOLLOW_TABLE_ ENGAGE_DISENG_AXIS (FB) [2]	AflImportFollowAxis	AflImportFollowAxis-TableEngageDiseng	
STANDARD_FOLLOW_TABLE_ START_AUTOSTOP_AXIS (FB) [2]	AflImportFollowAxis	AflImportFollowAxis-TableStartAutoStop	
STANDARD_WINDER_DANCER_ AXIS (FB) [2]	AflImportStandardWinder	AflImportStandardWinderDancer	
STANDARD_WINDER_TORQUE_ AXIS (FB) [2]	AflImportStandardWinder	AflImportStandardWinderTorque	

[1] All standard blocks of the type CONTROL require the 'AMK CONTROL STRUCTURE' 'ST_CONTROL (STRUCT)'. 'ST_CONTROL (STRUCT)' is imported with the macro AflImportBasicControl from the folder AflImportBasic. 'ST_CONTROL (STRUCT)' needs imported 1 x per PLC project.

[2] All standard blocks of the type AXIS call the FB 'BASIC_MOTION'. Der FB 'BASIC_MOTION' s imported with the macro AflImportBasicDrive from the folder AflImportBasic. FB 'BASIC_MOTION' needs imported 1 x per PLC project.

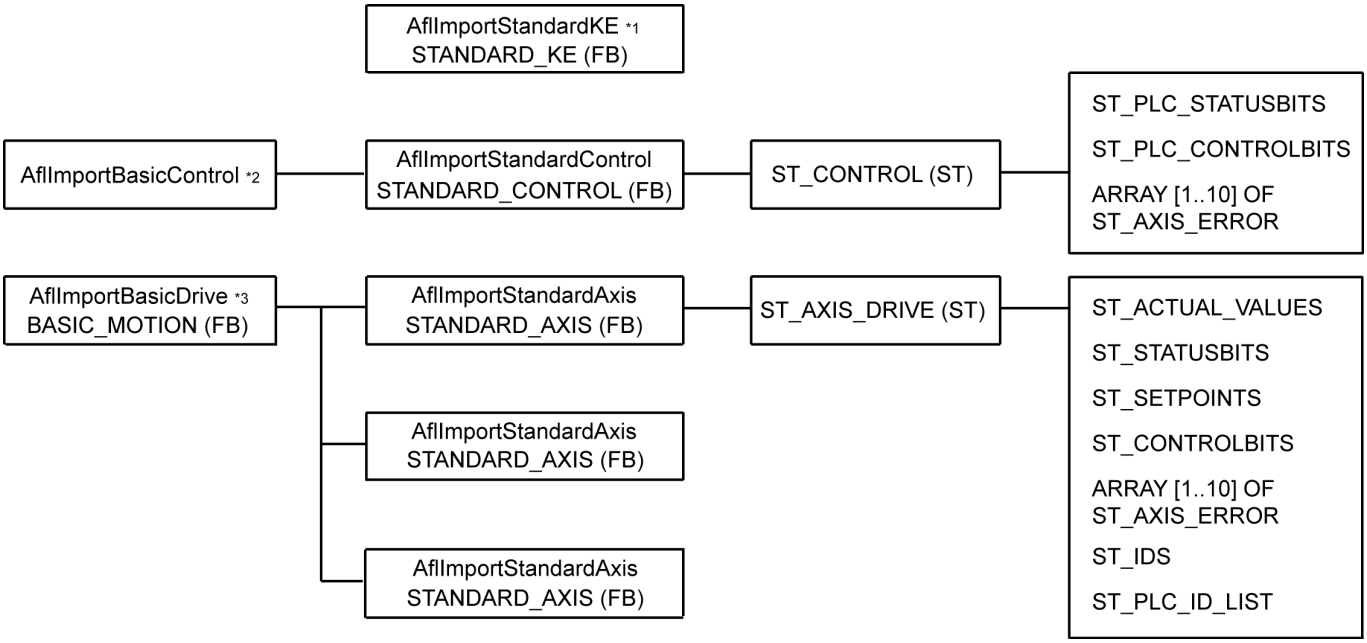
[3] All standard blocks with expanded functionality can be used with a modification for the ihX. Siehe 'Modified 'Standard axes' for ihX drives' auf Seite 208.

Application example:



Structure PLC program:

The macros 'AflImport ...' Importing the standard blocks and structures in the current PLC project.



*1 AflImportBasicControl import the structure 'ST_CONTROL'. It has to call one time for each project.

*2 Only if you use power supply module KE/KEN/KES (exception KEN 05-xx).
The devices AMKASMART iC and AMKASYN KEN 05-xx charge automatically the DC-Bus to supply the connected inverters.

*3 AflImportBasicDrive import 'BASIC_MOTION' (FB). It has to call one time for each project. 'BASIC_MOTION' must not be called by the user. It will be called automatically by the function block 'STANDARD_AXIS'.

You have to call for each project an instance of 'STANDARD_CONTROL' (for A-series) or 'STANDARD_CONTROL_iSA (for iSA) must be necessarily used in each PLC project.

The combination between AFL Standard block 'STANDARD_AXIS' and ID32796 'Source RF' Code 0: 'Controller enable (RF) via binary input' is not allowed.

The control signal RF 'inverter on' must be set via a function block.

Allowed:

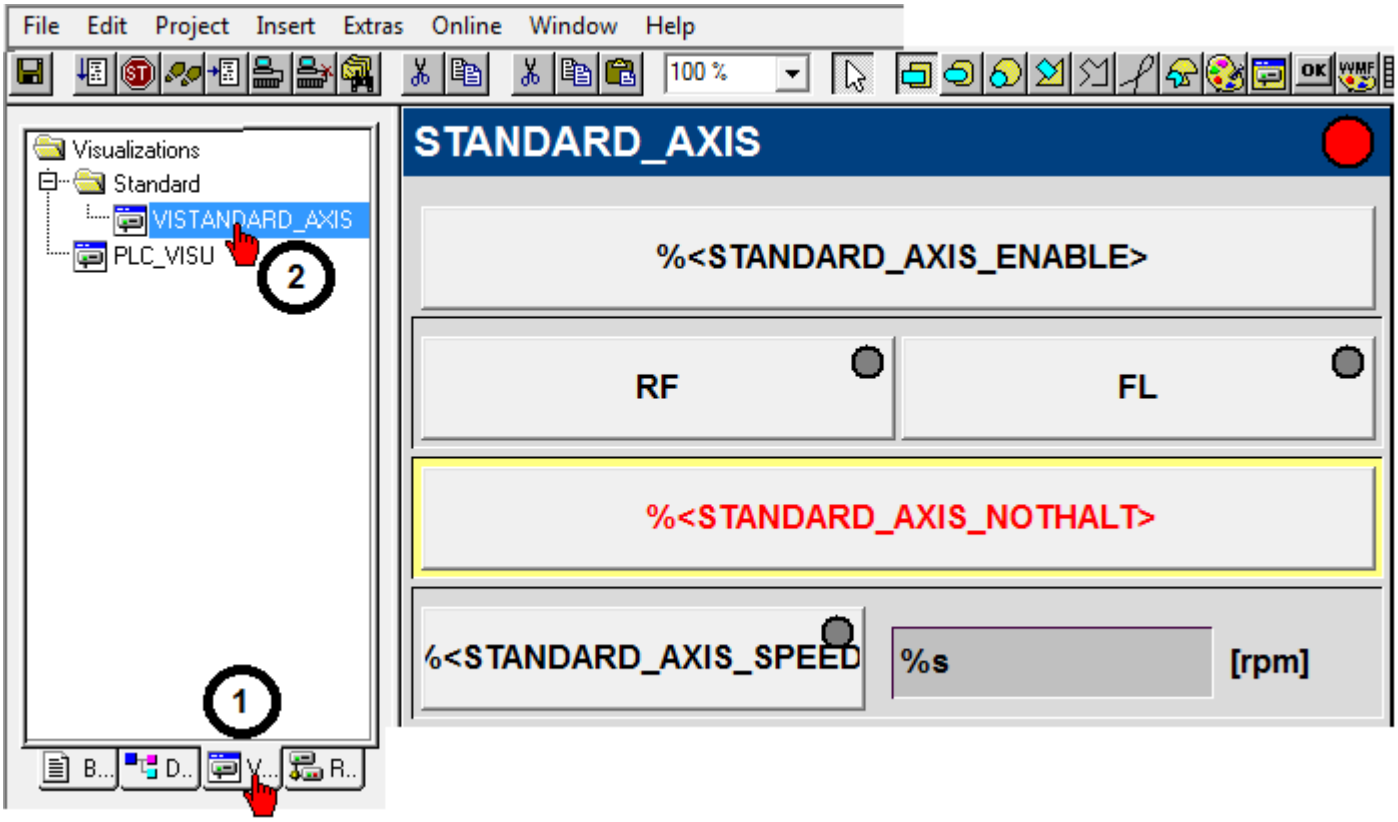
Code 5: 'Controller enable via EtherCAT'

Code 25: 'RF via EtherCAT AND-linked with the binary input RF'

4.1.2.4 Visualization standard blocks

The visualizations of the standard blocks will be imported into the tap/folder 'visualization'.

For each standard block an instance of the visualization must be generated.



4.1.2.5 Overview - Standard function blocks and axis structures

The standard function blocks will be integrated (with CODESYS V2) inside the actual project with ‘macros’. See CODESYS menu 'Edit' → 'Marcros'.

STANDARD_CONTROL (FB)

(Macro: AflImportBasic - AflImportBasicControl + AflImportStandard - AflImportStandardControl)

The following basic functionalities for the controller are available:

- Status bits (SBM System ready message, QFL Acknowl. error cleared, network and bus information)
- The controlling of FL
- The reading out of diagnostic messages

Siehe 'STANDARD_CONTROL (FB)' auf Seite 195.

STANDARD_CONTROL_iSA (FB)

(Macro: AflImportBasic - AflImportBasicControl + AflImportStandard - AflImportStandardControl_iSA)

The following basic functionalities for the controller with supply are available:

- The importing of status bit information (network ready, bus ready, bus warning, bus error, SBM, QUE and QFL)
- The controlling of FL

- The reading out of diagnostic messages

Siehe 'STANDARD_CONTROL_iSA (FB)' auf Seite 197.

STANDARD_KE (FB)
(Macro: AflImportStandard - AflImportStandardKE)
The following basic functionalities for the input are available:

- Status bits (QUE Acknowl. DC bus ON, SBM System ready message)
- Control bits (UE DC bus ON, FL Clear error)
- Output of diagnostics number

Siehe 'STANDARD_KE (FB)' auf Seite 199.

STANDARD_AXIS (FB)
STANDARD_IDT4 (FB)
STANDARD_ihX (FB)
STANDARD_KWZ (FB)
(Macro: AflImportBasic - AflImportBasicDrive + AflImportStandard - AflImportStandardAxis)
The following basic functionalities for the axis are available:

Inputs/outputs on the function block

- Control bits (RF Controller enable, FL Clear error)
- Emergency stop
- Speed control
- Homing cycle
- Positioning (absolute/relative)
- Jog mode (minus/plus)
- Status signals of axis functions

Access via program code

- Configuration of status messages (ID26 'Configuration status bits')
- Reading of parameters according to a specifications list
- Actual position cached
- Control with feedforward (torque and speed)

Siehe 'STANDARD_AXIS (FB) (STANDARD_AXIS_KWZ, STANDARD_AXIS_IDT4, STANDARD_AXIS_ihX)' auf Seite 201.

4.1.2.5.1 Advanced standard axes

The advanced axes feature the same basic functionalities as function block 'STANDARD_AXIS'.

It is possible to implement a dancer winder or sensorless rewinder and unwinder with 'Winder' type axes.
It is possible to implement a printing mark control or interposed positioning with 'Position' type axes.
It is possible to implement a printing mark control with 'Printmark' type axes.
'Follow' type axes are available. These follow a setpoint value (master) synchronously.
In addition tables can also be calculated for cam functions, for example.

STANDARD_AXIS_POSITION_INTERPOSED (FB)
(Macro: AflImportAdvancedPositioning - AflImportAdvancedPositioning)
Expanded functionality:

- Interposed positioning

Siehe 'STANDARD_AXIS_POSITION_INTERPOSED (FB)' auf Seite 208.

STANDARD_AXIS_POSITION_TO_THE_MARK (FB)
(Macro: AflImportPrintMarkAxis - AflImportStandardPrintMarkAxisPositionToTheMark)
Expanded functionality:

- Printing mark control

Siehe 'STANDARD_AXIS_POSITION_TO_THE_MARK (FB)' auf Seite 217.

STANDARD_AXIS_PRINTMARK_INSETTER_CONT (FB)

(Macro: AflImportPrintMarkAxis - AflImportStandardAxisPrintMarkInsetterCont)

Expanded functionality:

- Print mark control: The position of the web to be treated is controlled in relation to a master format (insetting)

Siehe 'STANDARD_AXIS_PRINTMARK_INSETTER_CONT (FB)' auf Seite 226.

STANDARD_AXIS_PRINTMARK_INSETTER_INTERVAL (FB)

(Macro: AflImportPrintMarkAxis - AflImportStandardAxisPrintMarkInsetterInterval)

Expanded functionality:

- Print mark control: The position of the web to be treated is controlled in relation to a master format (insetting)

Siehe 'STANDARD_AXIS_PRINTMARK_INSETTER_INTERVAL (FB)' auf Seite 237.

STANDARD_AXIS_PRINTMARK_REGISTER_CONTROL_CONT (FB)

(Macro: AflImportPrintMarkAxis - AflImportStandardAxisPrintMarkRegisterControlCont)

Expanded functionality:

- Print mark control: The position of the treating tool (e.g. cutter, punch) is controlled in relation to a master format (register control)

Siehe 'STANDARD_AXIS_PRINTMARK_REGISTER_CONTROL_CONT (FB)' auf Seite 249.

STANDARD_AXIS_PRINTMARK_REGISTER_CONTROL_DISCONT (FB)

(Macro: AflImportPrintMarkAxis - AflImportStandardAxisPrintMarkRegisterControlDiscont)

Expanded functionality:

- Print mark control: The position of the treating tool (e.g. cutter, punch) is controlled in relation to a master format (register control)

Siehe 'STANDARD_AXIS_PRINTMARK_REGISTER_CONTROL_DISCONT (FB)' auf Seite 260.

STANDARD_FOLLOW_AXIS_GEAR (FB)

(Macro: AflImportFollowAxis - AflImportFollowAxisGear)

Expanded functionality:

- The slave axis continuously follows a master.
- A gear ratio can be defined.

Siehe 'STANDARD_FOLLOW_AXIS_GEAR (FB)' auf Seite 271.

STANDARD_FOLLOW_AXIS_CONTINUOUS_TABLE1_TABLE2 (FB)

(Macro: AflImportFollowAxis - AflImportFollowAxisContinuousTable1Table2)

Expanded functionality:

- The slave axis continuously follows a freely selectable table.
- At the end of the table, a table switchover can be carried out.

Siehe 'STANDARD_FOLLOW_AXIS_CONTINUOUS_TABLE1_TABLE2 (FB)' auf Seite 279.

STANDARD_FOLLOW_AXIS_SWITCHING_TABLE_WITH_OUTGEAR (FB)

(Macro: AflImportFollowAxis - AflImportFollowAxisSwitchingTableWithOutgear)

Expanded functionality:

- The slave axis continuously follows a freely selectable table.
- At the end of the table, a table switchover can be carried out.
- The table can be modified via a gear ratio (compress and extend).

Siehe 'STANDARD_FOLLOW_AXIS_SWITCHING_TABLE_WITH_OUTGEAR (FB)' auf Seite 288.

STANDARD_FOLLOW_AXIS_SWITCHING_TABLE1_TABLE2 (FB)

(Macro: AflImportFollowAxis - AflImportFollowAxisSwitchingTable1Table2)

Expanded functionality:

- The slave axis continuously follows a freely selectable table with a phasing-in and phasing out table.
- At the end of the table, a table switchover can be carried out.

Siehe 'STANDARD_FOLLOW_AXIS_SWITCHING_TABLE1_TABLE2 (FB)' auf Seite 296.

STANDARD_FOLLOW_AXIS_TABLE_CONTINUOUS (FB)
(Macro: AflImportFollowAxis - AflImportFollowAxisContinuousTable)
Expanded functionality:

- The slave axis continuously follows a freely selectable table.

Siehe 'STANDARD_FOLLOW_AXIS_TABLE_CONTINUOUS (FB)' auf Seite 305.

STANDARD_FOLLOW_TABLE_ENGAGE_DISENG_AXIS (FB)
(Macro: AflImportFollowAxis - AflImportFollowAxisTableEngageDiseng)
Expanded functionality:

- The slave axis continuously follows a freely selectable table with a phasing-in and phasing out table.

Siehe 'STANDARD_FOLLOW_TABLE_ENGAGE_DISENG_AXIS (FB)' auf Seite 314.

STANDARD_FOLLOW_TABLE_START_AUTOSTOP_AXIS (FB)
(Macro: AflImportFollowAxis - AflImportFollowAxisTableStartAutoStop)
Expanded functionality:

- The slave axis continuously follows a freely selectable table with a phasing-in and phasing out table.

Siehe 'STANDARD_FOLLOW_TABLE_START_AUTOSTOP_AXIS (FB)' auf Seite 322.

STANDARD_WINDER_DANCER_AXIS (FB)
(Macro: AflImportFollowAxis - AflImportFollowAxisTableEngageDiseng)
Expanded functionality:
The function block 'STANDARD_WINDER_DANCER_AXIS' realizes a dancer winder with the following characteristics with following features:

- PID regulated dancer winder
- Operation as unwinder and rewinder
- Diameter calculation
- Search diameter at the start of the winder

Siehe 'STANDARD_WINDER_DANCER_AXIS (FB)' auf Seite 330.

STANDARD_WINDER_TORQUE_AXIS (FB)
(Macro: AflImportStandardWinder - AflImportStandardWinderTorque)
Expanded functionality:

- The function block 'STANDARD_WINDER_TORQUE_AXIS' realizes a sensorless rewinder and unwinder with following features:
- Torque controlled center winder
- Operation as rewinder and unwinder
- Diameter calculation
- Search diameter at the start of the winder
- Determine the reel of inertia of unknown diameter
- Synchronize onto a running web

(Further information: See documentation on application blocks part no. 203694 FB 'WINDER_TORQUE_CONTROL'
Siehe 'STANDARD_WINDER_TORQUE_AXIS (FB)' auf Seite 339.

4.1.2.5.2 Structures

The following access options are available:

PLC controller (ST_CONTROL):

- Status bits (SBM System ready message, network status)
- Control bits (FL Clear error, Read diagnostic message)
- Diagnostic array (memory for ten diagnostic messages)

Siehe 'ST_CONTROL (ST)' auf Seite 189.

Axis (ST_AXIS_DRIVE):

- Setpoint values (speed, position, torque, pilot control values)
- Actual values (speed, position, actual torque, probe values, etc.)
- Control bits (RF controller enable, FL Clear error, Read diagnostic message)
- Status bits (Speed window, Standstill window, etc.)
- Diagnostic array (memory for ten diagnostic messages)
- Read parameter list
- Configuration (PLC functions)

Siehe 'ST_AXIS_DRIVE (ST)' auf Seite 191.

4.1.3 Description Structures (ST)

4.1.3.1 ST_CONTROL (ST)

The control structure 'ST_CONTROL' contains the control-specific data.
It is divided into the following substructures:

- ST_PLC_STATUSBITS Siehe 'ST_PLC_STATUSBITS (ST)' auf Seite 189.)
- ST_PLC_CONTROLBITS (Siehe 'ST_PLC_CONTROLBITS (ST)' auf Seite 190.
- ARRAY [1..10] OF ST_AXIS_ERROR Siehe 'arstDiagnosis (ARR_ST)' auf Seite 190.

Storage location:
CODESYS V2
'Data types' → **'AMK_CONTROL_STRUCTURE'** → **'Interface'**

CODESYS V3
'POUs' → **'Control'** → **'Standard'** → **'Types'** → **'AMK_CONTROL_STRUCTURE'**

4.1.3.1.1 ST_PLC_STATUSBITS (ST)

The structure 'ST_PLC_STATUSBITS' is a substructure of 'ST_CONTROL'.
The status bits of the controller can be accessed via 'ST_CONTROL.ST_PLC_STATUSBITS'.
Contents:

Name	Description	Unit
boID33029_SBM	System ready message	-
boQFL	Acknowledgement Clear error	-
boNetworkReady	Network ready	-
boBusReady	Bus ready	-
boBusWarning	Bus warning	-
boBusError	Bus error	-
boDiagOk	Diagnosis completed	-

Storage location:
CODESYS V2
'Data types' → **'AMK_CONTROL_STRUCTURE'**

CODESYS V3
'POUs' → **'Control'** → **'Standard'** → **'Types'** → **'AMK_CONTROL_STRUCTURE'** → **'Interface'**

4.1.3.1.2 ST_PLC_CONTROLBITS (ST)

The structure 'ST_PLC_CONTROLBITS' is a substructure of 'ST_CONTROL'.
The control bits of the control can be set via 'ST_CONTROL.ST_PLC_CONTROLBITS'.
Contents:

Name	Description	Unit
boID32913_FL	Clear error	-
boDiagnosis	Reading the diagnostic messages	-

Storage location:
CODESYS V2
'Data types' → 'AMK_CONTROL_STRUCTURE'

CODESYS V3
'POUs' → 'Control' → 'Standard' → 'Types' → 'AMK_CONTROL_STRUCTURE' → 'Interface'

4.1.3.1.3 arstDiagnosis (ARR_ST)

The array 'arstDiagnosis' is part of the structure of 'ST_AXIS_DRIVE' and 'ST_CONTROL'.
The diagnostic messages can be accessed via 'ST_AXIS_DRIVE.arstDiagnosis [1-10]', for example.
The diagnostic array contains up to 10 diagnostic messages which are structured as follows:

Name	Description	Unit
arstDiagnosis [1]	boErr: Error bit for error 1 iDiagNo: Diagnostic number for error 1 diInfo1: Additional 1 for error 1 iAdr: Device address*	-
arstDiagnosis [2]	boErr: Error bit for error 2 iDiagNo: Diagnostic number for error 2 diInfo1: Additional 1 for error 2 iAdr: Device address*	-
...	...	
arstDiagnosis [10]	boErr: Error bit for error 10 iDiagNo: Diagnostic number for error 10 diInfo1: Additional 1 for error 10 iAdr: Device address*	-

* iAdr: Device address
An ACC-bus controller displays the device address when a slave error occurs; in all other cases, 0 (local error). EtherCAT always 0.

4.1.3.2 ST_AXIS_DRIVE (ST)

The axis structure 'ST_AXIS_DRIVE' contains the axis-specific data.
It is divided into the following substructures:

- ST_ACTUAL_VALUES Siehe 'ST_ACTUAL_VALUES (ST)' auf Seite 191.
- ST_STATUSBITS Siehe 'ST_STATUSBITS (ST)' auf Seite 192.
- ST_SETPOINTS Siehe 'ST_SETPOINTS (ST)' auf Seite 193.
- ST_CONTROLBITS Siehe 'ST_CONTROLBITS (ST)' auf Seite 193.
- ARRAY [1..10] OF ST_AXIS_ERROR (Siehe 'arstDiagnosis (ARR_ST) ' auf Seite 194.)
- ST_IDS Siehe 'ST_IDS (ST)' auf Seite 194.
- ST_PLC_ID_LIST (provisional) Siehe 'ST_PLC_ID_LIST (ST) (provisional)' auf Seite 194.

Storage location:
CODESYS V2
'Data types' → 'AMK_AXIS_STRUCTURE' → 'Interface'

CODESYS V3
'POUs' → 'Control' → 'Standard' → 'Types' → 'AMK_AXIS_STRUCTURE'

The structure 'ST_AXIS_DRIVE' is updated once the accompanying STANDARD_ACHSE is released.

4.1.3.2.1 ST_ACTUAL_VALUES (ST)

The structure 'ST_ACTUAL_VALUES' is a substructure of 'ST_AXIS_DRIVE'.
The actual axis values can be accessed via 'ST_AXIS_DRIVE.ST_ACTUAL_VALUES'.
Contents:

Name	Description	Unit
diID00040_Velocity_feedback_value	Actual speed value	0.0001 rpm
diID00051_Position_feedback_value	Actual position value	Increments
diID00084_Torque_feedb_val	Actual torque value	0.1 % Mn
diID00130_Probe1_val_p_edge	Probe value 1 positive edge [1)]	Increments
diID00131_Probe1_val_n_edge	Probe value 1 negative edge [1)]	Increments
diID00189_Following_dist	Following distance error [1)]	Increments
diRectangularPulsInput_X132	Input pulses of square-wave pulse input	Increments

1) Only available for EtherCAT!
The 'Probe values' and the 'Following distance error' are inside the plc program commented out. This is the reason while the values are not updated.
Commented out parts inside plc program see:
- Declaration editor Standard Axis: fbGetStatusActVal
- Program code editor : Evaluation variable 'boErr'
- Program code editor : Evaluation variable 'boErrID'
- Action actSync: PLC Program code editor: part 'Read status'

Storage location:
CODESYS V2
'Data types' → 'AMK_AXIS_STRUCTURE' → 'Interface'

CODESYS V3
'POUs' → 'Control' → 'Standard' → 'Types' → 'AMK_AXIS_STRUCTURE' → 'Interface'

4.1.3.2.2 ST_STATUSBITS (ST)

The structure 'ST_STATUSBITS' is a substructure of 'ST_AXIS_DRIVE'.

The status bits of the axis can be accessed via 'ST_AXIS_DRIVE.ST_STATUSBITS'.

Contents:

Name	Description	Unit
boID330_VelocityWindow	Code 330 nact = nset ID157 Velocity window ("To velocity")	-
boID331_ZeroVelocityWindow	Code 331 nact < nmin ID124 Zero velocity window	-
boID332_VelocityWindowLimit	Code 332 nact < nx ID125 Velocity limit nx	-
boID333_TorqueLimit	Code 333 Md>=Mdx ID126 Torque limit Mdx	-
boID334_NegTorqueLimit	Code 334 Mset >Mlimit Set torque, Limit torque (ID82/83)	-
boID335_NegVelocityLimit	335 nset >= mlimit velocity setpoint. Limit speed (ID38/39)	-
boID336_InPosition	In position [1)]	-
boID33029_SBM	System ready message	-
boID33030_QUE	Acknowledgement DC bus ON	-
boID33031_QRF	Acknowledgement controller enable	-
boID33036_RFP_known	Code 33036 Home position known	-
boID33074_Warning_active	Code 33074 Warning active group warning	-
boDiagOk	Diagnosis OK	-
boSetPosEnabAck	Secondary operation mode1 release position specifications	-
boSetVelocityEnabAck	Secondary operation mode2 release speed	-
boSetTorqueEnabAck	Secondary operation mode3 release torque setpoint	-
boResidDistDone	Residual distance cleared	-

1) The 'In position' bit will evaluate only with controller cards with internal interpolator. If you use an Ethernet controller card, the bit will only evaluate if you use the commands absolute or relative positioning (boPosAbs / boPosRel).

Storage location:

CODESYS V2

'Data types' → 'AMK_AXIS_STRUCTURE' → 'Interface'

CODESYS V3

'POUs' → 'Control' → 'Standard' → 'Types' → 'AMK_AXIS_STRUCTURE' → 'Interface'

4.1.3.2.3 ST_SETPOINTS (ST)

The structure 'ST_SETPOINTS' is a substructure of 'ST_AXIS_DRIVE'.
Setpoint values can be written to the axis via 'ST_AXIS_DRIVE.ST_SETPOINTS'.
Contents:

Name	Description	Unit
diID00036_Velocity_command_value	Velocity command value (ID36)	0.0001 rpm
diID00037_Velocity_command_value_addit	Velocity command value additive (ID37) 1)	0.0001 rpm
diID00047_Position_command_value	Position setpoint (ID47)	Increments
diID00080_Torque_command_value	Torque command value (ID80)	0.1 % Mn
diID00081_Torque_command_value_additiv	Torque command value additive (ID81) 1)	0.1 % Mn

1) The 'Command values' are inside the plc program commented out. This is the reason while the values are not updated.
Commented out parts inside plc program see:
- Declaration editor Standard Axis: fbMotion
- Action actSync: plc program code editor: part 'Writing the setpoint values'

Storage location:
CODESYS V2
'Data types' → 'AMK_AXIS_STRUCTURE' → 'Interface'

CODESYS V3
'POUs' → 'Control' → 'Standard' → 'Types' → 'AMK_AXIS_STRUCTURE' → 'Interface'

4.1.3.2.4 ST_CONTROLBITS (ST)

The structure 'ST_CONTROLBITS' is a substructure of 'ST_AXIS_DRIVE'.
The control bits of the axis can be set via 'ST_AXIS_DRIVE.ST_CONTROLBITS'.
Contents:

Name	Description	Unit
boID32904_RF	Controller enable on	-
boID32913_FL	Clear error	-
enSetPointMode	Setpoint value mode Type EN_SETPOINT AMK_SETPOINT_VELOCITY Speed specification AMK_SETPOINT_POSITION Position specification AMK_SETPOINT_TORQUE Torque specification	-
boDiagnosis	Reading the diagnostic messages	

Storage location:
CODESYS V2
'Data types' → 'AMK_AXIS_STRUCTURE' → 'Interface'

CODESYS V3
'POUs' → 'Control' → 'Standard' → 'Types' → 'AMK_AXIS_STRUCTURE' → 'Interface'

4.1.3.2.5 ST_IDS (ST)

The structure 'ST_IDS' is a substructure of 'ST_AXIS_DRIVE'.
Parameter values can be accessed via ST_AXIS_DRIVE.ST_IDS.
Contents:

Name	Description	Unit
diID00002_SERCOS_cycle	SERCOS cycle time	0.001 ms
diID00038_Pos_velocity_limit	Pos. velocity limit	0.0001 rpm
diID00039_Neg_velocity_limit	Neg. velocity limit	0.0001 rpm
diID00049_Positive_position_limit	Positive position limit value	Increments
diID00050_Negative_position_limit	Negative position limit value	Increments
diID00057_In_posit_window	Position window	Increments
diID00116_Resol_mot_encod	Resolution motor encoder	Increments
diID00228_Synchr_pos_window	Angle synchronous window	Increments
diID32771_Nom_torque	Nominal torque Mn	0.1 % Mn

Storage location:
CODESYS V2
'Data types' → 'AMK_AXIS_STRUCTURE' → 'IDs'

CODESYS V3
'POUs' → 'Control' → 'Standard' → 'Types' → 'AMK_AXIS_STRUCTURE' → 'IDs'

4.1.3.2.6 ST_PLC_ID_LIST (ST) (provisional)

The structure 'ST_PLC_ID_LIST' is a substructure of 'ST_AXIS_DRIVE'.
The required PLC parameters for the standard axes can be set via ST_AXIS_DRIVE.ST_PLC_ID_LIST.
Contents:

Name	Description	Unit
diID32919_00_Plc_List	PLC parameter list	-
diID32919_01_maximum	Parameter list length	-
diID32919_02_Resid_dist_wind	neg. Delete residual distance window Default value: 20000	Increments
diID32919_03_Sync_set_pulses_multip	Multiplier setpoint value Used to modify to mechanic conditions of the system Default value: 10000	Increments
diID32919_04_Sync_set_pulses_divider	Divider setpoint value Used to modify to mechanic conditions of the system Default value: 10000	Increments
diID32919_05_diEmergency_StopRamp	Emergency stop ramp	ms

Storage location:
CODESYS V2
'Data types' → 'AMK_AXIS_STRUCTURE' → 'IDs'

CODESYS V3
'POUs' → 'Control' → 'Standard' → 'Types' → 'AMK_AXIS_STRUCTURE' → 'IDs'

4.1.3.2.7 arstDiagnosis (ARR_ST)

The array 'arstDiagnosis' is part of the structure of 'ST_AXIS_DRIVE' and 'ST_CONTROL'.
The diagnostic messages can be accessed via 'ST_AXIS_DRIVE.arstDiagnosis [1-10]', for example.
The diagnostic array contains up to 10 diagnostic messages which are structured as follows:

Name	Description	Unit
arstDiagnosis [1]	boErr: Error bit for error 1 iDiagNo: Diagnostic number for error 1 diInfo1: Additional 1 for error 1 iAdr: Device address*	-
arstDiagnosis [2]	boErr: Error bit for error 2 iDiagNo: Diagnostic number for error 2 diInfo1: Additional 1 for error 2 iAdr: Device address*	-
...	...	
arstDiagnosis [10]	boErr: Error bit for error 10 iDiagNo: Diagnostic number for error 10 diInfo1: Additional 1 for error 10 iAdr: Device address*	-

* iAdr: Device address

An ACC-bus controller displays the device address when a slave error occurs; in all other cases, 0 (local error). EtherCAT always 0.

4.1.4 Description Standard blocks (FB)

4.1.4.1 Supply and PLC

4.1.4.1.1 STANDARD_CONTROL (FB)

The following basic functionalities for the controller are available:

- The importing of status bit information (network ready, bus ready, bus warning, bus error, SBM and QFL) from AMK controller
- The controlling of FL
- The reading out of diagnostic messages

User interface

		STANDARD_CONTROL			
FB enable	-	boEnable	boEnabAck	-	Ackn. "FB enable"
Clear error	-	boErrorReset	boSBM	-	System ready message
Bus instance	-	uiBusInstance	boNetworkReady	-	Network ready
			boBusReady	-	Bus ready
			boBusWarning	-	Bus warning
			boBusError	-	Bus error
			boErr	-	Error
			iErrID	-	Error ID
			enErrName	-	Name of faulty FB
Control structure	-	stControl	stControl	-	Control structure
Device structure	-	stDevice	stDevice	-	Device structure

Input variables

Name	Type	Description	Sync.	Async.
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.		x
boErrorReset	BOOL	Error Reset (FL = clear error)		x
uiBusInstance	UINT	Bus instance; default value 5		x

Output variables

Name	Type	Description	Sync.	Async.
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled		x
boSBM	BOOL	System ready message		x
boNetworkReady	BOOL	Network ready		x
boBusReady	BOOL	Bus ready (module, device ready, no errors or warnings, bus operational mode)		x
boBusWarning	BOOL	Bus warning		x
boBusError	BOOL	Bus error		x
boErr	BOOL	The function block is in an error state FALSE: No error (permitted commanding or warning) TRUE: Error		x
iErrID	INT	Error identity number: Diagnostic number is output iErrID = 0: No error iErrID ≠ 0, boErr = TRUE: Error iErrID ≠ 0, boErr = FALSE: Warning		x
enErrName	ENUM	EN_FB_NAME Name of faulty block for which the diagnostic number is output. The diagnostic number is explained in the description of the corresponding library.		x

Input and output variables

Name	Type	Description	Sync.	Async.
stDevice	STRUCT	ST_DEVICE The device description structure assigns the block a device. (See document Software description AmkBase Bibliothek, Part no.204986)		x
stControl	STRUCT	ST_CONTROL Siehe 'ST_CONTROL (ST)' auf Seite 189.		x

Usage note in the CODESYS program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
STANDARD_CONTROL	-

History

Library	AmkAfl	
System versions	**Functionality**	**Target**
AS-C V3.01/0827	Basic functionality (1st version)	≥ AS_CP_200_0812_T202053
AS-PL15 V3.01/0804	Basic functionality (1st version)	≥ AS_V313_0946_T202724

4.1.4.1.2 STANDARD_CONTROL_iSA (FB)

The following basic functionalities for the controller with supply are available:

- The importing of status bit information (network ready, bus ready, bus warning, bus error, SBM, QUE and QFL)
- The controlling of FL
- The reading out of diagnostic messages

Error message during automatic configuration: DEV_GET_ERROR_ID11
Reason: iSA description file does not contain the necessary entering.
Integrated from AIPEX PRO Version ≥ 3.03 2015/41 part-no. 205890

User interface

		STANDARD_CONTROL_iSA			
FB enable	-	boEnable	boEnabAck	-	Ackn. "FB enable"
Clear error	-	boErrorReset	boSBM	-	System ready message
Bus instance	-	uiBusInstance	boQUE	-	Ackn. "DC bus ON"
			wErrorID11	-	State ID11 'Status class 1-errors'
			boErrTemperature	-	Error temp. device *
			boErrDC_overvoltage	-	Over voltage DC bus *
			boNetworkReady	-	Network ready
			boBusReady	-	Bus ready
			boBusWarning	-	Bus warning
			boBusError	-	Bus error
			boErr	-	Error
			iErrID	-	Error ID
			enErrName	-	Name of faulty FB
Control structure	-	stControl	stControl	-	Control structure
Device structure	-	stDevice	stDevice	-	Device structure

Input variables

Name	Type	Description	Sync.	Async.
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.		x
boErrorReset	BOOL	Error Reset (FL = clear error)		x
uiBusInstance	UINT	Bus instance; default value 5		x

Output variables

Name	Type	Description	Sync.	Async.
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled		x
boSBM	BOOL	System ready message		x
boQUE	BOOL	Acknowledgement DC bus ON		x
wErrorID11	WORD	State ID11 'Status class 1-errors'		x
boErrTemperature	BOOL	Error temperature device *		x
boErrDC_overvoltage	BOOL	Over voltage DC bus *		x
boNetworkReady	BOOL	Network ready		x
boBusReady	BOOL	Bus ready (module, device ready, no errors or warnings, bus operational mode)		x
boBusWarning	BOOL	Bus warning		x
boBusError	BOOL	Bus error		x
boErr	BOOL	The function block is in an error state FALSE: No error (permitted commanding or warning) TRUE: Error		x
iErrID	INT	Error identity number: Diagnostic number is output iErrID = 0: No error iErrID ≠ 0, boErr = TRUE: Error iErrID ≠ 0, boErr = FALSE: Warning		x
enErrName	ENUM	EN_FB_NAME Name of faulty block for which the diagnostic number is output. The diagnostic number is explained in the description of the corresponding library.		x

Input and output variables

Name	Type	Description	Sync.	Async.
stDevice	STRUCT	ST_DEVICE The device description structure assigns the block a device. (See document Software description AmkBase Bibliothek, Part no.204986)		x
stControl	STRUCT	ST_CONTROL Siehe 'ST_CONTROL (ST)' auf Seite 189.		x

Usage note in the CODESYS program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
STANDARD_CONTROL_iSA	-

History

Library	AmkAfl	
System versions	**Functionality**	**Target**
AS-C V3.01/0827	Basic functionality (1st version)	≥ AS_CP_200_0812_T202053
AS-PL15 V3.01/0804	Basic functionality (1st version)	≥ AS_V313_0946_T202724

*

Reaction iSA:

No independent response to a temperature or over-voltage fault.

The user must initiate via the PLC program an appropriate action, e. g. activate STO.

4.1.4.1.3 STANDARD_KE (FB)

The following basic functionalities for the input are available:

- The importing of status bit information (QUE, SBM) from the AMK input
- The controlling of UE and FL
- The read out of the diagnostic number

The function block 'STANDARD_KE' contains a inhibit time for DC bus ON min. 15 sec off before switching back on.

See function block 'fbTof'.

User interface

		STANDARD_KE			
FB enable	-	boEnable	boEnabAck	-	Ackn. "FB enable"
DC bus On	-	boUE	uiDiagnosticNr	-	Diagnostics number
Clear error	-	boFL	boQUE	-	Ackn. "DC bus ON"
			boSBM	-	System ready message
			boErr	-	Error
			iErrID	-	Error ID
			enErrName	-	Name of faulty FB
Device structure	-	stDevice	stDevice	-	Device structure

Input variables

Name	Type	Description	Sync.	Async.
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.		x
boUE	BOOL	Bit for switching the converter (DC bus voltage on). The corresponding return message QUE is a precondition for RF. Relevant parameters: ID32795 'Source UE' Default: 0 (DC Bus Enable with the binary input) Setting: e.g. Code 5 (DC Bus Enable with the binary input)		x
boFL	BOOL	Bit for error deletion		x

Output variables

Name	Type	Description	Sync.	Async.
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled		x
uiDiagnosticNr	UINT	Diagnostics number		x
boQUE	BOOL	Acknowledgement DC bus ON		x
boSBM	BOOL	System ready message		x
boErr	BOOL	The function block is in an error state FALSE: No error (permitted commanding or warning) TRUE: Error		x
iErrID	INT	Error identity number: Diagnostic number is output iErrID = 0: No error iErrID ≠ 0, boErr = TRUE: Error iErrID ≠ 0, boErr = FALSE: Warning		x
enErrName	ENUM	EN_FB_NAME Name of faulty block for which the diagnostic number is output. The diagnostic number is explained in the description of the corresponding library.		x

Input and output variables

Name	Type	Description	Sync.	Async.
stDevice	STRUCT	ST_DEVICE The device description structure assigns the block a device. (See document Software description AmkBase Bibliothek, Part no.204986)		x

Usage note in the CODESYS program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
STANDARD_KE	-

History

Library	AmkAfl	
System versions	**Functionality**	**Target**
AS-C V3.01/0827	Basic functionality (1st version)	≥ AS_CP_200_0812_T202053
AS-PL15 V3.01/0804	Basic functionality (1st version)	≥ AS_V313_0946_T202724

4.1.4.2 Standard Axis

4.1.4.2.1 STANDARD_AXIS (FB) (STANDARD_AXIS_KWZ, STANDARD_AXIS_IDT4, STANDARD_AXIS_ihX)

The following basic functionalities for the axis are available:

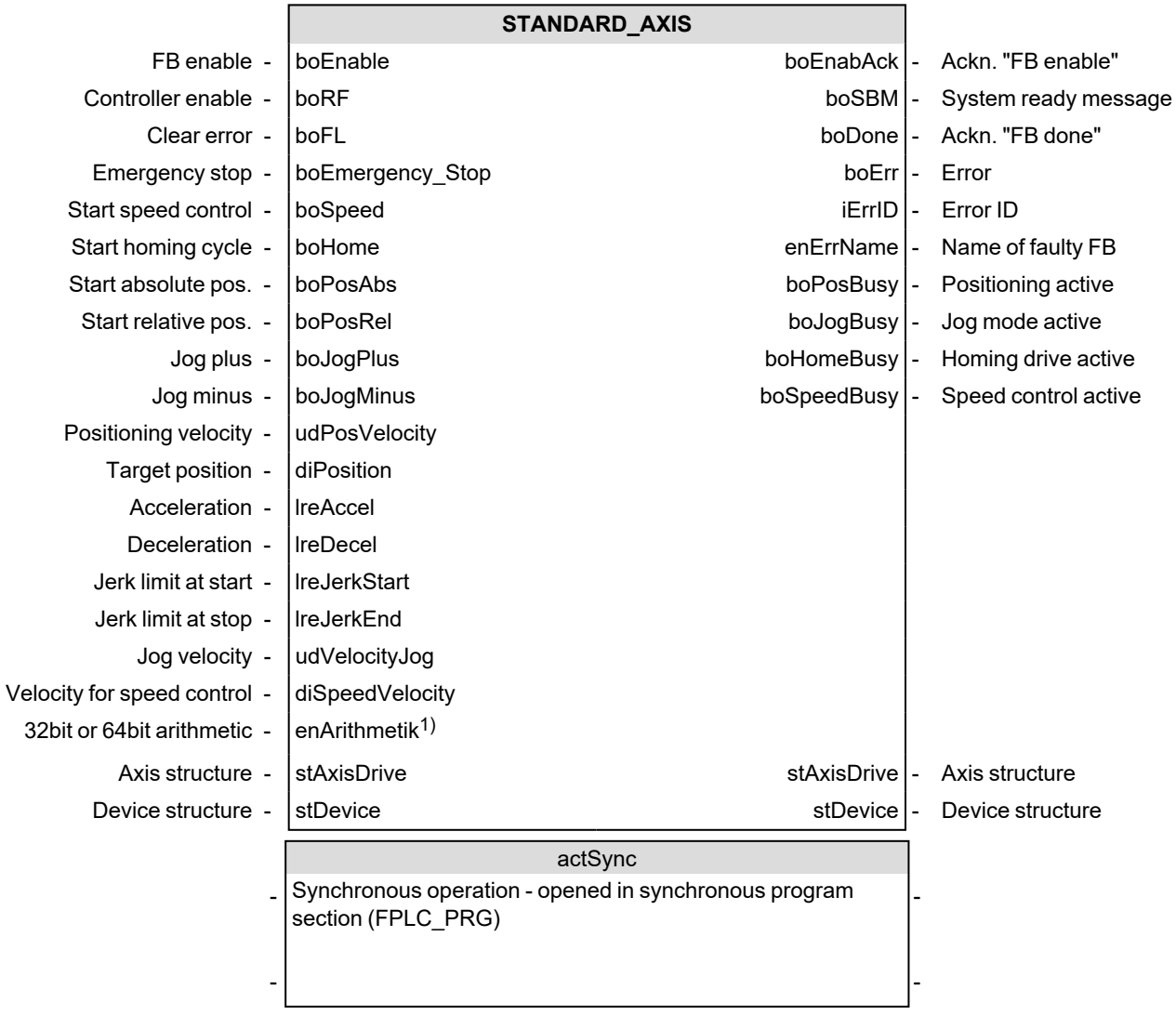
Inputs/outputs on the function block

- Control bits (RF Controller enable, FL Clear error)
- Emergency stop
- Speed control
- Homing cycle
- Positioning (absolute/relative)
- Jog mode (minus/plus)
- Status signals of axis functions

Access via program code

- Configuration of status messages (ID26 'Configuration status bits')
- Reading of parameters according to a specifications list
- Actual position cached
- Control with feedforward (torque and speed)

User interface



1) Only for CODESYS V3

Input variables

Name	Type	Description	Sync.	Async.
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.		x
boRF	BOOL	Bit for enabling the controller Relevant parameters: ID32796 'Source RF' Allowed: Code 5: 'Controller enable via EtherCAT' Code 25: RF 'via EtherCAT AND-linked with the binary input UE'  Code 0: 'Controller enable (RF) via binary input' is not allowed		x
boFL	BOOL	Bit for error deletion		x
boEmergency_ Stop	BOOL	EMERGENCY STOP: The setpoint of the velocity is decreased to zero along the emergency-stop ramp. Once initiated, an emergency stop cannot be aborted. Relevant variables: EMERGENCY STOP ramp: Ramp time in which the drive is slowed down from the current velocity to zero. EMERGENCY STOP ramp: ST_AXIS_DRIVE.ST_IDS.ST_ID32919_ PlcList. diID32919_05_diEmergency_StopRamp Unit: ms Default value: 100 ms		x

Name	Type	Description	Sync.	Async.
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). Relevant parameters: ID32802 'AMK secondary operating mode 2' [Operating mode: Speed control] [Setpoint source: diMainSetpoint (code 41h)] [Ramps: active / inactive (bit 6)] ID32780 'Acceleration ramp' ID32781'Deceleration ramp' Relevant variables: Speed setpoint: diSpeedVelocity Velocity ramp: STANDARD_AXIS.fbVelocity.diVelocityRamp [default value: 100 ms] Speed control active: boSpeedBusy		x
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. Relevant parameters: ID41 'Homing velocity' ID147 'Homing parameter' ID150 'Homing offset 1' ID32926 'AMK homing cycle parameter' Relevant variables: Homing drive active: boHomeBusy Homing done: boDone Homing point known: stAxisDrive.stStatus.boID33036_RFP_known		x

Name	Type	Description	Sync.	Async.
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 [Relevant parameters: see 'boPosRel']		x
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. Relevant parameters: ID32801 'AMK secondary operating mode 1' [Operating mode: position control] [Setpoint source: diMainSetpoint (code 41h)] Relevant variables: Positioning velocity: udPosVelocity Target position: diPosition Acceleration: lreAccel Deceleration: lreDecel Jerk limitation (start): lreJerkStart Jerk limitation (end): lreJerkEnd Positioning active: boPosBusy Positioning done: boDone		x
boJogPlus	BOOL	Jog in positive direction Enable signal: With a positive edge, the jogging function starts (positive direction). Jog mode is run as long as 'boJobPlus' = TRUE. Use a negative edge 'boJogPlus' = FALSE to cancel the current jog mode. [Relevant parameters: see 'boJogMinus']		x

Name	Type	Description	Sync.	Async.
boJogMinus	BOOL	Jogging in negative direction Enable signal: With a positive edge, the jogging function starts (negative direction). Jog mode is run as long as 'boJobMinus' = TRUE. Use a negative edge 'boJogMinus' = FALSE to cancel the current jog mode. Relevant parameters: ID32801 'AMK secondary operating mode 1' [Operating mode: position control] [Setpoint source: diMainSetpoint (code 41h)] Relevant variables: Jog velocity: udVelocityJog Acceleration: lreAccel Deceleration: lreDecel Jerk limitation (start): lreJerkStart Jerk limitation (end): lreJerkEnd Jog mode active: boJogBusy		x
udPosVelocity	UDINT	Positioning velocity Unit: incr/s Default value: 34133		x
diPosition	DINT	Target position Use the inputs 'boPosAbs' and 'boPosRel' to determine whether the target position is approached absolutely or relatively. Unit: Increments Default value: 30720		x
lreAccel	LREAL	Acceleration for positioning and jog mode Range: $1{,}43\ 10^{-13} < \lvert\text{'lreAccel'}\rvert < 1{,}43\ 10^{+16}$ Unit: incr/s^2 Default: 1000000		x
lreDecel	LREAL	Deceleration for positioning and jog mode Range: $1{,}43\ 10^{-13} < \lvert\text{'lreDecel'}\rvert < 1{,}43\ 10^{+16}$ Unit: incr/s^2 Default: 1000000		x
lreJerkStart	LREAL	Jerk limitation (start) for positioning and jog mode Unit: incr/s^3 Default: 10000000		x
lreJerkEnd	LREAL	Jerk limitation (end) for positioning and jog mode Unit: incr/s^3 Default: 10000000		x
udVelocityJog	UDINT	Jog velocity Unit: incr/s Default value: 34133		x
diSpeedVelocity	DINT	Speed setpoint for speed control Unit: rpm Default value: 100		x

<table>
<tr><th>Name</th><th>Type</th><th>Description</th><th>Sync.</th><th>Async.</th></tr>
<tr><td>enArithmetik[1)]</td><td>ENUM</td><td>EN_POS_J_ARITHMETIK
(This input is only available for CODESYS V3 function blocks.
CODESYS V2, the selection is always adjusted automatically)

With 'EN_POS_J_ARITHMETIK' the selection 32 bit or 64 bit arithmetic for position function blocks in position operation mode is set.

Options:
<table>
<tr><td>enAutomaticSelection</td><td>Automatic determination if 32 bit or 64 bit arithmetic (Default)
 
The default setting 'enAutomaticSelection' can only be used, if the PLC controller is on the first place in the CODESYS 'Device_ Config'. Is that not the case the selection has to be made manually by the user.</td></tr>
<tr><td>en64BitArithmetik</td><td>64 bit arithmetic</td></tr>
<tr><td>en32BitArithmetik</td><td>32 bit arithmetic</td></tr>
</table>
If the automatic determination does not work, the following diagnostic number is displayed and the selection must done manually by the user.
<table>
<tr><td>iErrID</td><td>5</td></tr>
</table>
Additional information about 'EN_POS_J_ARITHMETIK' and manual selection:
Siehe 'EN_POS_J_ARITHMETIK' auf Seite 601.</td><td></td><td>x</td></tr>
</table>

1) Only for CODESYS V3

Output variables

Name	Type	Description	Sync.	Async.
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled		x
boSBM	BOOL	System ready message		x
boDone	BOOL	Positioning done. Valid for 'boPosAbs', 'boPosRel' and 'boHome'		x
boErr	BOOL	The function block is in an error state FALSE: No error (permitted commanding or warning) TRUE: Error		x
iErrID	INT	Error identity number: Diagnostic number is output iErrID = 0: No error iErrID ≠ 0, boErr = TRUE: Error iErrID ≠ 0, boErr = FALSE: Warning Bus error when 'boErr' = TRUE: EtherCAT or CAN are not in operational mode.	x	
enErrName	ENUM	EN_FB_NAME Name of faulty block for which the diagnostic number is output. The diagnostic number is explained in the description of the corresponding library.		x
boPosBusy	BOOL	Positioning active		x
boJogBusy	BOOL	Jog mode active		x
boHomeBusy	BOOL	Home active		x
boSpeedBusy	BOOL	Speed control active		x

Input and output variables

Name	Type	Description	Sync.	Async.
stDevice	STRUCT	ST_DEVICE The device description structure assigns the block a device. (See document Software description AmkBase Bibliothek, Part no.204986)		x
stAxisDrive	STRUCT	ST_AXIS_DRIVE Siehe 'ST_AXIS_DRIVE (ST)' auf Seite 191.		x

Usage note in the CODESYS program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
STANDARD_AXIS	STANDARD_AXIS.actSync

History

Library	AmkAfl	
System versions	**Functionality**	**Target**
AS-C V3.01/0827	Basic functionality (1st version)	≥ AS_CP_200_0812_T202053
AS-PL15 V3.01/0804	Basic functionality (1st version)	≥ AS_V313_0946_T202724

4.1.4.2.2 Modified 'Standard axes' for ihX drives

All expanded 'Standard axes' can be used for ihX drives.

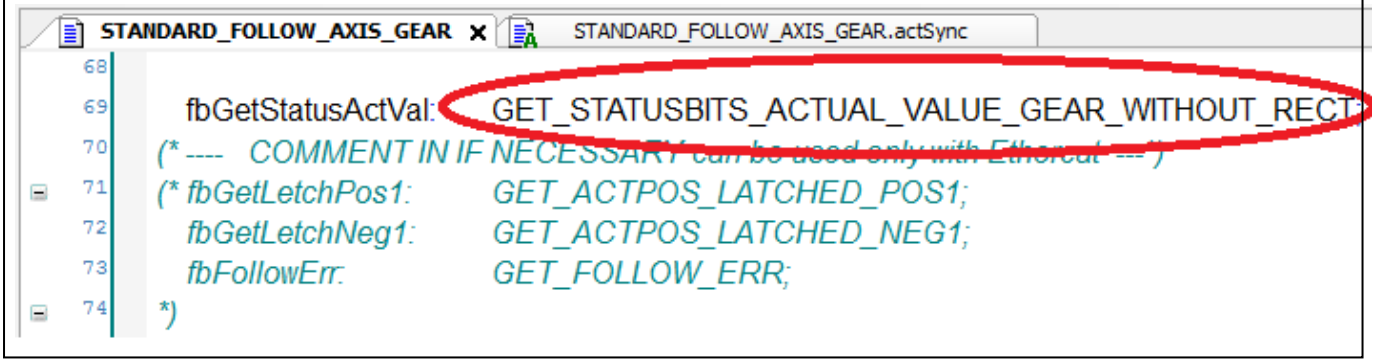
The ihX has no rectangular pulse input. Therefore, the evaluation of the rectangle pulse input must be deactivated.

About the declaration of the function block 'fbGetStatusActVal' the user can decide whether the rectangular pulse input is evaluated or not.

Without rectangular pulse input
Declaration for fbGetStatusActVal:
GET_STATUSBITS_ACTUAL_VALUE_GEAR

Example:

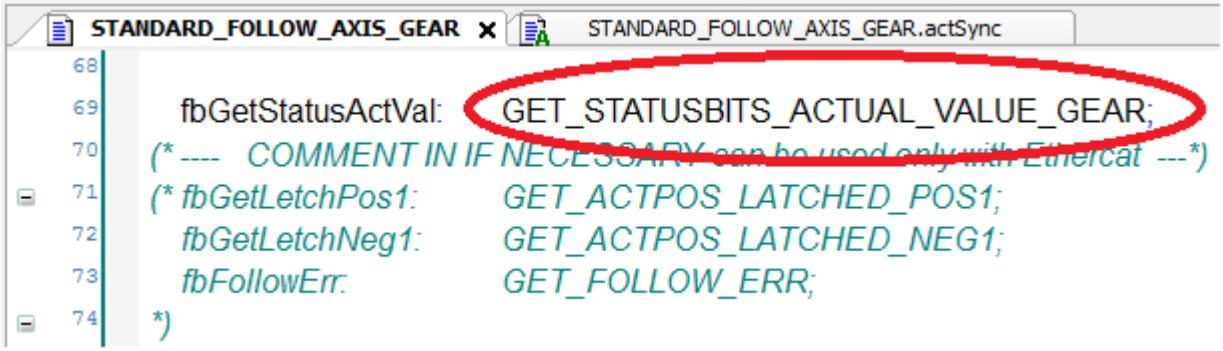
FB STANDARD_FOLLOW_AXIS_GEAR (manually modified for ihX)



With rectangular pulse input
Declaration for fbGetStatusActVal:
GET_STATUSBITS_ACTUAL_VALUE_GEAR

Example:

FB STANDARD_FOLLOW_AXIS_GEAR (original)



4.1.4.2.3 STANDARD_AXIS_POSITION_INTERPOSED (FB)

Expanded functionality:

- Interposed positioning

Siehe 'POSITION_RELATIV_INTERPOSED (FB)' auf Seite 367.

The following basic functionalities for the axis are available:

Inputs/outputs on the function block

- Control bits (RF Controller enable, FL Clear error)
- Emergency stop

- Speed control
- Homing cycle
- Positioning (absolute/relative)
- Jog mode (minus/plus)
- Status signals of axis functions

Access via program code

- Configuration of status messages (ID26 'Configuration status bits')
- Reading of parameters according to a specifications list
- Actual position cached
- Control with feedforward (torque and speed)

User interface

		STANDARD_AXIS_POSITION_INTERPOSED			
FB enable	-	boEnable	boEnabAck	-	Ackn. "FB enable"
Controller enable	-	boRF	boSBM	-	System ready message
Clear error	-	boFL	boDone	-	Ackn. "FB done"
Emergency stop	-	boEmergency_Stop	boErr	-	Error
Start speed control	-	boSpeed	iErrID	-	Error ID
Start homing cycle	-	boHome	enErrName	-	Name of faulty FB
Start absolute pos.	-	boPosAbs	boPosBusy	-	Positioning active
Start relative pos.	-	boPosRel	boJogBusy	-	Jog mode active
Jog plus	-	boJogPlus	boHomeBusy	-	Homing drive active
Jog minus	-	boJogMinus	boSpeedBusy	-	Speed control active
Positioning velocity	-	udPosVelocity	**boInPos_Interpose**	-	In position
Target position	-	diPosition	**boDone_Interpose**	-	Ackn. positioning done
Acceleration	-	lreAccel	**boBusy_Interpose**	-	Movement busy
Deceleration	-	lreDecel			
Jerk limitation at start	-	lreJerkStart			
Jerk limitation at stop	-	lreJerkEnd			
Jog velocity	-	udVelocityJog			
Velocity for speed control	-	diSpeedVelocity			
Start movement	-	**boExec_Interpose**			
Start interpose positioning	-	**boInterpose_Interpose**			
Relative position	-	**diPosition_Interpose**			
Offset correction	-	**diOffset_Interpose**			
Acceleration interpose	-	**udAccel_Interpose**			
Deceleration interpose	-	**udDecel_Interpose**			
Velocity interpose	-	**udVelocity_Interpose**			
Pos. window interpose	-	**diPosWindow_Interpose**			
32bit or 64bit arithmetic	-	enArithmetik[1)]			
Axis structure	-	stAxisDrive	stAxisDrive	-	Axis structure
Device structure	-	stDevice	stDevice	-	Device structure

	actSync	
-	Synchronous operation - opened in synchronous program section (FPLC_PRG)	-
-		-

1) Only for CODESYS V3

Input variables

Name	Type	Description	Sync.	Async.
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.		x
boRF	BOOL	Bit for enabling the controller Relevant parameters: ID32796 'Source RF' Allowed: Code 5: 'Controller enable via EtherCAT' Code 25: RF 'via EtherCAT AND-linked with the binary input UE'  Code 0: 'Controller enable (RF) via binary input' is not allowed		x
boFL	BOOL	Bit for error deletion		x
boEmergency_ Stop	BOOL	EMERGENCY STOP: The setpoint of the velocity is decreased to zero along the emergency-stop ramp. Once initiated, an emergency stop cannot be aborted. Relevant variables: EMERGENCY STOP ramp: Ramp time in which the drive is slowed down from the current velocity to zero. EMERGENCY STOP ramp: ST_AXIS_DRIVE.ST_IDS.ST_ID32919_ PlcList. diID32919_05_diEmergency_StopRamp Unit: ms Default value: 100 ms		x

Name	Type	Description	Sync.	Async.
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). Relevant parameters: ID32802 'AMK secondary operating mode 2' [Operating mode: Speed control] [Setpoint source: diMainSetpoint (code 41h)] [Ramps: active / inactive (bit 6)] ID32780 'Acceleration ramp' ID32781'Deceleration ramp' Relevant variables: Speed setpoint: diSpeedVelocity Velocity ramp: STANDARD_AXIS.fbVelocity.diVelocityRamp [default value: 100 ms] Speed control active: boSpeedBusy		x
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. Relevant parameters: ID41 'Homing velocity' ID147 'Homing parameter' ID150 'Homing offset 1' ID32926 'AMK homing cycle parameter' Relevant variables: Homing drive active: boHomeBusy Homing done: boDone Homing point known: stAxisDrive.stStatus.boID33036_RFP_known		x

Name	Type	Description	Sync.	Async.
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 [Relevant parameters: see 'boPosRel']		x
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. Relevant parameters: ID32801 'AMK secondary operating mode 1' [Operating mode: position control] [Setpoint source: diMainSetpoint (code 41h)] Relevant variables: Positioning velocity: udPosVelocity Target position: diPosition Acceleration: lreAccel Deceleration: lreDecel Jerk limitation (start): lreJerkStart Jerk limitation (end): lreJerkEnd Positioning active: boPosBusy Positioning done: boDone		x
boJogPlus	BOOL	Jog in positive direction Enable signal: With a positive edge, the jogging function starts (positive direction). Jog mode is run as long as 'boJobPlus' = TRUE. Use a negative edge 'boJogPlus' = FALSE to cancel the current jog mode. [Relevant parameters: see 'boJogMinus']		x

Name	Type	Description	Sync.	Async.
boJogMinus	BOOL	Jogging in negative direction Enable signal: With a positive edge, the jogging function starts (negative direction). Jog mode is run as long as 'boJobMinus' = TRUE. Use a negative edge 'boJogMinus' = FALSE to cancel the current jog mode. Relevant parameters: ID32801 'AMK secondary operating mode 1' [Operating mode: position control] [Setpoint source: diMainSetpoint (code 41h)] Relevant variables: Jog velocity: udVelocityJog Acceleration: lreAccel Deceleration: lreDecel Jerk limitation (start): lreJerkStart Jerk limitation (end): lreJerkEnd Jog mode active: boJogBusy		x
udPosVelocity	UDINT	Positioning velocity Unit: incr/s Default value: 34133		x
diPosition	DINT	Target position Use the inputs 'boPosAbs' and 'boPosRel' to determine whether the target position is approached absolutely or relatively. Unit: Increments Default value: 30720		x
lreAccel	LREAL	Acceleration for positioning and jog mode Range: $1{,}43\ 10^{-13} < \lvert\text{'lreAccel'}\rvert < 1{,}43\ 10^{+16}$ Unit: incr/s^2 Default: 1000000		x
lreDecel	LREAL	Deceleration for positioning and jog mode Range: $1{,}43\ 10^{-13} < \lvert\text{'lreDecel'}\rvert < 1{,}43\ 10^{+16}$ Unit: incr/s^2 Default: 1000000		x
lreJerkStart	LREAL	Jerk limitation (start) for positioning and jog mode Unit: incr/s^3 Default: 10000000		x
lreJerkEnd	LREAL	Jerk limitation (end) for positioning and jog mode Unit: incr/s^3 Default: 10000000		x
udVelocityJog	UDINT	Jog velocity Unit: incr/s Default value: 34133		x
diSpeedVelocity	DINT	Speed setpoint for speed control Unit: rpm Default value: 100		x

Name	Type	Description	Sync.	Async.
boExec_ Interpose	BOOL	Enable signal: With a positive edge, the axis moves continuous as long as a positive edge at input boInterpose_Interpose starts the interpose positioning.	x	
boInterpose_ Interpose	BOOL	Start interposed positioning (setpoint diPosition_Interpose)	x	
diPosition_ Interpose	DINT	Target position relative (after boInterpose) Unit: incr/s Default value: 60000	x	
diOffset_ Interpose	DINT	Setpoint position offset correction Unit: incr/s Default value: 0	x	
udAccel_ Interpose	UDINT	Deceleration for interpose positioning Unit: incr/s^2 Default value: 1000000	x	
udDecel_ Interpose	UDINT	Deceleration for interpose positioning Unit: incr/s^2 Default value: 1000000	x	
udVelocity_ Interpose	UDINT	Positioning velocity Unit: incr/s Default value: 34133	x	
diPosWindow_ Interpose	DINT	Range window for the message 'In position' (boInPos_Interpose) Unit: incr/s Default value: 1000	x	

<table>
<tr><th>Name</th><th>Type</th><th>Description</th><th>Sync.</th><th>Async.</th></tr>
<tr><td>enArithmetik[1)]</td><td>ENUM</td><td>EN_POS_J_ARITHMETIK
(This input is only available for CODESYS V3 function blocks.
CODESYS V2, the selection is always adjusted automatically)

With 'EN_POS_J_ARITHMETIK' the selection 32 bit or 64 bit arithmetic for position function blocks in position operation mode is set.

Options:
<table><tr><td>enAutomaticSelection</td><td>Automatic determination if 32 bit or 64 bit arithmetic (Default)
 
The default setting 'enAutomaticSelection' can only be used, if the PLC controller is on the first place in the CODESYS 'Device_ Config'. Is that not the case the selection has to be made manually by the user.</td></tr><tr><td>en64BitArithmetik</td><td>64 bit arithmetic</td></tr><tr><td>en32BitArithmetik</td><td>32 bit arithmetic</td></tr></table>
If the automatic determination does not work, the following diagnostic number is displayed and the selection must done manually by the user.
<table><tr><td>iErrID</td><td>5</td></tr></table>
Additional information about 'EN_POS_J_ARITHMETIK' and manual selection:
Siehe 'EN_POS_J_ARITHMETIK' auf Seite 601.</td><td></td><td>x</td></tr>
</table>

Output variables

Name	Type	Description	Sync.	Async.
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled		x
boSBM	BOOL	System ready message		x
boDone	BOOL	Positioning done. Valid for 'boPosAbs', 'boPosRel' and 'boHome'		x
boErr	BOOL	The function block is in an error state FALSE: No error (permitted commanding or warning) TRUE: Error		x
iErrID	INT	Error identity number: Diagnostic number is output iErrID = 0: No error iErrID ≠ 0, boErr = TRUE: Error iErrID ≠ 0, boErr = FALSE: Warning		x
enErrName	ENUM	EN_FB_NAME Name of faulty block for which the diagnostic number is output. The diagnostic number is explained in the description of the corresponding library.		x
boPosBusy	BOOL	Positioning active		x
boJogBusy	BOOL	Jog mode active		x
boHomeBusy	BOOL	Home active		x
boSpeedBusy	BOOL	Speed control active		x
boInPos_Interpose	BOOL	Message "InPosition": Actual position = target position within the position window (diPosWindow_Interpose).	x	
boDone_Interpose	BOOL	Response that the function block has been completely executed.	x	
boBusy_Interpose	BOOL	Execution message: This bit remains set as long as the block is being processed.	x	

Input and output variables

Name	Type	Description	Sync.	Async.
stDevice	STRUCT	ST_DEVICE The device description structure assigns the block a device. (See document Software description AmkBase Bibliothek, Part no.204986)		x
stAxisDrive	STRUCT	ST_AXIS_DRIVE Siehe 'ST_AXIS_DRIVE (ST)' auf Seite 191.		x

Usage note in the CODESYS program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
STANDARD_AXIS	STANDARD_AXIS.actSync

History

Library	AmkAfl	
System versions	**Functionality**	**Target**
AS-C V3.01/0827	Basic functionality (1st version)	≥ AS_CP_200_0812_T202053
AS-PL15 V3.01/0804	Basic functionality (1st version)	≥ AS_V313_0946_T202724

4.1.4.2.4 STANDARD_AXIS_POSITION_TO_THE_MARK (FB)

Expanded functionality:

- Printing mark control

Siehe 'POSITION_RELATIV_INTERPOSED (FB)' auf Seite 367.

The following basic functionalities for the axis are available:

Inputs/outputs on the function block

- Control bits (RF Controller enable, FL Clear error)
- Emergency stop
- Speed control
- Homing cycle
- Positioning (absolute/relative)
- Jog mode (minus/plus)
- Status signals of axis functions

Access via program code

- Configuration of status messages (ID26 'Configuration status bits')
- Reading of parameters according to a specifications list
- Actual position cached
- Control with feedforward (torque and speed)

PM function description
Abbreviation: PM printing mark

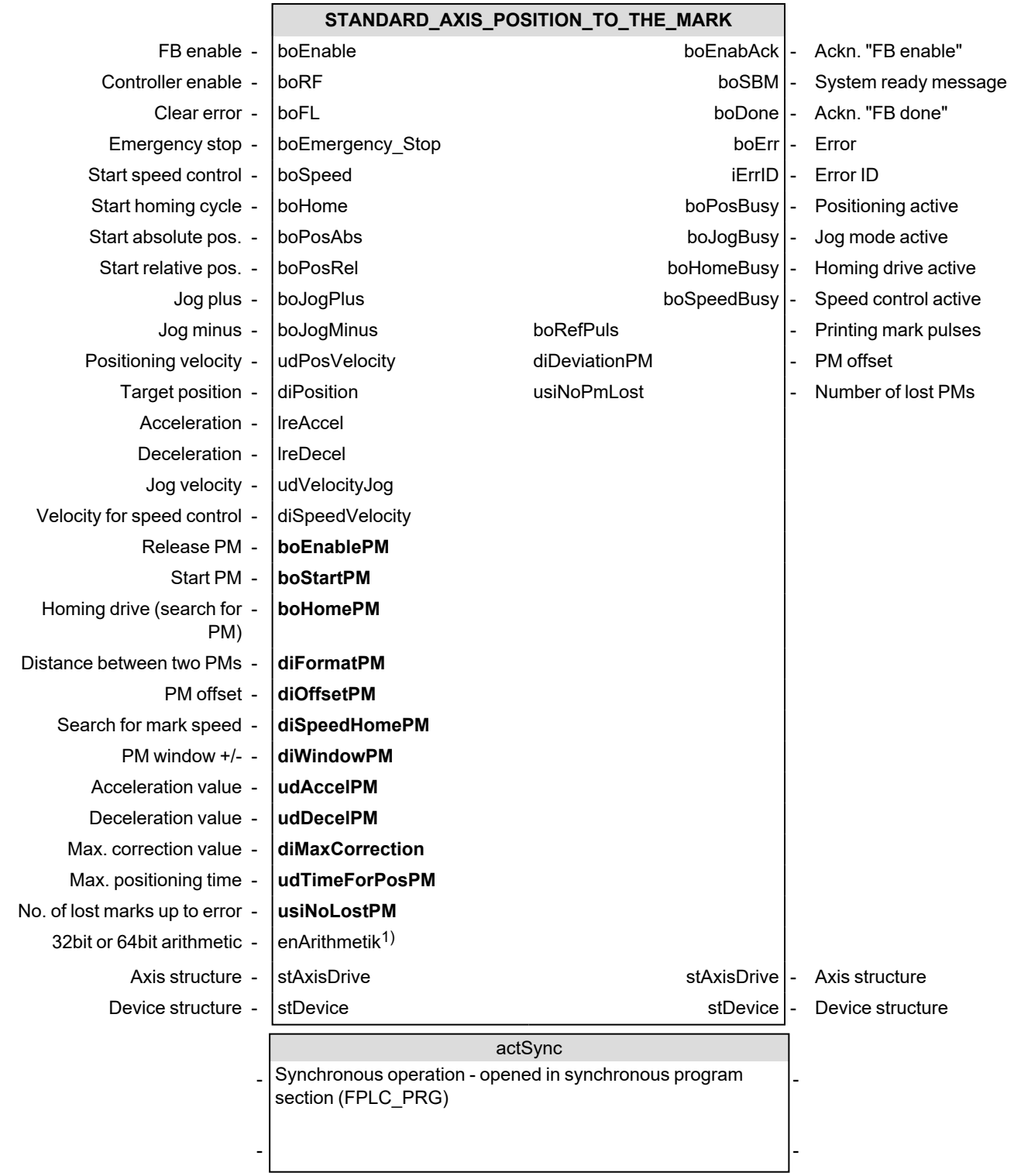
FB 'STANDARD_AXIS_POSITION_TO_THE_MARK' performs printing mark control relatively inconsistently.

Based on the

- acceleration and deceleration values **a** (udAccelPM and udDecelPM [s^2])
- movement **s** (diFormatPM [increments]) + offset [increments]
- and time **t** in which the positioning must be completed (udTimeForPosPM [ms])

the FB calculates a movement profile.

User interface



1) Only for CODESYS V3

Input variables

Name	Type	Description	Sync.	Async.
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.		x

Name	Type	Description	Sync.	Async.
boRF	BOOL	Bit for enabling the controller Relevant parameters: ID32796 'Source RF' Allowed: Code 5: 'Controller enable via EtherCAT' Code 25: RF 'via EtherCAT AND-linked with the binary input UE'  Code 0: 'Controller enable (RF) via binary input' is not allowed		x
boFL	BOOL	Bit for error deletion		x
boEmergency_Stop	BOOL	EMERGENCY STOP: The setpoint of the velocity is decreased to zero along the emergency-stop ramp. Once initiated, an emergency stop cannot be aborted. Relevant variables: EMERGENCY STOP ramp: Ramp time in which the drive is slowed down from the current velocity to zero. EMERGENCY STOP ramp: ST_AXIS_DRIVE.ST_IDS.ST_ID32919_ PlcList. diID32919_05_diEmergency_StopRamp Unit: ms Default value: 100 ms		x

Name	Type	Description	Sync.	Async.
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). Relevant parameters: ID32802 'AMK secondary operating mode 2' [Operating mode: Speed control] [Setpoint source: diMainSetpoint (code 41h)] [Ramps: active / inactive (bit 6)] ID32780 'Acceleration ramp' ID32781'Deceleration ramp' Relevant variables: Speed setpoint: diSpeedVelocity Velocity ramp: STANDARD_AXIS.fbVelocity.diVelocityRamp [default value: 100 ms] Speed control active: boSpeedBusy		x
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. Relevant parameters: ID41 'Homing velocity' ID147 'Homing parameter' ID150 'Homing offset 1' ID32926 'AMK homing cycle parameter' Relevant variables: Homing drive active: boHomeBusy Homing done: boDone Homing point known: stAxisDrive.stStatus.boID33036_RFP_known		x

Name	Type	Description	Sync.	Async.
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 [Relevant parameters: see 'boPosRel']		x
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. Relevant parameters: ID32801 'AMK secondary operating mode 1' [Operating mode: position control] [Setpoint source: diMainSetpoint (code 41h)] Relevant variables: Positioning velocity: udPosVelocity Target position: diPosition Acceleration: lreAccel Deceleration: lreDecel Jerk limitation (start): lreJerkStart Jerk limitation (end): lreJerkEnd Positioning active: boPosBusy Positioning done: boDone		x
boJogPlus	BOOL	Jog in positive direction Enable signal: With a positive edge, the jogging function starts (positive direction). Jog mode is run as long as 'boJobPlus' = TRUE. Use a negative edge 'boJogPlus' = FALSE to cancel the current jog mode. [Relevant parameters: see 'boJogMinus']		x

Name	Type	Description	Sync.	Async.
boJogMinus	BOOL	Jogging in negative direction Enable signal: With a positive edge, the jogging function starts (negative direction). Jog mode is run as long as 'boJobMinus' = TRUE. Use a negative edge 'boJogMinus' = FALSE to cancel the current jog mode. Relevant parameters: ID32801 'AMK secondary operating mode 1' [Operating mode: position control] [Setpoint source: diMainSetpoint (code 41h)] Relevant variables: Jog velocity: udVelocityJog Acceleration: lreAccel Deceleration: lreDecel Jerk limitation (start): lreJerkStart Jerk limitation (end): lreJerkEnd Jog mode active: boJogBusy		x
udPosVelocity	UDINT	Positioning velocity Unit: incr/s Default value: 34133		x
diPosition	DINT	Target position Use the inputs 'boPosAbs' and 'boPosRel' to determine whether the target position is approached absolutely or relatively. Unit: Increments Default value: 30720		x
lreAccel	LREAL	Acceleration for positioning and jog mode Range: $1{,}43\ 10^{-13} < \lvert\text{'lreAccel'}\rvert < 1{,}43\ 10^{+16}$ Unit: incr/s^2 Default: 1000000		x
lreDecel	LREAL	Deceleration for positioning and jog mode Range: $1{,}43\ 10^{-13} < \lvert\text{'lreDecel'}\rvert < 1{,}43\ 10^{+16}$ Unit: incr/s^2 Default: 1000000		x
lreJerkStart	LREAL	Jerk limitation (start) for positioning and jog mode Unit: incr/s^3 Default: 10000000		x
lreJerkEnd	LREAL	Jerk limitation (end) for positioning and jog mode Unit: incr/s^3 Default: 10000000		x
udVelocityJog	UDINT	Jog velocity Unit: incr/s Default value: 34133		x
diSpeedVelocity	DINT	Speed setpoint for speed control Unit: rpm Default value: 100		x

Name	Type	Description	Sync.	Async.
boEnablePM	BOOL	Enables the BS. The entered values are checked and the movement profile is calculated. Relevant parameters: ID32801 'AMK secondary operating mode 1' [Operating mode: position control] [Setpoint source: diMainSetpoint (code 41h)] ID32980 'Port 3 Bit 2'Code 401 ID32948 'Message 4x32' Code 24h ID169 'Probe control parameter'Code 1h Relevant variables: Setpoint specification: diInVal Table selection: boEnabOpTab2 Pointer to table 1: pstOpTab1 Pointer to table 2: pstOpTab2		x
boStartPM	BOOL	Start of PM positioning Relevant parameters: Distance between two PMs: diFormatPM Distance to PM: diOffsetPM PM window: diWindowPM Acceleration: udAccelPM Deceleration: udDecelPM Max. positioning time: iudTimeForPosPM Max. correction value: diMaxCorrection Max. lost PM: usiNoLostPM	x	
boHomePM	BOOL	Start of homing drive; axis move to PM + offset Relevant variables: Homing velocity: diSpeedHomePM Offset: diOffsetPM	x	
diFormatPM	DINT	Distance between two PMs Unit: Increments Default value: 20480	x	
diOffsetPM	DINT	Distance to PM (distance between PM sensor and stop position) Unit: Increments Default value: 10240	x	
diSpeedHomePM	DINT	PM homing drive velocity Unit: Increments/s Default value: 10000	x	
diWindowPM	DINT	Window in which the marks must be located for detection (+/- target position) Unit: Increments Default value: 20480	x	

<table>
<tr><th>Name</th><th>Type</th><th>Description</th><th>Sync.</th><th>Async.</th></tr>
<tr><td>udAccelPM</td><td>UDINT</td><td>PM acceleration
Unit: Increments/s²
Default value: 1000000</td><td>x</td><td></td></tr>
<tr><td>udDecelPM</td><td>UDINT</td><td>PM deceleration
Unit: Increments/s²
Default value: 1000000</td><td>x</td><td></td></tr>
<tr><td>udTimeForPosPM</td><td>UDINT</td><td>Time in which the positioning must be completed.
Unit: ms
Default value: 2000</td><td>x</td><td></td></tr>
<tr><td>diMaxCorrection</td><td>DINT</td><td>Max. correction value
Default 0 = The calculated correction value is always computed without limit.
Unit: Increments
Default value: 20480</td><td>x</td><td></td></tr>
<tr><td>usiNoLostPM</td><td>USINT</td><td>Once the preset number of marks is lost, an error message is generated.
Default 0 = Error message is never generated
Unit: Increments
Default value: 10</td><td>x</td><td></td></tr>
<tr><td>enArithmetik[1)]</td><td>ENUM</td><td>EN_POS_J_ARITHMETIK
(This input is only available for CODESYS V3 function blocks. CODESYS V2, the selection is always adjusted automatically)

With 'EN_POS_J_ARITHMETIK' the selection 32 bit or 64 bit arithmetic for position function blocks in position operation mode is set.

Options:
<table><tr><td>enAutomaticSelection</td><td>Automatic determination if 32 bit or 64 bit arithmetic (Default)
 
The default setting 'enAutomaticSelection' can only be used, if the PLC controller is on the first place in the CODESYS 'Device_ Config'. Is that not the case the selection has to be made manually by the user.</td></tr><tr><td>en64BitArithmetik</td><td>64 bit arithmetic</td></tr><tr><td>en32BitArithmetik</td><td>32 bit arithmetic</td></tr></table>
If the automatic determination does not work, the following diagnostic number is displayed and the selection must done manually by the user.
<table><tr><td>iErrID</td><td>5</td></tr></table>
Additional information about 'EN_POS_J_ARITHMETIK' and manual selection:
Siehe 'EN_POS_J_ARITHMETIK' auf Seite 601.</td><td></td><td>x</td></tr>
</table>

1) Only for CODESYS V3

Output variables

Name	Type	Description	Sync.	Async.
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled		x
boSBM	BOOL	System ready message		x
boDone	BOOL	Positioning done. Valid for 'boPosAbs', 'boPosRel' and 'boHome'		x
boErr	BOOL	The function block is in an error state FALSE: No error (permitted commanding or warning) TRUE: Error		x
iErrID	INT	Error identity number: Diagnostic number is output iErrID = 0: No error iErrID ≠ 0, boErr = TRUE: Error iErrID ≠ 0, boErr = FALSE: Warning		x
enErrName	ENUM	EN_FB_NAME Name of faulty block for which the diagnostic number is output. The diagnostic number is explained in the description of the corresponding library.		x
boPosBusy	BOOL	Positioning active		x
boJogBusy	BOOL	Jog mode active		x
boHomeBusy	BOOL	Home active		x
boSpeedBusy	BOOL	Speed control active		x
boDone	BOOL	Response that the function block has been completely executed.		x
boRefPuls	BOOL	For one cycle boRefPuls = TRUE when the latch value is changed.	x	
diDeviationPM	DINT	Deviation between actual and target position of the last found PM. Unit: Increments	x	
usiNoPmLost	USINT	Number of lost printing marks	x	

Input and output variables

Name	Type	Description	Sync.	Async.
stDevice	STRUCT	ST_DEVICE The device description structure assigns the block a device. (See document Software description AmkBase Bibliothek, Part no.204986)		x
stAxisDrive	STRUCT	ST_AXIS_DRIVE Siehe 'ST_AXIS_DRIVE (ST)' auf Seite 191.		x

Usage note in the CODESYS program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
STANDARD_AXIS	STANDARD_AXIS.actSync

History

Library	AmkAfl	
System versions	**Functionality**	**Target**
AS-C V3.01/0827	Basic functionality (1st version)	≥ AS_CP_200_0812_T202053
AS-PL15 V3.01/0804	Basic functionality (1st version)	≥ AS_V313_0946_T202724

4.1.4.2.5 STANDARD_AXIS_PRINTMARK_INSETTER_CONT (FB)

Expanded functionality:

- Print mark control: The position of the web to be treated is controlled in relation to a master format (insetting)

Siehe 'PRINT_MARK_INSETTER_CONT (FB)' auf Seite 490.

The following basic functionalities for the axis are available:

Inputs/outputs on the function block

- Control bits (RF Controller enable, FL Clear error)
- Emergency stop
- Speed control
- Homing cycle
- Positioning (absolute/relative)
- Jog mode (minus/plus)
- Status signals of axis functions

Access via program code

- Configuration of status messages (ID26 'Configuration status bits')
- Reading of parameters according to a specifications list
- Actual position cached
- Control with feedforward (torque and speed)

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

Setpoints and current values are transferred in a synchronous action. The synchronous action must be called in the synchronous program level FPLC_PRG.

User interface

	STANDARD_AXIS_PRINTMARK_INSETTER_CONT		
FB enable -	boEnable	boEnabAck	- Ackn. "FB enable"
Controller enable -	boRF	boSBM	- System ready message
Clear error -	boFL	boDone	- Ackn. "FB done"
Emergency stop -	boEmergency_Stop	boErr	- Error
Start speed control -	boSpeed	iErrID	- Error ID
Start homing cycle -	boHome	enErrName	- Name of faulty FB
Start absolute pos. -	boPosAbs	boPosBusy	- Positioning active
Start relative pos. -	boPosRel	boJogBusy	- Jog mode active
Jog plus -	boJogPlus	boHomeBusy	- Homing drive active
Jog minus -	boJogMinus	boSpeedBusy	- Speed control active
Positioning velocity -	udPosVelocity	**boEnabAckPmInsetterCont**	- Printmark block initialized and activated
Target position -	diPosition	**boParameterSetupDone**	- Parameterisation finished
Acceleration -	lreAccel	**boPmDetected**	- Print mark detected
Deceleration -	lreDecel	**boPmLost**	- Print mark lost
Jerk limit at start -	lreJerkStart	**boPmWnd**	- Print mark window active

Jerk limit at stop	-	lreJerkEnd	**diPmDeviation**	-	Deviation between setpoint and actual position
Jog velocity	-	udVelocityJog	**diOpOffsetOut**	-	Output operator offset
Velocity for speed control	-	diSpeedVelocity	**iActPmLost**	-	Number of lost print marks
Enable PM control	-	**boEnablePmInsetterCont**			
Reset output boPmLost	-	**boResetPmLost**			
Start output operator offset	-	**boSetupOffset**			
Start with print mark setpoint position	-	**boSetupPmPos**			
Master counts negative	-	**boMasterNegDir**			
Motor direction negative	-	**boMotorNegDir**			
Enable correction output	-	**boCorrOutControlOn**			
Operator offset	-	**diOpOffset**			
Format length	-	**diFormatLength**			
Window length	-	**diWindowLength**			
Number of marks until message 'mark lost'	-	**iNoPmLost**			
Correction range	-	**diCorrectionRange**			
Print mark position setpoint	-	**diPmSetupPos**			
Max. correction per format	-	**diCorrectionLimit**			
Start correction range 1	-	**diCorrRng1Start**			
End correction range 1	-	**diCorrRng1End**			
Offset correction control 1	-	**diCorrOutControl1PreSet**			
Start correction range 2	-	**diCorrRng2Start**			
End correction range 2	-	**diCorrRng2End**			
Offset correction control 2	-	**diCorrOutControl2PreSet**			
Start print mark detection	-	**boStartPmDet**			
Print mark detected	-	**boPmRefPulse**			
Master pulses	-	**diCount**			
Print mark offset	-	**diPmOffset**			
32bit or 64bit arithmetic	-	enArithmetik[1)]			
Axis structure	-	stAxisDrive	stAxisDrive	-	Axis structure
Device structure	-	stDevice	stDevice	-	Device structure

actSync	
Synchronous operation - opened in synchronous program section (FPLC_PRG)	-
	-

1) Only for CODESYS V3

Input variables

Name	Type	Description	Sync.	Async.
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.		x

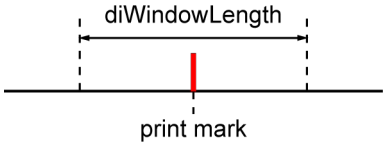
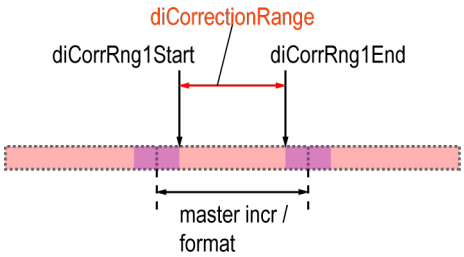
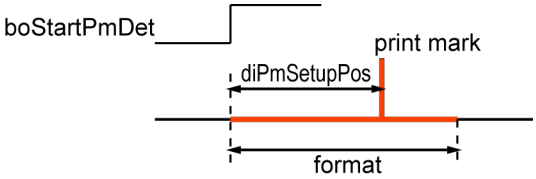
Name	Type	Description	Sync.	Async.
boRF	BOOL	Bit for enabling the controller Relevant parameters: ID32796 'Source RF' Allowed: Code 5: 'Controller enable via EtherCAT' Code 25: RF 'via EtherCAT AND-linked with the binary input UE'  Code 0: 'Controller enable (RF) via binary input' is not allowed		x
boFL	BOOL	Bit for error deletion		x
boEmergency_Stop	BOOL	EMERGENCY STOP: The setpoint of the velocity is decreased to zero along the emergency-stop ramp. Once initiated, an emergency stop cannot be aborted. Relevant variables: EMERGENCY STOP ramp: Ramp time in which the drive is slowed down from the current velocity to zero. EMERGENCY STOP ramp: ST_AXIS_DRIVE.ST_IDS.ST_ ID32919_PlcList. diID32919_05_diEmergency_StopRamp Unit: ms Default value: 100 ms		x

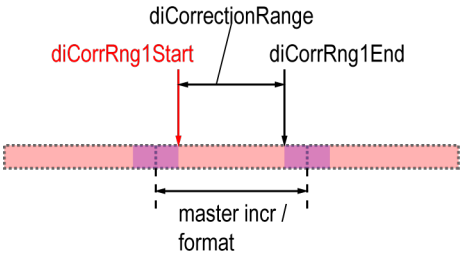
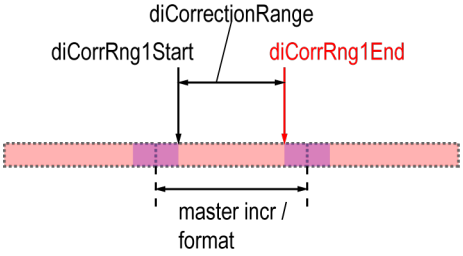
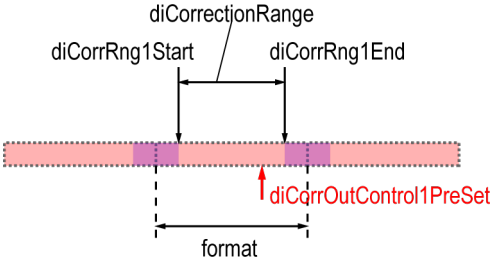
Name	Type	Description	Sync.	Async.
boSpeed	BOOL	Speed control Enable signal: With a positive edge, the speed control function starts. As long as 'boSpeed' = TRUE, speed control (speed setpoint) is carried out. Use a negative edge 'boSpeed' = FALSE to cancel the current speed control (with setpoint value = 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). Relevant parameters: ID32802 'AMK secondary operating mode 2' [Operating mode: Speed control] [Setpoint source: diMainSetpoint (code 41h)] [Ramps: active / inactive (bit 6)] ID32780 'Acceleration ramp' ID32781'Deceleration ramp' Relevant variables: Speed setpoint: diSpeedVelocity Velocity ramp: STANDARD_AXIS.fbVelocity.diVelocityRamp [default value: 100 ms] Speed control active: boSpeedBusy		x

Name	Type	Description	Sync.	Async.
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. Relevant parameters: ID41 'Homing velocity' ID147 'Homing parameter' ID150 'Homing offset 1' ID32926 'AMK homing cycle parameter' Relevant variables: Homing drive active: boHomeBusy Homing done: boDone Homing point known: stAxisDrive.stStatus.boID33036_RFP_known		x
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 [Relevant parameters: see 'boPosRel']		x
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. Relevant parameters: ID32801 'AMK secondary operating mode 1' [Operating mode: position control] [Setpoint source: diMainSetpoint (code 41h)] Relevant variables: Positioning velocity: udPosVelocity Target position: diPosition Acceleration: lreAccel Deceleration: lreDecel Jerk limitation (start): lreJerkStart Jerk limitation (end): lreJerkEnd Positioning active: boPosBusy Positioning done: boDone		x

Name	Type	Description	Sync.	Async.
boJogPlus	BOOL	Jog in positive direction Enable signal: With a positive edge, the jogging function starts (positive direction). Jog mode is run as long as 'boJobPlus' = TRUE. Use a negative edge 'boJogPlus' = FALSE to cancel the current jog mode. [Relevant parameters: see 'boJogMinus']		x
boJogMinus	BOOL	Jogging in negative direction Enable signal: With a positive edge, the jogging function starts (negative direction). Jog mode is run as long as 'boJobMinus' = TRUE. Use a negative edge 'boJogMinus' = FALSE to cancel the current jog mode. Relevant parameters: ID32801 'AMK secondary operating mode 1' [Operating mode: position control] [Setpoint source: diMainSetpoint (code 41h)] Relevant variables: Jog velocity: udVelocityJog Acceleration: lreAccel Deceleration: lreDecel Jerk limitation (start): lreJerkStart Jerk limitation (end): lreJerkEnd Jog mode active: boJogBusy		x
udPosVelocity	UDINT	Positioning velocity Unit: incr/s Default value: 34133		x
diPosition	DINT	Target position Use the inputs 'boPosAbs' and 'boPosRel' to determine whether the target position is approached absolutely or relatively. Unit: Increments Default value: 30720		x
lreAccel	LREAL	Acceleration for positioning and jog mode Range: $1{,}43\ 10^{-13} < \lvert\text{'lreAccel'}\rvert < 1{,}43\ 10^{+16}$ Unit: incr/s^2 Default: 1000000		x
lreDecel	LREAL	Deceleration for positioning and jog mode Range: $1{,}43\ 10^{-13} < \lvert\text{'lreDecel'}\rvert < 1{,}43\ 10^{+16}$ Unit: incr/s^2 Default: 1000000		x
lreJerkStart	LREAL	Jerk limitation (start) for positioning and jog mode Unit: incr/s^3 Default: 10000000		x

Name	Type	Description	Sync.	Async.
lreJerkEnd	LREAL	Jerk limitation (end) for positioning and jog mode Unit: incr/s^3 Default: 10000000		x
udVelocityJog	UDINT	Jog velocity Unit: incr/s Default value: 34133		x
diSpeedVelocity	DINT	Speed setpoint for speed control Unit: rpm Default value: 100		x
boEnablePmInsetterCont	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. The input increment detection is enabled. Increment changes at the input 'diInVal' lead to a change of the master reference point. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.		x
boResetPmLost	BOOL	Resetting the output 'boPmLost'		x
boSetupOffset	BOOL	Output start of the specified value of 'diOpOffset'. This value is used as an offset to shift the printing mark set position		x
boSetupPmPos	BOOL	Mode how to determine the nominal position of the print mark on a positive edge of 'boStartPmDet' 'boPmSetupPos' = FALSE: The position of the next print mark is set as nominal position. All following print marks are adjusted to this position. 'boPmSetupPos' = TRUE: The parameter 'diPmSetupPos' specifies the print mark relatively to the actual master position. The following print marks are adjusted to this position		x
boMasterNegDir	BOOL	Direction of master pulse negated 'boMasterNegDir' = FALSE: no negation 'boMasterNegDir' = TRUE: negation This input must only be set on a program start.		x
boMotorNegDir	BOOL	Direction of motor rotation negated 'boMotorNegDir' = FALSE: no negation 'boMotorNegDir' = TRUE: negation This input must only be set on a program start.		x

Name	Type	Description	Sync.	Async.
boCorrOutControlOn	BOOL	Correction output enabled 'boCorrOutControlOn' = FALSE: always correction output 'boCorrOutControlOn' = TRUE: correction output only in the ranges specified by 'diCorrRng1Start' and 'diCorrRng1End' resp. 'diCorrRng2Start' and 'diCorrRng2End'. The ranges are OR operated The modulo counting is started with a 0 -> 1 edge at 'boCorrOutControlOn' and can be preset by 'diCorrOutControl1PreSet' or 'diCorrOutControl2PreSet'		x
diOpOffset	DINT	Operator offset. Shifts the set position of the print mark. The shift takes place in the correction range and starts with a 0 → 1 edge at 'boSetupOffset'. It is independent of changes in the value of 'diCount'. The offset is calculated as		x
diFormatLength	DINT	Format length Nominal distance between two print marks [increments]		x
diWindowLength	DINT	Window length Length in front and after the nominal print mark position. In this range, the print mark sensor is activated and will accept a print mark. 		x
iNoPmLost	INT	Number of print marks to be missed in succession, before the output 'boPmLost' is set.		x
diCorrectionRange	DINT	Correction range The print mark control tries to output the correction in the specified range [% diFormatLength] 		x
diPmSetupPos	DINT	The parameter specifies the nominal position of the print mark relatively to the actual master position on a positive edge of 'boStartPmDet'. The detected print marks are controlled to this position [increments] 		x
diCorrectionLimit	DINT	Maximum correction output per format [increments] 'diCorrectionLimit' = 0: no limitation 'diCorrectionLimit' > 0: maximal value		x

Name	Type	Description	Sync.	Async.
diCorrRng1Start **diCorrRng2Start**	DINT	Correction range 1 / 2 Start value related to master increments / format [increments] 		x
diCorrRng1End **diCorrRng2End**	DINT	Correction range 1 / 2 End value related to master increments / format [increments] 		x
diCorrOutControl1PreSet **diCorrOutControl2PreSet**	DINT	Actual position value (zero offset) at start of the correction control, related to the master increments / format [increments] Example: If the controlled drive is positioned in the middle of the format when starting the correction control, >format/2< must be set on a 0 -> 1 edge of 'boCorrOutControlOn' 		x
boStartPmDet	BOOL	Start of the print mark control and specification of the print mark setpoint position by parameter 'diPmSetupPos' if 'boSetupPmPos' = TRUE	x	
boPmRefPulse	BOOL	Print mark pulse detected (reference pulse) (See document Software description AmkBase Bibliothek, Part no. 204986) BasicSupport TIME_TO_COUNT	x	
diCount	DINT	Counter value Masterimpulse [Inkremente] (See documentSoftware descriptionAmkBase Bibliothek, Part no. 204986) BasicSupport TIME_TO_COUNT	x	
diPmOffset	DINT	Print mark offset of the actual master position (See documentSoftware descriptionAmkBase Bibliothek, Part no. 204986) BasicSupport TIME_TO_COUNT	x	

<table>
<tr><th>Name</th><th>Type</th><th>Description</th><th>Sync.</th><th>Async.</th></tr>
<tr><td>enArithmetik[1)]</td><td>ENUM</td><td>EN_POS_J_ARITHMETIK
(This input is only available for CODESYS V3 function blocks. CODESYS V2, the selection is always adjusted automatically)

With 'EN_POS_J_ARITHMETIK' the selection 32 bit or 64 bit arithmetic for position function blocks in position operation mode is set.

Options:
<table><tr><td>enAutomaticSelection</td><td>Automatic determination if 32 bit or 64 bit arithmetic (Default)
 
The default setting 'enAutomaticSelection' can only be used, if the PLC controller is on the first place in the CODESYS 'Device_ Config'. Is that not the case the selection has to be made manually by the user.</td></tr><tr><td>en64BitArithmetik</td><td>64 bit arithmetic</td></tr><tr><td>en32BitArithmetik</td><td>32 bit arithmetic</td></tr></table>
If the automatic determination does not work, the following diagnostic number is displayed and the selection must done manually by the user.
<table><tr><td>iErrID</td><td>5</td></tr></table>
Additional information about 'EN_POS_J_ARITHMETIK' and manual selection:
Siehe 'EN_POS_J_ARITHMETIK' auf Seite 601.</td><td></td><td>x</td></tr>
</table>

1) Only for CODESYS V3

Output variables

Name	Type	Description	Sync.	Async.
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled		x
boSBM	BOOL	System ready message		x
boDone	BOOL	Positioning done. Valid for 'boPosAbs', 'boPosRel' and 'boHome'		x
boErr	BOOL	The function block is in an error state FALSE: No error (permitted commanding or warning) TRUE: Error		x
iErrID	INT	Error identity number: Diagnostic number is output iErrID = 0: No error iErrID ≠ 0, boErr = TRUE: Error iErrID ≠ 0, boErr = FALSE: Warning Bus error when 'boErr' = TRUE: EtherCAT or CAN are not in operational mode.		x
enErrName	ENUM	EN_FB_NAME Name of faulty block for which the diagnostic number is output. The diagnostic number is explained in the description of the corresponding library.		x
boPosBusy	BOOL	Positioning active		x
boJogBusy	BOOL	Jog mode active		x
boHomeBusy	BOOL	Home active		x
boSpeedBusy	BOOL	Speed control active		x
boEnabAckPmInsetterCont	BOOL	Print mark module initialized and active		x
boParameterSetupDone	BOOL	Parameterisation finished		x
boPmDetected	BOOL	Print mark detected, control activated		x
boPmLost	BOOL	Print mark lost activated when the number of print marks set in 'iNoPmLost' is not detected successively.		x
boPmWnd	BOOL	Mark window activates the print mark sensor		x
diPmDeviation	DINT	Deviation between setpoint and actual position of the actual print mark		x
diOpOffsetOut	DINT	Output value contains the actually output operator offset		x
iActPmLost	INT	Number of lost print marks		x

Input and output variables

Name	Type	Description	Sync.	Async.
stDevice	STRUCT	ST_DEVICE The device description structure assigns the block a device. (See document Software description AmkBase Bibliothek, Part no.204986)		x
stAxisDrive	STRUCT	ST_AXIS_DRIVE Siehe 'ST_AXIS_DRIVE (ST)' auf Seite 191.		x

Usage note in the CODESYS program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
STANDARD_AXIS_PRINTMARK_INSETTER_CONT	STANDARD_AXIS_PRINTMARK_INSETTER_CONT.actSync

History

Library	AmkAfl	
System versions	**Functionality**	**Target**
AS-C V3.01/0827	Basic functionality (1st version)	≥ AS_CP_200_0812_T202053
AS-PL15 V3.01/0804	Basic functionality (1st version)	≥ AS_V313_0946_T202724

4.1.4.2.6 STANDARD_AXIS_PRINTMARK_INSETTER_INTERVAL (FB)

Expanded functionality:

- Print mark control: The position of the web to be treated is controlled in relation to a master format (insetting)

Siehe 'PRINT_MARK_INSETTER_INTERVAL (FB)' auf Seite 495.

The following basic functionalities for the axis are available:

Inputs/outputs on the function block

- Control bits (RF Controller enable, FL Clear error)
- Emergency stop
- Speed control
- Homing cycle
- Positioning (absolute/relative)
- Jog mode (minus/plus)
- Status signals of axis functions

Access via program code

- Configuration of status messages (ID26 'Configuration status bits')
- Reading of parameters according to a specifications list
- Actual position cached
- Control with feedforward (torque and speed)

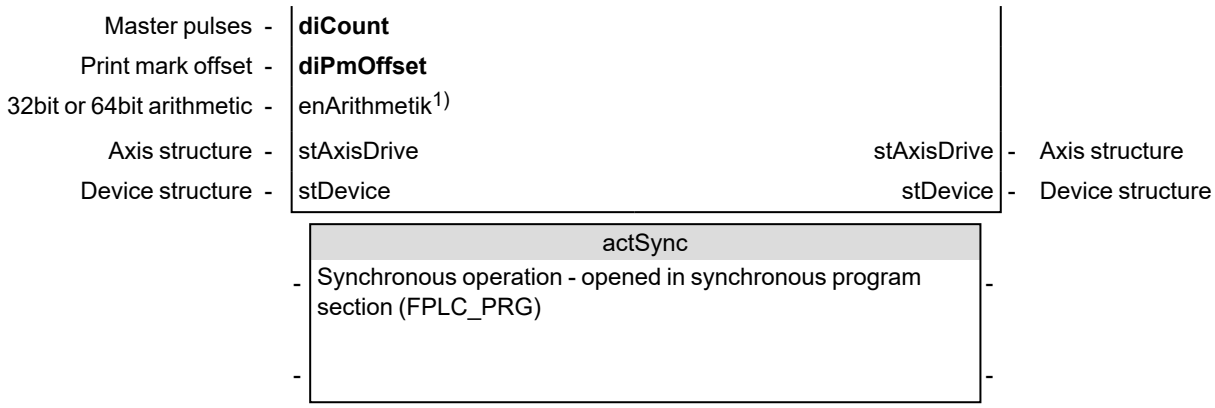
The function block is called in the asynchronous program level PLC_PRG.

Setpoints and current values are transferred in a synchronous action. The synchronous action must be called in the synchronous program level FPLC_PRG.

User interface

	STANDARD_AXIS_PRINTMARK_INSETTER_INTERVAL			
FB enable -	boEnable	boEnabAck	-	Ackn. "FB enable"
Controller enable -	boRF	boSBM	-	System ready message

Clear error	-	boFL	boDone	-	Ackn. "FB done"
Emergency stop	-	boEmergency_Stop	boErr	-	Error
Start speed control	-	boSpeed	iErrID	-	Error ID
Start homing cycle	-	boHome	enErrName	-	Name of faulty FB
Start absolute pos.	-	boPosAbs	boPosBusy	-	Positioning active
Start relative pos.	-	boPosRel	boJogBusy	-	Jog mode active
Jog plus	-	boJogPlus	boHomeBusy	-	Homing drive active
Jog minus	-	boJogMinus	boSpeedBusy	-	Speed control active
Positioning velocity	-	udPosVelocity	**boEnabAckPmInsetterInterval**	-	Printmark block initialized and activated
Target position	-	diPosition	**boParameterSetupDone**	-	Parameterisation finished
Acceleration	-	lreAccel	**boPmDetected**	-	Print mark detected
Deceleration	-	lreDecel	**boPmLost**	-	Print mark lost
Jerk limit at start	-	lreJerkStart	**boPmWnd**	-	Print mark window active
Jerk limit at stop	-	lreJerkEnd	**diPmDeviation**	-	Deviation between setpoint and actual position
Jog velocity	-	udVelocityJog	**diOpOffsetOut**	-	Output operator offset
Velocity for speed control	-	diSpeedVelocity	**iActPmLost**	-	Number of lost print marks
Enable PM control	-	**boEnablePmInsetterInterval**			
Reset output boPmLost	-	**boResetPmLost**			
Start output operator offset	-	**boSetupOffset**			
Start with print mark setpoint position	-	**boSetupPmPos**			
Master counts negative	-	**boMasterNegDir**			
Motor direction negative	-	**boMotorNegDir**			
Enable correction output	-	**boCorrOutControlOn**			
Operator offset	-	**diOpOffset**			
Format length	-	**diFormatLength**			
Window length	-	**diWindowLength**			
Number of marks until message 'mark lost'	-	**iNoPmLost**			
Correction range	-	**diCorrectionRange**			
Print mark position setpoint	-	**diPmSetupPos**			
Max. correction per format	-	**diCorrectionLimit**			
Start correction range 1	-	**diCorrRng1Start**			
End correction range 1	-	**diCorrRng1End**			
Offset correction control 1	-	**diCorrOutControl1PreSet**			
Start correction range 2	-	**diCorrRng2Start**			
End correction range 2	-	**diCorrRng2End**			
Offset correction control 2	-	**diCorrOutControl2PreSet**			
Cranc function curve shape	-	**enCrankShape**			
Cranc function start offset	-	**diCrancDriveOffset**			
Cranc function standstill range	-	**diCrankStillStand**			
Start print mark detection	-	**boStartPmDet**			
Start cranc function	-	**boCrankDriveStart**			
Print mark detected	-	**boPmRefPulse**			



1) Only for CODESYS V3

Input variables

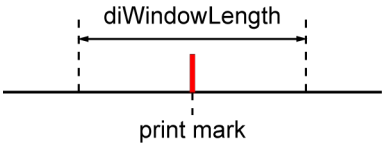
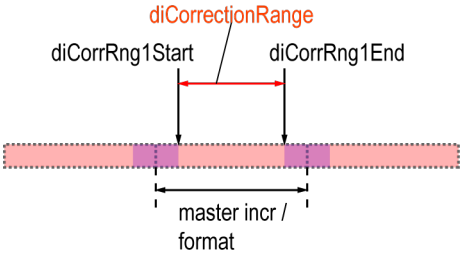
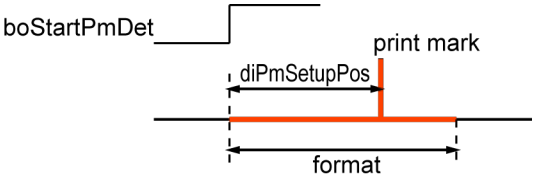
Name	Type	Description	Sync.	Async.
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.		x
boRF	BOOL	Bit for enabling the controller Relevant parameters: ID32796 'Source RF' Allowed: Code 5: 'Controller enable via EtherCAT' Code 25: RF 'via EtherCAT AND-linked with the binary input UE' Code 0: 'Controller enable (RF) via binary input' is not allowed		x
boFL	BOOL	Bit for error deletion		x
boEmergency_Stop	BOOL	EMERGENCY STOP: The setpoint of the velocity is decreased to zero along the emergency-stop ramp. Once initiated, an emergency stop cannot be aborted. Relevant variables: EMERGENCY STOP ramp: Ramp time in which the drive is slowed down from the current velocity to zero. EMERGENCY STOP ramp: ST_AXIS_DRIVE.ST_IDS.ST_ID32919_PlcList. diID32919_05_diEmergency_StopRamp Unit: ms Default value: 100 ms		x

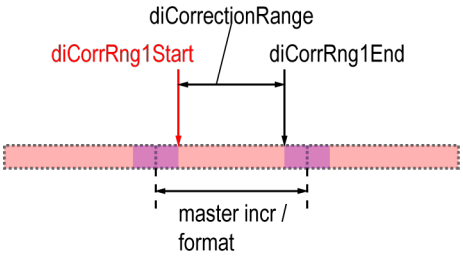
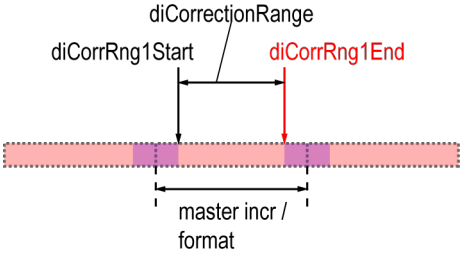
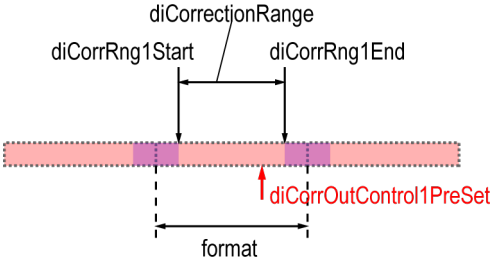
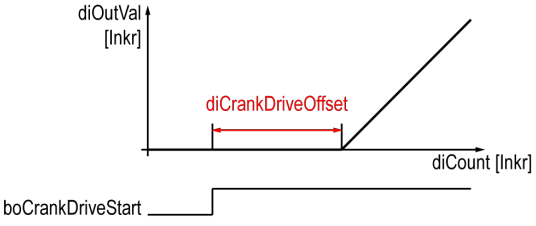
Name	Type	Description	Sync.	Async.
boSpeed	BOOL	Speed control Enable signal: With a positive edge, the speed control function starts. As long as 'boSpeed' = TRUE, speed control (speed setpoint) is carried out. Use a negative edge 'boSpeed' = FALSE to cancel the current speed control (with setpoint value = 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). Relevant parameters: ID32802 'AMK secondary operating mode 2' [Operating mode: Speed control] [Setpoint source: diMainSetpoint (code 41h)] [Ramps: active / inactive (bit 6)] ID32780 'Acceleration ramp' ID32781'Deceleration ramp' Relevant variables: Speed setpoint: diSpeedVelocity Velocity ramp: STANDARD_AXIS.fbVelocity.diVelocityRamp [default value: 100 ms] Speed control active: boSpeedBusy		x

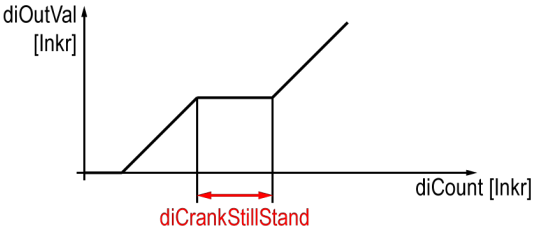
Name	Type	Description	Sync.	Async.
boHome	BOOL	Homing drive Enable signal: With a positive edge, the homing cycle function starts. As long as 'boHome' = TRUE, the homing drive is carried out. Use a negative edge 'boHome' = FALSE to cancel the current referencing or terminate the completed referencing. Relevant parameters: ID41 'Homing velocity' ID147 'Homing parameter' ID150 'Homing offset 1' ID32926 'AMK homing cycle parameter' Relevant variables: Homing drive active: boHomeBusy Homing done: boDone Homing point known: stAxisDrive.stStatus.boID33036_RFP_known		x
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 [Relevant parameters: see 'boPosRel']		x
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. Relevant parameters: ID32801 'AMK secondary operating mode 1' [Operating mode: position control] [Setpoint source: diMainSetpoint (code 41h)] Relevant variables: Positioning velocity: udPosVelocity Target position: diPosition Acceleration: lreAccel Deceleration: lreDecel Jerk limitation (start): lreJerkStart Jerk limitation (end): lreJerkEnd Positioning active: boPosBusy Positioning done: boDone		x

Name	Type	Description	Sync.	Async.
boJogPlus	BOOL	Jog in positive direction Enable signal: With a positive edge, the jogging function starts (positive direction). Jog mode is run as long as 'boJobPlus' = TRUE. Use a negative edge 'boJogPlus' = FALSE to cancel the current jog mode. [Relevant parameters: see 'boJogMinus']		x
boJogMinus	BOOL	Jogging in negative direction Enable signal: With a positive edge, the jogging function starts (negative direction). Jog mode is run as long as 'boJobMinus' = TRUE. Use a negative edge 'boJogMinus' = FALSE to cancel the current jog mode. Relevant parameters: ID32801 'AMK secondary operating mode 1' [Operating mode: position control] [Setpoint source: diMainSetpoint (code 41h)] Relevant variables: Jog velocity: udVelocityJog Acceleration: lreAccel Deceleration: lreDecel Jerk limitation (start): lreJerkStart Jerk limitation (end): lreJerkEnd Jog mode active: boJogBusy		x
udPosVelocity	UDINT	Positioning velocity Unit: incr/s Default value: 34133		x
diPosition	DINT	Target position Use the inputs 'boPosAbs' and 'boPosRel' to determine whether the target position is approached absolutely or relatively. Unit: Increments Default value: 30720		x
lreAccel	LREAL	Acceleration for positioning and jog mode Range: $1{,}43\ 10^{-13} < \lvert\text{'lreAccel'}\rvert < 1{,}43\ 10^{+16}$ Unit: incr/s^2 Default: 1000000		x
lreDecel	LREAL	Deceleration for positioning and jog mode Range: $1{,}43\ 10^{-13} < \lvert\text{'lreDecel'}\rvert < 1{,}43\ 10^{+16}$ Unit: incr/s^2 Default: 1000000		x
lreJerkStart	LREAL	Jerk limitation (start) for positioning and jog mode Unit: incr/s^3 Default: 10000000		x

Name	Type	Description	Sync.	Async.
lreJerkEnd	LREAL	Jerk limitation (end) for positioning and jog mode Unit: incr/s^3 Default: 10000000		x
udVelocityJog	UDINT	Jog velocity Unit: incr/s Default value: 34133		x
diSpeedVelocity	DINT	Speed setpoint for speed control Unit: rpm Default value: 100		x
boEnablePmInsetterInterval	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. The input increment detection is enabled. Increment changes at the input 'diInVal' lead to a change of the master reference point. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.		x
boResetPmLost	BOOL	Resetting the output 'boPmLost'		x
boSetupOffset	BOOL	Output start of the specified value of 'diOpOffset'. This value is used as an offset to shift the printing mark set position		x
boSetupPmPos	BOOL	Mode how to determine the nominal position of the print mark on a positive edge of 'boStartPmDet' 'boPmSetupPos' = FALSE: The position of the next print mark is set as nominal position. All following print marks are adjusted to this position. 'boPmSetupPos' = TRUE: The parameter 'diPmSetupPos' specifies the print mark relatively to the actual master position. The following print marks are adjusted to this position		x
boMasterNegDir	BOOL	Direction of master pulse negated 'boMasterNegDir' = FALSE: no negation 'boMasterNegDir' = TRUE: negation This input must only be set on a program start.		x
boMotorNegDir	BOOL	Direction of motor rotation negated 'boMotorNegDir' = FALSE: no negation 'boMotorNegDir' = TRUE: negation This input must only be set on a program start.		x

Name	Type	Description	Sync.	Async.
boCorrOutControlOn	BOOL	Correction output enabled 'boCorrOutControlOn' = FALSE: always correction output 'boCorrOutControlOn' = TRUE: correction output only in the ranges specified by 'diCorrRng1Start' and 'diCorrRng1End' resp. 'diCorrRng2Start' and 'diCorrRng2End'. The ranges are OR operated The modulo counting is started with a 0 -> 1 edge at 'boCorrOutControlOn' and can be preset by 'diCorrOutControl1PreSet' or 'diCorrOutControl2PreSet'		x
diOpOffset	DINT	Operator offset. Shifts the set position of the print mark. The shift takes place in the correction range and starts with a 0 → 1 edge at 'boSetupOffset'. It is independent of changes in the value of 'diCount'. The offset is calculated as		x
diFormatLength	DINT	Format length Nominal distance between two print marks [increments]		x
diWindowLength	DINT	Window length Length in front and after the nominal print mark position. In this range, the print mark sensor is activated and will accept a print mark. 		x
iNoPmLost	INT	Number of print marks to be missed in succession, before the output 'boPmLost' is set.		x
diCorrectionRange	DINT	Correction range The print mark control tries to output the correction in the specified range [% diFormatLength] 		x
diPmSetupPos	DINT	The parameter specifies the nominal position of the print mark relatively to the actual master position on a positive edge of 'boStartPmDet'. The detected print marks are controlled to this position [increments] 		x
diCorrectionLimit	DINT	Maximum correction output per format [increments] 'diCorrectionLimit' = 0: no limitation 'diCorrectionLimit' > 0: maximal value		x

Name	Type	Description	Sync.	Async.
diCorrRng1Start **diCorrRng2Start**	DINT	Correction range 1 / 2 Start value related to master increments / format [increments] 		x
diCorrRng1End **diCorrRng2End**	DINT	Correction range 1 / 2 End value related to master increments / format [increments] 		x
diCorrOutControl1PreSet **diCorrOutControl2PreSet**	DINT	Actual position value (zero offset) at start of the correction control, related to the master increments / format [increments] Example: If the controlled drive is positioned in the middle of the format when starting the correction control, >format/2< must be set on a 0 -> 1 edge of 'boCorrOutControlOn' 		x
enCrankShape	ENUM	EN_CRANK_DRIVE_SHAPE Shape of crank drive curve enCrankShapeLinear: linear 45° straight line enCrankShapeSine: SINUS - function enCrankShapeSquareSine: SINUS^2 - function		x
diCrancDriveOffset	DINT	Crank drive start offset. After the crank drive is started (boCrankDriveStart = TRUE), the function waits until the master has passed the offset 		x

Name	Type	Description	Sync.	Async.
diCrankStillStand	DINT	Crank drive standstill range 		x
boStartPmDet	BOOL	Start of the print mark control and specification of the print mark setpoint position by parameter 'diPmSetupPos' if 'boSetupPmPos' = TRUE	x	
boCrankDriveStart	BOOL	Start of the crank drive function, Output at 'diOutVal' starts according to the selected crank shape. The output is only done if 'diCount' changes	x	
boPmRefPulse	BOOL	Print mark pulse detected (reference pulse) (See document Software description AmkBase Bibliothek, Part no. 204986) BasicSupport TIME_TO_COUNT	x	
diCount	DINT	Counter value Masterimpulse [Inkremente] (See documentSoftware descriptionAmkBase Bibliothek, Part no. 204986) BasicSupport TIME_TO_COUNT	x	
diPmOffset	DINT	Print mark offset of the actual master position (See documentSoftware descriptionAmkBase Bibliothek, Part no. 204986) BasicSupport TIME_TO_COUNT	x	

<table>
<tr><th>Name</th><th>Type</th><th>Description</th><th>Sync.</th><th>Async.</th></tr>
<tr><td>enArithmetik[1)]</td><td>ENUM</td><td>EN_POS_J_ARITHMETIK
(This input is only available for CODESYS V3 function blocks. CODESYS V2, the selection is always adjusted automatically)

With 'EN_POS_J_ARITHMETIK' the selection 32 bit or 64 bit arithmetic for position function blocks in position operation mode is set.

Options:
<table><tr><td>enAutomaticSelection</td><td>Automatic determination if 32 bit or 64 bit arithmetic (Default)
 
The default setting 'enAutomaticSelection' can only be used, if the PLC controller is on the first place in the CODESYS 'Device_ Config'. Is that not the case the selection has to be made manually by the user.</td></tr><tr><td>en64BitArithmetik</td><td>64 bit arithmetic</td></tr><tr><td>en32BitArithmetik</td><td>32 bit arithmetic</td></tr></table>
If the automatic determination does not work, the following diagnostic number is displayed and the selection must done manually by the user.
<table><tr><td>iErrID</td><td>5</td></tr></table>
Additional information about 'EN_POS_J_ARITHMETIK' and manual selection:
Siehe 'EN_POS_J_ARITHMETIK' auf Seite 601.</td><td></td><td>x</td></tr>
</table>

1) Only for CODESYS V3

Output variables

Name	Type	Description	Sync.	Async.
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled		x
boSBM	BOOL	System ready message		x
boDone	BOOL	Positioning done. Valid for 'boPosAbs', 'boPosRel' and 'boHome'		x
boErr	BOOL	The function block is in an error state FALSE: No error (permitted commanding or warning) TRUE: Error		x
iErrID	INT	Error identity number: Diagnostic number is output iErrID = 0: No error iErrID ≠ 0, boErr = TRUE: Error iErrID ≠ 0, boErr = FALSE: Warning Bus error when 'boErr' = TRUE: EtherCAT or CAN are not in operational mode.		x
enErrName	ENUM	EN_FB_NAME Name of faulty block for which the diagnostic number is output. The diagnostic number is explained in the description of the corresponding library.		x
boPosBusy	BOOL	Positioning active		x
boJogBusy	BOOL	Jog mode active		x
boHomeBusy	BOOL	Home active		x
boSpeedBusy	BOOL	Speed control active		x
boEnabAckPmInsetterInterval	BOOL	Print mark module initialized and active		x
boParameterSetupDone	BOOL	Parameterisation finished		x
boPmDetected	BOOL	Print mark detected, control activated		x
boPmLost	BOOL	Print mark lost activated when the number of print marks set in 'iNoPmLost' is not detected successively.		x
boPmWnd	BOOL	Mark window activates the print mark sensor		x
diPmDeviation	DINT	Deviation between setpoint and actual position of the actual print mark		x
diOpOffsetOut	DINT	Output value contains the actually output operator offset		x
iActPmLost	INT	Number of lost print marks		x

Input and output variables

Name	Type	Description	Sync.	Async.
stDevice	STRUCT	ST_DEVICE The device description structure assigns the block a device. (See document Software description AmkBase Bibliothek, Part no.204986)		x
stAxisDrive	STRUCT	ST_AXIS_DRIVE Siehe 'ST_AXIS_DRIVE (ST)' auf Seite 191.		x

Usage note in the CODESYS program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
STANDARD_AXIS_PRINTMARK_INSETTER_INTERVAL	STANDARD_AXIS_PRINTMARK_INSETTER_INTERVAL.actSync

History

Library	AmkAfl	
System versions	**Functionality**	**Target**
AS-C V3.01/0827	Basic functionality (1st version)	≥ AS_CP_200_0812_T202053
AS-PL15 V3.01/0804	Basic functionality (1st version)	≥ AS_V313_0946_T202724

4.1.4.2.7 STANDARD_AXIS_PRINTMARK_REGISTER_CONTROL_CONT (FB)

Expanded functionality:

- Print mark control: The position of the treating tool (e.g. cutter, punch) is controlled in relation to a master format (register control)

Siehe 'PRINT_MARK_REGISTER_CONTROL_CONTINUOUS (FB)' auf Seite 506.

The following basic functionalities for the axis are available:

Inputs/outputs on the function block

- Control bits (RF Controller enable, FL Clear error)
- Emergency stop
- Speed control
- Homing cycle
- Positioning (absolute/relative)
- Jog mode (minus/plus)
- Status signals of axis functions

Access via program code

- Configuration of status messages (ID26 'Configuration status bits')
- Reading of parameters according to a specifications list
- Actual position cached
- Control with feedforward (torque and speed)

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

Setpoints and current values are transferred in a synchronous action. The synchronous action must be called in the synchronous program level FPLC_PRG.

User interface

		STD_AXIS_PRINTMARK_REGISTER_CONTROL_CONT			
FB enable	-	boEnable	boEnabAck	-	Ackn. "FB enable"
Controller enable	-	boRF	boSBM	-	System ready message

Clear error	-	boFL	boDone	-	Ackn. "FB done"
Emergency stop	-	boEmergency_Stop	boErr	-	Error
Start speed control	-	boSpeed	iErrID	-	Error ID
Start homing cycle	-	boHome	enErrName	-	Name of faulty FB
Start absolute pos.	-	boPosAbs	boPosBusy	-	Positioning active
Start relative pos.	-	boPosRel	boJogBusy	-	Jog mode active
Jog plus	-	boJogPlus	boHomeBusy	-	Homing drive active
Jog minus	-	boJogMinus	boSpeedBusy	-	Speed control active
Positioning velocity	-	udPosVelocity	**boEnabAckPmRegCtrlCont**	-	Printmark block initialized and activated
Target position	-	diPosition	**boParameterSetupDone**	-	Parameterisation finished
Acceleration	-	lreAccel	**boPmDetected**	-	Print mark detected
Deceleration	-	lreDecel	**boPmLost**	-	Print mark lost
Jerk limit at start	-	lreJerkStart	**boPmWnd**	-	Print mark window active
Jerk limit at stop	-	lreJerkEnd	**diPmDeviation**	-	Deviation between setpoint and actual position
Jog velocity	-	udVelocityJog	**diOpOffsetOut**	-	Output operator offset
Velocity for speed control	-	diSpeedVelocity	**iActPmLost**	-	Number of lost print marks
Enable PM control	-	**boEnablePmRegisterCtrlCont**			
Reset output boPmLost	-	**boResetPmLost**			
Start output operator offset	-	**boSetupOffset**			
Start with print mark setpoint position	-	**boSetupPmPos**			
Master counts negative	-	**boMasterNegDir**			
Motor direction negative	-	**boMotorNegDir**			
Enable correction output	-	**boCorrOutControlOn**			
Operator offset	-	**diOpOffset**			
Format length	-	**diFormatLength**			
Window length	-	**diWindowLength**			
Number of marks until message 'mark lost'	-	**iNoPmLost**			
Correction range	-	**diCorrectionRange**			
Print mark position setpoint	-	**diPmSetupPos**			
Max. correction per format	-	**diCorrectionLimit**			
Start correction range 1	-	**diCorrRng1Start**			
End correction range 1	-	**diCorrRng1End**			
Offset correction control 1	-	**diCorrOutControl1PreSet**			
Start correction range 2	-	**diCorrRng2Start**			
End correction range 2	-	**diCorrRng2End**			
Offset correction control 2	-	**diCorrOutControl2PreSet**			
Start print mark detection	-	**boStartPmDet**			
Print mark detected	-	**boPmRefPulse**			
Master pulses	-	**diCount**			
Print mark offset	-	**diPmOffset**			
Register controller pulses	-	**diRcCount**			
32bit or 64bit arithmetic	-	enArithmetik[1)]			
Axis structure	-	stAxisDrive	stAxisDrive	-	Axis structure
Device structure	-	stDevice	stDevice	-	Device structure

actSync
Synchronous operation - opened in synchronous program section (FPLC_PRG)

1) Only for CODESYS V3

Input variables

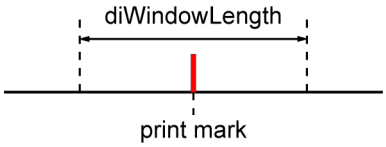
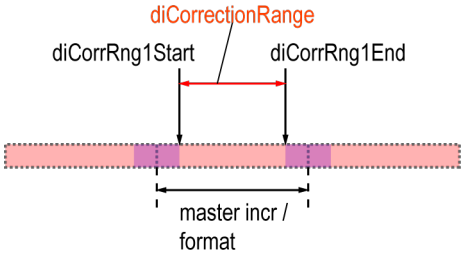
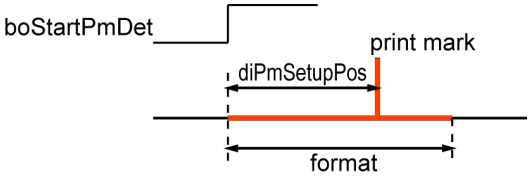
Name	Type	Description	Sync.	Async.
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.		x
boRF	BOOL	Bit for enabling the controller Relevant parameters: ID32796 'Source RF' Allowed: Code 5: 'Controller enable via EtherCAT' Code 25: RF 'via EtherCAT AND-linked with the binary input UE' Code 0: 'Controller enable (RF) via binary input' is not allowed		x
boFL	BOOL	Bit for error deletion		x
boEmergency_Stop	BOOL	EMERGENCY STOP: The setpoint of the velocity is decreased to zero along the emergency-stop ramp. Once initiated, an emergency stop cannot be aborted. Relevant variables: EMERGENCY STOP ramp: Ramp time in which the drive is slowed down from the current velocity to zero. EMERGENCY STOP ramp: ST_AXIS_DRIVE.ST_IDS.ST_ID32919_PlcList. diID32919_05_diEmergency_StopRamp Unit: ms Default value: 100 ms		x

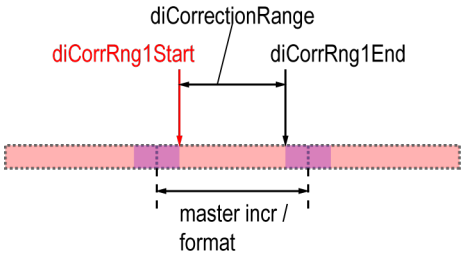
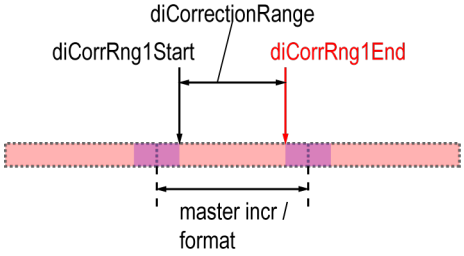
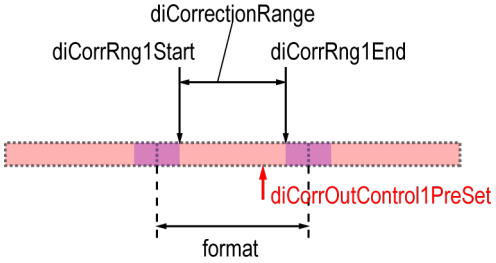
Name	Type	Description	Sync.	Async.
boSpeed	BOOL	Speed control Enable signal: With a positive edge, the speed control function starts. As long as 'boSpeed' = TRUE, speed control (speed setpoint) is carried out. Use a negative edge 'boSpeed' = FALSE to cancel the current speed control (with setpoint value = 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). Relevant parameters: ID32802 'AMK secondary operating mode 2' [Operating mode: Speed control] [Setpoint source: diMainSetpoint (code 41h)] [Ramps: active / inactive (bit 6)] ID32780 'Acceleration ramp' ID32781'Deceleration ramp' Relevant variables: Speed setpoint: diSpeedVelocity Velocity ramp: STANDARD_AXIS.fbVelocity.diVelocityRamp [default value: 100 ms] Speed control active: boSpeedBusy		x

Name	Type	Description	Sync.	Async.
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. Relevant parameters: ID41 'Homing velocity' ID147 'Homing parameter' ID150 'Homing offset 1' ID32926 'AMK homing cycle parameter' Relevant variables: Homing drive active: boHomeBusy Homing done: boDone Homing point known: stAxisDrive.stStatus.boID33036_RFP_known		x
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 [Relevant parameters: see 'boPosRel']		x
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. Relevant parameters: ID32801 'AMK secondary operating mode 1' [Operating mode: position control] [Setpoint source: diMainSetpoint (code 41h)] Relevant variables: Positioning velocity: udPosVelocity Target position: diPosition Acceleration: lreAccel Deceleration: lreDecel Jerk limitation (start): lreJerkStart Jerk limitation (end): lreJerkEnd Positioning active: boPosBusy Positioning done: boDone		x

Name	Type	Description	Sync.	Async.
boJogPlus	BOOL	Jog in positive direction Enable signal: With a positive edge, the jogging function starts (positive direction). Jog mode is run as long as 'boJobPlus' = TRUE. Use a negative edge 'boJogPlus' = FALSE to cancel the current jog mode. [Relevant parameters: see 'boJogMinus']		x
boJogMinus	BOOL	Jogging in negative direction Enable signal: With a positive edge, the jogging function starts (negative direction). Jog mode is run as long as 'boJobMinus' = TRUE. Use a negative edge 'boJogMinus' = FALSE to cancel the current jog mode. Relevant parameters: ID32801 'AMK secondary operating mode 1' [Operating mode: position control] [Setpoint source: diMainSetpoint (code 41h)] Relevant variables: Jog velocity: udVelocityJog Acceleration: lreAccel Deceleration: lreDecel Jerk limitation (start): lreJerkStart Jerk limitation (end): lreJerkEnd Jog mode active: boJogBusy		x
udPosVelocity	UDINT	Positioning velocity Unit: incr/s Default value: 34133		x
diPosition	DINT	Target position Use the inputs 'boPosAbs' and 'boPosRel' to determine whether the target position is approached absolutely or relatively. Unit: Increments Default value: 30720		x
lreAccel	LREAL	Acceleration for positioning and jog mode Range: $1{,}43\ 10^{-13} < \lvert\text{'lreAccel'}\rvert < 1{,}43\ 10^{+16}$ Unit: incr/s^2 Default: 1000000		x
lreDecel	LREAL	Deceleration for positioning and jog mode Range: $1{,}43\ 10^{-13} < \lvert\text{'lreDecel'}\rvert < 1{,}43\ 10^{+16}$ Unit: incr/s^2 Default: 1000000		x
lreJerkStart	LREAL	Jerk limitation (start) for positioning and jog mode Unit: incr/s^3 Default: 10000000		x

Name	Type	Description	Sync.	Async.
lreJerkEnd	LREAL	Jerk limitation (end) for positioning and jog mode Unit: incr/s^3 Default: 10000000		x
udVelocityJog	UDINT	Jog velocity Unit: incr/s Default value: 34133		x
diSpeedVelocity	DINT	Speed setpoint for speed control Unit: rpm Default value: 100		x
boEnablePmRegisterCtrlCont	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. The input increment detection is enabled. Increment changes at the input 'diInVal' lead to a change of the master reference point. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.		x
boResetPmLost	BOOL	Resetting the output 'boPmLost'		x
boSetupOffset	BOOL	Output start of the specified value of 'diOpOffset'. This value is used as an offset to shift the printing mark set position		x
boSetupPmPos	BOOL	Mode how to determine the nominal position of the print mark on a positive edge of 'boStartPmDet' 'boPmSetupPos' = FALSE: The position of the next print mark is set as nominal position. All following print marks are adjusted to this position. 'boPmSetupPos' = TRUE: The parameter 'diPmSetupPos' specifies the print mark relatively to the actual master position. The following print marks are adjusted to this position		x
boMasterNegDir	BOOL	Direction of master pulse negated 'boMasterNegDir' = FALSE: no negation 'boMasterNegDir' = TRUE: negation This input must only be set on a program start.		x
boMotorNegDir	BOOL	Direction of motor rotation negated 'boMotorNegDir' = FALSE: no negation 'boMotorNegDir' = TRUE: negation This input must only be set on a program start.		x

Name	Type	Description	Sync.	Async.
boCorrOutControlOn	BOOL	Correction output enabled 'boCorrOutControlOn' = FALSE: always correction output 'boCorrOutControlOn' = TRUE: correction output only in the ranges specified by 'diCorrRng1Start' and 'diCorrRng1End' resp. 'diCorrRng2Start' and 'diCorrRng2End'. The ranges are OR operated The modulo counting is started with a 0 -> 1 edge at 'boCorrOutControlOn' and can be preset by 'diCorrOutControl1PreSet' or 'diCorrOutControl2PreSet'		x
diOpOffset	DINT	Operator offset. Shifts the set position of the print mark. The shift takes place in the correction range and starts with a 0 → 1 edge at 'boSetupOffset'. It is independent of changes in the value of 'diCount'. The offset is calculated as		x
diFormatLength	DINT	Format length Nominal distance between two print marks [increments]		x
diWindowLength	DINT	Window length Length in front and after the nominal print mark position. In this range, the print mark sensor is activated and will accept a print mark. 		x
iNoPmLost	INT	Number of print marks to be missed in succession, before the output 'boPmLost' is set.		x
diCorrectionRange	DINT	Correction range The print mark control tries to output the correction in the specified range [% diFormatLength] 		x
diPmSetupPos	DINT	The parameter specifies the nominal position of the print mark relatively to the actual master position on a positive edge of 'boStartPmDet'. The detected print marks are controlled to this position [increments] 		x
diCorrectionLimit	DINT	Maximum correction output per format [increments] 'diCorrectionLimit' = 0: no limitation 'diCorrectionLimit' > 0: maximal value		x

Name	Type	Description	Sync.	Async.
diCorrRng1Start **diCorrRng2Start**	DINT	Correction range 1 / 2 Start value related to master increments / format [increments] 		x
diCorrRng1End **diCorrRng2End**	DINT	Correction range 1 / 2 End value related to master increments / format [increments] 		x
diCorrOutControl1PreSet **diCorrOutControl2PreSet**	DINT	Actual position value (zero offset) at start of the correction control, related to the master increments / format [increments] Example: If the controlled drive is positioned in the middle of the format when starting the correction control, >format/2< must be set on a 0 -> 1 edge of 'boCorrOutControlOn' 		x
boStartPmDet	BOOL	Start of the print mark control and specification of the print mark setpoint position by parameter 'diPmSetupPos' if 'boSetupPmPos' = TRUE	x	
boPmRefPulse	BOOL	Print mark pulse detected (reference pulse) (See document Software description AmkBase Bibliothek, Part no. 204986) BasicSupport TIME_TO_COUNT	x	
diCount	DINT	Counter value Masterimpulse [Inkremente] (See documentSoftware descriptionAmkBase Bibliothek, Part no. 204986) BasicSupport TIME_TO_COUNT	x	
diPmOffset	DINT	Print mark offset of the actual master position (See documentSoftware descriptionAmkBase Bibliothek, Part no. 204986) BasicSupport TIME_TO_COUNT	x	
diRcCount	DINT	Register controller pulses Pulses of the tool [Incr]	x	

<table>
<tr><th>Name</th><th>Type</th><th>Description</th><th>Sync.</th><th>Async.</th></tr>
<tr><td>enArithmetik[1)]</td><td>ENUM</td><td>EN_POS_J_ARITHMETIK
(This input is only available for CODESYS V3 function blocks. CODESYS V2, the selection is always adjusted automatically)

With 'EN_POS_J_ARITHMETIK' the selection 32 bit or 64 bit arithmetic for position function blocks in position operation mode is set.

Options:<table><tr><td>enAutomaticSelection</td><td>Automatic determination if 32 bit or 64 bit arithmetic (Default)
 
The default setting 'enAutomaticSelection' can only be used, if the PLC controller is on the first place in the CODESYS 'Device_ Config'. Is that not the case the selection has to be made manually by the user.</td></tr><tr><td>en64BitArithmetik</td><td>64 bit arithmetic</td></tr><tr><td>en32BitArithmetik</td><td>32 bit arithmetic</td></tr></table>If the automatic determination does not work, the following diagnostic number is displayed and the selection must done manually by the user.<table><tr><td>iErrID</td><td>5</td></tr></table>Additional information about 'EN_POS_J_ARITHMETIK' and manual selection:
Siehe 'EN_POS_J_ARITHMETIK' auf Seite 601.</td><td></td><td>x</td></tr>
</table>

1) Only for CODESYS V3

Output variables

Name	Type	Description	Sync.	Async.
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled		x
boSBM	BOOL	System ready message		x
boDone	BOOL	Positioning done. Valid for 'boPosAbs', 'boPosRel' and 'boHome'		x
boErr	BOOL	The function block is in an error state FALSE: No error (permitted commanding or warning) TRUE: Error		x
iErrID	INT	Error identity number: Diagnostic number is output iErrID = 0: No error iErrID ≠ 0, boErr = TRUE: Error iErrID ≠ 0, boErr = FALSE: Warning Bus error when 'boErr' = TRUE: EtherCAT or CAN are not in operational mode.		x
enErrName	ENUM	EN_FB_NAME Name of faulty block for which the diagnostic number is output. The diagnostic number is explained in the description of the corresponding library.		x
boPosBusy	BOOL	Positioning active		x
boJogBusy	BOOL	Jog mode active		x
boHomeBusy	BOOL	Home active		x
boSpeedBusy	BOOL	Speed control active		x
boEnabAckPmRegCtrlCont	BOOL	Print mark module initialized and active		x
boParameterSetupDone	BOOL	Parameterisation finished		x
boPmDetected	BOOL	Print mark detected, control activated		x
boPmLost	BOOL	Print mark lost activated when the number of print marks set in 'iNoPmLost' is not detected successively.		x
boPmWnd	BOOL	Mark window activates the print mark sensor		x
diPmDeviation	DINT	Deviation between setpoint and actual position of the actual print mark		x
diOpOffsetOut	DINT	Output value contains the actually output operator offset		x
iActPmLost	INT	Number of lost print marks		x

Input and output variables

Name	Type	Description	Sync.	Async.
stDevice	STRUCT	ST_DEVICE The device description structure assigns the block a device. (See document Software description AmkBase Bibliothek, Part no.204986)		x
stAxisDrive	STRUCT	ST_AXIS_DRIVE Siehe 'ST_AXIS_DRIVE (ST)' auf Seite 191.		x

Usage note in the CODESYS program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
STANDARD_AXIS_PRINTMARK_REGISTER_CONTROL_CONT	STANDARD_AXIS_PRINTMARK_REGISTER_CONTROL_CONT.actSync

History

Library	AmkAfl	
System versions	**Functionality**	**Target**
AS-C V3.01/0827	Basic functionality (1st version)	≥ AS_CP_200_0812_T202053
AS-PL15 V3.01/0804	Basic functionality (1st version)	≥ AS_V313_0946_T202724

4.1.4.2.8 STANDARD_AXIS_PRINTMARK_REGISTER_CONTROL_DISCONT (FB)

Expanded functionality:

- Print mark control: The position of the treating tool (e.g. cutter, punch) is controlled in relation to a master format (register control)

Siehe 'PRINT_MARK_REGISTER_CONTROL_DISCONT (FB)' auf Seite 512.

The following basic functionalities for the axis are available:

Inputs/outputs on the function block

- Control bits (RF Controller enable, FL Clear error)
- Emergency stop
- Speed control
- Homing cycle
- Positioning (absolute/relative)
- Jog mode (minus/plus)
- Status signals of axis functions

Access via program code

- Configuration of status messages (ID26 'Configuration status bits')
- Reading of parameters according to a specifications list
- Actual position cached
- Control with feedforward (torque and speed)

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

Setpoints and current values are transferred in a synchronous action. The synchronous action must be called in the synchronous program level FPLC_PRG.

User interface

Controller enable	-	boRF	boSBM	-	System ready message
Clear error	-	boFL	boDone	-	Ackn. "FB done"
Emergency stop	-	boEmergency_Stop	boErr	-	Error
Start speed control	-	boSpeed	iErrID	-	Error ID
Start homing cycle	-	boHome	enErrName	-	Name of faulty FB
Start absolute pos.	-	boPosAbs	boPosBusy	-	Positioning active
Start relative pos.	-	boPosRel	boJogBusy	-	Jog mode active
Jog plus	-	boJogPlus	boHomeBusy	-	Homing drive active
Jog minus	-	boJogMinus	boSpeedBusy	-	Speed control active
Positioning velocity	-	udPosVelocity	**boEnabAckPmRegisterCtrlDiscont**	-	Printmark block initialized and activated
Target position	-	diPosition	**boParameterSetupDone**	-	Parameterisation finished
Acceleration	-	lreAccel	**boPmDetected**	-	Print mark detected
Deceleration	-	lreDecel	**boPmLost**	-	Print mark lost
Jerk limit at start	-	lreJerkStart	**boPmWnd**	-	Print mark window active
Jerk limit at stop	-	lreJerkEnd	**diPmDeviation**	-	Deviation between setpoint and actual position
Jog velocity	-	udVelocityJog	**diOpOffsetOut**	-	Output operator offset
Velocity for speed control	-	diSpeedVelocity	**iActPmLost**	-	Number of lost print marks
Enable PM control	-	**boEnablePmRegisterCtrlDiscont**			
Reset output boPmLost	-	**boResetPmLost**			
Start output operator offset	-	**boSetupOffset**			
Start with print mark setpoint position	-	**boSetupPmPos**			
Master counts negative	-	**boMasterNegDir**			
Motor direction negative	-	**boMotorNegDir**			
Enable correction output	-	**boCorrOutControlOn**			
Operator offset	-	**diOpOffset**			
Format length	-	**diFormatLength**			
Window length	-	**diWindowLength**			
Number of marks until message 'mark lost'	-	**iNoPmLost**			
Correction range	-	**diCorrectionRange**			
Print mark position setpoint	-	**diPmSetupPos**			
Max. correction per format	-	**diCorrectionLimit**			
Start correction range 1	-	**diCorrRng1Start**			
End correction range 1	-	**diCorrRng1End**			

1) Only for CODESYS V3

Input variables

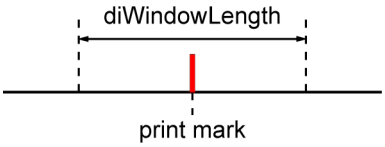
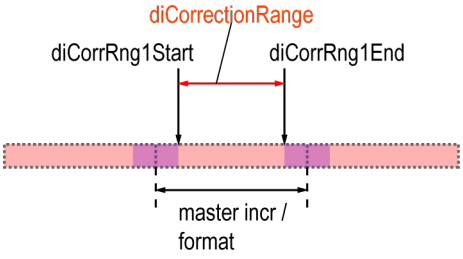
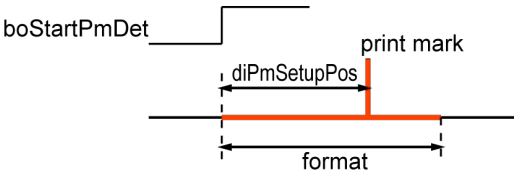
Name	Type	Description	Sync.	Async.
boEnable	BOOL	Enable signal: With a positive edge, the initialisation of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.		x
boRF	BOOL	Bit for enabling the controller Relevant parameters: ID32796 'Source RF' Allowed: Code 5: 'Controller enable via EtherCAT' Code 25: RF 'via EtherCAT AND-linked with the binary input UE' Code 0: 'Controller enable (RF) via binary input' is not allowed		x
boFL	BOOL	Bit for error deletion		x

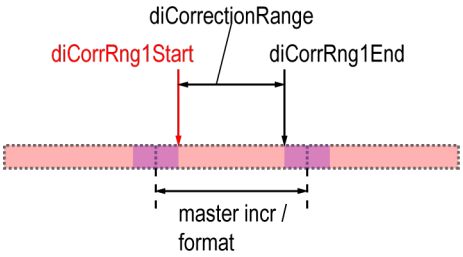
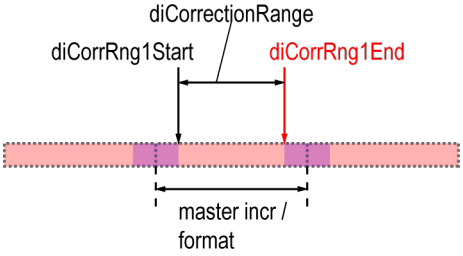
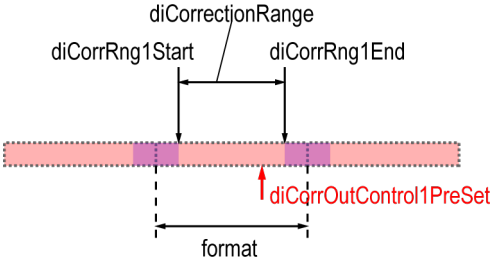
Name	Type	Description	Sync.	Async.
boEmergency_Stop	BOOL	EMERGENCY STOP: The setpoint of the velocity is decreased to zero along the emergency-stop ramp. Once initiated, an emergency stop cannot be aborted. Relevant variables: EMERGENCY STOP ramp: Ramp time in which the drive is slowed down from the current velocity to zero. EMERGENCY STOP ramp: ST_AXIS_DRIVE.ST_IDS.ST_ID32919_PlcList. diID32919_05_diEmergency_StopRamp Unit: ms Default value: 100 ms		x
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). Relevant parameters: ID32802 'AMK secondary operating mode 2' [Operating mode: Speed control] [Setpoint source: diMainSetpoint (code 41h)] [Ramps: active / inactive (bit 6)] ID32780 'Acceleration ramp' ID32781'Deceleration ramp' Relevant variables: Speed setpoint: diSpeedVelocity Velocity ramp: STANDARD_AXIS.fbVelocity.diVelocityRamp [default value: 100 ms] Speed control active: boSpeedBusy		x

Name	Type	Description	Sync.	Async.
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. Relevant parameters: ID41 'Homing velocity' ID147 'Homing parameter' ID150 'Homing offset 1' ID32926 'AMK homing cycle parameter' Relevant variables: Homing drive active: boHomeBusy Homing done: boDone Homing point known: stAxisDrive.stStatus.boID33036_RFP_known		x
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 [Relevant parameters: see 'boPosRel']		x
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. Relevant parameters: ID32801 'AMK secondary operating mode 1' [Operating mode: position control] [Setpoint source: diMainSetpoint (code 41h)] Relevant variables: Positioning velocity: udPosVelocity Target position: diPosition Acceleration: lreAccel Deceleration: lreDecel Jerk limitation (start): lreJerkStart Jerk limitation (end): lreJerkEnd Positioning active: boPosBusy Positioning done: boDone		x

Name	Type	Description	Sync.	Async.
boJogPlus	BOOL	Jog in positive direction Enable signal: With a positive edge, the jogging function starts (positive direction). Jog mode is run as long as 'boJobPlus' = TRUE. Use a negative edge 'boJogPlus' = FALSE to cancel the current jog mode. [Relevant parameters: see 'boJogMinus']		x
boJogMinus	BOOL	Jogging in negative direction Enable signal: With a positive edge, the jogging function starts (negative direction). Jog mode is run as long as 'boJobMinus' = TRUE. Use a negative edge 'boJogMinus' = FALSE to cancel the current jog mode. Relevant parameters: ID32801 'AMK secondary operating mode 1' [Operating mode: position control] [Setpoint source: diMainSetpoint (code 41h)] Relevant variables: Jog velocity: udVelocityJog Acceleration: lreAccel Deceleration: lreDecel Jerk limitation (start): lreJerkStart Jerk limitation (end): lreJerkEnd Jog mode active: boJogBusy		x
udPosVelocity	UDINT	Positioning velocity Unit: incr/s Default value: 34133		x
diPosition	DINT	Target position Use the inputs 'boPosAbs' and 'boPosRel' to determine whether the target position is approached absolutely or relatively. Unit: Increments Default value: 30720		x
lreAccel	LREAL	Acceleration for positioning and jog mode Range: $1{,}43\ 10^{-13} < \lvert\text{'lreAccel'}\rvert < 1{,}43\ 10^{+16}$ Unit: incr/s^2 Default: 1000000		x
lreDecel	LREAL	Deceleration for positioning and jog mode Range: $1{,}43\ 10^{-13} < \lvert\text{'lreDecel'}\rvert < 1{,}43\ 10^{+16}$ Unit: incr/s^2 Default: 1000000		x
lreJerkStart	LREAL	Jerk limitation (start) for positioning and jog mode Unit: incr/s^3 Default: 10000000		x

Name	Type	Description	Sync.	Async.
lreJerkEnd	LREAL	Jerk limitation (end) for positioning and jog mode Unit: incr/s^3 Default: 10000000		x
udVelocityJog	UDINT	Jog velocity Unit: incr/s Default value: 34133		x
diSpeedVelocity	DINT	Speed setpoint for speed control Unit: rpm Default value: 100		x
boEnablePmRegisterCtrlDiscont	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. The input increment detection is enabled. Increment changes at the input 'diInVal' lead to a change of the master reference point. In the state 'boEnable' = FALSE the block is no longer enabled and is thus no longer processed.		x
boResetPmLost	BOOL	Resetting the output 'boPmLost'		x
boSetupOffset	BOOL	Output start of the specified value of 'diOpOffset'. This value is used as an offset to shift the printing mark set position		x
boSetupPmPos	BOOL	Mode how to determine the nominal position of the print mark on a positive edge of 'boStartPmDet' 'boPmSetupPos' = FALSE: The position of the next print mark is set as nominal position. All following print marks are adjusted to this position. 'boPmSetupPos' = TRUE: The parameter 'diPmSetupPos' specifies the print mark relatively to the actual master position. The following print marks are adjusted to this position		x
boMasterNegDir	BOOL	Direction of master pulse negated 'boMasterNegDir' = FALSE: no negation 'boMasterNegDir' = TRUE: negation This input must only be set on a program start.		x
boMotorNegDir	BOOL	Direction of motor rotation negated 'boMotorNegDir' = FALSE: no negation 'boMotorNegDir' = TRUE: negation This input must only be set on a program start.		x

Name	Type	Description	Sync.	Async.
boCorrOutControlOn	BOOL	Correction output enabled 'boCorrOutControlOn' = FALSE: always correction output 'boCorrOutControlOn' = TRUE: correction output only in the ranges specified by 'diCorrRng1Start' and 'diCorrRng1End' resp. 'diCorrRng2Start' and 'diCorrRng2End'. The ranges are OR operated The modulo counting is started with a 0 -> 1 edge at 'boCorrOutControlOn' and can be preset by 'diCorrOutControl1PreSet' or 'diCorrOutControl2PreSet'		x
diOpOffset	DINT	Operator offset. Shifts the set position of the print mark. The shift takes place in the correction range and starts with a 0 → 1 edge at 'boSetupOffset'. It is independent of changes in the value of 'diCount'. The offset is calculated as		x
diFormatLength	DINT	Format length Nominal distance between two print marks [increments]		x
diWindowLength	DINT	Window length Length in front and after the nominal print mark position. In this range, the print mark sensor is activated and will accept a print mark. 		x
iNoPmLost	INT	Number of print marks to be missed in succession, before the output 'boPmLost' is set.		x
diCorrectionRange	DINT	Correction range The print mark control tries to output the correction in the specified range [% diFormatLength] 		x
diPmSetupPos	DINT	The parameter specifies the nominal position of the print mark relatively to the actual master position on a positive edge of 'boStartPmDet'. The detected print marks are controlled to this position [increments] 		x
diCorrectionLimit	DINT	Maximum correction output per format [increments] 'diCorrectionLimit' = 0: no limitation 'diCorrectionLimit' > 0: maximal value		x

Name	Type	Description	Sync.	Async.
diCorrRng1Start **diCorrRng2Start**	DINT	Correction range 1 / 2 Start value related to master increments / format [increments] 		x
diCorrRng1End **diCorrRng2End**	DINT	Correction range 1 / 2 End value related to master increments / format [increments] 		x
diCorrOutControl1PreSet **diCorrOutControl2PreSet**	DINT	Actual position value (zero offset) at start of the correction control, related to the master increments / format [increments] Example: If the controlled drive is positioned in the middle of the format when starting the correction control, >format/2< must be set on a 0 -> 1 edge of 'boCorrOutControlOn' 		x
boStartPmDet	BOOL	Start of the print mark control and specification of the print mark setpoint position by parameter 'diPmSetupPos' if 'boSetupPmPos' = TRUE	x	
boPmRefPulse	BOOL	Print mark pulse detected (reference pulse) (See document Software description AmkBase Bibliothek, Part no. 204986) BasicSupport TIME_TO_COUNT	x	
diCount	DINT	Counter value Masterimpulse [Inkremente] (See documentSoftware descriptionAmkBase Bibliothek, Part no. 204986) BasicSupport TIME_TO_COUNT	x	
diPmOffset	DINT	Print mark offset of the actual master position (See documentSoftware descriptionAmkBase Bibliothek, Part no. 204986) BasicSupport TIME_TO_COUNT	x	
diRcCount	DINT	Register controller pulses Pulses of the tool [Incr]	x	

<table>
<tr><th>Name</th><th>Type</th><th>Description</th><th>Sync.</th><th>Async.</th></tr>
<tr><td>enArithmetik[1)]</td><td>ENUM</td><td>EN_POS_J_ARITHMETIK
(This input is only available for CODESYS V3 function blocks. CODESYS V2, the selection is always adjusted automatically)

With 'EN_POS_J_ARITHMETIK' the selection 32 bit or 64 bit arithmetic for position function blocks in position operation mode is set.

Options:
<table><tr><td>enAutomaticSelection</td><td>Automatic determination if 32 bit or 64 bit arithmetic (Default)
 
The default setting 'enAutomaticSelection' can only be used, if the PLC controller is on the first place in the CODESYS 'Device_ Config'. Is that not the case the selection has to be made manually by the user.</td></tr><tr><td>en64BitArithmetik</td><td>64 bit arithmetic</td></tr><tr><td>en32BitArithmetik</td><td>32 bit arithmetic</td></tr></table>
If the automatic determination does not work, the following diagnostic number is displayed and the selection must done manually by the user.
<table><tr><td>iErrID</td><td>5</td></tr></table>
Additional information about 'EN_POS_J_ARITHMETIK' and manual selection:
Siehe 'EN_POS_J_ARITHMETIK' auf Seite 601.</td><td></td><td>x</td></tr>
</table>

1) Only for CODESYS V3

Output variables

Name	Type	Description	Sync.	Async.
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled		x
boSBM	BOOL	System ready message		x
boDone	BOOL	Positioning done. Valid for 'boPosAbs', 'boPosRel' and 'boHome'		x
boErr	BOOL	The function block is in an error state FALSE: No error (permitted commanding or warning) TRUE: Error		x
iErrID	INT	Error identity number: Diagnostic number is output iErrID = 0: No error iErrID ≠ 0, boErr = TRUE: Error iErrID ≠ 0, boErr = FALSE: Warning Bus error when 'boErr' = TRUE: EtherCAT or CAN are not in operational mode.		x
enErrName	ENUM	EN_FB_NAME Name of faulty block for which the diagnostic number is output. The diagnostic number is explained in the description of the corresponding library.		x
boPosBusy	BOOL	Positioning active		x
boJogBusy	BOOL	Jog mode active		x
boHomeBusy	BOOL	Home active		x
boSpeedBusy	BOOL	Speed control active		x
boEnabAckPmRegisterCtrlDiscont	BOOL	Print mark module initialized and active		x
boParameterSetupDone	BOOL	Parameterisation finished		x
boPmDetected	BOOL	Print mark detected, control activated		x
boPmLost	BOOL	Print mark lost activated when the number of print marks set in 'iNoPmLost' is not detected successively.		x
boPmWnd	BOOL	Mark window activates the print mark sensor		x
diPmDeviation	DINT	Deviation between setpoint and actual position of the actual print mark		x
diOpOffsetOut	DINT	Output value contains the actually output operator offset		x
iActPmLost	INT	Number of lost print marks		x

Input and output variables

Name	Type	Description	Sync.	Async.
stDevice	STRUCT	ST_DEVICE The device description structure assigns the block a device. (See document Software description AmkBase Bibliothek, Part no.204986)		x
stAxisDrive	STRUCT	ST_AXIS_DRIVE Siehe 'ST_AXIS_DRIVE (ST)' auf Seite 191.		x

Usage note in the CODESYS program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
STANDARD_AXIS_PRINTMARK_REGISTER_CONTROL_DISCONT	STANDARD_AXIS_PRINTMARK_REGISTER_CONTROL_DISCONT.actSync

History

Library	AmkAfl	
System versions	**Functionality**	**Target**
AS-C V3.01/0827	Basic functionality (1st version)	≥ AS_CP_200_0812_T202053
AS-PL15 V3.01/0804	Basic functionality (1st version)	≥ AS_V313_0946_T202724

4.1.4.2.9 STANDARD_FOLLOW_AXIS_GEAR (FB)

Expanded functionality:

- The slave axis continuously follows a master.
- A gear ratio can be defined.

The following basic functionalities for the axis are available:

Inputs/outputs on the function block

- Control bits (RF Controller enable, FL Clear error)
- Emergency stop
- Speed control
- Homing cycle
- Positioning (absolute/relative)
- Jog mode (minus/plus)
- Status signals of axis functions

Access via program code

- Configuration of status messages (ID26 'Configuration status bits')
- Reading of parameters according to a specifications list
- Actual position cached
- Control with feedforward (torque and speed)

User interface

1) Only for CODESYS V3

Input variables

Name	Type	Description	Sync.	Async.
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.		x

Name	Type	Description	Sync.	Async.
boRF	BOOL	Bit for enabling the controller Relevant parameters: ID32796 'Source RF' Allowed: Code 5: 'Controller enable via EtherCAT' Code 25: RF 'via EtherCAT AND-linked with the binary input UE'  Code 0: 'Controller enable (RF) via binary input' is not allowed  Behavior by RF inactive in the operating mode position control with mode following (boFollow = TRUE) and active SAK (ID32801 Bit 9 = TRUE) In the operating mode position control with active SAK (following error compensation), the function deceleration ramp after RF inactive (see ID32782 'Deceleration ramp RF inactive') can not be used. When the controller enable (boRF = FALSE) is de-activated, the setpoint output value is immediately stopped and the speed is maximum decelerated. Workaround for applications with deceleration ramps: Use the PLC to generate the setpoint values for the deceleration ramp. Remove the controller enable signal only at standstill.		x
boFL	BOOL	Bit for error deletion		x
boEmergency_ Stop	BOOL	EMERGENCY STOP: The setpoint of the velocity is decreased to zero along the emergency-stop ramp. Once initiated, an emergency stop cannot be aborted. Relevant variables: EMERGENCY STOP ramp: Ramp time in which the drive is slowed down from the current velocity to zero. EMERGENCY STOP ramp: ST_AXIS_DRIVE.ST_IDS.ST_ID32919_ PlcList. diID32919_05_diEmergency_StopRamp Unit: ms Default value: 100 ms		x

Name	Type	Description	Sync.	Async.
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). Relevant parameters: ID32802 'AMK secondary operating mode 2' [Operating mode: Speed control] [Setpoint source: diMainSetpoint (code 41h)] [Ramps: active / inactive (bit 6)] ID32780 'Acceleration ramp' ID32781'Deceleration ramp' Relevant variables: Speed setpoint: diSpeedVelocity Velocity ramp: STANDARD_AXIS.fbVelocity.diVelocityRamp [default value: 100 ms] Speed control active: boSpeedBusy		x
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. Relevant parameters: ID41 'Homing velocity' ID147 'Homing parameter' ID150 'Homing offset 1' ID32926 'AMK homing cycle parameter' Relevant variables: Homing drive active: boHomeBusy Homing done: boDone Homing point known: stAxisDrive.stStatus.boID33036_RFP_known		x

Name	Type	Description	Sync.	Async.
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 [Relevant parameters: see 'boPosRel']		x
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. Relevant parameters: ID32801 'AMK secondary operating mode 1' [Operating mode: position control] [Setpoint source: diMainSetpoint (code 41h)] Relevant variables: Positioning velocity: udPosVelocity Target position: diPosition Acceleration: lreAccel Deceleration: lreDecel Jerk limitation (start): lreJerkStart Jerk limitation (end): lreJerkEnd Positioning active: boPosBusy Positioning done: boDone		x
boJogPlus	BOOL	Jog in positive direction Enable signal: With a positive edge, the jogging function starts (positive direction). Jog mode is run as long as 'boJobPlus' = TRUE. Use a negative edge 'boJogPlus' = FALSE to cancel the current jog mode. [Relevant parameters: see 'boJogMinus']		x

Name	Type	Description	Sync.	Async.
boJogMinus	BOOL	Jogging in negative direction Enable signal: With a positive edge, the jogging function starts (negative direction). Jog mode is run as long as 'boJobMinus' = TRUE. Use a negative edge 'boJogMinus' = FALSE to cancel the current jog mode. Relevant parameters: ID32801 'AMK secondary operating mode 1' [Operating mode: position control] [Setpoint source: diMainSetpoint (code 41h)] Relevant variables: Jog velocity: udVelocityJog Acceleration: lreAccel Deceleration: lreDecel Jerk limitation (start): lreJerkStart Jerk limitation (end): lreJerkEnd Jog mode active: boJogBusy		x
boFollow	BOOL	Follow operation Enable signal: With a positive edge, the follow operation function starts. Follow operation is run as long as 'boFollow' = TRUE. Use a negative edge 'boFollow' = FALSE to cancel the ongoing follow operation. The axis follows the input increments via the preset gear ratio. Relevant parameters: ID32801 'AMK secondary operating mode 1' [Operating mode: position control] [Setpoint source: diMainSetpoint (code 41h)] Relevant variables: Setpoint specification: diInVal Multiplier: diMultiplier Divider: udDivider		x
diMultiplier	DINT	Use 'diMultiplier' and 'udDivider' to predefine any ratio. Setpoint value of follow operation = diInVal * (diMultiplier / udDivider)	x	
udDivider	UDINT		x	
udPosVelocity	UDINT	Positioning velocity Unit: incr/s Default value: 34133		x
diPosition	DINT	Target position Use the inputs 'boPosAbs' and 'boPosRel' to determine whether the target position is approached absolutely or relatively. Unit: Increments Default value: 30720		x
lreAccel	LREAL	Acceleration for positioning and jog mode Range: $1{,}43\ 10^{-13} < \lvert\text{'lreAccel'}\rvert < 1{,}43\ 10^{+16}$ Unit: $incr/s^2$ Default: 1000000		x

<table>
<tr><th>Name</th><th>Type</th><th>Description</th><th>Sync.</th><th>Async.</th></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><td></td><td>x</td></tr>
<tr><td>lreJerkStart</td><td>LREAL</td><td>Jerk limitation (start) for positioning and jog mode<table><tr><td>Unit</td><td>incr/s^3</td></tr><tr><td>Default</td><td>10000000</td></tr></table></td><td></td><td>x</td></tr>
<tr><td>lreJerkEnd</td><td>LREAL</td><td>Jerk limitation (end) for positioning and jog mode<table><tr><td>Unit</td><td>incr/s^3</td></tr><tr><td>Default</td><td>10000000</td></tr></table></td><td></td><td>x</td></tr>
<tr><td>udVelocityJog</td><td>UDINT</td><td>Jog velocity
Unit: incr/s
Default value: 34133</td><td></td><td>x</td></tr>
<tr><td>diSpeedVelocity</td><td>DINT</td><td>Speed setpoint for speed control
Unit: rpm
Default value: 100</td><td></td><td>x</td></tr>
<tr><td>diInVal</td><td>DINT</td><td>Setpoint specification for follow operation
(e.g., through master axis, FB VGEN)
Unit: Increments</td><td>x</td><td></td></tr>
<tr><td>enArithmetik[1)]</td><td>ENUM</td><td>EN_POS_J_ARITHMETIK
(This input is only available for CODESYS V3 function blocks.
CODESYS V2, the selection is always adjusted automatically)

With 'EN_POS_J_ARITHMETIK' the selection 32 bit or 64 bit arithmetic for position function blocks in position operation mode is set.

Options:<table><tr><td>enAutomaticSelection</td><td>Automatic determination if 32 bit or 64 bit arithmetic (Default)
 
The default setting 'enAutomaticSelection' can only be used, if the PLC controller is on the first place in the CODESYS 'Device_ Config'. Is that not the case the selection has to be made manually by the user.</td></tr><tr><td>en64BitArithmetik</td><td>64 bit arithmetic</td></tr><tr><td>en32BitArithmetik</td><td>32 bit arithmetic</td></tr></table>If the automatic determination does not work, the following diagnostic number is displayed and the selection must done manually by the user.<table><tr><td>iErrID</td><td>5</td></tr></table>Additional information about 'EN_POS_J_ARITHMETIK' and manual selection:
Siehe 'EN_POS_J_ARITHMETIK' auf Seite 601.</td><td></td><td>x</td></tr>
</table>

1) Only for CODESYS V3

Output variables

Name	Type	Description	Sync.	Async.
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled		x
boSBM	BOOL	System ready message		x
boDone	BOOL	Positioning done. Valid for 'boPosAbs', 'boPosRel' and 'boHome'		x
boErr	BOOL	The function block is in an error state FALSE: No error (permitted commanding or warning) TRUE: Error		x
iErrID	INT	Error identity number: Diagnostic number is output iErrID = 0: No error iErrID ≠ 0, boErr = TRUE: Error iErrID ≠ 0, boErr = FALSE: Warning		x
enErrName	ENUM	EN_FB_NAME Name of faulty block for which the diagnostic number is output. The diagnostic number is explained in the description of the corresponding library.		x
boPosBusy	BOOL	Positioning active		x
boJogBusy	BOOL	Jog mode active		x
boHomeBusy	BOOL	Home active		x
boSpeedBusy	BOOL	Speed control active		x
boFollowBusy	BOOL	Follow operation active		x

Input and output variables

Name	Type	Description	Sync.	Async.
stDevice	STRUCT	ST_DEVICE The device description structure assigns the block a device. (See document Software description AmkBase Bibliothek, Part no.204986)		x
stAxisDrive	STRUCT	ST_AXIS_DRIVE Siehe 'ST_AXIS_DRIVE (ST)' auf Seite 191.		x

Usage note in the CODESYS program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
STANDARD_AXIS	STANDARD_AXIS.actSync

History

Library	AmkAfl	
System versions	**Functionality**	**Target**
AS-C V3.01/0827	Basic functionality (1st version)	≥ AS_CP_200_0812_T202053
AS-PL15 V3.01/0804	Basic functionality (1st version)	≥ AS_V313_0946_T202724

4.1.4.2.10 STANDARD_FOLLOW_AXIS_CONTINUOUS_TABLE1_TABLE2 (FB)

Expanded functionality:

- The slave axis continuously follows a freely selectable table.
- At the end of the table, a table switchover can be carried out.

Siehe 'FOLLOW_AXIS_CONTINUOUS_TABLE1_TABLE2 (FB)' auf Seite 388.

The following basic functionalities for the axis are available:

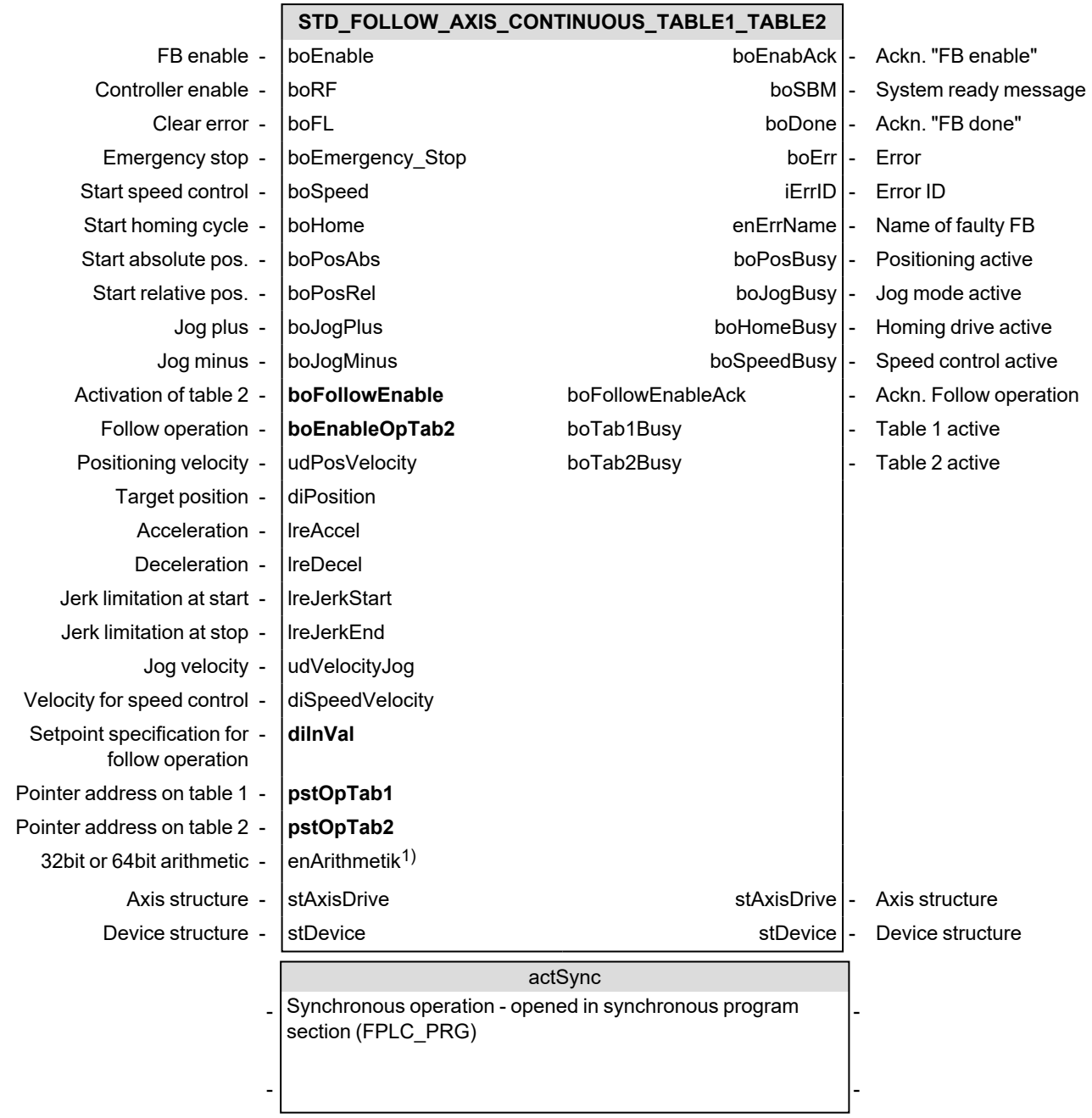
Inputs/outputs on the function block

- Control bits (RF Controller enable, FL Clear error)
- Emergency stop
- Speed control
- Homing cycle
- Positioning (absolute/relative)
- Jog mode (minus/plus)
- Status signals of axis functions

Access via program code

- Configuration of status messages (ID26 'Configuration status bits')
- Reading of parameters according to a specifications list
- Actual position cached
- Control with feedforward (torque and speed)

User interface



1) Only for CODESYS V3

Input variables

Name	Type	Description	Sync.	Async.
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.		x

Name	Type	Description	Sync.	Async.
boRF	BOOL	Bit for enabling the controller Relevant parameters: ID32796 'Source RF' Allowed: Code 5: 'Controller enable via EtherCAT' Code 25: RF 'via EtherCAT AND-linked with the binary input UE'  Code 0: 'Controller enable (RF) via binary input' is not allowed  Behavior by RF inactive in the operating mode position control with mode following (boFollow = TRUE) and active SAK (ID32801 Bit 9 = TRUE) In the operating mode position control with active SAK (following error compensation), the function deceleration ramp after RF inactive (see ID32782 'Deceleration ramp RF inactive') can not be used. When the controller enable (boRF = FALSE) is de-activated, the setpoint output value is immediately stopped and the speed is maximum decelerated. Workaround for applications with deceleration ramps: Use the PLC to generate the setpoint values for the deceleration ramp. Remove the controller enable signal only at standstill.		x
boFL	BOOL	Bit for error deletion		x
boEmergency_ Stop	BOOL	EMERGENCY STOP: The setpoint of the velocity is decreased to zero along the emergency-stop ramp. Once initiated, an emergency stop cannot be aborted. Relevant variables: EMERGENCY STOP ramp: Ramp time in which the drive is slowed down from the current velocity to zero. EMERGENCY STOP ramp: ST_AXIS_DRIVE.ST_IDS.ST_ID32919_ PlcList. diID32919_05_diEmergency_StopRamp Unit: ms Default value: 100 ms		x

Name	Type	Description	Sync.	Async.
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). Relevant parameters: ID32802 'AMK secondary operating mode 2' [Operating mode: Speed control] [Setpoint source: diMainSetpoint (code 41h)] [Ramps: active / inactive (bit 6)] ID32780 'Acceleration ramp' ID32781'Deceleration ramp' Relevant variables: Speed setpoint: diSpeedVelocity Velocity ramp: STANDARD_AXIS.fbVelocity.diVelocityRamp [default value: 100 ms] Speed control active: boSpeedBusy		x
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. Relevant parameters: ID41 'Homing velocity' ID147 'Homing parameter' ID150 'Homing offset 1' ID32926 'AMK homing cycle parameter' Relevant variables: Homing drive active: boHomeBusy Homing done: boDone Homing point known: stAxisDrive.stStatus.boID33036_RFP_known		x

Name	Type	Description	Sync.	Async.
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 [Relevant parameters: see 'boPosRel']		x
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. Relevant parameters: ID32801 'AMK secondary operating mode 1' [Operating mode: position control] [Setpoint source: diMainSetpoint (code 41h)] Relevant variables: Positioning velocity: udPosVelocity Target position: diPosition Acceleration: lreAccel Deceleration: lreDecel Jerk limitation (start): lreJerkStart Jerk limitation (end): lreJerkEnd Positioning active: boPosBusy Positioning done: boDone		x
boJogPlus	BOOL	Jog in positive direction Enable signal: With a positive edge, the jogging function starts (positive direction). Jog mode is run as long as 'boJobPlus' = TRUE. Use a negative edge 'boJogPlus' = FALSE to cancel the current jog mode. [Relevant parameters: see 'boJogMinus']		x

Name	Type	Description	Sync.	Async.
boJogMinus	BOOL	Jogging in negative direction Enable signal: With a positive edge, the jogging function starts (negative direction). Jog mode is run as long as 'boJobMinus' = TRUE. Use a negative edge 'boJogMinus' = FALSE to cancel the current jog mode. Relevant parameters: ID32801 'AMK secondary operating mode 1' [Operating mode: position control] [Setpoint source: diMainSetpoint (code 41h)] Relevant variables: Jog velocity: udVelocityJog Acceleration: lreAccel Deceleration: lreDecel Jerk limitation (start): lreJerkStart Jerk limitation (end): lreJerkEnd Jog mode active: boJogBusy		x
boFollowEnable	BOOL	Follow operation Enable signal: With a positive edge, the follow operation function starts. Follow operation is run as long as 'boFollowEnable' = TRUE. Use a negative edge 'boFollowEnable' = FALSE to cancel the ongoing follow operation. The axis follows the input increments and the transferred table. Relevant parameters: ID32801 'AMK secondary operating mode 1' [Operating mode: position control] [Setpoint source: diMainSetpoint (code 41h)] Relevant variables: Setpoint specification: diInVal Table selection: boEnabOpTab2 Pointer to table 1: pstOpTab1 Pointer to table 2: pstOpTab2		x
boEnabOpTab2	BOOL	• For 'boEnabOpTab2' = TRUE, the second operating table is enabled; switching is performed at the coordinate origin. • With 'boEnabOpTab2' = FALSE, operating table 1 is used again. This switch is also performed at the coordinate origin.	x	
udPosVelocity	UDINT	Positioning velocity Unit: incr/s Default value: 34133		x
diPosition	DINT	Target position Use the inputs 'boPosAbs' and 'boPosRel' to determine whether the target position is approached absolutely or relatively. Unit: Increments Default value: 30720		x

Name	Type	Description	Sync.	Async.
lreAccel	LREAL	Acceleration for positioning and jog mode Range: $1{,}43\ 10^{-13} < \lvert\text{lreAccel}\rvert < 1{,}43\ 10^{+16}$ Unit: incr/s^2 Default: 1000000		x
lreDecel	LREAL	Deceleration for positioning and jog mode Range: $1{,}43\ 10^{-13} < \lvert\text{lreDecel}\rvert < 1{,}43\ 10^{+16}$ Unit: incr/s^2 Default: 1000000		x
lreJerkStart	LREAL	Jerk limitation (start) for positioning and jog mode Unit: incr/s^3 Default: 10000000		x
lreJerkEnd	LREAL	Jerk limitation (end) for positioning and jog mode Unit: incr/s^3 Default: 10000000		x
udVelocityJog	UDINT	Jog velocity Unit: incr/s Default value: 34133		x
diSpeedVelocity	DINT	Speed setpoint for speed control Unit: rpm Default value: 100		x
diInVal	DINT	Setpoint specification for follow operation (e.g., through master axis, FB VGEN) Unit: Increments	x	
pstOpTab1	POINTER	POINTER TO ST_PROF_TAB Point to operating table. (See document Software description AmkBase Bibliothek, Part no. 204986)	x	
pstOpTab2	POINTER	POINTER TO ST_PROF_TAB Point to operating table. (See document Software description AmkBase Bibliothek, Part no. 204986)	x	

<table>
<tr><th>Name</th><th>Type</th><th>Description</th><th>Sync.</th><th>Async.</th></tr>
<tr><td>enArithmetik[1)]</td><td>ENUM</td><td>EN_POS_J_ARITHMETIK
(This input is only available for CODESYS V3 function blocks. CODESYS V2, the selection is always adjusted automatically)

With 'EN_POS_J_ARITHMETIK' the selection 32 bit or 64 bit arithmetic for position function blocks in position operation mode is set.

Options:
<table><tr><td>enAutomaticSelection</td><td>Automatic determination if 32 bit or 64 bit arithmetic (Default)
 
The default setting 'enAutomaticSelection' can only be used, if the PLC controller is on the first place in the CODESYS 'Device_ Config'. Is that not the case the selection has to be made manually by the user.</td></tr><tr><td>en64BitArithmetik</td><td>64 bit arithmetic</td></tr><tr><td>en32BitArithmetik</td><td>32 bit arithmetic</td></tr></table>
If the automatic determination does not work, the following diagnostic number is displayed and the selection must done manually by the user.
<table><tr><td>iErrID</td><td>5</td></tr></table>
Additional information about 'EN_POS_J_ARITHMETIK' and manual selection:
Siehe 'EN_POS_J_ARITHMETIK' auf Seite 601.</td><td></td><td>x</td></tr>
</table>

1) Only for CODESYS V3

Output variables

Name	Type	Description	Sync.	Async.
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled		x
boSBM	BOOL	System ready message		x
boDone	BOOL	Positioning done. Valid for 'boPosAbs', 'boPosRel' and 'boHome'		x
boErr	BOOL	The function block is in an error state FALSE: No error (permitted commanding or warning) TRUE: Error		x
iErrID	INT	Error identity number: Diagnostic number is output iErrID = 0: No error iErrID ≠ 0, boErr = TRUE: Error iErrID ≠ 0, boErr = FALSE: Warning		x
enErrName	ENUM	EN_FB_NAME Name of faulty block for which the diagnostic number is output. The diagnostic number is explained in the description of the corresponding library.		x
boPosBusy	BOOL	Positioning active		x
boJogBusy	BOOL	Jog mode active		x
boHomeBusy	BOOL	Home active		x
boSpeedBusy	BOOL	Speed control active		x
boFollowEnableAck	BOOL	Acknowledgement: Follow operation		x
boTab1Busy	BOOL	pstOpTab1: Table 1 (cam 1) enabled	x	
boTab2Busy	BOOL	pstOpTab2: Table 2 (cam 2) enabled	x	

Input and output variables

Name	Type	Description	Sync.	Async.
stDevice	STRUCT	ST_DEVICE The device description structure assigns the block a device. (See document Software description AmkBase Bibliothek, Part no.204986)		x
stAxisDrive	STRUCT	ST_AXIS_DRIVE Siehe 'ST_AXIS_DRIVE (ST)' auf Seite 191.		x

Usage note in the CODESYS program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
STANDARD_AXIS	STANDARD_AXIS.actSync

History

Library	AmkAfl	
System versions	**Functionality**	**Target**
AS-C V3.01/0827	Basic functionality (1st version)	≥ AS_CP_200_0812_T202053
AS-PL15 V3.01/0804	Basic functionality (1st version)	≥ AS_V313_0946_T202724

4.1.4.2.11 STANDARD_FOLLOW_AXIS_SWITCHING_TABLE_WITH_OUTGEAR (FB)

Expanded functionality:

- The slave axis continuously follows a freely selectable table.
- At the end of the table, a table switchover can be carried out.
- The table can be modified via a gear ratio (compress and extend).

Siehe 'FOLLOW_AXIS_SWITCHING_TABLE1_TABLE2 (FB)' auf Seite 392.

The following basic functionalities for the axis are available:

Inputs/outputs on the function block

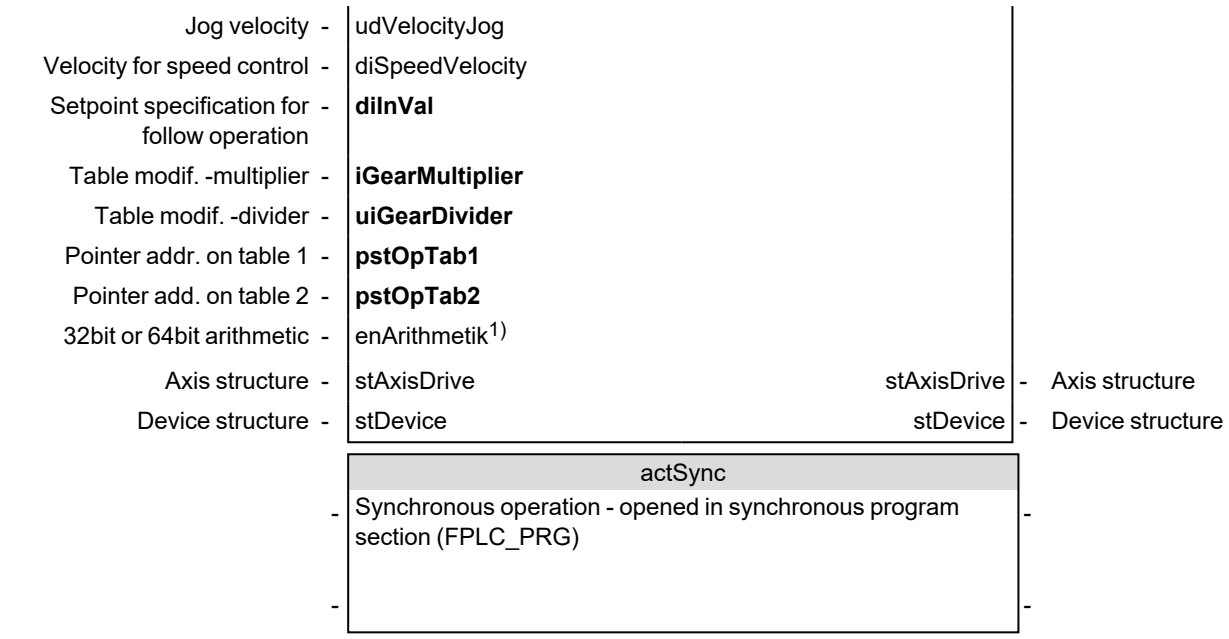
- Control bits (RF Controller enable, FL Clear error)
- Emergency stop
- Speed control
- Homing cycle
- Positioning (absolute/relative)
- Jog mode (minus/plus)
- Status signals of axis functions

Access via program code

- Configuration of status messages (ID26 'Configuration status bits')
- Reading of parameters according to a specifications list
- Actual position cached
- Control with feedforward (torque and speed)

User interface

	STD_...._WITH_OUTGEAR			
FB enable -	boEnable	boEnabAck	-	Ackn. "FB enable"
Controller enable -	boRF	boSBM	-	System ready message
Clear error -	boFL	boDone	-	Ackn. "FB done"
Emergency stop -	boEmergency_Stop	boErr	-	Error
Start speed control -	boSpeed	iErrID	-	Error ID
Start homing cycle -	boHome	enErrName	-	Name of faulty FB
Start absolute pos. -	boPosAbs	boPosBusy	-	Positioning active
Start relative pos. -	boPosRel	boJogBusy	-	Jog mode active
Jog plus -	boJogPlus	boHomeBusy	-	Homing drive active
Jog minus -	boJogMinus	boSpeedBusy	-	Speed control active
Activation of table 2 -	**boFollowEnable**	boFollowEnableAck	-	Follow operation release
Control of tab. function -	**boEnableOpTab2**	boTab1Busy	-	Table 1 active
Positioning velocity -	udPosVelocity	boTab2Busy	-	Table 2 active
Target position -	diPosition			
Acceleration -	lreAccel			
Deceleration -	lreDecel			
Jerk limitation at start -	lreJerkStart			
Jerk limitation at stop -	lreJerkEnd			



1) Only for CODESYS V3

Input variables

Name	Type	Description	Sync.	Async.
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.		x
boRF	BOOL	Bit for enabling the controller Relevant parameters: ID32796 'Source RF' Allowed: Code 5: 'Controller enable via EtherCAT' Code 25: RF 'via EtherCAT AND-linked with the binary input UE' Code 0: 'Controller enable (RF) via binary input' is not allowed Behavior by RF inactive in the operating mode position control with mode following (boFollow = TRUE) and active SAK (ID32801 Bit 9 = TRUE) In the operating mode position control with active SAK (following error compensation), the function deceleration ramp after RF inactive (see ID32782 'Deceleration ramp RF inactive') can not be used. When the controller enable (boRF = FALSE) is de-activated, the setpoint output value is immediately stopped and the speed is maximum decelerated. Workaround for applications with deceleration ramps: Use the PLC to generate the setpoint values for the deceleration ramp. Remove the controller enable signal only at standstill.		x
boFL	BOOL	Bit for error deletion		x

Name	Type	Description	Sync.	Async.
boEmergency_ Stop	BOOL	EMERGENCY STOP: The setpoint of the velocity is decreased to zero along the emergency-stop ramp. Once initiated, an emergency stop cannot be aborted. Relevant variables: EMERGENCY STOP ramp: Ramp time in which the drive is slowed down from the current velocity to zero. EMERGENCY STOP ramp: ST_AXIS_DRIVE.ST_IDS.ST_ID32919_ PlcList. diID32919_05_diEmergency_StopRamp Unit: ms Default value: 100 ms		x
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). Relevant parameters: ID32802 'AMK secondary operating mode 2' [Operating mode: Speed control] [Setpoint source: diMainSetpoint (code 41h)] [Ramps: active / inactive (bit 6)] ID32780 'Acceleration ramp' ID32781'Deceleration ramp' Relevant variables: Speed setpoint: diSpeedVelocity Velocity ramp: STANDARD_AXIS.fbVelocity.diVelocityRamp [default value: 100 ms] Speed control active: boSpeedBusy		x

Name	Type	Description	Sync.	Async.
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. Relevant parameters: ID41 'Homing velocity' ID147 'Homing parameter' ID150 'Homing offset 1' ID32926 'AMK homing cycle parameter' Relevant variables: Homing drive active: boHomeBusy Homing done: boDone Homing point known: stAxisDrive.stStatus.boID33036_RFP_known		x
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 [Relevant parameters: see 'boPosRel']		x
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. Relevant parameters: ID32801 'AMK secondary operating mode 1' [Operating mode: position control] [Setpoint source: diMainSetpoint (code 41h)] Relevant variables: Positioning velocity: udPosVelocity Target position: diPosition Acceleration: lreAccel Deceleration: lreDecel Jerk limitation (start): lreJerkStart Jerk limitation (end): lreJerkEnd Positioning active: boPosBusy Positioning done: boDone		x

Name	Type	Description	Sync.	Async.
boJogPlus	BOOL	Jog in positive direction Enable signal: With a positive edge, the jogging function starts (positive direction). Jog mode is run as long as 'boJobPlus' = TRUE. Use a negative edge 'boJogPlus' = FALSE to cancel the current jog mode. [Relevant parameters: see 'boJogMinus']		x
boJogMinus	BOOL	Jogging in negative direction Enable signal: With a positive edge, the jogging function starts (negative direction). Jog mode is run as long as 'boJobMinus' = TRUE. Use a negative edge 'boJogMinus' = FALSE to cancel the current jog mode. Relevant parameters: ID32801 'AMK secondary operating mode 1' [Operating mode: position control] [Setpoint source: diMainSetpoint (code 41h)] Relevant variables: Jog velocity: udVelocityJog Acceleration: lreAccel Deceleration: lreDecel Jerk limitation (start): lreJerkStart Jerk limitation (end): lreJerkEnd Jog mode active: boJogBusy		x
boFollowEnable	BOOL	Follow operation Enable signal: With a positive edge, the follow operation function starts. Follow operation is run as long as 'boFollowEnable' = TRUE. Use a negative edge 'boFollowEnable' = FALSE to cancel the ongoing follow operation. The axis follows the input increments and the transferred table. Relevant parameters: ID32801 'AMK secondary operating mode 1' [Operating mode: position control] [Setpoint source: diMainSetpoint (code 41h)] Relevant variables: Setpoint specification: diInVal Table selection: boEnabOpTab2 Pointer to table 1: pstOpTab1 Pointer to table 2: pstOpTab2 Table modification multiplier: iGearMultiplier Table modification divider: uiGearDivider	x	
boEnabOpTab2	BOOL	Activation of operating table 2 boEnabOpTab2:= FALSE (pstOpTab1 enabled) boEnabOpTab2:= TRUE (pstOpTab2 enabled)	x	

Name	Type	Description	Sync.	Async.
udPosVelocity	UDINT	Positioning velocity Unit: incr/s Default value: 34133		x
diPosition	DINT	Target position Use the inputs 'boPosAbs' and 'boPosRel' to determine whether the target position is approached absolutely or relatively. Unit: Increments Default value: 30720		x
lreAccel	LREAL	Acceleration for positioning and jog mode Range: $1{,}43\ 10^{-13} < \lvert\text{lreAccel}\rvert < 1{,}43\ 10^{+16}$ Unit: incr/s^2 Default: 1000000		x
lreDecel	LREAL	Deceleration for positioning and jog mode Range: $1{,}43\ 10^{-13} < \lvert\text{lreDecel}\rvert < 1{,}43\ 10^{+16}$ Unit: incr/s^2 Default: 1000000		x
lreJerkStart	LREAL	Jerk limitation (start) for positioning and jog mode Unit: incr/s^3 Default: 10000000		x
lreJerkEnd	LREAL	Jerk limitation (end) for positioning and jog mode Unit: incr/s^3 Default: 10000000		x
udVelocityJog	UDINT	Jog velocity Unit: incr/s Default value: 34133		x
diSpeedVelocity	DINT	Speed setpoint for speed control Unit: rpm Default value: 100		x
diInVal	DINT	Setpoint specification for follow operation (e.g., through master axis, FB VGEN) Unit: Increments	x	
iGearMultiplier	INT	Table modification multiplier to compress/extend the cam function	x	
uiGearDivider	UINT	Table modification divider to compress/extend the cam function	x	
pstOpTab1	POINTER	POINTER TO ST_PROF_TAB Point to operating table. (See document Software description AmkBase Bibliothek, Part no. 204986)	x	
pstOpTab2	POINTER	POINTER TO ST_PROF_TAB Point to operating table. (See document Software description AmkBase Bibliothek, Part no. 204986)	x	

<table>
<tr><th>Name</th><th>Type</th><th>Description</th><th>Sync.</th><th>Async.</th></tr>
<tr><td>enArithmetik[1)]</td><td>ENUM</td><td>EN_POS_J_ARITHMETIK
(This input is only available for CODESYS V3 function blocks. CODESYS V2, the selection is always adjusted automatically)

With 'EN_POS_J_ARITHMETIK' the selection 32 bit or 64 bit arithmetic for position function blocks in position operation mode is set.

Options:
<table>
<tr><td>enAutomaticSelection</td><td>Automatic determination if 32 bit or 64 bit arithmetic (Default)
 
The default setting 'enAutomaticSelection' can only be used, if the PLC controller is on the first place in the CODESYS 'Device_ Config'. Is that not the case the selection has to be made manually by the user.</td></tr>
<tr><td>en64BitArithmetik</td><td>64 bit arithmetic</td></tr>
<tr><td>en32BitArithmetik</td><td>32 bit arithmetic</td></tr>
</table>
If the automatic determination does not work, the following diagnostic number is displayed and the selection must done manually by the user.
<table>
<tr><td>iErrID</td><td>5</td></tr>
</table>
Additional information about 'EN_POS_J_ARITHMETIK' and manual selection:
Siehe 'EN_POS_J_ARITHMETIK' auf Seite 601.</td><td></td><td>x</td></tr>
</table>

1) Only for CODESYS V3

Output variables

Name	Type	Description	Sync.	Async.
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled		x
boSBM	BOOL	System ready message		x
boDone	BOOL	Positioning done. Valid for 'boPosAbs', 'boPosRel' and 'boHome'		x
boErr	BOOL	The function block is in an error state FALSE: No error (permitted commanding or warning) TRUE: Error		x
iErrID	INT	Error identity number: Diagnostic number is output iErrID = 0: No error iErrID ≠ 0, boErr = TRUE: Error iErrID ≠ 0, boErr = FALSE: Warning		x
enErrName	ENUM	EN_FB_NAME Name of faulty block for which the diagnostic number is output. The diagnostic number is explained in the description of the corresponding library.		x
boPosBusy	BOOL	Positioning active		x
boJogBusy	BOOL	Jog mode active		x
boHomeBusy	BOOL	Home active		x
boSpeedBusy	BOOL	Speed control active		x
boFollowEnableAck	BOOL	Acknowledgement: Follow operation		x
boTab1Busy	BOOL	pstOpTab1: Table 1 (cam 1) enabled	x	
boTab2Busy	BOOL	pstOpTab2: Table 2 (cam 2) enabled	x	

Input and output variables

Name	Type	Description	Sync.	Async.
stDevice	STRUCT	ST_DEVICE The device description structure assigns the block a device. (See document Software description AmkBase Bibliothek, Part no.204986)		x
stAxisDrive	STRUCT	ST_AXIS_DRIVE Siehe 'ST_AXIS_DRIVE (ST)' auf Seite 191.		x

Usage note in the CODESYS program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
STANDARD_AXIS	STANDARD_AXIS.actSync

History

Library	AmkAfl	
System versions	**Functionality**	**Target**
AS-C V3.01/0827	Basic functionality (1st version)	≥ AS_CP_200_0812_T202053
AS-PL15 V3.01/0804	Basic functionality (1st version)	≥ AS_V313_0946_T202724

4.1.4.2.12 STANDARD_FOLLOW_AXIS_SWITCHING_TABLE1_TABLE2 (FB)

Expanded functionality:

- The slave axis continuously follows a freely selectable table with a phasing-in and phasing out table.
- At the end of the table, a table switchover can be carried out.

Siehe 'FOLLOW_AXIS_SWITCHING_TABLE1_TABLE2 (FB)' auf Seite 392.

The following basic functionalities for the axis are available:

Inputs/outputs on the function block

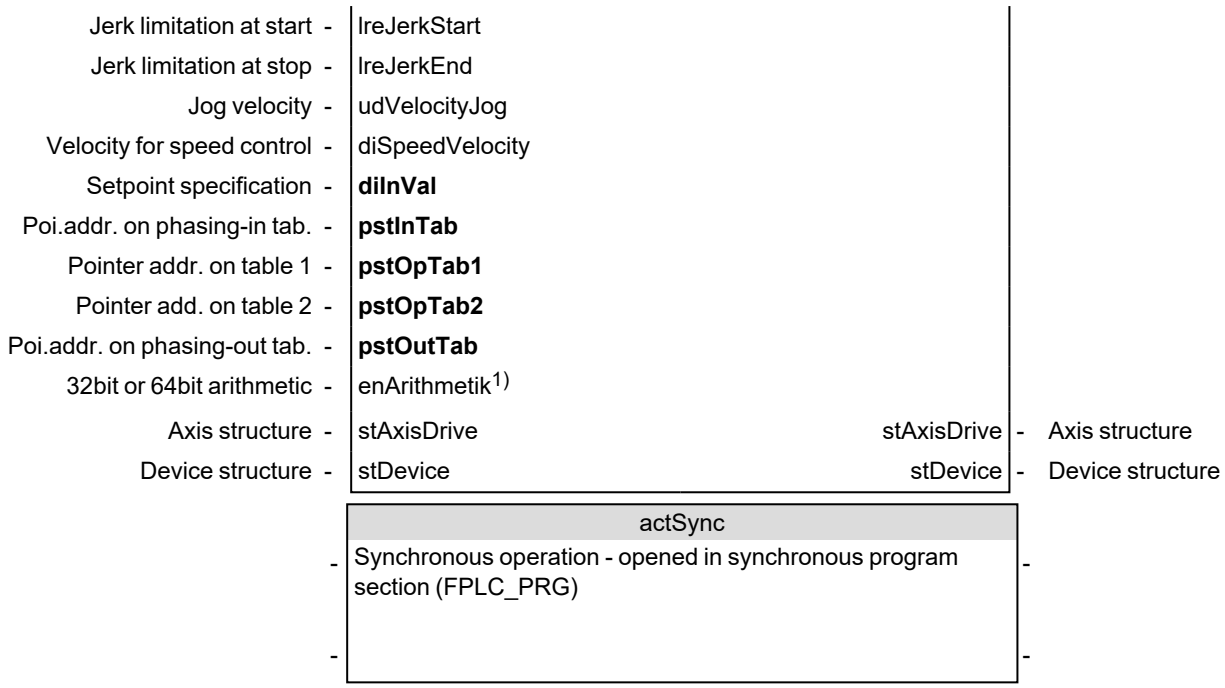
- Control bits (RF Controller enable, FL Clear error)
- Emergency stop
- Speed control
- Homing cycle
- Positioning (absolute/relative)
- Jog mode (minus/plus)
- Status signals of axis functions

Access via program code

- Configuration of status messages (ID26 'Configuration status bits')
- Reading of parameters according to a specifications list
- Actual position cached
- Control with feedforward (torque and speed)

User interface

	STD_FOLLOW_AXIS_SWITCHING_TABLE1_TABLE2			
FB enable -	boEnable	boEnabAck	-	Ackn. "FB enable"
Controller enable -	boRF	boSBM	-	System ready message
Clear error -	boFL	boDone	-	Ackn. "FB done"
Emergency stop -	boEmergency_Stop	boErr	-	Error
Start speed control -	boSpeed	iErrID	-	Error ID
Start homing cycle -	boHome	enErrName	-	Name of faulty FB
Start absolute pos. -	boPosAbs	boPosBusy	-	Positioning active
Start relative pos. -	boPosRel	boJogBusy	-	Jog mode active
Jog plus -	boJogPlus	boHomeBusy	-	Homing drive active
Jog minus -	boJogMinus	boSpeedBusy	-	Speed control active
Activation of table 2 -	**boFollowEnable**	boFollowEnableAck	-	Ackn. Follow operation
Follow operation -	**boTableControl**	boTab1Busy	-	Table 1 active
Control of tab. function -	**boEnableOpTab2**	boTab2Busy	-	Table 2 active
Phasing in angle -	**udInAngle**			
Phasing out angle -	**udOutAngle**			
Positioning velocity -	udPosVelocity			
Target position -	diPosition			
Acceleration -	lreAccel			
Deceleration -	lreDecel			



1) Only for CODESYS V3

Input variables

Name	Type	Description	Sync.	Async.
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.		x
boRF	BOOL	Bit for enabling the controller Relevant parameters: ID32796 'Source RF' Allowed: Code 5: 'Controller enable via EtherCAT' Code 25: RF 'via EtherCAT AND-linked with the binary input UE' Code 0: 'Controller enable (RF) via binary input' is not allowed Behavior by RF inactive in the operating mode position control with mode following (boFollow = TRUE) and active SAK (ID32801 Bit 9 = TRUE) In the operating mode position control with active SAK (following error compensation), the function deceleration ramp after RF inactive (see ID32782 'Deceleration ramp RF inactive') can not be used. When the controller enable (boRF = FALSE) is de-activated, the setpoint output value is immediately stopped and the speed is maximum decelerated. Workaround for applications with deceleration ramps: Use the PLC to generate the setpoint values for the deceleration ramp. Remove the controller enable signal only at standstill.		x

Name	Type	Description	Sync.	Async.
boFL	BOOL	Bit for error deletion		x
boEmergency_ Stop	BOOL	EMERGENCY STOP: The setpoint of the velocity is decreased to zero along the emergency-stop ramp. Once initiated, an emergency stop cannot be aborted. Relevant variables: EMERGENCY STOP ramp: Ramp time in which the drive is slowed down from the current velocity to zero. EMERGENCY STOP ramp: ST_AXIS_DRIVE.ST_IDS.ST_ID32919_ PlcList. diID32919_05_diEmergency_StopRamp Unit: ms Default value: 100 ms		x
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). Relevant parameters: ID32802 'AMK secondary operating mode 2' [Operating mode: Speed control] [Setpoint source: diMainSetpoint (code 41h)] [Ramps: active / inactive (bit 6)] ID32780 'Acceleration ramp' ID32781'Deceleration ramp' Relevant variables: Speed setpoint: diSpeedVelocity Velocity ramp: STANDARD_AXIS.fbVelocity.diVelocityRamp [default value: 100 ms] Speed control active: boSpeedBusy		x

Name	Type	Description	Sync.	Async.
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. Relevant parameters: ID41 'Homing velocity' ID147 'Homing parameter' ID150 'Homing offset 1' ID32926 'AMK homing cycle parameter' Relevant variables: Homing drive active: boHomeBusy Homing done: boDone Homing point known: stAxisDrive.stStatus.boID33036_RFP_known		x
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 [Relevant parameters: see 'boPosRel']		x
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. Relevant parameters: ID32801 'AMK secondary operating mode 1' [Operating mode: position control] [Setpoint source: diMainSetpoint (code 41h)] Relevant variables: Positioning velocity: udPosVelocity Target position: diPosition Acceleration: lreAccel Deceleration: lreDecel Jerk limitation (start): lreJerkStart Jerk limitation (end): lreJerkEnd Positioning active: boPosBusy Positioning done: boDone		x

Name	Type	Description	Sync.	Async.
boJogPlus	BOOL	Jog in positive direction Enable signal: With a positive edge, the jogging function starts (positive direction). Jog mode is run as long as 'boJobPlus' = TRUE. Use a negative edge 'boJogPlus' = FALSE to cancel the current jog mode. [Relevant parameters: see 'boJogMinus']		x
boJogMinus	BOOL	Jogging in negative direction Enable signal: With a positive edge, the jogging function starts (negative direction). Jog mode is run as long as 'boJobMinus' = TRUE. Use a negative edge 'boJogMinus' = FALSE to cancel the current jog mode. Relevant parameters: ID32801 'AMK secondary operating mode 1' [Operating mode: position control] [Setpoint source: diMainSetpoint (code 41h)] Relevant variables: Jog velocity: udVelocityJog Acceleration: lreAccel Deceleration: lreDecel Jerk limitation (start): lreJerkStart Jerk limitation (end): lreJerkEnd Jog mode active: boJogBusy		x
boFollowEnable	BOOL	Follow operation Enable signal: With a positive edge, the follow operation function starts. Follow operation is run as long as 'boFollowEnable' = TRUE. Use a negative edge 'boFollowEnable' = FALSE to cancel the ongoing follow operation. The axis follows the input increments and the transferred table. Relevant parameters: ID32801 'AMK secondary operating mode 1' [Operating mode: position control] [Setpoint source: diMainSetpoint (code 41h)] Relevant variables: Setpoint specification: diInVal Phasing-in range: udInVal Phasing-out range: udOutVal Table selection: boEnabOpTab2 Table function control: boTableControl Pointer to phasing-in table: pstInTab Pointer to phasing-out table: pstOutTab Pointer to table 1: pstOpTab1 Pointer to table 2: pstOpTab2		x

Name	Type	Description	Sync.	Async.
boTableControl	BOOL	Table function control Use boTableControl to control the processing of operating tables. (Prerequisite: The follow operation is enable 'boFollowEnable' = TRUE and the master reference is generated). Enable signal: Use a positive edge at 'boTableControl' to launch the function; the phasing-in table is processed (prerequisite: diInVal within phasing-in range) At the end of the phasing in table, an automatic transition occurs to the operating table. As long as 'boTableControl' = TRUE, the operating table is processed. When 'boTableControl' = False, a switchover to the phasing-out table (prerequisite: diOutVal is within phasing-out range) is made at the end of the phasing-out table. Table processing automatically stops at the end of the phasing-out table. The master reference is not lost in the process.	x	
boEnabOpTab2	BOOL	• For 'boEnabOpTab2' = TRUE, the second operating table is enabled; switching is performed at the coordinate origin. • With 'boEnabOpTab2' = FALSE, operating table 1 is used again. This switch is also performed at the coordinate origin.	x	
udInAngle	UDINT	Phasing in angle Maximum permissible position on the x axis up to which phasing in is permitted Unit: Increments Default value: 1000	x	
udOutAngle	UDINT	Phasing out angle Maximum permissible position on the x axis up to which phasing out is permitted Unit: Increments Default value: 1000	x	
udPosVelocity	UDINT	Positioning velocity Unit: incr/s Default value: 34133		x
diPosition	DINT	Target position Use the inputs 'boPosAbs' and 'boPosRel' to determine whether the target position is approached absolutely or relatively. Unit: Increments Default value: 30720		x
lreAccel	LREAL	Acceleration for positioning and jog mode Range: $1{,}43\ 10^{-13} < \lvert\text{'lreAccel'}\rvert < 1{,}43\ 10^{+16}$ Unit: incr/s^2 Default: 1000000		x
lreDecel	LREAL	Deceleration for positioning and jog mode Range: $1{,}43\ 10^{-13} < \lvert\text{'lreDecel'}\rvert < 1{,}43\ 10^{+16}$ Unit: incr/s^2 Default: 1000000		x
lreJerkStart	LREAL	Jerk limitation (start) for positioning and jog mode Unit: incr/s^3 Default: 10000000		x

Name	Type	Description	Sync.	Async.
lreJerkEnd	LREAL	Jerk limitation (end) for positioning and jog mode Unit: incr/s^3 Default: 10000000		x
udVelocityJog	UDINT	Jog velocity Unit: incr/s Default value: 34133		x
diSpeedVelocity	DINT	Speed setpoint for speed control Unit: rpm Default value: 100		x
diInVal	DINT	Setpoint specification for follow operation (e.g., through master axis, FB VGEN) Unit: Increments	x	
pstInTab	POINTER	POINTER TO ST_PROF_TAB Pointer to phasing-in table. (See document Software description AmkBase Bibliothek, Part no. 204986)	x	
pstOpTab1	POINTER	POINTER TO ST_PROF_TAB Point to operating table. (See document Software description AmkBase Bibliothek, Part no. 204986)	x	
pstOpTab2	POINTER	POINTER TO ST_PROF_TAB Point to operating table. (See document Software description AmkBase Bibliothek, Part no. 204986)	x	
pstOutTab	POINTER	POINTER TO ST_PROF_TAB Pointer to phasing-out table. (See document Software description AmkBase Bibliothek, Part no. 204986)	x	

<table>
<tr><th>Name</th><th>Type</th><th>Description</th><th>Sync.</th><th>Async.</th></tr>
<tr><td>enArithmetik[1)]</td><td>ENUM</td><td>EN_POS_J_ARITHMETIK
(This input is only available for CODESYS V3 function blocks. CODESYS V2, the selection is always adjusted automatically)

With 'EN_POS_J_ARITHMETIK' the selection 32 bit or 64 bit arithmetic for position function blocks in position operation mode is set.

Options:
<table>
<tr><td>enAutomaticSelection</td><td>Automatic determination if 32 bit or 64 bit arithmetic (Default)
 
The default setting 'enAutomaticSelection' can only be used, if the PLC controller is on the first place in the CODESYS 'Device_ Config'. Is that not the case the selection has to be made manually by the user.</td></tr>
<tr><td>en64BitArithmetik</td><td>64 bit arithmetic</td></tr>
<tr><td>en32BitArithmetik</td><td>32 bit arithmetic</td></tr>
</table>
If the automatic determination does not work, the following diagnostic number is displayed and the selection must done manually by the user.
<table>
<tr><td>iErrID</td><td>5</td></tr>
</table>
Additional information about 'EN_POS_J_ARITHMETIK' and manual selection:
Siehe 'EN_POS_J_ARITHMETIK' auf Seite 601.</td><td></td><td>x</td></tr>
</table>

1) Only for CODESYS V3

Output variables

Name	Type	Description	Sync.	Async.
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled		x
boSBM	BOOL	System ready message		x
boDone	BOOL	Positioning done. Valid for 'boPosAbs', 'boPosRel' and 'boHome'		x
boErr	BOOL	The function block is in an error state FALSE: No error (permitted commanding or warning) TRUE: Error		x
iErrID	INT	Error identity number: Diagnostic number is output iErrID = 0: No error iErrID ≠ 0, boErr = TRUE: Error iErrID ≠ 0, boErr = FALSE: Warning		x
enErrName	ENUM	EN_FB_NAME Name of faulty block for which the diagnostic number is output. The diagnostic number is explained in the description of the corresponding library.		x
boPosBusy	BOOL	Positioning active		x
boJogBusy	BOOL	Jog mode active		x
boHomeBusy	BOOL	Home active		x
boSpeedBusy	BOOL	Speed control active		x
boFollowEnableAck	BOOL	Acknowledgement: Follow operation		x
boTab1Busy	BOOL	pstOpTab1: Table 1 (cam 1) enabled	x	
boTab2Busy	BOOL	pstOpTab2: Table 2 (cam 2) enabled	x	

Input and output variables

Name	Type	Description	Sync.	Async.
stDevice	STRUCT	ST_DEVICE The device description structure assigns the block a device. (See document Software description AmkBase Bibliothek, Part no.204986)		x
stAxisDrive	STRUCT	ST_AXIS_DRIVE Siehe 'ST_AXIS_DRIVE (ST)' auf Seite 191.		x

Usage note in the CODESYS program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
STANDARD_AXIS	STANDARD_AXIS.actSync

History

Library	AmkAfl	
System versions	**Functionality**	**Target**
AS-C V3.01/0827	Basic functionality (1st version)	≥ AS_CP_200_0812_T202053
AS-PL15 V3.01/0804	Basic functionality (1st version)	≥ AS_V313_0946_T202724

4.1.4.2.13 STANDARD_FOLLOW_AXIS_TABLE_CONTINUOUS (FB)

Expanded functionality:

- The slave axis continuously follows a freely selectable table.

Siehe 'FOLLOW_AXIS_TABLE_CONTINUOUS (FB)' auf Seite 394.

The following basic functionalities for the axis are available:

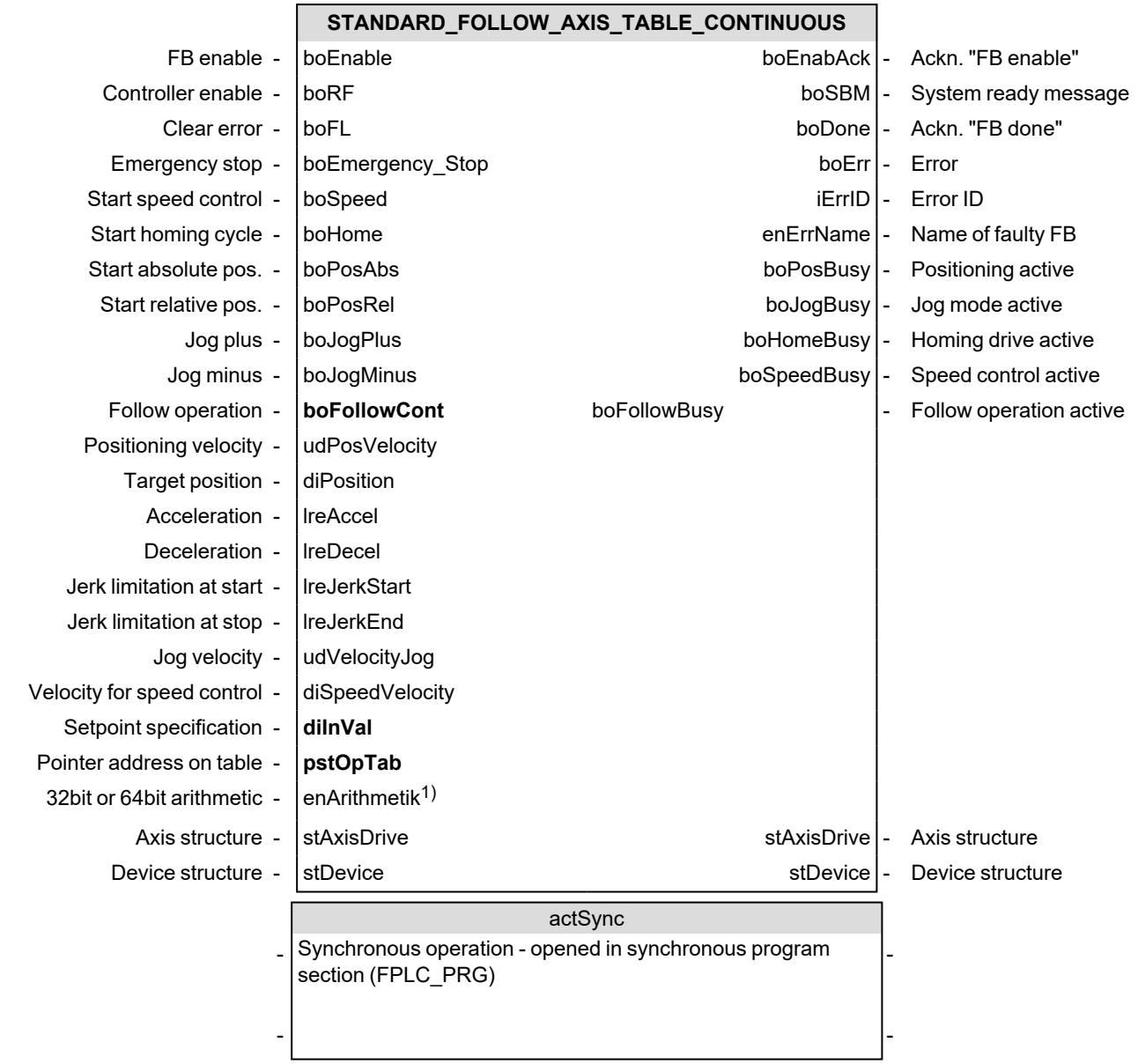
Inputs/outputs on the function block

- Control bits (RF Controller enable, FL Clear error)
- Emergency stop
- Speed control
- Homing cycle
- Positioning (absolute/relative)
- Jog mode (minus/plus)
- Status signals of axis functions

Access via program code

- Configuration of status messages (ID26 'Configuration status bits')
- Reading of parameters according to a specifications list
- Actual position cached
- Control with feedforward (torque and speed)

User interface



1) Only for CODESYS V3

Input variables

Name	Type	Description	Sync.	Async.
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.		x

Name	Type	Description	Sync.	Async.
boRF	BOOL	Bit for enabling the controller Relevant parameters: ID32796 'Source RF' Allowed: Code 5: 'Controller enable via EtherCAT' Code 25: RF 'via EtherCAT AND-linked with the binary input UE'  Code 0: 'Controller enable (RF) via binary input' is not allowed  Behavior by RF inactive in the operating mode position control with mode following (boFollow = TRUE) and active SAK (ID32801 Bit 9 = TRUE) In the operating mode position control with active SAK (following error compensation), the function deceleration ramp after RF inactive (see ID32782 'Deceleration ramp RF inactive') can not be used. When the controller enable (boRF = FALSE) is de-activated, the setpoint output value is immediately stopped and the speed is maximum decelerated. Workaround for applications with deceleration ramps: Use the PLC to generate the setpoint values for the deceleration ramp. Remove the controller enable signal only at standstill.		x
boFL	BOOL	Bit for error deletion		x
boEmergency_ Stop	BOOL	EMERGENCY STOP: The setpoint of the velocity is decreased to zero along the emergency-stop ramp. Once initiated, an emergency stop cannot be aborted. Relevant variables: EMERGENCY STOP ramp: Ramp time in which the drive is slowed down from the current velocity to zero. EMERGENCY STOP ramp: ST_AXIS_DRIVE.ST_IDS.ST_ID32919_ PlcList. diID32919_05_diEmergency_StopRamp Unit: ms Default value: 100 ms		x

Name	Type	Description	Sync.	Async.
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). Relevant parameters: ID32802 'AMK secondary operating mode 2' [Operating mode: Speed control] [Setpoint source: diMainSetpoint (code 41h)] [Ramps: active / inactive (bit 6)] ID32780 'Acceleration ramp' ID32781'Deceleration ramp' Relevant variables: Speed setpoint: diSpeedVelocity Velocity ramp: STANDARD_AXIS.fbVelocity.diVelocityRamp [default value: 100 ms] Speed control active: boSpeedBusy		x
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. Relevant parameters: ID41 'Homing velocity' ID147 'Homing parameter' ID150 'Homing offset 1' ID32926 'AMK homing cycle parameter' Relevant variables: Homing drive active: boHomeBusy Homing done: boDone Homing point known: stAxisDrive.stStatus.boID33036_RFP_known		x

Name	Type	Description	Sync.	Async.
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 [Relevant parameters: see 'boPosRel']		x
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. Relevant parameters: ID32801 'AMK secondary operating mode 1' [Operating mode: position control] [Setpoint source: diMainSetpoint (code 41h)] Relevant variables: Positioning velocity: udPosVelocity Target position: diPosition Acceleration: lreAccel Deceleration: lreDecel Jerk limitation (start): lreJerkStart Jerk limitation (end): lreJerkEnd Positioning active: boPosBusy Positioning done: boDone		x
boJogPlus	BOOL	Jog in positive direction Enable signal: With a positive edge, the jogging function starts (positive direction). Jog mode is run as long as 'boJobPlus' = TRUE. Use a negative edge 'boJogPlus' = FALSE to cancel the current jog mode. [Relevant parameters: see 'boJogMinus']		x

Name	Type	Description	Sync.	Async.
boJogMinus	BOOL	Jogging in negative direction Enable signal: With a positive edge, the jogging function starts (negative direction). Jog mode is run as long as 'boJobMinus' = TRUE. Use a negative edge 'boJogMinus' = FALSE to cancel the current jog mode. Relevant parameters: ID32801 'AMK secondary operating mode 1' [Operating mode: position control] [Setpoint source: diMainSetpoint (code 41h)] Relevant variables: Jog velocity: udVelocityJog Acceleration: lreAccel Deceleration: lreDecel Jerk limitation (start): lreJerkStart Jerk limitation (end): lreJerkEnd Jog mode active: boJogBusy		x
boFollowCont	BOOL	Follow operation Enable signal: With a positive edge, the follow operation function starts. Follow operation is run as long as 'boFollowCont' = TRUE. Use a negative edge 'boFollowCont' = FALSE to abort the ongoing follow operation. The axis follows the input increments and the transferred table. Relevant parameters: ID32801 'AMK secondary operating mode 1' [Operating mode: position control] [Setpoint source: diMainSetpoint (code 41h)] Relevant variables: Setpoint specification: diInVal Pointer to table: pstOpTab		x
udPosVelocity	UDINT	Positioning velocity Unit: incr/s Default value: 34133		x
diPosition	DINT	Target position Use the inputs 'boPosAbs' and 'boPosRel' to determine whether the target position is approached absolutely or relatively. Unit: Increments Default value: 30720		x
lreAccel	LREAL	Acceleration for positioning and jog mode Range: $1{,}43\ 10^{-13} < \lvert\text{'lreAccel'}\rvert < 1{,}43\ 10^{+16}$ Unit: incr/s^2 Default: 1000000		x

Name	Type	Description	Sync.	Async.
lreDecel	LREAL	Deceleration for positioning and jog mode Range: $1{,}43\ 10^{-13} < \lvert'lreDecel'\rvert < 1{,}43\ 10^{+16}$ Unit: $incr/s^2$ Default: 1000000		x
lreJerkStart	LREAL	Jerk limitation (start) for positioning and jog mode Unit: $incr/s^3$ Default: 10000000		x
lreJerkEnd	LREAL	Jerk limitation (end) for positioning and jog mode Unit: $incr/s^3$ Default: 10000000		x
udVelocityJog	UDINT	Jog velocity Unit: incr/s Default value: 34133		x
diSpeedVelocity	DINT	Speed setpoint for speed control Unit: rpm Default value: 100		x
diInVal	DINT	Setpoint specification for follow operation (e.g., through master axis, FB VGEN) Unit: Increments	x	
pstOpTab	POINTER	POINTER TO ST_PROF_TAB Point to operating table. (See document Software description AmkBase Bibliothek, Part no. 204986)	x	

<table>
<tr><th>Name</th><th>Type</th><th>Description</th><th>Sync.</th><th>Async.</th></tr>
<tr><td>enArithmetik[1)]</td><td>ENUM</td><td>EN_POS_J_ARITHMETIK
(This input is only available for CODESYS V3 function blocks. CODESYS V2, the selection is always adjusted automatically)

With 'EN_POS_J_ARITHMETIK' the selection 32 bit or 64 bit arithmetic for position function blocks in position operation mode is set.

Options:
<table>
<tr><td>enAutomaticSelection</td><td>Automatic determination if 32 bit or 64 bit arithmetic (Default)
 
The default setting 'enAutomaticSelection' can only be used, if the PLC controller is on the first place in the CODESYS 'Device_ Config'. Is that not the case the selection has to be made manually by the user.</td></tr>
<tr><td>en64BitArithmetik</td><td>64 bit arithmetic</td></tr>
<tr><td>en32BitArithmetik</td><td>32 bit arithmetic</td></tr>
</table>
If the automatic determination does not work, the following diagnostic number is displayed and the selection must done manually by the user.
<table>
<tr><td>iErrID</td><td>5</td></tr>
</table>
Additional information about 'EN_POS_J_ARITHMETIK' and manual selection:
Siehe 'EN_POS_J_ARITHMETIK' auf Seite 601.</td><td></td><td>x</td></tr>
</table>

1) Only for CODESYS V3

Output variables

Name	Type	Description	Sync.	Async.
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled		x
boSBM	BOOL	System ready message		x
boDone	BOOL	Positioning done. Valid for 'boPosAbs', 'boPosRel' and 'boHome'		x
boErr	BOOL	The function block is in an error state FALSE: No error (permitted commanding or warning) TRUE: Error		x
iErrID	INT	Error identity number: Diagnostic number is output iErrID = 0: No error iErrID ≠ 0, boErr = TRUE: Error iErrID ≠ 0, boErr = FALSE: Warning		x
enErrName	ENUM	EN_FB_NAME Name of faulty block for which the diagnostic number is output. The diagnostic number is explained in the description of the corresponding library.		x
boPosBusy	BOOL	Positioning active		x
boJogBusy	BOOL	Jog mode active		x
boHomeBusy	BOOL	Home active		x
boSpeedBusy	BOOL	Speed control active		x
boFollowBusy	BOOL	Follow operation active		x

Input and output variables

Name	Type	Description	Sync.	Async.
stDevice	STRUCT	ST_DEVICE The device description structure assigns the block a device. (See document Software description AmkBase Bibliothek, Part no.204986)		x
stAxisDrive	STRUCT	ST_AXIS_DRIVE Siehe 'ST_AXIS_DRIVE (ST)' auf Seite 191.		x

Usage note in the CODESYS program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
STANDARD_AXIS	STANDARD_AXIS.actSync

History

Library	AmkAfl	
System versions	**Functionality**	**Target**
AS-C V3.01/0827	Basic functionality (1st version)	≥ AS_CP_200_0812_T202053
AS-PL15 V3.01/0804	Basic functionality (1st version)	≥ AS_V313_0946_T202724

4.1.4.2.14 STANDARD_FOLLOW_TABLE_ENGAGE_DISENG_AXIS (FB)

Expanded functionality:

- The slave axis continuously follows a freely selectable table with a phasing-in and phasing out table.

Siehe 'FOLLOW_AXIS_TABLE_ENGAGE_DISENGAGE (FB)' auf Seite 395.

The following basic functionalities for the axis are available:

Inputs/outputs on the function block

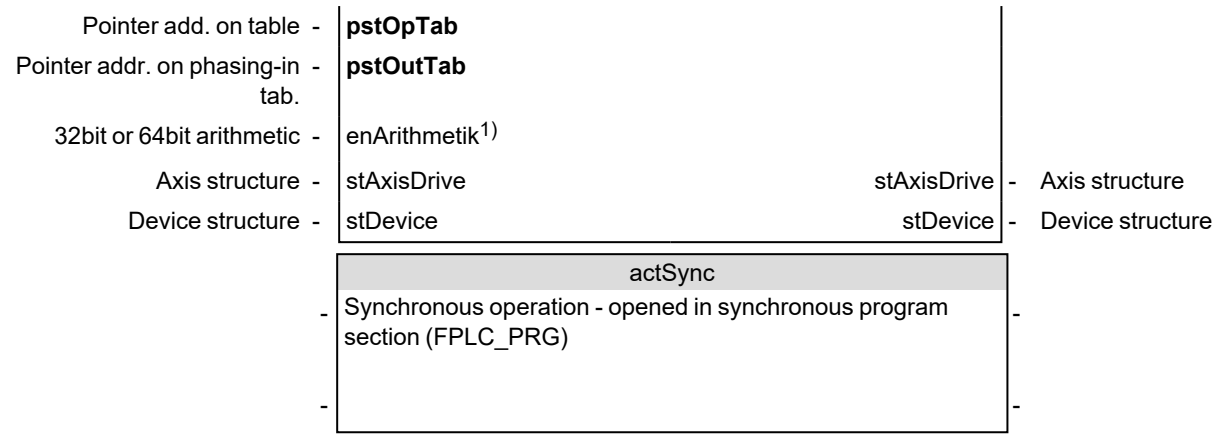
- Control bits (RF Controller enable, FL Clear error)
- Emergency stop
- Speed control
- Homing cycle
- Positioning (absolute/relative)
- Jog mode (minus/plus)
- Status signals of axis functions

Access via program code

- Configuration of status messages (ID26 'Configuration status bits')
- Reading of parameters according to a specifications list
- Actual position cached
- Control with feedforward (torque and speed)

User interface

		STD_FOLLOW_TABLE_ENGAGE_DISENG_AXIS			
FB enable	-	boEnable	boEnabAck	-	Ackn. "FB enable"
Controller enable	-	boRF	boSBM	-	System ready message
Clear error	-	boFL	boDone	-	Ackn. "FB done"
Emergency stop	-	boEmergency_Stop	boErr	-	Error
Start speed control	-	boSpeed	iErrID	-	Error ID
Start homing cycle	-	boHome	enErrName	-	Name of faulty FB
Start absolute pos.	-	boPosAbs	boPosBusy	-	Positioning active
Start relative pos.	-	boPosRel	boJogBusy	-	Jog mode active
Jog plus	-	boJogPlus	boHomeBusy	-	Homing drive active
Jog minus	-	boJogMinus	boSpeedBusy	-	Speed control active
Follow operation	-	**boFollowEnable**	boFollowEnableAck	-	Ackn. Follow operation
Control of tab. function	-	**boTableControl**	boTableBusy	-	Table editing active
Phasing in angle	-	**udInAngle**			
Phasing out angle	-	**udOutAngle**			
Positioning velocity	-	udPosVelocity			
Target position	-	diPosition			
Acceleration	-	lreAccel			
Deceleration	-	lreDecel			
Jerk limitation at start	-	lreJerkStart			
Jerk limitation at stop	-	lreJerkEnd			
Jog velocity	-	udVelocityJog			
Velocity for speed control	-	diSpeedVelocity			
Setpoint specification for follow operation	-	**diInVal**			
Pointer addr. on phasing-in tab.	-	**pstInTab**			



1) Only for CODESYS V3

Input variables

Name	Type	Description	Sync.	Async.
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.		x
boRF	BOOL	Bit for enabling the controller Relevant parameters: ID32796 'Source RF' Allowed: Code 5: 'Controller enable via EtherCAT' Code 25: RF 'via EtherCAT AND-linked with the binary input UE' Code 0: 'Controller enable (RF) via binary input' is not allowed Behavior by RF inactive in the operating mode position control with mode following (boFollow = TRUE) and active SAK (ID32801 Bit 9 = TRUE) In the operating mode position control with active SAK (following error compensation), the function deceleration ramp after RF inactive (see ID32782 'Deceleration ramp RF inactive') can not be used. When the controller enable (boRF = FALSE) is de-activated, the setpoint output value is immediately stopped and the speed is maximum decelerated. Workaround for applications with deceleration ramps: Use the PLC to generate the setpoint values for the deceleration ramp. Remove the controller enable signal only at standstill.		x
boFL	BOOL	Bit for error deletion		x

Name	Type	Description	Sync.	Async.
boEmergency_ Stop	BOOL	EMERGENCY STOP: The setpoint of the velocity is decreased to zero along the emergency-stop ramp. Once initiated, an emergency stop cannot be aborted. Relevant variables: EMERGENCY STOP ramp: Ramp time in which the drive is slowed down from the current velocity to zero. EMERGENCY STOP ramp: ST_AXIS_DRIVE.ST_IDS.ST_ID32919_ PlcList. diID32919_05_diEmergency_StopRamp Unit: ms Default value: 100 ms		x
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). Relevant parameters: ID32802 'AMK secondary operating mode 2' [Operating mode: Speed control] [Setpoint source: diMainSetpoint (code 41h)] [Ramps: active / inactive (bit 6)] ID32780 'Acceleration ramp' ID32781'Deceleration ramp' Relevant variables: Speed setpoint: diSpeedVelocity Velocity ramp: STANDARD_AXIS.fbVelocity.diVelocityRamp [default value: 100 ms] Speed control active: boSpeedBusy		x

Name	Type	Description	Sync.	Async.
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. Relevant parameters: ID41 'Homing velocity' ID147 'Homing parameter' ID150 'Homing offset 1' ID32926 'AMK homing cycle parameter' Relevant variables: Homing drive active: boHomeBusy Homing done: boDone Homing point known: stAxisDrive.stStatus.boID33036_RFP_known		x
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 [Relevant parameters: see 'boPosRel']		x
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. Relevant parameters: ID32801 'AMK secondary operating mode 1' [Operating mode: position control] [Setpoint source: diMainSetpoint (code 41h)] Relevant variables: Positioning velocity: udPosVelocity Target position: diPosition Acceleration: lreAccel Deceleration: lreDecel Jerk limitation (start): lreJerkStart Jerk limitation (end): lreJerkEnd Positioning active: boPosBusy Positioning done: boDone		x

Name	Type	Description	Sync.	Async.
boJogPlus	BOOL	Jog in positive direction Enable signal: With a positive edge, the jogging function starts (positive direction). Jog mode is run as long as 'boJobPlus' = TRUE. Use a negative edge 'boJogPlus' = FALSE to cancel the current jog mode. [Relevant parameters: see 'boJogMinus']		x
boJogMinus	BOOL	Jogging in negative direction Enable signal: With a positive edge, the jogging function starts (negative direction). Jog mode is run as long as 'boJobMinus' = TRUE. Use a negative edge 'boJogMinus' = FALSE to cancel the current jog mode. Relevant parameters: ID32801 'AMK secondary operating mode 1' [Operating mode: position control] [Setpoint source: diMainSetpoint (code 41h)] Relevant variables: Jog velocity: udVelocityJog Acceleration: lreAccel Deceleration: lreDecel Jerk limitation (start): lreJerkStart Jerk limitation (end): lreJerkEnd Jog mode active: boJogBusy		x
boFollowEnable	BOOL	Follow operation Enable signal: With a positive edge, the follow operation function starts. Follow operation is run as long as 'boFollowEnable' = TRUE. Use a negative edge 'boFollowEnable' = FALSE to cancel the ongoing follow operation. The axis follows the input increments and the transferred table. Relevant parameters: ID32801 'AMK secondary operating mode 1' [Operating mode: position control] [Setpoint source: diMainSetpoint (code 41h)] Relevant variables: Setpoint specification: diInVal Phasing-in range: udInVal Phasing-out range: udOutVal Table function control: boTableControl Pointer to phasing-in table: pstInTab Pointer to phasing-out table: pstOutTab Pointer to table: pstOpTab		x

Name	Type	Description	Sync.	Async.
boTableControl	BOOL	Table function control Use boTableControl to control the processing of operating tables. (Prerequisite: The follow operation is enable 'boFollowEnable' = TRUE and the master reference is generated). Enable signal: Use a positive edge at 'boTableControl' to launch the function; the phasing-in table is processed (prerequisite: diInVal within phasing-in range) At the end of the phasing in table, an automatic transition occurs to the operating table. As long as 'boTableControl' = TRUE, the operating table is processed. When 'boTableControl' = False, a switchover to the phasing-out table (prerequisite: diOutVal is within phasing-out range) is made at the end of the phasing-out table. Table processing automatically stops at the end of the phasing-out table. The master reference is not lost in the process.	x	
udInAngle	UDINT	Phasing in angle Maximum permissible position on the x axis up to which phasing in is permitted Unit: Increments Default value: 1000	x	
udOutAngle	UDINT	Phasing out angle Maximum permissible position on the x axis up to which phasing out is permitted Unit: Increments Default value: 1000	x	
udPosVelocity	UDINT	Positioning velocity Unit: incr/s Default value: 34133		x
diPosition	DINT	Target position Use the inputs 'boPosAbs' and 'boPosRel' to determine whether the target position is approached absolutely or relatively. Unit: Increments Default value: 30720		x
lreAccel	LREAL	Acceleration for positioning and jog mode Range: $1{,}43\ 10^{-13} < \lvert\text{'lreAccel'}\rvert < 1{,}43\ 10^{+16}$ Unit: incr/s^2 Default: 1000000		x
lreDecel	LREAL	Deceleration for positioning and jog mode Range: $1{,}43\ 10^{-13} < \lvert\text{'lreDecel'}\rvert < 1{,}43\ 10^{+16}$ Unit: incr/s^2 Default: 1000000		x
lreJerkStart	LREAL	Jerk limitation (start) for positioning and jog mode Unit: incr/s^3 Default: 10000000		x
lreJerkEnd	LREAL	Jerk limitation (end) for positioning and jog mode Unit: incr/s^3 Default: 10000000		x

<table>
<tr><th>Name</th><th>Type</th><th>Description</th><th>Sync.</th><th>Async.</th></tr>
<tr><td>udVelocityJog</td><td>UDINT</td><td>Jog velocity
Unit: incr/s
Default value: 34133</td><td></td><td>x</td></tr>
<tr><td>diSpeedVelocity</td><td>DINT</td><td>Speed setpoint for speed control
Unit: rpm
Default value: 100</td><td></td><td>x</td></tr>
<tr><td>diInVal</td><td>DINT</td><td>Setpoint specification for follow operation
(e.g., through master axis, FB VGEN)
Unit: Increments</td><td>x</td><td></td></tr>
<tr><td>pstInTab</td><td>POINTER</td><td>POINTER TO ST_PROF_TAB
Pointer to phasing-in table.
(See document Software description AmkBase Bibliothek, Part no. 204986)</td><td>x</td><td></td></tr>
<tr><td>pstOpTab</td><td>POINTER</td><td>POINTER TO ST_PROF_TAB
Point to operating table.
(See document Software description AmkBase Bibliothek, Part no. 204986)</td><td>x</td><td></td></tr>
<tr><td>pstOutTab</td><td>POINTER</td><td>POINTER TO ST_PROF_TAB
Pointer to phasing-out table.
(See document Software description AmkBase Bibliothek, Part no. 204986)</td><td>x</td><td></td></tr>
<tr><td>enArithmetik[1)]</td><td>ENUM</td><td>EN_POS_J_ARITHMETIK
(This input is only available for CODESYS V3 function blocks. CODESYS V2, the selection is always adjusted automatically)

With 'EN_POS_J_ARITHMETIK' the selection 32 bit or 64 bit arithmetic for position function blocks in position operation mode is set.

Options:
<table><tr><td>enAutomaticSelection</td><td>Automatic determination if 32 bit or 64 bit arithmetic (Default)
 
The default setting 'enAutomaticSelection' can only be used, if the PLC controller is on the first place in the CODESYS 'Device_ Config'. Is that not the case the selection has to be made manually by the user.</td></tr><tr><td>en64BitArithmetik</td><td>64 bit arithmetic</td></tr><tr><td>en32BitArithmetik</td><td>32 bit arithmetic</td></tr></table>
If the automatic determination does not work, the following diagnostic number is displayed and the selection must done manually by the user.
<table><tr><td>iErrID</td><td>5</td></tr></table>
Additional information about 'EN_POS_J_ARITHMETIK' and manual selection:
Siehe 'EN_POS_J_ARITHMETIK' auf Seite 601.</td><td></td><td>x</td></tr>
</table>

1) Only for CODESYS V3

Output variables

Name	Type	Description	Sync.	Async.
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled		x
boSBM	BOOL	System ready message		x
boDone	BOOL	Positioning done. Valid for 'boPosAbs', 'boPosRel' and 'boHome'		x
boErr	BOOL	The function block is in an error state FALSE: No error (permitted commanding or warning) TRUE: Error		x
iErrID	INT	Error identity number: Diagnostic number is output iErrID = 0: No error iErrID ≠ 0, boErr = TRUE: Error iErrID ≠ 0, boErr = FALSE: Warning		x
enErrName	ENUM	EN_FB_NAME Name of faulty block for which the diagnostic number is output. The diagnostic number is explained in the description of the corresponding library.		x
boPosBusy	BOOL	Positioning active		x
boJogBusy	BOOL	Jog mode active		x
boHomeBusy	BOOL	Home active		x
boSpeedBusy	BOOL	Speed control active		x
boFollowEnableAck	BOOL	Acknowledgement: Follow operation		x
boTableBusy	BOOL	Table editing active	x	

Input and output variables

Name	Type	Description	Sync.	Async.
stDevice	STRUCT	ST_DEVICE The device description structure assigns the block a device. (See document Software description AmkBase Bibliothek, Part no.204986)		x
stAxisDrive	STRUCT	ST_AXIS_DRIVE Siehe 'ST_AXIS_DRIVE (ST)' auf Seite 191.		x

Usage note in the CODESYS program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
STANDARD_AXIS	STANDARD_AXIS.actSync

History

Library	AmkAfl	
System versions	**Functionality**	**Target**
AS-C V3.01/0827	Basic functionality (1st version)	≥ AS_CP_200_0812_T202053
AS-PL15 V3.01/0804	Basic functionality (1st version)	≥ AS_V313_0946_T202724

4.1.4.2.15 STANDARD_FOLLOW_TABLE_START_AUTOSTOP_AXIS (FB)

Expanded functionality:

- The slave axis continuously follows a freely selectable table with a phasing-in and phasing out table.

Siehe 'FOLLOW_AXIS_TABLE_START_AUTOSTOP (FB)' auf Seite 397.

The following basic functionalities for the axis are available:

Inputs/outputs on the function block

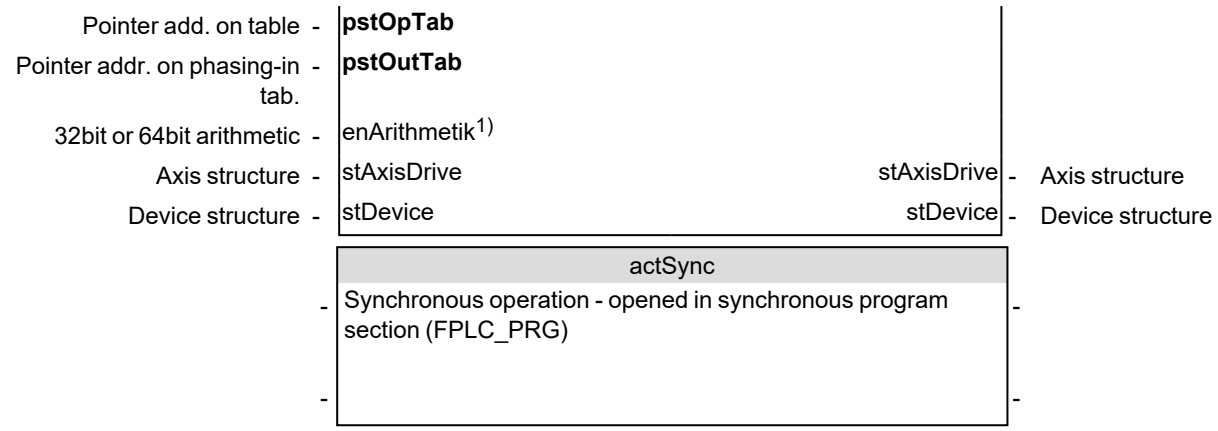
- Control bits (RF Controller enable, FL Clear error)
- Emergency stop
- Speed control
- Homing cycle
- Positioning (absolute/relative)
- Jog mode (minus/plus)
- Status signals of axis functions

Access via program code

- Configuration of status messages (ID26 'Configuration status bits')
- Reading of parameters according to a specifications list
- Actual position cached
- Control with feedforward (torque and speed)

User interface

	STD_FOLLOW_TABLE_START_AUTOSTOP_AXIS			
FB enable -	boEnable	boEnabAck	-	Ackn. "FB enable"
Controller enable -	boRF	boSBM	-	System ready message
Clear error -	boFL	boDone	-	Ackn. "FB done"
Emergency stop -	boEmergency_Stop	boErr	-	Error
Start speed control -	boSpeed	iErrID	-	Error ID
Start homing cycle -	boHome	enErrName	-	Name of faulty FB
Start absolute pos. -	boPosAbs	boPosBusy	-	Positioning active
Start relative pos. -	boPosRel	boJogBusy	-	Jog mode active
Jog plus -	boJogPlus	boHomeBusy	-	Homing drive active
Jog minus -	boJogMinus	boSpeedBusy	-	Speed control active
Follow operation -	**boFollowEnable**	**boFollowEnabAck**	-	Ackn. Follow operation
Control of tab. function -	**boTableControl**	**boTableBusy**	-	Table editing active
Phasing in angle -	**udInAngle**			
Number of table operations -	**uiOpNo**			
Positioning velocity -	udPosVelocity			
Target position -	diPosition			
Acceleration -	lreAccel			
Deceleration -	lreDecel			
Jerk limitation at start -	lreJerkStart			
Jerk limitation at stop -	lreJerkEnd			
Jog velocity -	udVelocityJog			
Velocity for speed control -	diSpeedVelocity			
Setpoint specification for follow operation -	**diInVal**			
Pointer addr. on phasing-in tab. -	**pstInTab**			



1) Only for CODESYS V3

Input variables

Name	Type	Description	Sync.	Async.
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.		x
boRF	BOOL	Bit for enabling the controller Relevant parameters: ID32796 'Source RF' Allowed: Code 5: 'Controller enable via EtherCAT' Code 25: RF 'via EtherCAT AND-linked with the binary input UE' Code 0: 'Controller enable (RF) via binary input' is not allowed Behavior by RF inactive in the operating mode position control with mode following (boFollow = TRUE) and active SAK (ID32801 Bit 9 = TRUE) In the operating mode position control with active SAK (following error compensation), the function deceleration ramp after RF inactive (see ID32782 'Deceleration ramp RF inactive') can not be used. When the controller enable (boRF = FALSE) is de-activated, the setpoint output value is immediately stopped and the speed is maximum decelerated. Workaround for applications with deceleration ramps: Use the PLC to generate the setpoint values for the deceleration ramp. Remove the controller enable signal only at standstill.		x
boFL	BOOL	Bit for error deletion		x

Name	Type	Description	Sync.	Async.
boEmergency_ Stop	BOOL	EMERGENCY STOP: The setpoint of the velocity is decreased to zero along the emergency-stop ramp. Once initiated, an emergency stop cannot be aborted. Relevant variables: EMERGENCY STOP ramp: Ramp time in which the drive is slowed down from the current velocity to zero. EMERGENCY STOP ramp: ST_AXIS_DRIVE.ST_IDS.ST_ID32919_ PlcList. diID32919_05_diEmergency_StopRamp Unit: ms Default value: 100 ms		x
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). Relevant parameters: ID32802 'AMK secondary operating mode 2' [Operating mode: Speed control] [Setpoint source: diMainSetpoint (code 41h)] [Ramps: active / inactive (bit 6)] ID32780 'Acceleration ramp' ID32781'Deceleration ramp' Relevant variables: Speed setpoint: diSpeedVelocity Velocity ramp: STANDARD_AXIS.fbVelocity.diVelocityRamp [default value: 100 ms] Speed control active: boSpeedBusy		x

Name	Type	Description	Sync.	Async.
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. Relevant parameters: ID41 'Homing velocity' ID147 'Homing parameter' ID150 'Homing offset 1' ID32926 'AMK homing cycle parameter' Relevant variables: Homing drive active: boHomeBusy Homing done: boDone Homing point known: stAxisDrive.stStatus.boID33036_RFP_known		x
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 [Relevant parameters: see 'boPosRel']		x
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. Relevant parameters: ID32801 'AMK secondary operating mode 1' [Operating mode: position control] [Setpoint source: diMainSetpoint (code 41h)] Relevant variables: Positioning velocity: udPosVelocity Target position: diPosition Acceleration: lreAccel Deceleration: lreDecel Jerk limitation (start): lreJerkStart Jerk limitation (end): lreJerkEnd Positioning active: boPosBusy Positioning done: boDone		x

Name	Type	Description	Sync.	Async.
boJogPlus	BOOL	Jog in positive direction Enable signal: With a positive edge, the jogging function starts (positive direction). Jog mode is run as long as 'boJobPlus' = TRUE. Use a negative edge 'boJogPlus' = FALSE to cancel the current jog mode. [Relevant parameters: see 'boJogMinus']		x
boJogMinus	BOOL	Jogging in negative direction Enable signal: With a positive edge, the jogging function starts (negative direction). Jog mode is run as long as 'boJobMinus' = TRUE. Use a negative edge 'boJogMinus' = FALSE to cancel the current jog mode. Relevant parameters: ID32801 'AMK secondary operating mode 1' [Operating mode: position control] [Setpoint source: diMainSetpoint (code 41h)] Relevant variables: Jog velocity: udVelocityJog Acceleration: lreAccel Deceleration: lreDecel Jerk limitation (start): lreJerkStart Jerk limitation (end): lreJerkEnd Jog mode active: boJogBusy		x
boFollowEnable	BOOL	Follow operation Enable signal: With a positive edge, the follow operation function starts. Follow operation is run as long as 'boFollowEnable' = TRUE. Use a negative edge 'boFollowEnable' = FALSE to cancel the ongoing follow operation. The axis follows the input increments and the transferred table. Relevant parameters: ID32801 'AMK secondary operating mode 1' [Operating mode: position control] [Setpoint source: diMainSetpoint (code 41h)] Relevant variables: Setpoint specification: diInVal Phasing-in range: udInVal Phasing-out range: udOutVal Table function control: boTableControl Pointer to phasing-in table: pstInTab Pointer to phasing-out table: pstOutTab Pointer to table: pstOpTab		x

Name	Type	Description	Sync.	Async.
boTableControl	BOOL	Table function control Use boTableControl to control the processing of operating tables. (Prerequisite: The follow operation is enable 'boFollowEnable' = TRUE and the master reference is generated). Enable signal: Use a positive edge at 'boTableControl' to launch the function; the phasing-in table is processed (prerequisite: diInVal within phasing-in range) At the end of the phasing in table, an automatic transition occurs to the operating table. As long as 'boTableControl' = TRUE, the operating table is processed. When 'boTableControl' = False, a switchover to the phasing-out table (prerequisite: diOutVal is within phasing-out range) is made at the end of the phasing-out table. Table processing automatically stops at the end of the phasing-out table. The master reference is not lost in the process.	x	
udInAngle	UDINT	Phasing in angle Maximum permissible position on the x axis up to which phasing in is permitted Unit: Increments Default value: 1000		x
uiOpNo	UINT	Number of operating table cycles to be processed [1]		x
udPosVelocity	UDINT	Positioning velocity Unit: incr/s Default value: 34133		x
diPosition	DINT	Target position Use the inputs 'boPosAbs' and 'boPosRel' to determine whether the target position is approached absolutely or relatively. Unit: Increments Default value: 30720		x
lreAccel	LREAL	Acceleration for positioning and jog mode Range: $1{,}43\ 10^{-13} < \lvert\text{'lreAccel'}\rvert < 1{,}43\ 10^{+16}$ Unit: incr/s^2 Default: 1000000		x
lreDecel	LREAL	Deceleration for positioning and jog mode Range: $1{,}43\ 10^{-13} < \lvert\text{'lreDecel'}\rvert < 1{,}43\ 10^{+16}$ Unit: incr/s^2 Default: 1000000		x
lreJerkStart	LREAL	Jerk limitation (start) for positioning and jog mode Unit: incr/s^3 Default: 10000000		x
lreJerkEnd	LREAL	Jerk limitation (end) for positioning and jog mode Unit: incr/s^3 Default: 10000000		x
udVelocityJog	UDINT	Jog velocity Unit: incr/s Default value: 34133		x

<table>
<tr><th>Name</th><th>Type</th><th>Description</th><th>Sync.</th><th>Async.</th></tr>
<tr><td>diSpeedVelocity</td><td>DINT</td><td>Speed setpoint for speed control
Unit: rpm
Default value: 100</td><td></td><td>x</td></tr>
<tr><td>diInVal</td><td>DINT</td><td>Setpoint specification for follow operation
(e.g., through master axis, FB VGEN)
Unit: Increments</td><td>x</td><td></td></tr>
<tr><td>pstInTab</td><td>POINTER</td><td>POINTER TO ST_PROF_TAB
Pointer to phasing-in table.
(See document Software description AmkBase Bibliothek, Part no. 204986)</td><td></td><td>x</td></tr>
<tr><td>pstOpTab</td><td>POINTER</td><td>POINTER TO ST_PROF_TAB
Point to operating table.
(See document Software description AmkBase Bibliothek, Part no. 204986)</td><td></td><td>x</td></tr>
<tr><td>pstOutTab</td><td>POINTER</td><td>POINTER TO ST_PROF_TAB
Pointer to phasing-out table.
(See document Software description AmkBase Bibliothek, Part no. 204986)</td><td></td><td>x</td></tr>
<tr><td>enArithmetik[1)]</td><td>ENUM</td><td>EN_POS_J_ARITHMETIK
(This input is only available for CODESYS V3 function blocks. CODESYS V2, the selection is always adjusted automatically)

With 'EN_POS_J_ARITHMETIK' the selection 32 bit or 64 bit arithmetic for position function blocks in position operation mode is set.

Options:<table><tr><td>enAutomaticSelection</td><td>Automatic determination if 32 bit or 64 bit arithmetic (Default)
 
The default setting 'enAutomaticSelection' can only be used, if the PLC controller is on the first place in the CODESYS 'Device_ Config'. Is that not the case the selection has to be made manually by the user.</td></tr><tr><td>en64BitArithmetik</td><td>64 bit arithmetic</td></tr><tr><td>en32BitArithmetik</td><td>32 bit arithmetic</td></tr></table>If the automatic determination does not work, the following diagnostic number is displayed and the selection must done manually by the user.<table><tr><td>iErrID</td><td>5</td></tr></table>Additional information about 'EN_POS_J_ARITHMETIK' and manual selection:
Siehe 'EN_POS_J_ARITHMETIK' auf Seite 601.</td><td></td><td>x</td></tr>
</table>

1) Only for CODESYS V3

Output variables

Name	Type	Description	Sync.	Async.
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled		x
boSBM	BOOL	System ready message		x
boDone	BOOL	Positioning done. Valid for 'boPosAbs', 'boPosRel' and 'boHome'		x
boErr	BOOL	The function block is in an error state FALSE: No error (permitted commanding or warning) TRUE: Error		x
iErrID	INT	Error identity number: Diagnostic number is output iErrID = 0: No error iErrID ≠ 0, boErr = TRUE: Error iErrID ≠ 0, boErr = FALSE: Warning		x
enErrName	ENUM	EN_FB_NAME Name of faulty block for which the diagnostic number is output. The diagnostic number is explained in the description of the corresponding library.		x
boPosBusy	BOOL	Positioning active		x
boJogBusy	BOOL	Jog mode active		x
boHomeBusy	BOOL	Home active		x
boSpeedBusy	BOOL	Speed control active		x
boFollowEnableAck	BOOL	Acknowledgement: Follow operation	x	
boTableBusy	BOOL	Table editing active	x	

Input and output variables

Name	Type	Description	Sync.	Async.
stDevice	STRUCT	ST_DEVICE The device description structure assigns the block a device. (See document Software description AmkBase Bibliothek, Part no.204986)		x
stAxisDrive	STRUCT	ST_AXIS_DRIVE Siehe 'ST_AXIS_DRIVE (ST)' auf Seite 191.		x

Usage note in the CODESYS program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
STANDARD_FOLLOW_TABLE_START_AUTOSTOP_AXIS	STANDARD_FOLLOW_TABLE_START_AUTOSTOP_AXIS.actSync

History

Library	AmkAfl	
System versions	**Functionality**	**Target**
AS-C V3.01/0827	Basic functionality (1st version)	≥ AS_CP_200_0812_T202053
AS-PL15 V3.01/0804	Basic functionality (1st version)	≥ AS_V313_0946_T202724

4.1.4.2.16 STANDARD_WINDER_DANCER_AXIS (FB)

Expanded functionality:

The function block 'STANDARD_WINDER_DANCER_AXIS' realizes a dancer winder with the following characteristics with following features:

- PID regulated dancer winder
- Operation as unwinder and rewinder
- Diameter calculation
- Search diameter at the start of the winder

Siehe 'WINDER_DANCER_CONTROL (FB)' auf Seite 540.

The following basic functionalities for the axis are available:

Inputs/outputs on the function block

- Control bits (RF Controller enable, FL Clear error)
- Emergency stop
- Speed control
- Homing cycle
- Positioning (absolute/relative)
- Jog mode (minus/plus)
- Status signals of axis functions

Access via program code

- Configuration of status messages (ID26 'Configuration status bits')
- Reading of parameters according to a specifications list
- Actual position cached
- Control with feedforward (torque and speed)

Setpoints and current values are transferred in a synchronous action. The synchronous action must be called in the synchronous program level FPLC_PRG.

User interface

		STANDARD_WINDER_DANCER_AXIS			
FB enable	-	boEnable	boEnabAck	-	Ackn. "FB enable"
Controller enable	-	boRF	boSBM	-	System ready message
Clear error	-	boFL	boDone	-	Ackn. "FB done"
Emergency stop	-	boEmergency_Stop	boErr	-	Error
Start speed control	-	boSpeed	iErrID	-	Error ID
Start homing cycle	-	boHome	enErrName	-	Name of faulty FB
Start absolute pos.	-	boPosAbs	boPosBusy	-	Positioning active
Start relative pos.	-	boPosRel	boJogBusy	-	Jog mode active
Jog plus	-	boJogPlus	boHomeBusy	-	Homing drive active
Jog minus	-	boJogMinus	boSpeedBusy	-	Speed control active
Positioning velocity	-	udPosVelocity	**boEnabAcknWinder**	-	Winder initialised and enabled
Target position	-	diPosition	**boDmOk**	-	Diameter ok, status "known"
Acceleration	-	lreAccel	**boDmLimit**	-	Diameter was limited
Deceleration	-	lreDecel	**iActDm**	-	Actual value diameter
Jerk limit at start	-	lreJerkStart	**diVWebSpeed**	-	Actual value web speed
Jerk limit at stop	-	lreJerkEnd	**diSetValMotorSpeed**	-	Speed setpoint motor
Jog velocity	-	udVelocityJog			
Velocity for speed control	-	diSpeedVelocity			
Enable winder	-	**boEnableWinder**			
Activation of the diameter calculation	-	**boDmCalc**			
Dancer in middle position	-	**boDncMidPos**			
Dancer mode "Follow actual value"	-	**boDncTraceMod**			
Activate gain ramp	-	**boRaiseGain**			
Activate output web speed	-	**boOutputSpeed**			
Set diameter, status "known"	-	**boDmOkPreset**			
Set diameter, status "unknown"	-	**boDmNokPreset**			
Set diameter status of "known" → "Unknown"	-	**boDmReset**			
Set output 'boDmOk'	-	**boSetDmOk**			
Dancer controller delete i component	-	**boClearI**			
Diameter preset value	-	**iDmPreset**			
Actual reel position	-	**diActReelPos**			
Actual web position	-	**diActWebPos**			
Actual value dancer position	-	**iActDncPos**			
32bit or 64bit arithmetic	-	enArithmetik[1)]			
Axis structure	-	stAxisDrive	stAxisDrive	-	Axis structure
Device structure	-	stDevice	stDevice	-	Device structure
Configuration structure winder	-	**stWinderConfig**	**stWinderConfig**	-	Configuration structure winder

actSync
Synchronous operation - opened in synchronous program section (FPLC_PRG)

1) Only for CODESYS V3

Input variables

Name	Type	Description	Sync.	Async.
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.		x
boRF	BOOL	Bit for enabling the controller Relevant parameters: ID32796 'Source RF' Allowed: Code 5: 'Controller enable via EtherCAT' Code 25: RF 'via EtherCAT AND-linked with the binary input UE' Code 0: 'Controller enable (RF) via binary input' is not allowed		x
boFL	BOOL	Bit for error deletion		x
boEmergency_ Stop	BOOL	EMERGENCY STOP: The setpoint of the velocity is decreased to zero along the emergency-stop ramp. Once initiated, an emergency stop cannot be aborted. Relevant variables: EMERGENCY STOP ramp: Ramp time in which the drive is slowed down from the current velocity to zero. EMERGENCY STOP ramp: ST_AXIS_DRIVE.ST_IDS.ST_ID32919_ PlcList. diID32919_05_diEmergency_StopRamp Unit: ms Default value: 100 ms		x

Name	Type	Description	Sync.	Async.
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). Relevant parameters: ID32802 'AMK secondary operating mode 2' [Operating mode: Speed control] [Setpoint source: diMainSetpoint (code 41h)] [Ramps: active / inactive (bit 6)] ID32780 'Acceleration ramp' ID32781'Deceleration ramp' Relevant variables: Speed setpoint: diSpeedVelocity Velocity ramp: STANDARD_AXIS.fbVelocity.diVelocityRamp [default value: 100 ms] Speed control active: boSpeedBusy		x
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. Relevant parameters: ID41 'Homing velocity' ID147 'Homing parameter' ID150 'Homing offset 1' ID32926 'AMK homing cycle parameter' Relevant variables: Homing drive active: boHomeBusy Homing done: boDone Homing point known: stAxisDrive.stStatus.boID33036_RFP_known		x

Name	Type	Description	Sync.	Async.
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 [Relevant parameters: see 'boPosRel']		x
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. Relevant parameters: ID32801 'AMK secondary operating mode 1' [Operating mode: position control] [Setpoint source: diMainSetpoint (code 41h)] Relevant variables: Positioning velocity: udPosVelocity Target position: diPosition Acceleration: lreAccel Deceleration: lreDecel Jerk limitation (start): lreJerkStart Jerk limitation (end): lreJerkEnd Positioning active: boPosBusy Positioning done: boDone		x
boJogPlus	BOOL	Jog in positive direction Enable signal: With a positive edge, the jogging function starts (positive direction). Jog mode is run as long as 'boJobPlus' = TRUE. Use a negative edge 'boJogPlus' = FALSE to cancel the current jog mode. [Relevant parameters: see 'boJogMinus']		x

Name	Type	Description	Sync.	Async.
boJogMinus	BOOL	Jogging in negative direction Enable signal: With a positive edge, the jogging function starts (negative direction). Jog mode is run as long as 'boJobMinus' = TRUE. Use a negative edge 'boJogMinus' = FALSE to cancel the current jog mode. Relevant parameters: ID32801 'AMK secondary operating mode 1' [Operating mode: position control] [Setpoint source: diMainSetpoint (code 41h)] Relevant variables: Jog velocity: udVelocityJog Acceleration: lreAccel Deceleration: lreDecel Jerk limitation (start): lreJerkStart Jerk limitation (end): lreJerkEnd Jog mode active: boJogBusy		x
udPosVelocity	UDINT	Positioning velocity Unit: incr/s Default value: 34133		x
diPosition	DINT	Target position Use the inputs 'boPosAbs' and 'boPosRel' to determine whether the target position is approached absolutely or relatively. Unit: Increments Default value: 30720		x
lreAccel	LREAL	Acceleration for positioning and jog mode Range: $1{,}43\ 10^{-13} < \lvert\text{'lreAccel'}\rvert < 1{,}43\ 10^{+16}$ Unit: incr/s^2 Default: 1000000		x
lreDecel	LREAL	Deceleration for positioning and jog mode Range: $1{,}43\ 10^{-13} < \lvert\text{'lreDecel'}\rvert < 1{,}43\ 10^{+16}$ Unit: incr/s^2 Default: 1000000		x
lreJerkStart	LREAL	Jerk limitation (start) for positioning and jog mode Unit: incr/s^3 Default: 10000000		x
lreJerkEnd	LREAL	Jerk limitation (end) for positioning and jog mode Unit: incr/s^3 Default: 10000000		x
udVelocityJog	UDINT	Jog velocity Unit: incr/s Default value: 34133		x
diSpeedVelocity	DINT	Speed setpoint for speed control Unit: rpm Default value: 100		x

Name	Type	Description	Sync.	Async.
boEnableWinder	BOOL	Enable signal: With a positive edge, the initialisation of the winder block starts. As long as 'boEnableWinder' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnableWinder' = FALSE the block is no longer enabled and is thus no longer processed.		x
boDmCalc	BOOL	Activation of the diameter calculation		x
boDncMidPos	BOOL	Dancers move to the middle position		x
boDncTraceMod	BOOL	Dancer mode Follow actual value, the actual value of the dancer is accepted as setpoint at the PID controller.  Only enabled when 'boDncMidPos' = FALSE		x
boRaiseGain	BOOL	Position dependent dancers gain ramp active		x
boOutputSpeed	BOOL	Output web speed as the speed setpoint for motor		x
boDmOkPreset	BOOL	Diameter value of input variable 'iDmPreset' set. The input bit 'boDmPreset' is automatically reset after set. Diameter status "known": The diameter value is interpreted as valid diameter, that is, acceleration and friction torques are calculated.		x
boDmNokPreset	BOOL	Diameter value of input variable 'iDmPreset' set. The input bit 'boDmPreset' is automatically reset after set. Diameter status "unknown": The diameter value is interpreted as invalid diameter, that is, acceleration and friction torques are calculated.		x
boDmReset	BOOL	Reset output 'boDmOk', → diameter status "unknown"		x
boSetDmOk	BOOL	Set output 'boDmOk', → diameter status "known"		x
boClearI	BOOL	Dancer controller Delete i component		x
iDmPreset	INT	Diameter preset value Value to which the diameter with 'boDmNokPreset' = TRUE or 'boDmOkPreset' =TRUE is set.[mm]		x
diActReelPos	DINT	Actual reel position [increments]	x	
diActWebPos	DINT	Actual web position [increments]	x	
iActDncPos	INT	Actual value dancer position [mV]	x	

<table>
<tr><th>Name</th><th>Type</th><th>Description</th><th>Sync.</th><th>Async.</th></tr>
<tr><td>enArithmetik[1)]</td><td>ENUM</td><td>EN_POS_J_ARITHMETIK
(This input is only available for CODESYS V3 function blocks.
CODESYS V2, the selection is always adjusted automatically)

With 'EN_POS_J_ARITHMETIK' the selection 32 bit or 64 bit arithmetic for position function blocks in position operation mode is set.

Options:
<table><tr><td>enAutomaticSelection</td><td>Automatic determination if 32 bit or 64 bit arithmetic (Default)
 
The default setting 'enAutomaticSelection' can only be used, if the PLC controller is on the first place in the CODESYS 'Device_ Config'. Is that not the case the selection has to be made manually by the user.</td></tr><tr><td>en64BitArithmetik</td><td>64 bit arithmetic</td></tr><tr><td>en32BitArithmetik</td><td>32 bit arithmetic</td></tr></table>
If the automatic determination does not work, the following diagnostic number is displayed and the selection must done manually by the user.
<table><tr><td>iErrID</td><td>5</td></tr></table>
Additional information about 'EN_POS_J_ARITHMETIK' and manual selection:
Siehe 'EN_POS_J_ARITHMETIK' auf Seite 601.</td><td></td><td>x</td></tr>
</table>

1) Only for CODESYS V3

Output variables

Name	Type	Description	Sync.	Async.
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled		x
boSBM	BOOL	System ready message		x
boDone	BOOL	Positioning done. Valid for 'boPosAbs', 'boPosRel' and 'boHome'		x
boErr	BOOL	The function block is in an error state FALSE: No error (permitted commanding or warning) TRUE: Error		x
iErrID	INT	Error identity number: Diagnostic number is output iErrID = 0: No error iErrID ≠ 0, boErr = TRUE: Error iErrID ≠ 0, boErr = FALSE: Warning Bus error when 'boErr' = TRUE: EtherCAT or CAN are not in operational mode.	x	
enErrName	ENUM	EN_FB_NAME Name of faulty block for which the diagnostic number is output. The diagnostic number is explained in the description of the corresponding library.		x
boPosBusy	BOOL	Positioning active		x
boJogBusy	BOOL	Jog mode active		x
boHomeBusy	BOOL	Home active		x
boSpeedBusy	BOOL	Speed control active		x
boEnabAcknWinder	BOOL	Acknowledgement: Winder is initialised and enabled		x
boDmOk	BOOL	Known diameter		x
boDmLimit	BOOL	Diameter was calculated, and the result was outside of 'ST_WINDER_DANCER_CONTROL_CONFIG.iDmMin' and 'ST_WINDER_DANCER_CONTROL_CONFIG.iDmMax', the value was limited accordingly		x
iActDm	INT	Actual diameter [mm]		x
diVWebSpeed	DINT	Actual value web speed [mm/s]	x	
diSetValMotorSpeed	DINT	Speed setpoint motor [0.0001 r/min]	x	

Input and output variables

Name	Type	Description	Sync.	Async.
stDevice	STRUCT	ST_DEVICE The device description structure assigns the block a device. (See document Software description AmkBase Bibliothek, Part no.204986)		x
stAxisDrive	STRUCT	ST_AXIS_DRIVE Siehe 'ST_AXIS_DRIVE (ST)' auf Seite 191.		x
stWinderConfig	STRUCT	ST_WINDER_DANCER_CONTROL_CONFIG Configuration structure of the winder (See document Software description AFL - AMK Function Library, part 2, Part no.203694)		x

Usage note in the CODESYS program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
STANDARD_WINDER_DANCER_AXIS	STANDARD_WINDER_DANCER_AXIS.actSync

History

Library	AmkAfl	
System versions	**Functionality**	**Target**
AS-C V3.01/0827	Basic functionality (1st version)	≥ AS_CP_200_0812_T202053
AS-PL15 V3.01/0804	Basic functionality (1st version)	≥ AS_V313_0946_T202724

4.1.4.2.17 STANDARD_WINDER_TORQUE_AXIS (FB)

Expanded functionality:

- The function block 'STANDARD_WINDER_TORQUE_AXIS' realizes a sensorless rewinder and unwinder with following features:
- Torque controlled center winder
- Operation as rewinder and unwinder
- Diameter calculation
- Search diameter at the start of the winder
- Determine the reel of inertia of unknown diameter
- Synchronize onto a running web

Siehe 'WINDER_TORQUE_CONTROL (FB)' auf Seite 550.

The following basic functionalities for the axis are available:

Inputs/outputs on the function block

- Control bits (RF Controller enable, FL Clear error)
- Emergency stop
- Speed control
- Homing cycle
- Positioning (absolute/relative)
- Jog mode (minus/plus)
- Status signals of axis functions

Access via program code

- Configuration of status messages (ID26 'Configuration status bits')
- Reading of parameters according to a specifications list
- Actual position cached
- Control with feedforward (torque and speed)

Setpoints and current values are transferred in a synchronous action. The synchronous action must be called in the synchronous program level FPLC_PRG.

User interface

		STANDARD_WINDER_TORQUE_AXIS			
FB enable	-	boEnable	boEnabAck	-	Ackn. "FB enable"
Controller enable	-	boRF	boSBM	-	System ready message
Clear error	-	boFL	boDone	-	Ackn. "FB done"
Emergency stop	-	boEmergency_Stop	boErr	-	Error
Start speed control	-	boSpeed	iErrID	-	Error ID
Start homing cycle	-	boHome	enErrName	-	Name of faulty FB
Start absolute pos.	-	boPosAbs	boPosBusy	-	Positioning active
Start relative pos.	-	boPosRel	boJogBusy	-	Jog mode active
Jog plus	-	boJogPlus	boHomeBusy	-	Homing drive active
Jog minus	-	boJogMinus	boSpeedBusy	-	Speed control active
Positioning velocity	-	udPosVelocity	**boEnabAcknWinder**	-	Winder initialised and enabled
Target position	-	diPosition	**boDmOk**	-	Diameter ok, status "known"
Acceleration	-	lreAccel	**boFindInertiaDone**	-	Inertia moment determined
Deceleration	-	lreDecel	**boWindRuns**	-	Wind runs
Jerk limit at start	-	lreJerkStart	**boSyncRuns**	-	Synchronize runs
Jerk limit at stop	-	lreJerkEnd	**boDmLimit**	-	Diameter was limited
Jog velocity	-	udVelocityJog	**iActDm**	-	Actual value diameter
Velocity for speed control	-	diSpeedVelocity	**diActValWebSpeed**	-	Actual value web speed
Enable winder	-	**boEnableWinder**	**diActMotorSpeedAbs**	-	Actual motor speed absolute
Activation of the diameter calculation	-	**boDmCalc**	**diActReelPosAbs**	-	Actual reel position absolute
Determination the inertia moment of the reel	-	**boFindInertia**			
Wind on	-	**boWind**			
Synchronize on	-	**boSync**			
Set diameter	-	**boDmPreset**			
Set diameter, status "known"	-	**boDmOkPreset**			
Set diameter, status "unknown"	-	**boDmNOkPreset**			
Set diameter status of "known" → "Unknown"	-	**boDmReset**			
Set output 'boDmOk'	-	**boSetDmOk**			
Setpoint web tension	-	**diWebTension**			
Diameter preset value	-	**iDmPreset**			
Actual speed motor	-	**diActMotorSpeed**			
Actual reel position	-	**diActReelPos**			
Actual web position	-	**diActWebPos**			
32bit or 64bit arithmetic	-	enArithmetik[1)]			
Axis structure	-	stAxisDrive	stAxisDrive	-	Axis structure
Device structure	-	stDevice	stDevice	-	Device structure
Configuration structure winder	-	**stWinderConfig**	**stWinderConfig**	-	Configuration structure winder

actSync
Synchronous operation - opened in synchronous program section (FPLC_PRG)

1) Only for CODESYS V3

Input variables

Name	Type	Description	Sync.	Async.
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.		x
boRF	BOOL	Bit for enabling the controller Relevant parameters: ID32796 'Source RF' Allowed: Code 5: 'Controller enable via EtherCAT' Code 25: RF 'via EtherCAT AND-linked with the binary input UE' Code 0: 'Controller enable (RF) via binary input' is not allowed		x
boFL	BOOL	Bit for error deletion		x
boEmergency_Stop	BOOL	EMERGENCY STOP: The setpoint of the velocity is decreased to zero along the emergency-stop ramp. Once initiated, an emergency stop cannot be aborted. Relevant variables: EMERGENCY STOP ramp: Ramp time in which the drive is slowed down from the current velocity to zero. EMERGENCY STOP ramp: ST_AXIS_DRIVE.ST_IDS.ST_ID32919_ PlcList. diID32919_05_diEmergency_StopRamp Unit: ms Default value: 100 ms		x

Name	Type	Description	Sync.	Async.
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). Relevant parameters: ID32802 'AMK secondary operating mode 2' [Operating mode: Speed control] [Setpoint source: diMainSetpoint (code 41h)] [Ramps: active / inactive (bit 6)] ID32780 'Acceleration ramp' ID32781'Deceleration ramp' Relevant variables: Speed setpoint: diSpeedVelocity Velocity ramp: STANDARD_AXIS.fbVelocity.diVelocityRamp [default value: 100 ms] Speed control active: boSpeedBusy		x
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. Relevant parameters: ID41 'Homing velocity' ID147 'Homing parameter' ID150 'Homing offset 1' ID32926 'AMK homing cycle parameter' Relevant variables: Homing drive active: boHomeBusy Homing done: boDone Homing point known: stAxisDrive.stStatus.boID33036_RFP_known		x

Name	Type	Description	Sync.	Async.
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 [Relevant parameters: see 'boPosRel']		x
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. Relevant parameters: ID32801 'AMK secondary operating mode 1' [Operating mode: position control] [Setpoint source: diMainSetpoint (code 41h)] Relevant variables: Positioning velocity: udPosVelocity Target position: diPosition Acceleration: lreAccel Deceleration: lreDecel Jerk limitation (start): lreJerkStart Jerk limitation (end): lreJerkEnd Positioning active: boPosBusy Positioning done: boDone		x
boJogPlus	BOOL	Jog in positive direction Enable signal: With a positive edge, the jogging function starts (positive direction). Jog mode is run as long as 'boJobPlus' = TRUE. Use a negative edge 'boJogPlus' = FALSE to cancel the current jog mode. [Relevant parameters: see 'boJogMinus']		x

Name	Type	Description	Sync.	Async.
boJogMinus	BOOL	Jogging in negative direction Enable signal: With a positive edge, the jogging function starts (negative direction). Jog mode is run as long as 'boJobMinus' = TRUE. Use a negative edge 'boJogMinus' = FALSE to cancel the current jog mode. Relevant parameters: ID32801 'AMK secondary operating mode 1' [Operating mode: position control] [Setpoint source: diMainSetpoint (code 41h)] Relevant variables: Jog velocity: udVelocityJog Acceleration: lreAccel Deceleration: lreDecel Jerk limitation (start): lreJerkStart Jerk limitation (end): lreJerkEnd Jog mode active: boJogBusy		x
udPosVelocity	UDINT	Positioning velocity Unit: incr/s Default value: 34133		x
diPosition	DINT	Target position Use the inputs 'boPosAbs' and 'boPosRel' to determine whether the target position is approached absolutely or relatively. Unit: Increments Default value: 30720		x
lreAccel	LREAL	Acceleration for positioning and jog mode Range: $1{,}43\ 10^{-13} < \lvert\text{'lreAccel'}\rvert < 1{,}43\ 10^{+16}$ Unit: incr/s^2 Default: 1000000		x
lreDecel	LREAL	Deceleration for positioning and jog mode Range: $1{,}43\ 10^{-13} < \lvert\text{'lreDecel'}\rvert < 1{,}43\ 10^{+16}$ Unit: incr/s^2 Default: 1000000		x
lreJerkStart	LREAL	Jerk limitation (start) for positioning and jog mode Unit: incr/s^3 Default: 10000000		x
lreJerkEnd	LREAL	Jerk limitation (end) for positioning and jog mode Unit: incr/s^3 Default: 10000000		x
udVelocityJog	UDINT	Jog velocity Unit: incr/s Default value: 34133		x
diSpeedVelocity	DINT	Speed setpoint for speed control Unit: rpm Default value: 100		x

Name	Type	Description	Sync.	Async.
boEnableWinder	BOOL	Enable signal: With a positive edge, the initialisation of the winder block starts. As long as 'boEnableWinder' = TRUE, the block remains enabled and is processed by the PLC. In the state 'boEnableWinder' = FALSE the block is no longer enabled and is thus no longer processed.		x
boDmCalc	BOOL	Activation of the diameter calculation		x
boFindInertia	BOOL	Determination the inertia moment of the reel and tightening web on		x
boWind	BOOL	Wind on		x
boSync	BOOL	Synchronize on		x
boDmPreset	BOOL	Set diameter value of the input 'iDmPreset'. The input bit is automatically reset after set.		x
boDmOkPreset	BOOL	Diameter value of input variable 'iDmPreset' set. The input bit 'boDmPreset' is automatically reset after set. Diameter status "known": The diameter value is interpreted as valid diameter, that is, acceleration and friction torques are calculated.		x
boDmNOkPreset	BOOL	Diameter value of input variable 'iDmPreset' set. The input bit 'boDmPreset' is automatically reset after set. Diameter status "unknown": The diameter value is interpreted as invalid diameter, that is, acceleration and friction torques are calculated.		x
boDmReset	BOOL	Reset output 'boDmOk'		x
boSetDmOk	BOOL	Set output 'boDmOk'		x
diWebTension	BOOL	Absolute web tension moment [0.1 Nm]		x
iDmPreset	INT	Diameter preset value Value to which the diameter with 'boDmNokPreset' = TRUE or 'boDmOkPreset' =TRUE is set.[mm]		x
diActMotorSpeed	DINT	Actual speed value motor [0.0001 r/min]	x	
diActReelPos	DINT	Actual reel position [increments]	x	
diActWebPos	INT	Actual web position [increments]	x	

<table>
<tr><th>Name</th><th>Type</th><th>Description</th><th>Sync.</th><th>Async.</th></tr>
<tr><td>enArithmetik[1)]</td><td>ENUM</td><td>EN_POS_J_ARITHMETIK
(This input is only available for CODESYS V3 function blocks. CODESYS V2, the selection is always adjusted automatically)

With 'EN_POS_J_ARITHMETIK' the selection 32 bit or 64 bit arithmetic for position function blocks in position operation mode is set.

Options:
<table>
<tr><td>enAutomaticSelection</td><td>Automatic determination if 32 bit or 64 bit arithmetic (Default)
 
The default setting 'enAutomaticSelection' can only be used, if the PLC controller is on the first place in the CODESYS 'Device_ Config'. Is that not the case the selection has to be made manually by the user.</td></tr>
<tr><td>en64BitArithmetik</td><td>64 bit arithmetic</td></tr>
<tr><td>en32BitArithmetik</td><td>32 bit arithmetic</td></tr>
</table>
If the automatic determination does not work, the following diagnostic number is displayed and the selection must done manually by the user.
<table>
<tr><td>iErrID</td><td>5</td></tr>
</table>
Additional information about 'EN_POS_J_ARITHMETIK' and manual selection:
Siehe 'EN_POS_J_ARITHMETIK' auf Seite 601.</td><td></td><td>x</td></tr>
</table>

1) Only for CODESYS V3

Output variables

Name	Type	Description	Sync.	Async.
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled		x
boSBM	BOOL	System ready message		x
boDone	BOOL	Positioning done. Valid for 'boPosAbs', 'boPosRel' and 'boHome'		x
boErr	BOOL	The function block is in an error state FALSE: No error (permitted commanding or warning) TRUE: Error		x
iErrID	INT	Error identity number: Diagnostic number is output iErrID = 0: No error iErrID ≠ 0, boErr = TRUE: Error iErrID ≠ 0, boErr = FALSE: Warning Bus error when 'boErr' = TRUE: EtherCAT or CAN are not in operational mode.	x	
enErrName	ENUM	EN_FB_NAME Name of faulty block for which the diagnostic number is output. The diagnostic number is explained in the description of the corresponding library.		x
boPosBusy	BOOL	Positioning active		x
boJogBusy	BOOL	Jog mode active		x
boHomeBusy	BOOL	Home active		x
boSpeedBusy	BOOL	Speed control active		x
boEnabAcknWinder	BOOL	Acknowledgement: Winder is initialised and enabled		x
boDmOk	BOOL	Known diameter		x
boFindInertiaDone	BOOL	Winder moment determined, tightened web		x
boWindRuns	BOOL	Wind runs		x
boSyncRuns	BOOL	Synchronize runs		x
boDmLimit	BOOL	Diameter was calculated 'ST_WINDER_TORQUE_CONTROL_CONFIG.iDmMin' and 'ST_WINDER_TORQUE_CONTROL_CONFIG.iDmMax', the value was limited		x
iActDm	INT	Actual diameter [mm]		x
diActValWebSpeed	DINT	Actual value web speed [mm/s]		x
diActMotorSpeedAbs	DINT	Actual motor speed absolute [0.0001 r/min]	x	
diActReelPosAbs	DINT	Actual reel position absolute [increments]	x	

Input and output variables

Name	Type	Description	Sync.	Async.
stDevice	STRUCT	ST_DEVICE The device description structure assigns the block a device. (See document Software description AmkBase Bibliothek, Part no.204986)		x
stAxisDrive	STRUCT	ST_AXIS_DRIVE Siehe 'ST_AXIS_DRIVE (ST)' auf Seite 191.		x
stWinderConfig	STRUCT	ST_WINDER_TORQUE_CONTROL_CONFIG Configuration structure of the winder (See document Software description AFL - AMK Function Library, part 2, Part no.203694)		x

Usage note in the CODESYS program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
STANDARD_WINDER_TORQUE_AXIS	STANDARD_WINDER_TORQUE_AXIS.actSync

History

Library	AmkAfl	
System versions	**Functionality**	**Target**
AS-C V3.01/0827	Basic functionality (1st version)	≥ AS_CP_200_0812_T202053
AS-PL15 V3.01/0804	Basic functionality (1st version)	≥ AS_V313_0946_T202724

4.2 AMK AFL Application blocks

4.2.1 Library description AmkAfl.lib

4.2.1.1 05_StandardLevel

4.2.1.1.1 01_Comparison Functions

4.2.1.1.1.1 COMPARE_2VALUES (FB)

The function block 'COMPARE_2VALUES' sets an output bit when the condition Value1 = Value2 is fulfilled within a window.

The call of this function block is permissible in both the asynchronous program section PLC_PRG as well as the synchronous program section FPLC_PRG.

User interface

		COMPARE_2VALUES			
FB enable	-	boEnable	boEnabAck	-	Ackn. "FB enable"
Size of window	-	diWindow	boEquality	-	Equality
Comparative value 1	-	diVal1			
Comparative value 2	-	diVal2			

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.
diWindow	DINT	Parameter for entry of the window size [same unit as the comparison values]
diVal1	DINT	Comparison value 1 or 2 [same unit as the comparison value]
diVal2	DINT	Comparison value 1 or 2 [same unit as the comparison value]

Output variables

Name	Type	Description
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled
boEquality	BOOL	Comparison value 1 = comparison value 2 within the specified window

Usage note in the CoDeSys program

The call of this function block is permissible in both the asynchronous program section PLC_PRG as well as the synchronous program section FPLC_PRG.

Associated with this function block, a visualisation is prepared in CoDeSys.

Siehe 'Visualization of AFL blocks' auf Seite 806.

4.2.1.1.1.2 fboCompare_ActVal_GT_Threshold (FB)

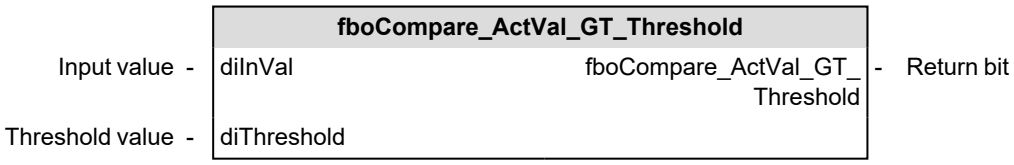
The function block 'fboCompare_ActVal_GT_Threshold' compares the input value with the threshold value.

The return value is TRUE if the input value > threshold, at threshold >0.

The return value is TRUE if the input value < threshold, at threshold <0.

The call of this function block is permissible in both the asynchronous program section PLC_PRG as well as the synchronous program section FPLC_PRG.

User interface



Input variables

Name	Type	Description
diInVal	DINT	Input value
diThreshold	DINT	Threshold value

Output variables

Name	Type	Description
fboCompare_ActVal_GT_Threshold	BOOL	Return bit

4.2.1.1.2 03_Position

4.2.1.1.2.1 Support

POSITION_1_A4 (FB)

The function block 'POSITION_1_A4' is a basic module for relative positioning. It positions an AMK axis relatively resp. relatively and retriggerably.

The function block is called in the synchronous program level FPLC_PRG.

'Support' functions and function blocks will not be used directly. They are called as subroutines by other functions or function blocks.

Anwender Interface

		POSITION_1_A4			
Selection mode	-	enMode	boDone	-	Ackn. "FB done"
FB enable	-	boEnable	boEnabAck	-	Ackn. "FB enable"
Execution started	-	boStart	bo0Vel	-	No setpoint output
Stop	-	boStop	boSetVel	-	Setpoint velocity reached
Emergency stop	-	boNotHalt	boSetPos	-	Setpoint position reached (retriggerable)
Target position	-	diPosition	boErr	-	Error
Acceleration	-	lreAccel	iErrId	-	Error ID
Deceleration	-	lreDecel	diOutOffs	-	Offset
Emergency stop ramp	-	lreDecelNotHalt	lreActVelocity	-	Actual velocity
Velocity	-	lreVelocity	lreActAccel	-	Actual acceleration
Jerk on start	-	lreJerkStart	diOutVal	-	Output value
Jerk on stop	-	lreJerkEnd			
Device structure	-	stDevice	stDevice	-	Device structure

Input variables

Name	Type	Description
enMode	ENUM	EN_POS_J_MODE Selection mode POS_REL: relative Positionierung POS_REL_RETRIG_EXT: relative Positionierung mit Nachtrigger-Modus und erweiterter Funktionalität
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	Stop all movements (transition to speed control with setpoint speed = 0).
boNotHalt	BOOL	EMERGENCY STOP: The setpoint of the velocity is decreased to zero along the emergency-stop ramp. Once initiated, an emergency stop cannot be aborted.
diPosition	DINT	Modulo position

Name	Type	Description
lreAccel	LREAL	Acceleration with which the target velocity is run
lreDecel	LREAL	Deceleration with which a lower target velocity is achieved
lreDecelNotHalt	LREAL	Emergency stop ramp; deceleration in case of emergency stop. The value entered is valid for deceleration from actual speed to speed 0 Unit: incr/s^2
lreVelocity	LREAL	Velocity with which the positioning is performed. [increments/s]
lreJerkStart	LREAL	Jerk limitation (start) for positioning and jog mode Unit: incr/s^3 Default: 10000000
lreJerkEnd	LREAL	Jerk limitation (end) for positioning and jog mode Unit: incr/s^3 Default: 10000000

Output variables

Name	Type	Description
boDone	BOOL	Response that the function block has been completely executed.
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled
bo0Vel	BOOL	When 'bo0Vel' is active, no setpoint is output.
boSetVel	BOOL	When 'boSetVel' is active, the target velocity has been reached.
boSetPos	BOOL	In retrigger mode the signal for a cycle becomes active when the retrigger position has been reached.
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
diOutOffs	DINT	Offset value before the retrigger is started; only in retrigger mode
lreActVelocity	LREAL	Actual velocity [increments/s]
lreActAccel	LREAL	Actual acceleration value [increments/s^2]
diOutVal	DINT	Output value [Inkremente]

Input and output variables

Name	Type	Description
stDevice	STRUCT	ST_DEVICE The device description structure assigns the block a device. (See document Software description AmkBase Bibliothek, Part no.204986)

Usage not in CoDeSys program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
	POSITION_1_A4

POSITION_JERK_1_A5 (FB)

The function block 'POSITION_JERK_1_A5' is a basic module for relative positioning with jerk limitation. It positions an AMK axis relatively resp. relatively and retriggerably.

The function block is called in the synchronous program level FPLC_PRG.

'Support' functions and function blocks will not be used directly. They are called as subroutines by other functions or function blocks.

User Interface

		POSITION_JERK_1_A5			
Selection mode	-	enMode	boDone	-	Ackn. "FB done"
FB enable	-	boEnable	boEnabAck	-	Ackn. "FB enable"
Execution started	-	boStart	bo0Vel	-	No setpoint output
Stop	-	boStop	boSetVel	-	Setpoint velocity reached
Emergency stop	-	boNotHalt	boSetPos	-	Setpoint position reached (retriggerable)
Target position	-	diPosition	boErr	-	Error
Positioning acceleration	-	lreAccel	iErrID	-	Error ID
Positioning deceleration	-	lreDecel	diOutOffs	-	Offset
Emergency stop ramp	-	lreDecelNotHalt	lreActVelocity	-	Actual velocity value
Positioning velocity	-	lreVelocity	lreActAccel	-	Actual acceleration
Jerk on start	-	lreJerkStart	diOutVal		Output value
Jerk on stop	-	lreJerkEnd			
Device structure	-	stDevice	stDevice	-	Device structure

Input variables

Name	Type	Description
enMode	ENUM	EN_POS_J_MODE Selection mode POS_REL: Relative positioning POS_REL_RETRIG_EXT: Relative positioning retriggerable, extended functionality POS_J_REL_RETRIG: Relative positioning retriggerable with jerk limitation
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.
boNotHalt	BOOL	EMERGENCY STOP: The setpoint of the velocity is decreased to zero along the emergency-stop ramp. Once initiated, an emergency stop cannot be aborted.
diPosition	DINT	Target of positioning [increments]
lreAccel / lreDecel	LREAL	Acceleration/deceleration ramp [increments/s^2]
lreDecelNotHalt	LREAL	EMERGENCY-STOP ramp: Ramp time in which the drive is slowed down from the current velocity to zero [ms].

Name	Type	Description
lreVelocity	LREAL	Velocity with which the positioning is performed. [increments/s]
lreJerkStart / lreJerkEnd	LREAL	Jerk for the acceleration/deceleration [increments/s^3]

Output variables

Name	Type	Description
boDone	BOOL	Response that the function block has been completely executed.
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled
bo0Vel	BOOL	When 'bo0Vel' is active, no setpoint is output.
boSetVel	BOOL	When 'boSetVel' is active, the target velocity has been reached.
boSetPos	BOOL	In retrigger mode the signal for a cycle becomes active when the retrigger position has been reached.
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 1: Input parameter not appropriate 2: Positioning velocity automatically adapted 3: Target velocity automatically adapted 4: Jerk parameter automatically adapted
diOutOffs	DINT	Offset value before the retrigger is started; only in retrigger mode
lreActVelocity	LREAL	Actual velocity [increments/s] (Feed-forward value)
lreActAccel	LREAL	Actual acceleration value [increments/s^2] (Feed-forward value)
diOutVal	DINT	Output position

Input and output varables

Name	Type	Description
stDevice	STRUCT	ST_DEVICE The device description structure assigns the block a device. (See document Software description AmkBase Bibliothek, Part no.204986)

Usage note in the CoDeSys program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
	POSITION_JERK_1_A5

4.2.1.1.2.2 POSITION_ABSOLUT (FB)

The function block 'POSITION_ABSOLUT' positions an AMK axis absolutely.
The function block is called in the synchronous program level FPLC_PRG.

User interface

		POSITION_ABSOLUT			
FB execution	-	boExec	boDone	-	Ackn. "FB done"
Emergency stop	-	boEmergency_Stop	boBusy	-	FB in progress
Emergency stop ramp	-	diEmergency_StopRamp	boInPos	-	In position
Target position	-	diPosition	boErr	-	Error
Positioning velocity	-	lreVelocity	iErrID	-	Error ID
Positioning acceleration	-	lreAccel	enErrName	-	Name of faulty FB
Target position window	-	diPosWindow			
Actual position value	-	diActPosition			
32bit or 64bit arithmetic	-	enArithmetik[1)]			
Position setpoint	-	diSetPosition	diSetPosition	-	Position setpoint
Device structure	-	stDevice	stDevice	-	Device structure

1) Only for CODESYS V3

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>boEmergency_Stop</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>diEmergency_StopRamp</td><td>DINT</td><td>EMERGENCY-STOP ramp: Ramp time in which the drive is slowed down from the current velocity to zero [ms].</td></tr>
<tr><td>diPosition</td><td>DINT</td><td>Target of positioning [increments]</td></tr>
<tr><td>lreAccel</td><td>LREAL</td><td>Acceleration/deceleration ramp [increments/s^2]</td></tr>
<tr><td>lreVelocity</td><td>LREAL</td><td>Velocity with which the positioning is performed. [increments/s]</td></tr>
<tr><td>diPosWindow</td><td>DINT</td><td>Position window: Window within which the InPos bit is set. [Increments]
Default: 1000 increments</td></tr>
<tr><td>diActPosition</td><td>DINT</td><td>Actual position feedback value [increments]</td></tr>
<tr><td>enArithmetik[1)]</td><td>ENUM</td><td>EN_POS_J_ARITHMETIK
(This input is only available for CODESYS V3 function blocks.
CODESYS V2, the selection is always adjusted automatically)

With 'EN_POS_J_ARITHMETIK' the selection 32 bit or 64 bit arithmetic for position function blocks in position operation mode is set.

Options:
<table>
<tr><td>enAutomaticSelection</td><td>Automatic determination if 32 bit or 64 bit arithmetic (Default)
 
The default setting 'enAutomaticSelection' can only be used, if the PLC controller is on the first place in the CODESYS 'Device_Config'. Is that not the case the selection has to be made manually by the user.</td></tr>
<tr><td>en64BitArithmetik</td><td>64 bit arithmetic</td></tr>
<tr><td>en32BitArithmetik</td><td>32 bit arithmetic</td></tr>
</table>

If the automatic determination does not work, the following diagnostic number is displayed and the selection must done manually by the user.
<table>
<tr><td>iErrID</td><td>5</td></tr>
</table>
Additional information about 'EN_POS_J_ARITHMETIK' and manual selection:
Siehe 'EN_POS_J_ARITHMETIK' auf Seite 601.</td></tr>
</table>

1) Only for CODESYS V3

Output variables

Name	Type	Description
boDone	BOOL	Response that the function block has been completely executed.
boBusy	BOOL	Execution message: This bit remains set as long as the block is being processed.
boInPos	BOOL	Message "InPosition": Actual position = target position within the position window
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
enErrName	ENUM	EN_FB_NAME Name of faulty block for which the diagnostic number is output. The diagnostic number is explained in the description of the corresponding library. Block: POS_1 — Library: AmkBase.lib

Input and output variables

Name	Type	Description
stDevice	STRUCT	ST_DEVICE The device description structure assigns the block a device. (See document Software description AmkBase Bibliothek, Part no.204986)
diSetPosition	DINT	Specification of the position setpoint (position setpoint system) [increments]

Usage note in the CoDeSys program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
	POSITION_ABSOLUT

Associated with this function block, a visualisation is prepared in CoDeSys.

Siehe 'Visualization of AFL blocks' auf Seite 806.

4.2.1.1.2.3 POSITION_ABSOLUT_VAJ (FB)

The function block 'POSITION_ABSOLUT_VAJ' positions an AMK axis absolutely. In addition to position, speed and acceleration, the user can also specify the jerk.

$$v = \frac{\Delta s}{\Delta t}$$ Velocity

$$a = \frac{\Delta v}{\Delta t} = \frac{\Delta s}{\Delta t^2}$$ Acceleration

$$j = \frac{\Delta a}{\Delta t} = \frac{\Delta v}{\Delta t^2} = \frac{\Delta s}{\Delta t^3}$$ Jerk

The function block is called in the synchronous program level FPLC_PRG.

In the AMKAMAC A4 controller the jerk limitation is not supported.

User interface

		POSITION_ABSOLUT_VAJ			
FB execution	-	boExec	boDone	-	Ackn. "FB done"
Emergency stop	-	boEmergency_Stop	boBusy	-	FB in progress
Emergency stop ramp	-	diEmergency_StopRamp	boInPos	-	In position
Target position	-	diPosition	lreActVelocity	-	Positioning velocity
Positioning acceleration	-	lreAccel	lreActAccel	-	Positioning acceleration
Positioning deceleration	-	lreDecel	boErr	-	Error
Positioning velocity	-	lreVelocity	iErrID	-	Error ID
Jerk on start	-	lreJerkStart	enErrName	-	Name of faulty FB
Jerk on stop	-	lreJerkEnd			
Target position window	-	diPosWindow			
Actual position value	-	diActPosition			
32bit or 64bit arithmetic	-	enArithmetik[1)]			
Position setpoint	-	diSetPosition	diSetPosition	-	Position setpoint
Device structure	-	stDevice	stDevice	-	Device structure

1) Only for CODESYS V3

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.
boEmergency_Stop	BOOL	EMERGENCY STOP: The setpoint of the velocity is decreased to zero along the emergency-stop ramp. Once initiated, an emergency stop cannot be aborted.
diEmergency_StopRamp	DINT	EMERGENCY-STOP ramp: Ramp time in which the drive is slowed down from the current velocity to zero [ms].
diPosition	DINT	Target of positioning [increments]
lreAccel / lreDecel	LREAL	Acceleration/deceleration ramp [increments/s^2]
lreVelocity	LREAL	Velocity with which the positioning is performed. [increments/s]
lreJerkStart / lreJerkEnd	LREAL	Jerk for the acceleration/deceleration [increments/s^3]

Name	Type	Description
diPosWindow	DINT	Position window: Window within which the InPos bit is set. [Increments] Default: 1000 increments
diActPosition	DINT	Actual position feedback value [increments]
enArithmetik[1)]	ENUM	EN_POS_J_ARITHMETIK (This input is only available for CODESYS V3 function blocks. CODESYS V2, the selection is always adjusted automatically) With 'EN_POS_J_ARITHMETIK' the selection 32 bit or 64 bit arithmetic for position function blocks in position operation mode is set. Options: enAutomaticSelection: Automatic determination if 32 bit or 64 bit arithmetic (Default) The default setting 'enAutomaticSelection' can only be used, if the PLC controller is on the first place in the CODESYS 'Device_Config'. Is that not the case the selection has to be made manually by the user. en64BitArithmetik: 64 bit arithmetic en32BitArithmetik: 32 bit arithmetic If the automatic determination does not work, the following diagnostic number is displayed and the selection must done manually by the user. iErrID: 5 Additional information about 'EN_POS_J_ARITHMETIK' and manual selection: Siehe 'EN_POS_J_ARITHMETIK' auf Seite 601.

1) Only for CODESYS V3

Output variables

Name	Type	Description
boDone	BOOL	Response that the function block has been completely executed.
boBusy	BOOL	Execution message: This bit remains set as long as the block is being processed.
boInPos	BOOL	Message "InPosition": Actual position = target position within the position window
lreActVelocity	LREAL	Velocity with which the positioning is performed. [increments/s]
lreActAccel	LREAL	Actual acceleration value [increments/s^2]
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

Name	Type	Description
enErrName	ENUM	EN_FB_NAME Name of faulty block for which the diagnostic number is output. The diagnostic number is explained in the description of the corresponding library. **Block** / **Library** POSITION_JERK_1 / AmkAfl.lib

Input and output variables

Name	Type	Description
diSetPosition	DINT	Specification of the position setpoint (position setpoint system) [increments]
stDevice	STRUCT	ST_DEVICE The device description structure assigns the block a device. (See document Software description AmkBase Bibliothek, Part no.204986)

Usage note in the CoDeSys program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
	POSITION_ABSOLUT_VAJ

Associated with this function block, a visualisation is prepared in CoDeSys.
Siehe 'Visualization of AFL blocks' auf Seite 806.

4.2.1.1.2.4 POSITION_CHECK_LIMIT_SWITCH (FB)

The function block 'POSITION_CHECK_LIMIT_SWITCH' checks a target position for end-position limits.

If the target position is within the specified limits, this is indicated by the output bit 'boLimitOk'. This message can be used to start absolute positioning, for instance. The corresponding target position is available at the output 'diTargetPosition'.

By setting the input bit 'boAdaptTargetPosition', the target position is modified to the limits. This means, if the target is outside of the specified limits, the output target position at the output 'diTargetPosition' will be modified to the corresponding limit. If the input bit 'boAdaptTargetPosition' is not set, just the end-position check will be performed and the target position will be passed through.

The call of this function block is permissible in both the asynchronous program section PLC_PRG as well as the synchronous program section FPLC_PRG.

User interface

	POSITION_CHECK_LIMIT_SWITCH		
FB enable -	boEnable	boEnabAck	- Ackn. "FB enable"
Adapt target position -	boAdaptTargetPosition	boLimitOk	- Target position in limit
Limit of end position -	diLimit1	diTargetPosition	- Output target position
Limit of end position -	diLimit2		
Positioning target -	diTarget		

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.
boAdaptTargetPosition	BOOL	Modification of the target position according to the end position when exceeding the end position
diLimit1	DINT	End position 1: smallest position [increments]
diLimit2	DINT	End position 2: largest position [increments]
diTarget	DINT	Target to be checked for end positions for positioning

Output variables

Name	Type	Description
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled
boLimitOk	BOOL	The output target position is in the end-position limits. This bit can be used to initiate positioning.
diTargetPosition	DINT	Output of the checked target position. This output can be used as target position for positioning.

Usage note in the CoDeSys program

The call of this function block is permissible in both the asynchronous program section PLC_PRG as well as the synchronous program section FPLC_PRG.

Associated with this function block, a visualisation is prepared in CoDeSys.
Siehe 'Visualization of AFL blocks' auf Seite 806.

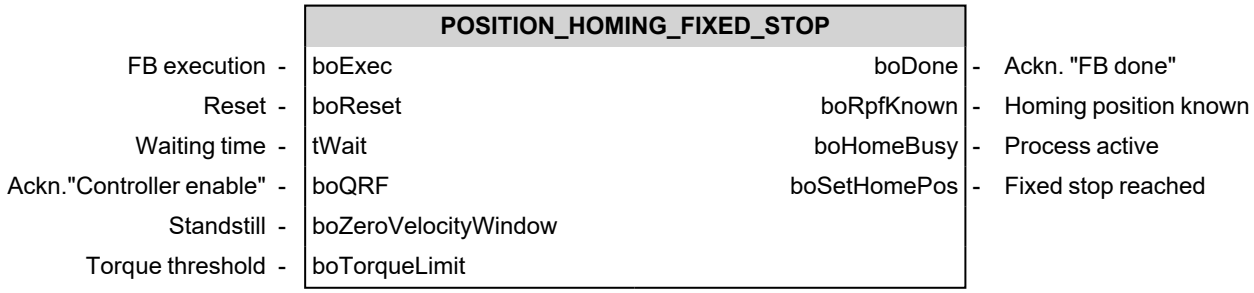
4.2.1.1.2.5 POSITION_HOMING_FIXED_STOP (FB)

The function block 'POSITION_HOMING_FIXED_STOP' recognises a fixed stop.

The movement up to the fixed stop under speed control has to be realised outside this function blocks, e.g. with 'STANDARD_AXIS'. The output 'boHomeBusy' can be used for activating the speed control.

The function block is called in the synchronous program level FPLC_PRG.

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. With a positive edge on this input, the output 'boHomeBusy' is set.

Name	Type	Description
boReset	BOOL	Resets the output message 'boRpfKnown' (Homing position known)
tWait	TIME	Dwell time at fixed stop until the outputs 'boRpfKnown' and 'boSetHomePos' are set. [ms] Default: 500 ms
boQRF	BOOL	Acknowledgement controller enable
boTorqueLimit	BOOL	Torque threshold reached (ID126 'Torque threshold', code 333 Md ≥ Mdx)
boZeroVelocityWindow	BOOL	Speed within standstill window (ID124 'Zero velocity window', code 331 nact < nmin)

Output variables

Name	Type	Description
boDone	BOOL	Acknowledgement: Function block is initialised and enabled
boRpfKnown	BOOL	Homing position known If the fixed stop is reached (ID333 'Message torque: actual value ≥ threshold' und ID331 'Message speed: actual value < minimal value' are TRUE for longer than 'tWait'), 'boRpfKnown' is set and remains until it is reset with a positive edge at 'boReset'.
boHomeBusy	BOOL	Processing is active This output can be used to set the drive to the operation mode speed control, e.g. boHomeBusy -> STANDARD_AXIS.boSpeed
boSetHomePos	BOOL	Fixed stop reached -> set homing position If the fixed stop is reached (ID333 'Message torque: actual value ≥ threshold' und ID331 'Message speed: actual value < minimal value' are TRUE for longer than 'tWait'), 'boSetHomePos' is set for one cycle.

Usage note in the CoDeSys program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
	POSITION_HOMING_FIXED_STOP

Associated with this function block, a visualisation is prepared in CoDeSys.
Siehe 'Visualization of AFL blocks' auf Seite 806.

4.2.1.1.2.6 POSITION_JERK_1 (FB)

The function block 'POSITION_JERK_1' is a base object for relative positioning with jerk limitation. It positions an AMK axis relatively or relatively and retriggerably.

The function block is called in the synchronous program level FPLC_PRG.

User interface

		POSITION_JERK_1			
Selection mode	-	enMode	boDone	-	Ackn. "FB done"
FB enable	-	boEnable	boEnabAck	-	Ackn. "FB enable"
Execution started	-	boStart	bo0Vel	-	No setpoint output
Stop	-	boStop	boSetVel	-	Setpoint velocity reached
Emergency stop	-	boNotHalt	boSetPos	-	Setpoint position reached (retriggerable)
Target position	-	diPosition	boErr	-	Error
Positioning acceleration	-	lreAccel	iErrID	-	Error ID
Positioning deceleration	-	lreDecel	diOutOffs	-	Offset
Emergency stop ramp	-	lreDecelNotHalt	lreActVelocity	-	Actual velocity value
Positioning velocity	-	lreVelocity	lreActAccel	-	Actual acceleration
Jerk on start	-	lreJerkStart	diOutVal	-	Output value
Jerk on stop	-	lreJerkEnd			
32bit or 64bit arithmetic	-	enArithmetik[1)]			
Device structure	-	stDevice	stDevice	-	Device structure

1) Only for CODESYS V3

Input variables

Name	Type	Description
enMode	ENUM	EN_POS_J_MODE Selection mode POS_J_REL: Relative positioning POS_J_REL_RETRIG_EXT: Relative positioning retriggerable
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.
boNotHalt	BOOL	EMERGENCY STOP: The setpoint of the velocity is decreased to zero along the emergency-stop ramp. Once initiated, an emergency stop cannot be aborted.
diPosition	DINT	Target of positioning [increments]
lreAccel / lreDecel	LREAL	Acceleration/deceleration ramp [increments/s^2]
lreDecelNotHalt	LREAL	EMERGENCY-STOP ramp: Ramp time in which the drive is slowed down from the current velocity to zero [ms].
lreVelocity	LREAL	Velocity with which the positioning is performed. [increments/s]
lreJerkStart / lreJerkEnd	LREAL	Jerk for the acceleration/deceleration [increments/s^3]

Name	Type	Description
enArithmetik[1)]	ENUM	EN_POS_J_ARITHMETIK (This input is only available for CODESYS V3 function blocks. CODESYS V2, the selection is always adjusted automatically) With 'EN_POS_J_ARITHMETIK' the selection 32 bit or 64 bit arithmetic for position function blocks in position operation mode is set. Options: enAutomaticSelection: Automatic determination if 32 bit or 64 bit arithmetic (Default) The default setting 'enAutomaticSelection' can only be used, if the PLC controller is on the first place in the CODESYS 'Device_Config'. Is that not the case the selection has to be made manually by the user. en64BitArithmetik: 64 bit arithmetic en32BitArithmetik: 32 bit arithmetic If the automatic determination does not work, the following diagnostic number is displayed and the selection must done manually by the user. iErrID: 5 Additional information about 'EN_POS_J_ARITHMETIK' and manual selection: Siehe 'EN_POS_J_ARITHMETIK' auf Seite 601.

1) Only for CODESYS V3

Output variables

Name	Type	Description
boDone	BOOL	Response that the function block has been completely executed.
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled
bo0Vel	BOOL	When 'bo0Vel' is active, no setpoint is output.
boSetVel	BOOL	When 'boSetVel' is active, the target velocity has been reached.
boSetPos	BOOL	In retrigger mode the signal for a cycle becomes active when the retrigger position has been reached.
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 1: Input parameters do not make sense 2: Positioning velocity is automatically adapted 3: End velocity was automatically adapted 4: Jerk parameter was automatically adapted 5: Control type could not be determined. Please 'ST_DEVICE' connect with the controller in AIPEX PRO and start autoconfiguration.
diOutOffs	DINT	Offset value before the retrigger is started; only in retrigger mode

Name	Type	Description
lreActVelocity	LREAL	Actual velocity [increments/s] (Pilot control value)
lreActAccel	LREAL	Actual acceleration value [increments/s^2] (Pilot control value)
diOutVal	DINT	Output position

Input and output variables

Name	Type	Description
stDevice	STRUCT	ST_DEVICE The device description structure assigns the block a device. (See document Software description AmkBase Bibliothek, Part no.204986)

Usage note in the CoDeSys program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
	POSITION_JERK_1

Associated with this function block, a visualisation is prepared in CoDeSys.
Siehe 'Visualization of AFL blocks' auf Seite 806.

4.2.1.1.2.7 POSITION_RELATIV (FB)

The function block 'POSITION_RELATIV' positions an AMK axis relatively.
The function block is called in the synchronous program level FPLC_PRG.

User interface

		POSITION_RELATIV			
FB execution	-	boExec	boDone	-	Ackn. "FB done"
Emergency stop	-	boEmergency_Stop	boBusy	-	FB in progress
Emergency stop ramp	-	diEmergency_StopRamp	boInPos	-	In position
Target position	-	diPosition	boErr	-	Error
Positioning velocity	-	lreVelocity	iErrID	-	Error ID
Positioning acceleration	-	lreAccel	enErrName	-	Name of faulty FB
Target position window	-	diPosWindow			
Actual position value	-	diActPosition			
32bit or 64bit arithmetic	-	enArithmetik[1)]			
Position setpoint	-	diSetPosition	diSetPosition	-	Position setpoint
Device structure	-	stDevice	stDevice	-	Device structure

1) Only for CODESYS V3

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>boEmergency_Stop</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>diEmergency_StopRamp</td><td>DINT</td><td>EMERGENCY-STOP ramp: Ramp time in which the drive is slowed down from the current velocity to zero [ms].</td></tr>
<tr><td>diPosition</td><td>DINT</td><td>Target of positioning [increments]</td></tr>
<tr><td>lreAccel</td><td>LREAL</td><td>Acceleration/deceleration ramp [increments/s^2]</td></tr>
<tr><td>lreVelocity</td><td>LREAL</td><td>Velocity with which the positioning is performed. [increments/s]</td></tr>
<tr><td>diPosWindow</td><td>DINT</td><td>Position window: Window within which the InPos bit is set. [Increments]
Default: 1000 increments</td></tr>
<tr><td>diActPosition</td><td>DINT</td><td>Actual position feedback value [increments]</td></tr>
<tr><td>enArithmetik[1)]</td><td>ENUM</td><td>EN_POS_J_ARITHMETIK
(This input is only available for CODESYS V3 function blocks.
CODESYS V2, the selection is always adjusted automatically)

With 'EN_POS_J_ARITHMETIK' the selection 32 bit or 64 bit arithmetic for position function blocks in position operation mode is set.

Options:
<table>
<tr><td>enAutomaticSelection</td><td>Automatic determination if 32 bit or 64 bit arithmetic (Default)
 
The default setting 'enAutomaticSelection' can only be used, if the PLC controller is on the first place in the CODESYS 'Device_Config'. Is that not the case the selection has to be made manually by the user.</td></tr>
<tr><td>en64BitArithmetik</td><td>64 bit arithmetic</td></tr>
<tr><td>en32BitArithmetik</td><td>32 bit arithmetic</td></tr>
</table>

If the automatic determination does not work, the following diagnostic number is displayed and the selection must done manually by the user.
<table>
<tr><td>iErrID</td><td>5</td></tr>
</table>

Additional information about 'EN_POS_J_ARITHMETIK' and manual selection:
Siehe 'EN_POS_J_ARITHMETIK' auf Seite 601.</td></tr>
</table>

1) Only for CODESYS V3

Output variables

Name	Type	Description
boDone	BOOL	Response that the function block has been completely executed.
boBusy	BOOL	Execution message: This bit remains set as long as the block is being processed.
boInPos	BOOL	Message "InPosition": Actual position = target position within the position window
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
enErrName	ENUM	EN_FB_NAME Name of faulty block for which the diagnostic number is output. The diagnostic number is explained in the description of the corresponding library. Block: POS_1 — Library: AmkBase.lib

Input and output variables

Name	Type	Description
stDevice	STRUCT	ST_DEVICE The device description structure assigns the block a device. (See document Software description AmkBase Bibliothek, Part no.204986)
diSetPosition	DINT	Specification of the position setpoint (position setpoint system) [increments]

Usage note in the CoDeSys program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
	POSITION_RELATIV

Associated with this function block, a visualisation is prepared in CoDeSys.
Siehe 'Visualization of AFL blocks' auf Seite 806.

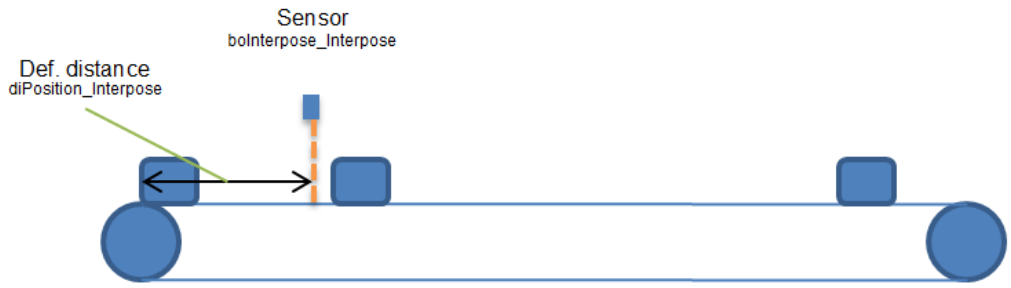
4.2.1.1.2.8 POSITION_RELATIV_INTERPOSED (FB)

The function block 'POSITION_RELATIV_INTERPOSED' moves the axis continuously after a positive edge of 'boExec' until the interposed positioning is started by a positive edge of 'boInterpose' . If the relative target position is reached, the function block stops.

The function block is called in the synchronous program level FPLC_PRG.

Example:

When a transport belt parts are deposited irregularly. After triggering the sensor, the parts are stopped at a defined position and removed from the belt.



User interface

		POSITION_RELATIV_INTERPOSED			
FB execution	-	boExec	boDone	-	Ackn. "FB done"
Start interposed positioning	-	boInterpose	boBusy	-	FB in progress
Emergency stop	-	boEmergency_Stop	boErr	-	Error
Emergency stop ramp	-	diEmergency_StopRamp	iErrID	-	Error ID
Target position (relative)	-	diPosition	enErrName	-	Name of faulty FB
Correction of position setpoint	-	diOffset	boInPos	-	In position
Acceleration	-	udAccel	bo0Vel	-	Velocity = 0
Deceleration	-	udDecel	boSetVel	-	Setpoint velocity reached
Velocity	-	udVelocity			
Target position window	-	diPosWindow			
Actual position value	-	diActPosition			
Position setpoint	-	diSetPosition	diSetPosition	-	Position setpoint
Device structure	-	stDevice	stDevice	-	Device structure

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.
boInterpose	BOOL	Start interposed positioning
boEmergency_Stop	BOOL	EMERGENCY STOP: The setpoint of the velocity is decreased to zero along the emergency-stop ramp. Once initiated, an emergency stop cannot be aborted.
diEmergency_StopRamp	DINT	EMERGENCY-STOP ramp: Ramp time in which the drive is slowed down from the current velocity to zero [ms].
diPosition	DINT	Target position relative (after boInterpose) Unit: incr/s Default value: 60000

Name	Type	Description
diOffset	DINT	Setpoint position offset correction Unit: incr/s Default value: 0
udAccel	UDINT	Acceleration with which the target velocity is run
udDecel	UDINT	Deceleration with which a lower target velocity is achieved
udVelocity	UDINT	Velocity with which the positioning is performed. [increments/s]
diPosWindow	DINT	Window for the message "In Position" [increments]
diActPosition	DINT	Actual position feedback value [increments]

Output variables

Name	Type	Description
boDone	BOOL	Response that the function block has been completely executed.
boBusy	BOOL	Execution message: This bit remains set as long as the block is being processed.
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
enErrName	ENUM	EN_FB_NAME Name of faulty block for which the diagnostic number is output. The diagnostic number is explained in the description of the corresponding library. Baustein: POS_1 — Bibliothek: AmkBase.lib
boInPos	BOOL	Message "InPosition": Actual position = target position within the position window
bo0Vel	BOOL	When 'bo0Vel' is active, no setpoint is output.
boSetVel	BOOL	When 'boSetVel' is active, the target velocity has been reached.

Input and output variables

Name	Type	Description
diSetPosition	DINT	Specification of the position setpoint (position setpoint system) [increments]
stDevice	STRUCT	ST_DEVICE The device description structure assigns the block a device. (See document Software description AmkBase Bibliothek, Part no.204986)

Usage not in the CoDeSys program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
	POSITION_RELATIV_INTERPOSED

Associated with this function block, a visualisation is prepared in CoDeSys.
Siehe 'Visualization of AFL blocks' auf Seite 806.

4.2.1.1.2.9 POSITION_RELATIV_RETRIGGER_VAJ (FB)

The function block 'POSITION_RELATIV_RETRIGGER_VAJ' positions an AMK drive relatively and retriggerably.
The function block is called in the synchronous program level FPLC_PRG.

In the AMKAMAC A4 controller the jerk limitation is not supported.

User interface

		POSITION_RELATIV_RETRIGGER_VAJ			
FB enable	-	boEnable	boEnabAck	-	Ackn. "FB enable"
Execution started	-	boStart	boDone	-	Ackn. "FB done"
Stop	-	boStop	boBusy	-	FB in progress
Emergency stop	-	boEmergency_Stop	boInPos	-	In position
Emergency stop ramp	-	diEmergency_StopRamp	lreActVelocity	-	Actual velocity value
Target position	-	diPosition	lreActAccel	-	Actual acceleration
Acceleration	-	lreAccel	bo0Vel	-	No setpoint output
Deceleration	-	lreDecel	boSetVel	-	Setpoint velocity reached
Positioning velocity	-	lreVelocity	boSetPos	-	Setpoint position reached (retriggerable)
Jerk on start	-	lreJerkStart	diOutOffs	-	Offset
Jerk on stop	-	lreJerkEnd	boErr	-	Error
Target position window	-	diPosWindow	iErrID	-	Error ID
Actual position value	-	diActPosition	enErrName	-	Name of faulty FB
32bit or 64bit arithmetic	-	enArithmetik[1)]			
Position setpoint	-	diSetPosition	diSetPosition	-	Position setpoint
Device structure	-	stDevice	stDevice	-	Device structure

1) Only for CODESYS V3

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.
boEmergency_Stop	BOOL	EMERGENCY STOP: The setpoint of the velocity is decreased to zero along the emergency-stop ramp. Once initiated, an emergency stop cannot be aborted.
diEmergency_StopRamp	DINT	EMERGENCY-STOP ramp: Ramp time in which the drive is slowed down from the current velocity to zero [ms].
diPosition	DINT	Target of positioning [increments]
lreAccel / lreDecel	LREAL	Acceleration/deceleration ramp [increments/s^2]
lreVelocity	LREAL	Velocity with which the positioning is performed. [increments/s]
lreJerkStart / lreJerkEnd	LREAL	Jerk for the acceleration/deceleration [increments/s^3]
diPosWindow	DINT	Window for the message "In Position" [increments]

Name	Type	Description
diActPosition	DINT	Actual position feedback value [increments]
enArithmetik[1)]	ENUM	EN_POS_J_ARITHMETIK (This input is only available for CODESYS V3 function blocks. CODESYS V2, the selection is always adjusted automatically) With 'EN_POS_J_ARITHMETIK' the selection 32 bit or 64 bit arithmetic for position function blocks in position operation mode is set. Options: enAutomaticSelection: Automatic determination if 32 bit or 64 bit arithmetic (Default) The default setting 'enAutomaticSelection' can only be used, if the PLC controller is on the first place in the CODESYS 'Device_Config'. Is that not the case the selection has to be made manually by the user. en64BitArithmetik: 64 bit arithmetic en32BitArithmetik: 32 bit arithmetic If the automatic determination does not work, the following diagnostic number is displayed and the selection must done manually by the user. iErrID: 5 Additional information about 'EN_POS_J_ARITHMETIK' and manual selection: Siehe 'EN_POS_J_ARITHMETIK' auf Seite 601.

1) Only for CODESYS V3

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.
boBusy	BOOL	Execution message: This bit remains set as long as the block is being processed.
boInPos	BOOL	Message "InPosition": Actual position = target position within the position window
lreActVelocity	LREAL	Actual velocity [increments/s]
lreActAccel	LREAL	Actual acceleration value [increments/s^2]
bo0Vel	BOOL	When 'bo0Vel' is active, no setpoint is output.
boSetVel	BOOL	When 'boSetVel' is active, the target velocity has been reached.
boSetPos	BOOL	In retrigger mode the signal for a cycle becomes active when the retrigger position has been reached.
diOutOffs	DINT	Offset value before the retrigger is started; only in retrigger mode
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
enErrName	ENUM	EN_FB_NAME Name of faulty block for which the diagnostic number is output. The diagnostic number is explained in the description of the corresponding library. Block: POSITION_JERK_1 — Library: AmkAfl.lib

Input and output variables

Name	Type	Description
diSetPosition	DINT	Specification of the position setpoint (position setpoint system) [increments]
stDevice	STRUCT	ST_DEVICE The device description structure assigns the block a device. (See document Software description AmkBase Bibliothek, Part no.204986)

Usage note in the CoDeSys program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
	POSITION_RELATIV_RETRIGGER_VAJ

Associated with this function block, a visualisation is prepared in CoDeSys.
Siehe 'Visualization of AFL blocks' auf Seite 806.

4.2.1.1.2.10 POSITION_RELATIV_VAJ (FB)

The function block 'POSITION_RELATIV_VAJ' positions an AMK axis relatively. In addition to position, speed and acceleration, the user can also specify the jerk.

$v = \frac{\Delta s}{\Delta t}$ Velocity

$a = \frac{\Delta v}{\Delta t} = \frac{\Delta s}{\Delta t^2}$ Acceleration

$j = \frac{\Delta a}{\Delta t} = \frac{\Delta v}{\Delta t^2} = \frac{\Delta s}{\Delta t^3}$ Jerk

The function block is called in the synchronous program level FPLC_PRG.

In the AMKAMAC A4 controller the jerk limitation is not supported.

User interface

		POSITION_RELATIV_VAJ			
FB execution	-	boExec	boDone	-	Ackn. "FB done"
Emergency stop	-	boEmergency_Stop	boBusy	-	FB in progress
Emergency stop ramp	-	diEmergency_StopRamp	boInPos	-	In position
Target position	-	diPosition	lreActVelocity	-	Positioning velocity
Positioning acceleration	-	lreAccel	lreActAccel	-	Positioning acceleration
Positioning deceleration	-	lreDecel	boErr	-	Error
Positioning velocity	-	lreVelocity	iErrID	-	Error ID
Jerk on start	-	lreJerkStart	enErrName	-	Name of faulty FB
Jerk on stop	-	lreJerkEnd			
Target position window	-	diPosWindow			
Actual position value	-	diActPosition			
32bit or 64bit arithmetic	-	enArithmetik[1)]			
Position setpoint	-	diSetPosition	diSetPosition	-	Position setpoint
Device structure	-	stDevice	stDevice	-	Device structure

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.
boEmergency_Stop	BOOL	EMERGENCY STOP: The setpoint of the velocity is decreased to zero along the emergency-stop ramp. Once initiated, an emergency stop cannot be aborted.
diEmergency_StopRamp	DINT	EMERGENCY-STOP ramp: Ramp time in which the drive is slowed down from the current velocity to zero [ms].
diPosition	DINT	Target of positioning [increments]
lreAccel / lreDecel	LREAL	Acceleration/deceleration ramp [increments/s^2]
lreVelocity	LREAL	Velocity with which the positioning is performed. [increments/s]
lreJerkStart / lreJerkEnd	LREAL	Jerk for the acceleration/deceleration [increments/s^3]
diPosWindow	DINT	Position window: Window within which the InPos bit is set. [Increments] Default: 1000 increments
diActPosition	DINT	Actual position feedback value [increments]

Name	Type	Description
enArithmetik[1)]	ENUM	EN_POS_J_ARITHMETIK (This input is only available for CODESYS V3 function blocks. CODESYS V2, the selection is always adjusted automatically) With 'EN_POS_J_ARITHMETIK' the selection 32 bit or 64 bit arithmetic for position function blocks in position operation mode is set. Options: enAutomaticSelection: Automatic determination if 32 bit or 64 bit arithmetic (Default) The default setting 'enAutomaticSelection' can only be used, if the PLC controller is on the first place in the CODESYS 'Device_Config'. Is that not the case the selection has to be made manually by the user. en64BitArithmetik: 64 bit arithmetic en32BitArithmetik: 32 bit arithmetic If the automatic determination does not work, the following diagnostic number is displayed and the selection must done manually by the user. iErrID: 5 Additional information about 'EN_POS_J_ARITHMETIK' and manual selection: Siehe 'EN_POS_J_ARITHMETIK' auf Seite 601.

1) Only for CODESYS V3

Output variables

Name	Type	Description
boDone	BOOL	Response that the function block has been completely executed.
boBusy	BOOL	Execution message: This bit remains set as long as the block is being processed.
boInPos	BOOL	Message "InPosition": Actual position = target position within the position window
lreActVelocity	LREAL	Velocity with which the positioning is performed. [increments/s]
lreActAccel	LREAL	Actual acceleration value [increments/s^2]
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
enErrName	ENUM	EN_FB_NAME Name of faulty block for which the diagnostic number is output. The diagnostic number is explained in the description of the corresponding library. Block: POSITION_JERK_1 — Library: AmkAfl.lib

Input and output variables

Name	Type	Description
diSetPosition	DINT	Specification of the position setpoint (position setpoint system) [increments]
stDevice	STRUCT	ST_DEVICE The device description structure assigns the block a device. (See document Software description AmkBase Bibliothek, Part no.204986)

Usage note in the CoDeSys program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
	POSITION_RELATIV_VAJ

Associated with this function block, a visualisation is prepared in CoDeSys.
Siehe 'Visualization of AFL blocks' auf Seite 806.

4.2.1.1.3 04_Manual

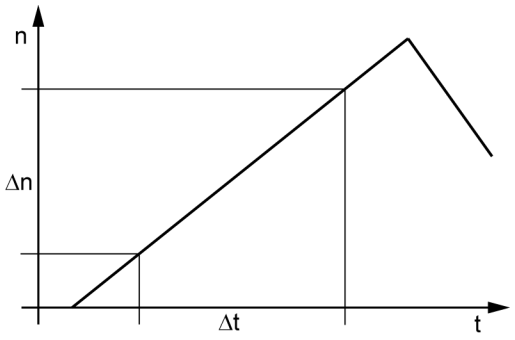
4.2.1.1.3.1 MANUAL_DETERMINE_INERTIA (FB)

The function block 'MANUAL_DETERMINE_INERTIA' calculates the moment of inertia of the drive and the associated mechanicas. To do so, the drive accelerates at the specified torque limit tot he specified maximum velocity.

$J = \frac{M}{\alpha}$ Inertia

$\alpha = \frac{\Delta \omega}{\Delta t}$ Angular acceleration

$\omega = 2 * \pi * n$ Angular velocity
n – speed / 1/s



The function block is called in the asynchronous program level PLC_PRG.
Setpoints and current values are transferred in a synchronous action. The synchronous action must be called in the synchronous program level FPLC_PRG.

User interface

	MANUAL_DETERMINE_INERTIA			
FB execution -	boExec	boDone	-	Ackn. "FB done"
Maximum torque -	diMaxTorque	boBusy	-	FB in progress
Time out -	tTimeOut	boErr	-	Error
		iErrID	-	Error ID
		enErrName	-	Name of faulty FB
Device structure -	stDevice	stDevice	-	Device structure

	act sync			
Maximum velocity -	diMaxVelocity	reInertia	-	Inertia
Actual velocity value -	diActVelocity	diSetSpeed	-	Velocity setpoint

Input variables of the asynchronous program levels (PLC_PRG)

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.
diMaxTorque	DINT	Maximum torque [0.1% Mn]
tTimeOut	TIME	Timeout

Input variables of the synchronous program levels (FPLC_PRG)

Name	Type	Description
diMaxVelocity	DINT	Maximum velocity [0.0001 rpm]
diActVelocity	DINT	Actual velocity [0.0001 rpm]

Output variables of the asynchronous program levels (PLC_PRG)

Name	Type	Description
boDone	BOOL	Response that the function block has been completely executed.
boBusy	BOOL	Execution message: This bit remains set as long as the block is being processed.
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 **enErrName = NONE** iErrID 1: Incorrect entry: Required input variables contain values ≤ 0 => check input variables iErrID 2: Timeout: Within 'tTimeOut', the drive did not accelerate to 'diMaxVelocity' => increase 'diMaxTorque' if necessary **enErrName ≠ NONE** Error number according to 'enErrName'
enErrName	ENUM	EN_FB_NAME Name of faulty block for which the diagnostic number is output. The diagnostic number is explained in the description of the corresponding library. READ_ID_DINT: AmkSystem.lib WRITE_ID_DINT_TMP: AmkSystem.lib
strFbInstance	STRING	Path of the function block

Output variables of the synchronous program levels (FPLC_PRG)

Name	Type	Description
reInertia	REAL	Determined moment of inertia [kg x m^2]
diSetSpeed	DINT	Velocity setpoint [0.0001 rpm]

Input and output variables

Name	Type	Description
stDevice	=>	ST_DEVICE The device description structure assigns the block a device. (See document Software description AmkBase Bibliothek, Part no.204986)

Usage note in the CoDeSys program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
MANUAL_DETERMINE_INERTIA	MANUAL_DETERMINE_INERTIA.actSync

Associated with this function block, a visualisation is prepared in CoDeSys.
Siehe 'Visualization of AFL blocks' auf Seite 806.

4.2.1.1.3.2 MANUAL_JOG (FB)

The function block 'MANUAL_JOG' realises the jog operation (positive/negative) in position control.
The function block is called in the synchronous program level FPLC_PRG.

User interface

		MANUAL_JOG			
FB enable	-	boEnable	boEnabAck	-	Ackn. "FB enable"
Emergency stop	-	boEmergency_Stop	boBusy	-	FB in progress
Emergency stop ramp	-	diEmergency_StopRamp	boErr	-	Error
Jog mode positive	-	boPlus	iErrID	-	Error ID
Jog mode negative	-	boMinus	enErrName	-	Name of faulty FB
Jog velocity	-	udVelocity			
Jog acceleration	-	udAccel			
Position setpoint	-	diSetPosition	diSetPosition	-	Position setpoint
Device structure	-	stDevice	stDevice	-	Device structure

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.
boEmergency_Stop	BOOL	EMERGENCY STOP: The setpoint of the velocity is decreased to zero along the emergency-stop ramp. Once initiated, an emergency stop cannot be aborted.
diEmergency_StopRamp	DINT	EMERGENCY-STOP ramp: Ramp time in which the drive is slowed down from the current velocity to zero [ms].
boPlus / boMinus	BOOL	Jog in positive or negative direction.
udVelocity	UDINT	Velocity for jog mode [increments/s]
udAccel	UDINT	Acceleration/deceleration ramp [increments/s^2]

Output variables

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

Name	Type	Description
boBusy	BOOL	Execution message: This bit remains set as long as the block is being processed.
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
enErrName	ENUM	EN_FB_NAME Name of faulty block for which the diagnostic number is output. The diagnostic number is explained in the description of the corresponding library. Block: VGEN_A — Library: AmkBase.lib Block: RATIO_INC — Library: AmkBase.lib

Input and output variables

Name	Type	Description
diSetPosition	DINT	Specification of the position setpoint (position setpoint system) [increments]
stDevice	STRUCT	ST_DEVICE The device description structure assigns the block a device. (See document Software description AmkBase Bibliothek, Part no.204986)

Usage note in the CoDeSys program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
	MANUAL_JOG

Associated with this function block, a visualisation is prepared in CoDeSys.
Siehe 'Visualization of AFL blocks' auf Seite 806.

4.2.1.1.3.3 MANUAL_JOG_VAJ (FB)

The function block 'MANUAL_JOG_VAJ' realises the jog operation (plus/minus) in position control. In addition to position, speed and acceleration, the user can also specify the jerk.

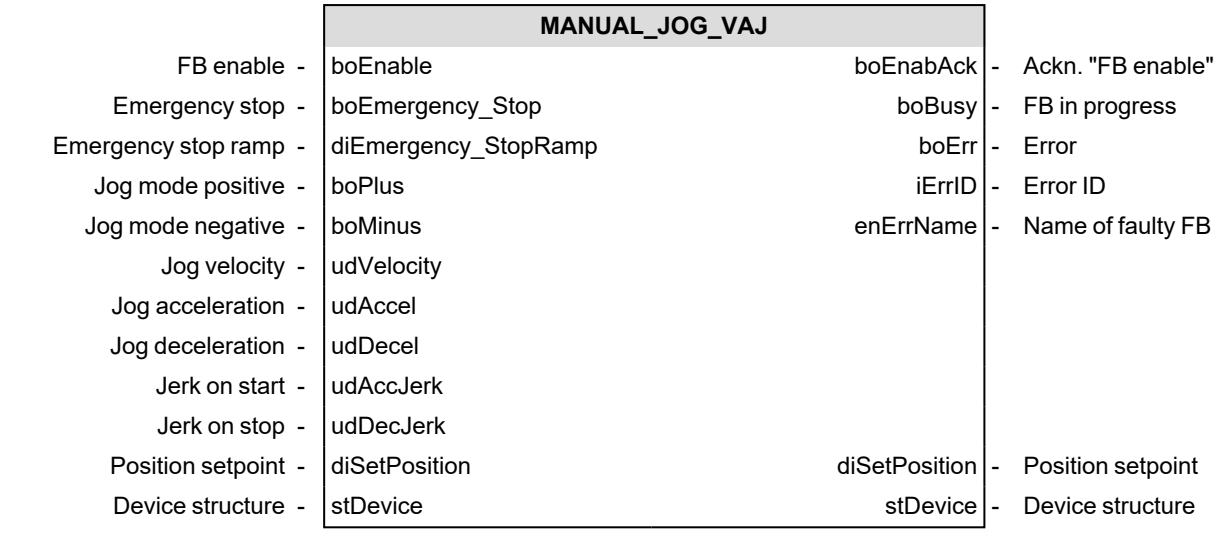
$$v = \frac{\Delta s}{\Delta t}$$ Velocity

$$a = \frac{\Delta v}{\Delta t} = \frac{\Delta s}{\Delta t^2}$$ Acceleration

$$j = \frac{\Delta a}{\Delta t} = \frac{\Delta v}{\Delta t^2} = \frac{\Delta s}{\Delta t^3}$$ Jerk

The function block is called in the synchronous program level FPLC_PRG.

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.
boEmergency_Stop	BOOL	EMERGENCY STOP: The setpoint of the velocity is decreased to zero along the emergency-stop ramp. Once initiated, an emergency stop cannot be aborted.
diEmergency_StopRamp	DINT	EMERGENCY-STOP ramp: Ramp time in which the drive is slowed down from the current velocity to zero [ms].
boPlus / boMinus	BOOL	Jog in positive or negative direction.
udVelocity	UDINT	Velocity for jog mode [increments/s]
udAccel / udDecel	UDINT	Acceleration/deceleration ramp [increments/s^2]
udAccJerk / udDecJerk	UDINT	Jerk for the acceleration/deceleration [increments/s^3]

Output variables

Name	Type	Description
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled
boBusy	BOOL	Execution message: This bit remains set as long as the block is being processed.
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

Name	Type	Description
enErrName	ENUM	EN_FB_NAME Name of faulty block for which the diagnostic number is output. The diagnostic number is explained in the description of the corresponding library. **Block** / **Library** VGEN_AJ / AmkBase.lib RATIO_INC / AmkBase.lib

Input and output variables

Name	Type	Description
diSetPosition	DINT	Specification of the position setpoint (position setpoint system) [increments]
stDevice	STRUCT	ST_DEVICE The device description structure assigns the block a device. (See document Software description AmkBase Bibliothek, Part no.204986)

Usage note in the CoDeSys program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
	MANUAL_JOG_VAJ

Associated with this function block, a visualisation is prepared in CoDeSys.
Siehe 'Visualization of AFL blocks' auf Seite 806.

4.2.1.1.3.4 MANUAL_STEP (FB)

The function block 'MANUAL_STEP' realises step operation (plus/minus) in position control.
The function block is called in the synchronous program level FPLC_PRG.

User interface

		MANUAL_STEP			
FB enable	-	boEnable	boEnabAck	-	Ackn. "FB enable"
Emergency stop	-	boEmergency_Stop	boErr	-	Error
Emergency stop ramp	-	diEmergency_StopRamp	iErrID	-	Error ID
Jog mode positive	-	boPlus	enErrName	-	Name of faulty FB
Jog mode negative	-	boMinus			
Increments	-	diStep			
Step velocity	-	udVelocity			
Step acceleration	-	udAccel			
Position setpoint	-	diSetPosition	diSetPosition	-	Position setpoint
Device structure	-	stDevice	stDevice	-	Device structure

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.
boEmergency_Stop	BOOL	EMERGENCY STOP: The setpoint of the velocity is decreased to zero along the emergency-stop ramp. Once initiated, an emergency stop cannot be aborted.

Name	Type	Description
diEmergency_StopRamp	DINT	EMERGENCY-STOP ramp: Ramp time in which the drive is slowed down from the current velocity to zero [ms].
boPlus / boMinus	BOOL	Jog in positive or negative direction. The drive stops positioning when the signal drops out before reaching the end position.
diStep	DINT	Step width: Maximum distance that is travelled when holding down the Jog button
udVelocity	UDINT	Velocity for step mode [increments/s]
udAccel	UDINT	Acceleration/deceleration ramp [increments/s^2]

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
enErrName	ENUM	EN_FB_NAME Name of faulty block for which the diagnostic number is output. The diagnostic number is explained in the description of the corresponding library. Block: POS_1 — Library: AmkBase.lib

Input and output variables

Name	Type	Description
diSetPosition	DINT	Specification of the position setpoint (position setpoint system) [increments]
stDevice	STRUCT	ST_DEVICE The device description structure assigns the block a device. (See document Software description AmkBase Bibliothek, Part no.204986)

Usage note in the CoDeSys program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
	MANUAL_STEP

Associated with this function block, a visualisation is prepared in CoDeSys.
Siehe 'Visualization of AFL blocks' auf Seite 806.

4.2.1.1.3.5 MANUAL_VELOCITY (FB)

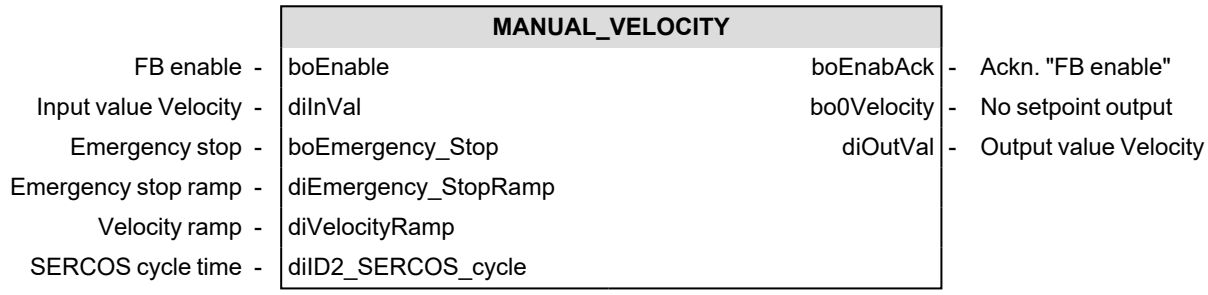
The function block 'MANUAL_VELOCITY' realises speed specification with emergency-stop function. The ramp time 'diVelocityRamp' is constant. Thus, the different setpoint speeds are always reached at the same time.

Example 1: Jump at the input 'diInVal' 0 → 100 [r/min]
Ramp time 1000 [ms]
It follows: The output value 'diOutVal' is ramp within 1s of 0 → 100.0000 [0.0001 r/min].

Example 2: Jump at the input 'diInVal' 0 → 1000 [r/min]
Ramp time 1000 [ms]
It follows: The output value 'diOutVal' is ramp within 1s of 0 → 1000.0000 [0.0001 r/min].

The function block is called in the synchronous program level FPLC_PRG.

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	Specification of the velocity setpoint [rpm] Entry of the velocity setpoint in rpm!
boEmergency_Stop	BOOL	EMERGENCY STOP: The setpoint of the velocity is decreased to zero along the emergency-stop ramp. Once initiated, an emergency stop cannot be aborted.
diEmergency_StopRamp	DINT	EMERGENCY-STOP ramp: Ramp time in which the drive is slowed down from the current velocity to zero [ms].
diVelocityRamp	DINT	Velocity ramp [ms]
diID2_SERCOS_cycle	DINT	ID2 'SERCOS cycle time' [µs]

Output variables

Name	Type	Description
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled
bo0Velocity	BOOL	When 'bo0Vel' is active, no setpoint is output.
diOutVal	DINT	Velocity setpoint to be transferred to the drive [0.0001 rpm] Output of the velocity setpoint in 0.0001 rpm!

Usage note in the CoDeSys program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
	MANUAL_VELOCITY

Associated with this function block, a visualisation is prepared in CoDeSys.
Siehe 'Visualization of AFL blocks' auf Seite 806.

4.2.1.1.3.6 MANUAL_VELOCITY_1 (FB)

The function block 'MANUAL_VELOCITY_1' realises speed specification with emergency-stop function. The rice of the predetermined ramp in 'diVelocityRamp' is constant. Thus, the different setpoint speeds can be reached in different times.

Example 1: Jump at the input 'diInVal' 0 → 100 [r/min]
Ramp rise 100.0000 [0.0001 r/s]
It follows: The output value 'diOutVal' is ramp within 1s of 0 → 100.0000 [0.0001 r/min].

Example 2: Jump at the input 'diInVal' 0 → 1000 [r/min]
Ramp rise 100.0000 [0.0001 r/s]
It follows: The output value 'diOutVal' is ramp within 10s of 0 → 1000.0000 [0.0001 r/min].

The function block is called in the synchronous program level FPLC_PRG.

User interface

		MANUAL_VELOCITY_1			
FB enable	-	boEnable	boEnabAck	-	Ackn. "FB enable"
Emergency stop	-	boEmergency_Stop	bo0Velocity	-	No setpoint output
Emergency stop ramp	-	diEmergency_StopRamp	diOutVal	-	Output value Velocity
Input value Velocity	-	diInVal			
Velocity ramp	-	diVelocityRamp			
SERCOS cycle time	-	diID2_SERCOS_cycle			

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	Specification of the velocity setpoint [rpm] Entry of the velocity setpoint in rpm!
boEmergency_Stop	BOOL	EMERGENCY STOP: The setpoint of the velocity is decreased to zero along the emergency-stop ramp. Once initiated, an emergency stop cannot be aborted.
diEmergency_StopRamp	DINT	EMERGENCY-STOP ramp: Ramp time in which the drive is slowed down from the current velocity to zero [ms].
diVelocityRamp	DINT	Ramp rise [0.0001 r/s]
diID2_SERCOS_cycle	DINT	ID2 'SERCOS cycle time' [µs]

Output variables

Name	Type	Description
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled
bo0Velocity	BOOL	When 'bo0Vel' is active, no setpoint is output.
diOutVal	DINT	Velocity setpoint to be transferred to the drive [0.0001 rpm] Output of the velocity setpoint in 0.0001 rpm!

Usage note in the CoDeSys program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
	MANUAL_VELOCITY_1

Associated with this function block, a visualisation is prepared in CoDeSys.
Siehe 'Visualization of AFL blocks' auf Seite 806.

4.2.1.1.4 06_Follow

4.2.1.1.4.1 FOLLOW_AXIS_CAM_CONT (FB)

The function block 'FOLLOW_AXIS_CAM_CONT' implements the configurable output of a binary signal as a function of the input variable 'diInVal' (e.g. a path information) in the form of a cam switch. That means that the signal levels of the binary output can be defined by a cam preset e.g. depending on actual position values.

The switching points can be defined as input values by presetting the cam switch-on and switch-off positions ('di..CamPositionOn' / 'di..CamPositionOff'). Up to 16 cams can be distributed. The switching points of the cams can be modified while the function block is active ('boEnable' = TRUE).

The following properties can be set by input variables:

- enMode: operation mode
- tFilter: filter time constant
- tDelay delay time
- uiHyst: hysteresis

The function block is called in the synchronous program level FPLC_PRG.

Operation mode

The operation mode is divided to

- enMode = CAM_CONT_INC; incremental evaluation of input values
  The input variable 'diInVal' is processed as 32 bit signed integer value.
  The function block adds up the differences of each two successive input values to a positive 32 bit value. If it exceeds a configurable modulo value (e.g. increments per rotation), the output is reset to 0.
- enMode = CAM_CONT_ABS; absolute evaluation of input values
  The input variable 'diInVal' is processed as 32 bit signed integer value.
  Overshoot at the end of the permitted range will be limited.

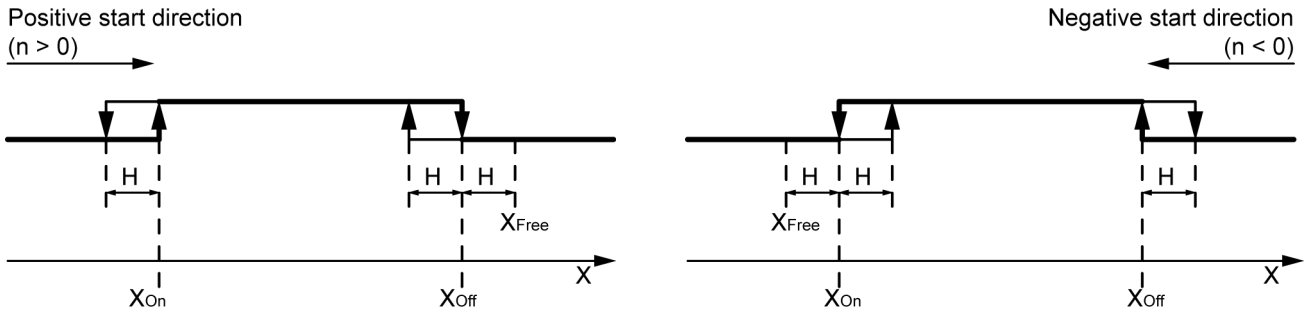
Filter by dead time compensation

In order to minimise the influence of speed changes, the filter averages over several actual speed values. The time constant 'tFilter' leads to the number of speed values to be averaged

uiCycleTime = ID2 'SERCOS cycle time'

Hysteresis

The calculation of the hysteresis (X_{On}, X_{Off}) is shown in the following figures.



Positive start direction (n > 0, X incrementing)

1. Start position $X < X_{On}$ => output cam information = 0
2. If $X_{On} \leq X < X_{Off}$ => output cam information = 1
3. Turn back to $X \geq (X_{On} - H)$ => output cam information = 1
   Further turn back to $X < (X_{On} - H)$ => output cam information = 0
4. Turn forward up to $X \geq X_{Off}$ => output cam information = 0
5. If turn back starts for $X \leq X_{Free}$ ($X_{Free} = X_{Off} + H$) => output cam information 0 until $X \leq (X_{Off} - H)$
   (For further turn back $X \leq (X_{Off} - H)$ => output cam information = 1)
6. If turn back starts for $X \geq X_{Free}$, the cam signal is built according to the 'negative start direction'.

Negative start direction (n < 0; X decrementing)

1. Start position $X \geq X_{Off}$ => output cam information = 0
2. If $X_{On} \leq X < X_{Off}$ => output cam information = 1
3. Turn forward up to $X < (X_{Off} + H)$ => output cam information = 1
   Further turn forward to $X \geq (X_{Off} + H)$ => output cam information = 0
4. Turn back to $X < X_{On}$ => output cam information = 0
5. If turn forward starts for $X > X_{Free}$ ($X_{Free} = X_{On} - H$) => output cam information 0 until $X > (X_{On} + H)$
   (For further turn forward $X \geq (X_{On} + H)$ => output cam information = 1)
6. If turn forward starts for $X < X_{Free}$, the cam signal is built according to the 'positive start direction'.

The changeover from positive to negative start direction and vice versa takes place if the cam has reached or exceeded the position X_{Free} with the same direction the cam was approached.

User interface

		FOLLOW_AXIS_CAM_CONT			
FB enable	-	boEnable	boEnabAck	-	Ackn. "FB enable"
Selection mode Betriebsart	-	enMode	boErr	-	Error
Input value	-	diInVal	iErrID	-	Error ID
Modulo value	-	udModulo	enErrName	-	Name of faulty FB
Filter time	-	tFilter	boOutVal	-	Output signal
Delay time	-	tDelay			
Hysteresis	-	uiHyst			
Number of cams	-	uiNumberOfCams			
Switch-on position cam 1	-	di1CamPositionOn			
Switch-off position cam 1	-	di1CamPositionOff			
	-	:			
	-	:			
Switch-on position cam 16	-	di16CamPositionOn			
Switch-off position cam 16	-	di16CamPositionOff			
Device structure	-	stDevice	stDevice	-	Device structure

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_CAM_CONT_MODE Operation mode (see above) CAM_CONT_INC: incremental evaluation of input values (default) CAM_CONT_ABS: absolute evaluation of input values
diInVal	DINT	Input value (path relation) [increments]
udModulo	UDINT	Modulo value In mode 'enMode' = CAM_CONT_INC, this is the value at which cam table evaluation restarts at "0" Range: 0 … $+2^{31}$-1 Default: 20000
tFilter	TIME	Filter time constant Attenuation of the effect of velocity changes within the dead time compensation. (Default = 1 ms)
tDelay	TIME	Dead-time constant . Time to calculate the offset of the binary information which depends on the actual velocity (default = 0 = inactive)
uiHyst	UINT	Hysteresis (see above) Hysteresis value , enclosing the on and off edges (X_{On}, X_{Off}) of a cam signal [increments] (default = 0 = inactive) In conjunction with a dead time compensation, the hysteresis must be set to a higher value than the dead time compensation distance X_{tot}. X_{tot} = dead time compensation distance [increments] T_{tot} = dead time [ms] n = speed [rpm] G = encoder resolution [increments/round] If 'enMode' = CAM_CONT_INC: $X_{Off} > X_{On}$: H < Modulowert – ($X_{Off} – X_{On}$) $X_{Off} < X_{On}$: H < $X_{On} – X_{Off}$
uiNumberOfCams	UINT	Number of active cams (default = 1)
di1CamPositionOn ... di16CamPositionOn	DINT	Switch-on positions of cam 1 ... 16 [increments]
di1CamPositionOff ... di16CamPositionOff	DINT	Switch-off positions of cam 1 ... 16 [increments]

Output variables

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

Name	Type	Description
boOutVal	BOOL	Output signal of the cam tracks
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
enErrName	ENUM	EN_FB_NAME Name of faulty block for which the diagnostic number is output. The diagnostic number is explained in the description of the corresponding library. Function block: CAM_CONT; Library: AmkBase.lib

Input and output variables

Name	Type	Description
stDevice	STRUCT	ST_DEVICE The device description structure assigns the block a device. (See document Software description AmkBase Bibliothek, Part no.204986)

Usage note in the CoDeSys program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
	FOLLOW_AXIS_CAM_CONT

Associated with this function block, a visualisation is prepared in CoDeSys.
Siehe 'Visualization of AFL blocks' auf Seite 806.

4.2.1.1.4.2 FOLLOW_AXIS_CONTINUOUS_TAB1_TAB2_OUTGEAR32 (FB)

The function block 'FOLLOW_AXIS_CONTINUOUS_TAB1_TAB2_OUTGEAR32' realises continuous follow operation over table functions (cam).

- For this purpose, two different operating tables can be transferred.
- The selection of the active operating table is performed with a binary signal.
- With the input bit 'boEnabOpTab2' = TRUE, operating table 2 is processed.
- Table switching is always performed at the origin to prevent abrupt compensation movements.
- If no second operating table is specified or the input bit 'boEnabOpTab2' = FALSE, operating table 1 is processed.
- The minimum requirement is operating table 1.

It is possible to transfer operating tables with more than 360 support points to the block.

In addition, the block also has an output gear stage. This makes it possible to modify standardised cams to various situations.

The calculation of the gear stage is performed in 32-bit mode, meaning that a maximum of 16-bit values can be transferred for the multiplier/divider.

The function block is called in the synchronous program level FPLC_PRG.

User interface

	FOLLOW_AXIS_CONTINUOUS_TAB1_TAB2_OUTGEAR32			
FB enable -	boEnable	boEnabAck	-	Ackn. "FB enable"
Enable operating table -	boEnabOpTab2	boTab1Busy	-	Table in progress
Emergency stop -	boEmergency_Stop	boTab2Busy	-	Table in progress
Emergency stop ramp -	diEmergency_StopRamp	boDone	-	Ackn. "table done"
SERCOS cycle time -	diID2_SERCOS_cycle	boErr	-	Error
Gear multiplier -	iMultiplier	iErrID	-	Error ID
Gear divider -	uiDivider	enErrName	-	Name of faulty FB
Input value -	diInVal			
Operating table -	pstOpTab1			
Operating table -	pstOpTab2			
Position setpoint -	diSetPosition	diSetPosition	-	Position setpoint

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.
boEnabOpTab2	BOOL	• For 'boEnabOpTab2' = TRUE, the second operating table is enabled; switching is performed at the coordinate origin. • With 'boEnabOpTab2' = FALSE, operating table 1 is used again. This switch is also performed at the coordinate origin.
boEmergency_Stop	BOOL	EMERGENCY STOP: The setpoint of the velocity is decreased to zero along the emergency-stop ramp. Once initiated, an emergency stop cannot be aborted.
diEmergency_StopRamp	DINT	EMERGENCY-STOP ramp: Ramp time in which the drive is slowed down from the current velocity to zero [ms].
diID2_SERCOS_cycle	DINT	ID2 'SERCOS cycle time' [µs]
iMultiplier	INT	Gear multiplier/factor [1]
uiDivider	UINT	Gear divider [1]
diInVal	DINT	Input increments from master encoder (increment source that should be followed) [increments]
pstOpTab1 / pstOpTab2	POINTER	POINTER TO ST_PROF_TAB Reference to operating table (table-supported cam) (See document Software description AmkBase Bibliothek, Part no. 204986)

Output variables

Name	Type	Description
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled
boTab1Busy / boTab2Busy	BOOL	Table processing is ongoing.
boDone	BOOL	Table processing is complete.
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
enErrName	ENUM	EN_FB_NAME Name of faulty block for which the diagnostic number is output. The diagnostic number is explained in the description of the corresponding library. Block: CAM_PROF — Library: AmkBase.lib

Input and output variables

Name	Type	Description
diSetPosition	DINT	Specification of the position setpoint (position setpoint system) [increments]

Usage note in the CoDeSys program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
	FOLLOW_AXIS_CONTINUOUS_TAB1_TAB2_OUTGEAR32

Associated with this function block, a visualisation is prepared in CoDeSys.
Siehe 'Visualization of AFL blocks' auf Seite 806.

4.2.1.1.4.3 FOLLOW_AXIS_CONTINUOUS_TABLE1_TABLE2 (FB)

The function block FOLLOW_AXIS_CONTINUOUS_TABLE1_TABLE2 realises continuous follow operation over table functions (cam).

- For this purpose, two different operating tables can be transferred.
- The selection of the active operating table is performed with a binary signal.
- With the input bit 'boEnabOpTab2' = TRUE, operating table 2 is processed.
- Table switching is always performed at the origin to prevent abrupt compensation movements.
- If no second operating table is specified or the input bit 'boEnabOpTab2' = FALSE, operating table 1 is processed.
- The minimum requirement is operating table 1.

It is possible to transfer operating tables with more than 360 support points to the block.

The function block is called in the synchronous program level FPLC_PRG.

User interface

	FOLLOW_AXIS_CONTINUOUS_TABLE1_TABLE2			
FB enable -	boEnable	boEnabAck	-	Ackn. "FB enable"
Enable operating table -	boEnabOpTab2	boTab1Busy	-	Table in progress
Emergency stop -	boEmergency_Stop	boTab2Busy	-	Table in progress
Emergency stop ramp -	diEmergency_StopRamp	boDone	-	Ackn. "table done"
SERCOS cycle time -	diID2_SERCOS_cycle	boErr	-	Error
Input value -	diInVal	iErrID	-	Error ID
Operating table -	pstOpTab1	enErrName	-	Name of faulty FB
Operating table -	pstOpTab2			
Position setpoint -	diSetPosition	diSetPosition	-	Position setpoint

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.
boEnabOpTab2	BOOL	• For 'boEnabOpTab2' = TRUE, the second operating table is enabled; switching is performed at the coordinate origin. • With 'boEnabOpTab2' = FALSE, operating table 1 is used again. This switch is also performed at the coordinate origin.
boEmergency_Stop	BOOL	EMERGENCY STOP: The setpoint of the velocity is decreased to zero along the emergency-stop ramp. Once initiated, an emergency stop cannot be aborted.
diEmergency_StopRamp	DINT	EMERGENCY-STOP ramp: Ramp time in which the drive is slowed down from the current velocity to zero [ms].
diID2_SERCOS_cycle	DINT	ID2 'SERCOS cycle time' [µs]
diInVal	DINT	Input increments from master encoder (increment source that should be followed) [increments]
pstOpTab1 / pstOpTab2	POINTER	POINTER TO ST_PROF_TAB Reference to operating table (table-supported cam) (See document Software description AmkBase Bibliothek, Part no. 204986)

Output variables

Name	Type	Description
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled
boTab1Busy / boTab2Busy	BOOL	Table processing is ongoing.
boDone	BOOL	Table processing is complete.
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
enErrName	ENUM	EN_FB_NAME Name of faulty block for which the diagnostic number is output. The diagnostic number is explained in the description of the corresponding library. Block: CAM_PROF — Library: AmkBase.lib

Input and output variables

Name	Type	Description
diSetPosition	DINT	Specification of the position setpoint (position setpoint system) [increments]

Usage note in the CoDeSys program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
	FOLLOW_AXIS_CONTINUOUS_TABLE1_TABLE2

Associated with this function block, a visualisation is prepared in CoDeSys.
Siehe 'Visualization of AFL blocks' auf Seite 806.

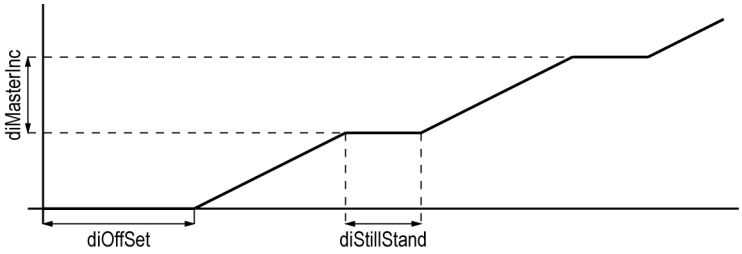
4.2.1.1.4.4 FOLLOW_AXIS_CRANK_DRIVE_EMULATION (FB)

The function block 'FOLLOW_AXIS_CRANK_DRIVE_EMULATION' emulates a crank gearing drive.

The function block provides a way-way-function with standstill area. It can be used for example in film transport and cut with stationary blade.

- The curve starts always at the end of the standstill area.
- By parameterisation, different junctions can be selected:
  - 45° curve
  - Sinus curve
  - Sinus² curve

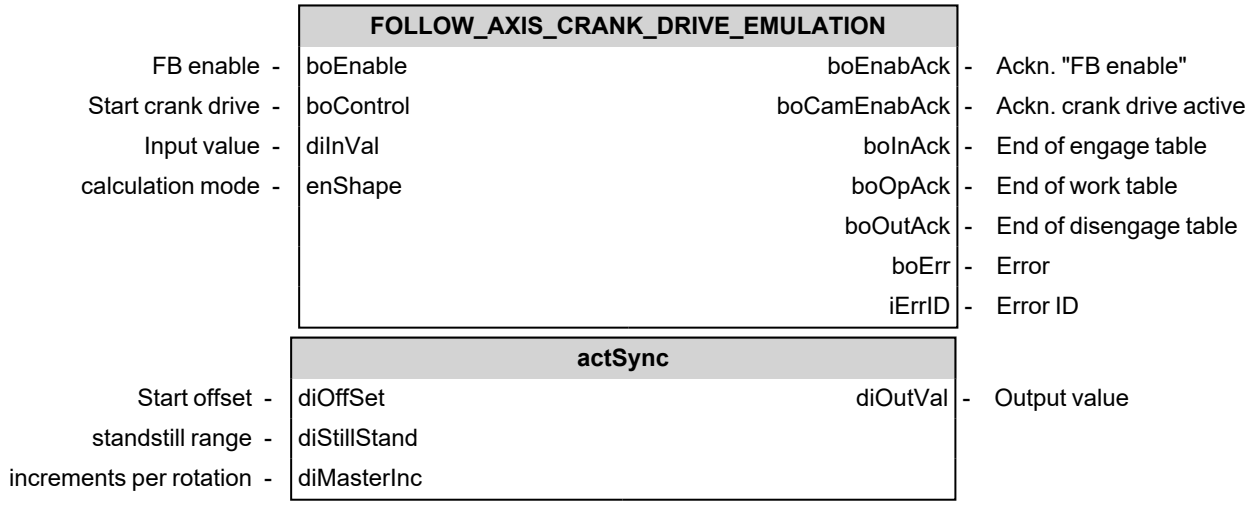
By means of an offset at the start of the movement, the curve can be synchronised to an external starting point.



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

Setpoints and current values are transferred in a synchronous action. The synchronous action must be called in the synchronous program level FPLC_PRG.

User interface



Input variables of the asynchronous part of the program (PLC_PRG)

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 of the crank drive function
diInVal	DINT	Input value pulses [increments]

Name	Type	Description
enShape	ENUM	EN_CRANK_DRIVE_SHAPE Shape of crank drive curve enCrankShapeLinear: linear 45° straight line enCrankShapeSine: SINUS - function enCrankShapeSquareSine: $SINUS^2$ - function

Input variables of the synchronous part of the program (FPLC_PRG)

Name	Type	Description
diOffSet	DINT	Offset at the start of the first revolution. [Inkr] The offset moves the movement function to the back.
diStillStand	DINT	Stability range at the end of the revolutions [incr]
diMasterInc	DINT	Number of increments per revolution (without start - offset) [incr.]

Output variables of the asynchronous part of the program (PLC_PRG)

Name	Type	Description
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled
boCamEnabAck	BOOL	Acknowledgement: Cranc drive active
boInAck	BOOL	Acknowledge end of phasing in table Pulse at end of phasing in table; 'boInAck' = TRUE for 2 sampling time points
boOpAck	BOOL	Acknowledge end of operating table Pulse at end of operating table; 'boOpAck' = TRUE for 2 sampling time points
boOutAck	BOOL	Acknowledge end of phasing out table Pulse at end of phasing out table; 'boOutAck' = TRUE for 2 sampling time points
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 1 - 99: Error code according to 'CAM_PROF' (AmkBase.lib) 100: Incorrect value in variable 'enShape' 101: Parameter 'diMasterInc' ≤ 0 102: Parameter 'diStillStand' to large ('diStillStand' ≤ 'diMasterInc' / 2) 103: Parameter 'diOffSet' < 0

Output variables of the synchronous part of the program (FPLC_PRG)

Name	Type	Description
diOutVal	DINT	Output value - impuls [Incr] (can, for example, be issued with 'SET_VAL_I' to the drive)

Usage note in the CoDeSys program

PLC_PRG (asynchronous part of the programm)	FPLC_PRG (synchronous part of the programm)
FOLLOW_AXIS_CRANK_DRIVE_EMULATION	FOLLOW_AXIS_CRANK_DRIVE_EMULATION.actSync

Associated with this function block, a visualisation is prepared in CoDeSys.
Siehe 'Visualization of AFL blocks' auf Seite 806.

4.2.1.1.4.5 FOLLOW_AXIS_SWITCHING_TABLE1_TABLE2 (FB)

The function block FOLLOW_AXIS_SWITCHING_TABLE1_TABLE2 realises continuous follow operation over table functions (cam).

- Via a phasing in/phasing out table, the operating table is phased in or out.
- If no phasing in table is transferred, phasing in is performed with the selected operating table.
- The same applies for the phasing out procedure: Without a transferred phasing out table, phasing out is performed with the active operating table when the execution signal 'boExec' is withdrawn.
- Two different operating tables can be transferred.
- With the input bit 'boEnabOpTab2' = TRUE, operating table 2 is activated. Table switching is always performed at the origin to prevent abrupt compensation movements.
- If no second operating table is specified or the input bit 'boEnabOpTab2' = FALSE, operating table 1 is processed.
- With the positive edge of 'boEnable', detection of input increments is activated. This means that, starting at this point in time, increment changes to the input variable 'diInVal' cause a change of the master reference point (X).
- The phasing in. operating and phasing out tables assigned by 'pstInTab', 'pstOpTab' and 'pstOutTab' are evaluated. The minimum requirement is the operating table.
- Control of the phase in/phase out procedure is performed with the input 'boExec'.
- The following applies: For a positive edge at 'boExec' and X ≤ 'udInAngle', the table interpolator is activated. This means, at the setpoint sink 'diSetPosition', increment changes are output according to the phasing in table.
- At the end of the phasing in table, a transition occurs to the operating table.
- The operating table is processed until a negative edge is detected at 'boExec' and X ≤ 'udOutAngle'. Then a transition to the phasing out table occurs with automatic stop of the interpolation at the end of the phasing out table.

It is possible to transfer operating tables with more than 360 support points to the block.

The function block is called in the synchronous program level FPLC_PRG.

User interface

		FOLLOW_AXIS_SWITCHING_TABLE1_TABLE2			
FB enable	-	boEnable	boEnabAck	-	Ackn. "FB enable"
Enable operating table	-	boEnabOpTab2	boTab1Busy	-	Table in progress
FB execution	-	boExec	boTab2Busy	-	Table in progress
Emergency stop	-	boEmergency_Stop	boDone	-	Ackn. "table done"
Emergency stop ramp	-	diEmergency_StopRamp	boErr	-	Error
SERCOS cycle time	-	diID2_SERCOS_cycle	iErrID	-	Error ID
Input value	-	diInVal	enErrName	-	Name of faulty FB
Phasing in angle	-	udInAngle			
Phasing out angle	-	udOutAngle			
Phasing in table	-	pstInTab			
Operating table	-	pstOpTab1			
Operating table	-	pstOpTab2			
Phasing out table	-	pstOutTab			
Position setpoint	-	diSetPosition	diSetPosition	-	Position setpoint

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.
boEnabOpTab2	BOOL	• For 'boEnabOpTab2' = TRUE, the second operating table is enabled; switching is performed at the coordinate origin. • With 'boEnabOpTab2' = FALSE, operating table 1 is used again. This switch is also performed at the coordinate origin.
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.
boEmergency_Stop	BOOL	EMERGENCY STOP: The setpoint of the velocity is decreased to zero along the emergency-stop ramp. Once initiated, an emergency stop cannot be aborted.
diEmergency_StopRamp	DINT	EMERGENCY-STOP ramp: Ramp time in which the drive is slowed down from the current velocity to zero [ms].
diID2_SERCOS_cycle	DINT	ID2 'SERCOS cycle time' [µs]
diInVal	DINT	Input increments from master encoder (increment source that should be followed) [increments]
udInAngle / udOutAngle	UDINT	Phase in or phase out range on the table X-axis. This values specifies the maximum table X-position from the coordinate origin (0) at which phasing in or phasing out is still permissible. (0... udInAngle)
pstInTab / pstOutTab	POINTER	POINTER TO ST_PROF_TAB Reference to phasing in or phasing out table (table-supported cam) (See document Software description AmkBase Bibliothek, Part no.204986)
pstOpTab1 / pstOpTab2	POINTER	POINTER TO ST_PROF_TAB Reference to operating table (table-supported cam) (See document Software description AmkBase Bibliothek, Part no.204986)

Output variables

Name	Type	Description
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled
boTab1Busy / boTab2Busy	BOOL	Table processing is ongoing.
boDone	BOOL	Table processing is complete.
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

Name	Type	Description
enErrName	ENUM	EN_FB_NAME Name of faulty block for which the diagnostic number is output. The diagnostic number is explained in the description of the corresponding library. **Block** / **Library** CAM_PROF / AmkBase.lib

Input and output variables

Name	Type	Description
diSetPosition	DINT	Specification of the position setpoint (position setpoint system) [increments]

Usage note in the CoDeSys program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
	FOLLOW_AXIS_SWITCHING_TABLE1_TABLE2

Associated with this function block, a visualisation is prepared in CoDeSys.
Siehe 'Visualization of AFL blocks' auf Seite 806.

4.2.1.1.4.6 FOLLOW_AXIS_TABLE_CONTINUOUS (FB)

The function block 'FOLLOW_AXIS_TABLE_CONTINUOUS' realises continuous follow operation over table functions (cam).

Here the increments of the master (of the encoder that is to be followed) at the input 'diInVal' and the follow table to be processed are to be transferred as pointer at the input 'pstOpTab'. It is possible to transfer operating tables with more than 360 support points to the block.

The function block is called in the synchronous program level FPLC_PRG.

User interface

		FOLLOW_AXIS_TABLE_CONTINUOUS			
FB enable	-	boEnable	boEnabAck	-	Ackn. "FB enable"
Input value	-	diInVal	boErr	-	Error
Emergency stop	-	boEmergency_Stop	iErrID	-	Error ID
Emergency stop ramp	-	diEmergency_StopRamp	enErrName	-	Name of faulty FB
SERCOS cycle time	-	diID2_SERCOS_cycle		-	
Operating table	-	pstOpTab		-	
Position setpoint	-	diSetPosition	diSetPosition	-	Position setpoint

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 increments from master encoder (increment source that should be followed) [increments]
boEmergency_Stop	BOOL	EMERGENCY STOP: The setpoint of the velocity is decreased to zero along the emergency-stop ramp. Once initiated, an emergency stop cannot be aborted.
diEmergency_StopRamp	DINT	EMERGENCY-STOP ramp: Ramp time in which the drive is slowed down from the current velocity to zero [ms].
diID2_SERCOS_cycle	DINT	ID2 'SERCOS cycle time' [µs]

Name	Type	Description
pstOpTab	POINTER	POINTER TO ST_PROF_TAB Reference to operating table (table-supported cam) (See document Software description AmkBase Bibliothek, Part no. 204986)

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
enErrName	ENUM	EN_FB_NAME Name of faulty block for which the diagnostic number is output. The diagnostic number is explained in the description of the corresponding library. Block: CAM_PROF; Library: AmkBase.lib

Input and output variables

Name	Type	Description
diSetPosition	DINT	Specification of the position setpoint (position setpoint system) [increments]

Usage note in the CoDeSys program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
	FOLLOW_AXIS_TABLE_CONTINUOUS

Associated with this function block, a visualisation is prepared in CoDeSys.
Siehe 'Visualization of AFL blocks' auf Seite 806.

4.2.1.1.4.7 FOLLOW_AXIS_TABLE_ENGAGE_DISENGAGE (FB)

The function block 'FOLLOW_AXIS_TABLE_ENGAGE_DISENGAGE' realises continuous follow operation over table functions (cam).

- Via a phasing in/phasing out table, the operating table can be phased in or out.
- With the positive edge of 'boEnable', detection of input increments is activated. This means that, starting at this point in time, increment changes to the input variables 'diInVal' cause a change of the master reference point (X).
- The phasing in. operating and phasing out tables assigned by 'pstInTab', 'pstOpTab' and 'pstOutTab' are evaluated. The minimum requirement is the operating table.
- Control of the phase in/phase out procedure is performed with the input 'boExec' or 'boTableControl'.
- The following applies: For a positive edge at 'boExec' (etg. 'boTableControl') and "X ≤ udInAngle", the table interpolator is activated. This means, at the setpoint sink 'diSetPosition', increment changes are output according to the phasing in table.
- At the end of the phasing in table, a transition occurs to the operating table. The operating table is processed until a negative edge is detected at 'boExec' and "X ≤ udOutAngle" or listed under 'uiOpNo' number is processed
- Then a transition to the phasing out table occurs with automatic stop of the interpolation at the end of the phasing out table.

It is possible to transfer operating tables with more than 360 support points to the block.

The function block is called in the synchronous program level FPLC_PRG.

User interface

	FOLLOW_AXIS_TABLE_ENGAGE_DISENGAGE			
FB enable -	boEnable	boEnabAck	-	Ackn. "FB enable"
FB execution -	boExec	boBusy	-	FB in progress
Emergency stop -	boEmergency_Stop	boDone	-	Ackn. "table done"
Emergency stop ramp -	diEmergency_StopRamp	boErr	-	Error
SERCOS cycle time -	diID2_SERCOS_cycle	iErrID	-	Error ID
Input value -	diInVal	enErrName	-	Name of faulty FB
Phasing in angle -	udInAngle			
Phasing out angle -	udOutAngle			
Phasing in table -	pstInTab			
Operating table -	pstOpTab			
Phasing out table -	pstOutTab			
Position setpoint -	diSetPosition	diSetPosition	-	Position setpoint

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.
boEmergency_Stop	BOOL	EMERGENCY STOP: The setpoint of the velocity is decreased to zero along the emergency-stop ramp. Once initiated, an emergency stop cannot be aborted.
diEmergency_StopRamp	DINT	EMERGENCY-STOP ramp: Ramp time in which the drive is slowed down from the current velocity to zero [ms].
diID2_SERCOS_cycle	DINT	ID2 'SERCOS cycle time' [µs]
diInVal	DINT	Input increments from master encoder (increment source that should be followed) [increments]
udInAngle / udOutAngle	UDINT	Phase in or phase out range on the table X-axis. This values specifies the maximum table X-position from the coordinate origin (0) at which phasing in or phasing out is still permissible. (0... udInAngle)
pstInTab / pstOutTab	POINTER	POINTER TO ST_PROF_TAB Reference to phasing in or phasing out table (table-supported cam) (See document Software description AmkBase Bibliothek, Part no. 204986)
pstOpTab	POINTER	POINTER TO ST_PROF_TAB Reference to operating table (table-supported cam) (See document Software description AmkBase Bibliothek, Part no. 204986)

Output variables

Name	Type	Description
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled
boBusy	BOOL	Table processing is ongoing.
boDone	BOOL	Table processing is complete.

<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></td></tr>
<tr><td>enErrName</td><td>ENUM</td><td>EN_FB_NAME
Name of faulty block for which the diagnostic number is output.
The diagnostic number is explained in the description of the corresponding library.<table><tr><th>Block</th><th>Library</th></tr><tr><td>CAM_PROF</td><td>AmkBase.lib</td></tr></table></td></tr>
</table>

Input and output variables

Name	Type	Description
diSetPosition	DINT	Specification of the position setpoint (position setpoint system) [increments]

Usage note in the CoDeSys program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
	FOLLOW_AXIS_TABLE_ENGAGE_DISENGAGE

Associated with this function block, a visualisation is prepared in CoDeSys.
Siehe 'Visualization of AFL blocks' auf Seite 806.

4.2.1.1.4.8 FOLLOW_AXIS_TABLE_START_AUTOSTOP (FB)

The function block 'FOLLOW_AXIS_TABLE_START_AUTOSTOP' realises continuous follow operation over table functions (cam).

- Via a phasing in/phasing out table, the operating table can be phased in and, after processing of the specified table cycles, automatically phased out.
- With the positive edge of 'boEnable', detection of input increments is activated. This means that, starting at this point in time, increment changes to the input variable 'diInVal' cause a change of the master reference point (X).
- The phasing in. operating and phasing out tables assigned by 'pstInTab', 'pstOpTab' and 'pstOutTab' are evaluated. The minimum requirement is the operating table.
- Control of the phase in/phase out procedure is performed with the input 'boExec'.
- The following applies: For a positive edge at 'boExec' and "X ≤ udInAngle", the table interpolator is activated. This means, at the setpoint sink 'diSetPosition', increment changes are output according to the phasing in table.
- At the end of the phasing in table, a transition occurs to the operating table. The operating table is processed until the number of cycles specified under 'uiOpNo' has been reached.
- Then a transition to the phasing out table occurs with automatic stop of the interpolation at the end of the phasing out table.

It is possible to transfer operating tables with more than 360 support points to the block.

The function block is called in the synchronous program level FPLC_PRG.

User interface

	FOLLOW_AXIS_TABLE_START_AUTOSTOP		
FB enable -	boEnable	boEnabAck	- Ackn. "FB enable"
FB execution -	boExec	boBusy	- FB in progress
Emergency stop -	boEmergency_Stop	boDone	- Ackn. "table done"
Emergency stop ramp -	diEmergency_StopRamp	boErr	- Error
SERCOS cycle time -	diID2_SERCOS_cycle	iErrID	- Error ID
Input value -	diInVal	enErrName	- Name of faulty FB
Number of table operations -	uiOpNo		
Phasing in angle -	udInAngle		
Phasing in table -	pstInTab		
Operating table -	pstOpTab		
Phasing out table -	pstOutTab		
Position setpoint -	diSetPosition	diSetPosition	- Position setpoint

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.
boEmergency_Stop	BOOL	EMERGENCY STOP: The setpoint of the velocity is decreased to zero along the emergency-stop ramp. Once initiated, an emergency stop cannot be aborted.
diEmergency_StopRamp	DINT	EMERGENCY-STOP ramp: Ramp time in which the drive is slowed down from the current velocity to zero [ms].
diID2_SERCOS_cycle	DINT	ID2 'SERCOS cycle time' [µs]
diInVal	DINT	Input increments from master encoder (increment source that should be followed) [increments]
uiOpNo	UINT	Number of operating table cycles to be processed [1]
udInAngle	UDINT	Phase in or phase out range on the table X-axis. This values specifies the maximum table X-position from the coordinate origin (0) at which phasing in or phasing out is still permissible. (0... udInAngle)
pstInTab / pstOutTab	POINTER	POINTER TO ST_PROF_TAB Reference to phasing in or phasing out table (table-supported cam) (See document Software description AmkBase Bibliothek, Part no. 204986)
pstOpTab	POINTER	POINTER TO ST_PROF_TAB Reference to operating table (table-supported cam) (See document Software description AmkBase Bibliothek, Part no. 204986)

Output variables

Name	Type	Description
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled
boBusy	BOOL	Table processing is ongoing.

Name	Type	Description
boDone	BOOL	Table processing is complete.
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
enErrName	ENUM	EN_FB_NAME Name of faulty block for which the diagnostic number is output. The diagnostic number is explained in the description of the corresponding library. Block: CAM_PROF — Library: AmkBase.lib

Input and output variables

Name	Type	Description
diSetPosition	DINT	Specification of the position setpoint (position setpoint system) [increments]

Usage note in the CoDeSys program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
	FOLLOW_AXIS_TABLE_START_AUTOSTOP

Associated with this function block, a visualisation is prepared in CoDeSys.
Siehe 'Visualization of AFL blocks' auf Seite 806.

4.2.1.1.4.9 FOLLOW_AXIS_TABLE_START_AUTOSTOP_1 (FB)

The function block 'FOLLOW_AXIS_TABLE_START_AUTOSTOP_1' realises continuous follow operation over table functions (cam).

- Via a phasing in/phasing out table, the operating table can be phased in and, after processing of the specified table cycles, automatically phased out.
- With the positive edge of 'boEnable', detection of input increments is activated. This means that, starting at this point in time, increment changes to the input variable 'diInVal' cause a change of the master reference point (X).
- The phasing in. operating and phasing out tables assigned by 'pstInTab', 'pstOpTab' and 'pstOutTab' are evaluated. The minimum requirement is the operating table.
- Control of the phase in/phase out procedure is performed with the input 'boExec'.
- The following applies: For a positive edge at 'boExec' and "X ≤ udInAngle", the table interpolator is activated. This means, at the setpoint sink 'diSetPosition', increment changes are output according to the phasing in table.
- At the end of the phasing in table, a transition occurs to the operating table. The operating table is processed until the number of cycles specified under 'uiOpNo' has been reached.
- Then a transition to the phasing out table occurs with automatic stop of the interpolation at the end of the phasing out table.

The function block 'FOLLOW_AXIS_TABLE_START_AUTOSTOP_1' is substantially equal to the function block 'FOLLOW_AXIS_TABLE_START_AUTOSTOP'.

The two blocks differ as follows:

- The pointer to the following tables are constantly accept when 'boEnable' = TRUE. An 0→1 edge change at 'boEnable' is no longer necessary
- If only one operating table passed, the output value will stop automatically at the end of the operating table. The table is so executed at least once. The phasing in and phasing out cycle is eliminated. In this case, the operating table is used for the phasing in and phasing out cycle of the function block 'FOLLOW_AXIS_TABLE_START_AUTOSTOP'. The automatic stop is possible after a minimum of 3 passes.

It is possible to transfer operating tables with more than 360 support points to the block.

The function block is called in the synchronous program level FPLC_PRG.

User interface

		FOLLOW_AXIS_TABLE_START_AUTOSTOP_1			
FB enable	-	boEnable	boEnabAck	-	Ackn. "FB enable"
FB execution	-	boExec	boBusy	-	FB in progress
Emergency stop	-	boEmergency_Stop	boDone	-	Ackn. "table done"
Emergency stop ramp	-	diEmergency_StopRamp	boErr	-	Error
SERCOS cycle time	-	diID2_SERCOS_cycle	iErrID	-	Error ID
Input value	-	diInVal	enErrName	-	Name of faulty FB
Number of table operations	-	uiOpNo			
Phasing in angle	-	udInAngle			
Phasing in table	-	pstInTab			
Operating table	-	pstOpTab			
Phasing out table	-	pstOutTab			
Position setpoint	-	diSetPosition	diSetPosition	-	Position setpoint

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
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.
boEmergency_Stop	BOOL	EMERGENCY STOP: The setpoint of the velocity is decreased to zero along the emergency-stop ramp. Once initiated, an emergency stop cannot be aborted.
diEmergency_StopRamp	DINT	EMERGENCY-STOP ramp: Ramp time in which the drive is slowed down from the current velocity to zero [ms].
diID2_SERCOS_cycle	DINT	ID2 'SERCOS cycle time' [µs]
diInVal	DINT	Input increments from master encoder (increment source that should be followed) [increments]
uiOpNo	UINT	Number of operating table cycles to be processed [1]
udInAngle	UDINT	Phase in or phase out range on the table X-axis. This values specifies the maximum table X-position from the coordinate origin (0) at which phasing in or phasing out is still permissible. (0... udInAngle)
pstInTab / pstOutTab	POINTER	POINTER TO ST_PROF_TAB Reference to phasing in or phasing out table (table-supported cam) (See document Software description AmkBase Bibliothek, Part no. 204986)
pstOpTab	POINTER	POINTER TO ST_PROF_TAB Reference to operating table (table-supported cam) (See document Software description AmkBase Bibliothek, Part no. 204986)

Output variables

Name	Type	Description
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled
boBusy	BOOL	Table processing is ongoing.
boDone	BOOL	Table processing is complete.
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
enErrName	ENUM	EN_FB_NAME Name of faulty block for which the diagnostic number is output. The diagnostic number is explained in the description of the corresponding library. Block: CAM_PROF — Library: AmkBase.lib

Input and output variables

Name	Type	Description
diSetPosition	DINT	Specification of the position setpoint (position setpoint system) [increments]

Usage note in the CoDeSys program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
	FOLLOW_AXIS_TABLE_START_AUTOSTOP_1

Associated with this function block, a visualisation is prepared in CoDeSys.
Siehe 'Visualization of AFL blocks' auf Seite 806.

4.2.1.1.4.10 FOLLOW_RATIO_INC_64 (FB)

The function block 'FOLLOW_RATIO_INC_64' realises electronic gearing.

The input increments are modified by a multiplier and divider so that the corresponding ratio is available at the output. The calculated of the ratio is performed with 64 bits, so that multipliers and dividers that are 32-bit can be used.

The function block is called in the synchronous program level FPLC_PRG.

User interface

		FOLLOW_RATIO_INC_64			
FB enable	-	boEnable	boEnabAck	-	Ackn. "FB enable"
Emergency stop	-	boEmergency_Stop	boErr	-	Error
Emergency stop ramp	-	diEmergency_StopRamp	iErrID	-	Error ID
SERCOS cycle time	-	diID2_SERCOS_cycle	enErrName	-	Name of faulty FB
Input value	-	diInVal			
Gear multiplier	-	diMultiplier			
Gear divider	-	udDivider			
Position setpoint	-	diSetPosition	diSetPosition	-	Position setpoint

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.
boEmergency_Stop	BOOL	EMERGENCY STOP: The setpoint of the velocity is decreased to zero along the emergency-stop ramp. Once initiated, an emergency stop cannot be aborted.
diEmergency_StopRamp	DINT	EMERGENCY-STOP ramp: Ramp time in which the drive is slowed down from the current velocity to zero [ms].
diID2_SERCOS_cycle	DINT	ID2 'SERCOS cycle time' [µs]
diInVal	DINT	Input increments from master encoder (increment source that should be followed) [increments]
diMultiplier	DINT	Gear multiplier/factor [1]
udDivider	UDINT	Gear divider [1]

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></td></tr>
<tr><td>enErrName</td><td>ENUM</td><td>EN_FB_NAME
Name of faulty block for which the diagnostic number is output.
The diagnostic number is explained in the description of the corresponding library.<table><tr><th>Block</th><th>Library</th></tr><tr><td>RATIO_INC_1</td><td>AmkBase.lib</td></tr></table></td></tr>
</table>

Input and output variables

Name	Type	Description
diSetPosition	DINT	Specification of the position setpoint (position setpoint system) [increments]

Usage note in the CoDeSys program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
	FOLLOW_RATIO_INC_64

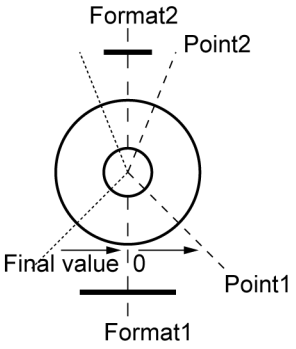
Associated with this function block, a visualisation is prepared in CoDeSys.
Siehe 'Visualization of AFL blocks' auf Seite 806.

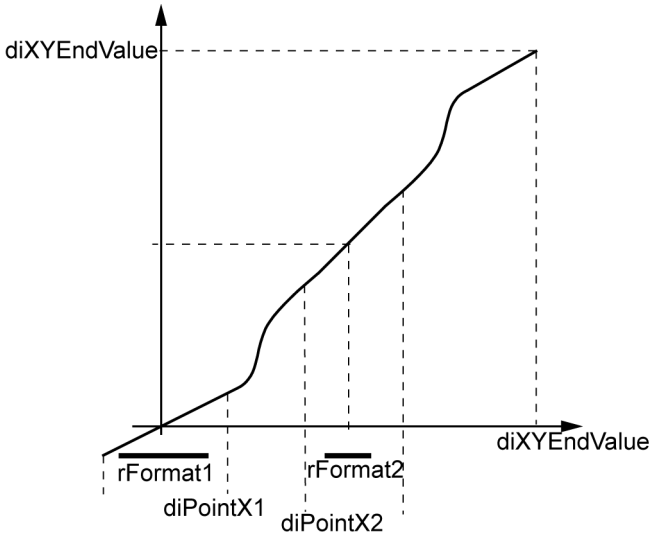
4.2.1.1.5 07_Convert/Calc

4.2.1.1.5.1 CALC_PROF_2_FORMAT_POLY (FB)

The function block 'CALC_PROF_2_FORMAT_POLY' is used to calculate a table profile.

Two formats, which oppose each other, are connected to each other with a compensation cycle according to a 5th order polynomial. The calculation results are stored in an ARRAY.





The call of this function block is permissible in both the asynchronous program section PLC_PRG as well as the synchronous program section FPLC_PRG.

User interface

	CALC_PROF_2_FORMAT_POLY			
FB execution -	boExec	boDone	-	Ackn. "FB done"
End value X and Y axis -	diXYEndValue			
Format -	rFormat1			
Format -	rFormat2			
Perimeter -	rPerimeter			
Point on X axis -	diPointX1			
Point on X axis -	diPointX2			
Table value -	arTab_A	arTab_A	-	Table value

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.
diXYEndValue	DINT	End value of the X and Y axis of the table
rFormat1	REAL	Format length 1 [mm]
rFormat2	REAL	Format length 2 [mm]
rPerimeter	REAL	Circumference of the roller [mm]

Name	Type	Description
diPointX1	DINT	Point 1 on the X-axis [increments] This point describes the end of the synchronous movement and the start point of the adjustment movement
diPointX2	DINT	Point 2 on the X-axis [increments] This point describes the end of the adjustment movement and the start point of the second synchronous movement

Output variables

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

Input and output variables

Name	Type	Description
arTab_A	ARRAY	ARRAY[0..5] OF SMC_CAMXYVA Support point array (storage of the calculated 6 support points)

Usage note in the CoDeSys program

The call of this function block is permissible in both the asynchronous program section PLC_PRG as well as the synchronous program section FPLC_PRG.

Associated with this function block, a visualisation is prepared in CoDeSys.

Siehe 'Visualization of AFL blocks' auf Seite 806.

4.2.1.1.5.2 CONVERT_ANALOG_TO_DIGITAL (FB)

The function block 'CONVERT_ANALOG_TO_DIGITAL' calculates the corresponding digital value form an analog value. The conversion is performed linearly. The associated line is calculated from the input parameters.

$y = m * x + n$ Linear equation

$m = \frac{\Delta y}{\Delta x} = \frac{y_1 - y_0}{x_1 - x_0}$ Gradient

$n = y_0 - m * x_0$ Offset

The call of this function block is permissible in both the asynchronous program section PLC_PRG as well as the synchronous program section FPLC_PRG.

User interface

	CONVERT_ANALOG_TO_DIGITAL			
FB enable -	boEnable	boEnabAck	-	Ackn. "FB enable"
Analog input value -	reAnalogInput	wDigitalOutput	-	Digital output value
Digital start point -	wDigitalStartPoint			
Analog start point -	reAnalogStartPoint			
Digital end point -	wDigitalEndPoint			
Analog end point -	reAnalogEndPoint			

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.
reAnalogInput	REAL	Analog input value that is to be converted into a digital value
wDigitalStartPoint	WORD	Digital value of the 1st value pair to determine the conversion line; y_0
reAnalogStartPoint	REAL	Analog value of the 1st value pair to determine the conversion line; x_0
wDigitalEndPoint	WORD	Digital value of the 2nd value pair to determine the conversion line; y_1
reAnalogEndPoint	REAL	Analog value of the 2nd value pair to determine the conversion line; x_1

Output variables

Name	Type	Description
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled
wDigitalOutput	WORD	Calculated output value boEnable=FALSE => wDigitalOutput=0

Usage note in the CoDeSys program

The call of this function block is permissible in both the asynchronous program section PLC_PRG as well as the synchronous program section FPLC_PRG.

Associated with this function block, a visualisation is prepared in CoDeSys.

Siehe 'Visualization of AFL blocks' auf Seite 806.

4.2.1.1.5.3 CONVERT_DIGITAL_TO_ANALOG (FB)

The function block 'CONVERT_DIGITAL_TO_ANALOG' calculates the corresponding analog value form a digital value. The conversion is performed linearly. The associated line is calculated from the input parameters.

$y = m * x + n$ Linear equation

$m = \frac{\Delta y}{\Delta x} = \frac{y_1 - y_0}{x_1 - x_0}$ Gradient

$n = y_0 - m * x_0$ Offset

The call of this function block is permissible in both the asynchronous program section PLC_PRG as well as the synchronous program section FPLC_PRG.

User interface

	CONVERT_DIGITAL_TO_ANALOG			
FB enable -	boEnable	boEnabAck	-	Ackn. "FB enable"
Digital input value -	wDigitalInput	reAnalogOutput	-	Analog output value
Digital start point -	wDigitalStartPoint			
Analog start point -	reAnalogStartPoint			
Digital end point -	wDigitalEndPoint			
Analog end point -	reAnalogEndPoint			

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.
wDigitalInput	WORD	Digital input value that is to be converted into an analog value
wDigitalStartPoint	WORD	Digital value of the 1st value pair to determine the conversion line; x_0
reAnalogStartPoint	REAL	Analog value of the 1st value pair to determine the conversion line; y_0
wDigitalEndPoint	WORD	Digital value of the 2nd value pair to determine the conversion line; x_1
reAnalogEndPoint	REAL	Analog value of the 2nd value pair to determine the conversion line; y_1

Output variables

Name	Type	Description
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled
reAnalogOutput	REAL	Calculated output value boEnable=FALSE => reAnalogOutput=0

Usage note in the CoDeSys program

The call of this function block is permissible in both the asynchronous program section PLC_PRG as well as the synchronous program section FPLC_PRG.

Associated with this function block, a visualisation is prepared in CoDeSys.

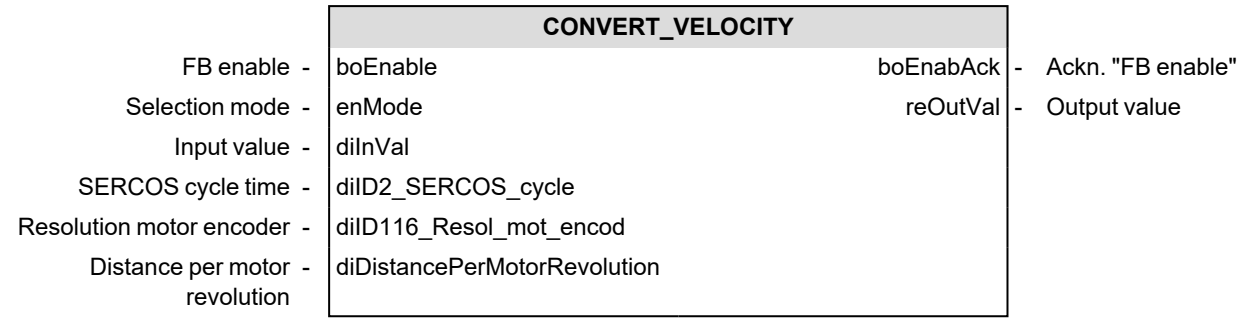
Siehe 'Visualization of AFL blocks' auf Seite 806.

4.2.1.1.5.4 CONVERT_VELOCITY (FB)

The function block 'CONVERT_VELOCITY' is used to convert velocities. The selection of the velocity conversion is performed using modes.

The call of this function block is permissible in both the asynchronous program section PLC_PRG as well as the synchronous program section FPLC_PRG.

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
enMode	ENUM	EN_VELOCITY_CONVERT_MODE Selection mode

V_CONVERT_RPM_TO_INC_ID2	Rpm=> Incr/ID2 Input variables: - diID2_SERCOS_cycle - diID116_Resol_mot_encod - diInVal
V_CONVERT_INC_ID2_TO_RPM	Incr/ID2 => rpm Input variables: - diID2_SERCOS_cycle - diID116_Resol_mot_encod - diInVal
V_CONVERT_U_MIN_TO_INC_S	Rpm => Incr/s Input variables: - diID116_Resol_mot_encod - diInVal
V_CONVERT_INC_S_TO_U_MIN	Incr/s => rpm Input variables: - diID116_Resol_mot_encod - diInVal
V_CONVERT_MM_S_TO_U_MIN	mm/s => rpm Input variables: - diDistancePerMotorRevolution - diInVal
V_CONVERT_U_MIN_TO_MM_S	rpm => mm/s Input variables: - diDistancePerMotorRevolution - diInVal

diInVal	DINT	Input value
diID2_SERCOS_cycle	DINT	ID2 'SERCOS cycle time' [µs]
diID116_Resol_mot_encod	DINT	ID116 'Resolution motor encoder' [Increments]
diDistancePerMotor Revolution	DINT	Distance for one motor revolution [mm]

Output variables

Name	Type	Description
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled
reOutVal	REAL	Output value

Usage note in the CoDeSys program

The call of this function block is permissible in both the asynchronous program section PLC_PRG as well as the synchronous program section FPLC_PRG.

Associated with this function block, a visualisation is prepared in CoDeSys.

Siehe 'Visualization of AFL blocks' auf Seite 806.

4.2.1.1.5.5 fbrInertiaHollowCylinder (F)

The function 'fbrInertiaHollowCylinder' calculates the moment of inertia of a hollow cylinder.

$J_{Hz} = J_a - J_i$ Inertia

$J_{Hz} = \frac{1}{32} \pi h \rho (d_a^4 - d_i^4)$ hollow cylinder

$J = \frac{1}{8} m * d^2$ Inertia

$m = V * \rho$ Mass

$V = \frac{1}{4} \pi d^2 * h$ Volume

The call of this function is permissible in both the asynchronous program section PLC_PRG as well as the synchronous program section FPLC_PRG.

User interface

	fbrInertiaHollowCylinder			
Height -	reHeight	fbrInertiaHollowCylinder	-	Inertia
Outer diameter -	reOuterDiameter			
Inner diameter -	reInnerDiameter			
Density -	reDensity			

Input variables

Name	Type	Description
reHeight	REAL	Height h [m]
reOuterDiameter	REAL	Outer diameter d_o [m]
reInnerDiameter	REAL	Inside diameter d_i [m]
reDensity	REAL	Spec. density ρ [kg/m^3]

Output variable

Name	Type	Description
fbrInertiaHollowCylinder	REAL	Moment of inertia J_{Hz} [kg x m^2]

4.2.1.1.5.6 fbrInertiaPlainCylinder (F)

The function 'fbrInertiaPlainCylinder' calculates the moment of inertia of a full cylinder.

$J_{Vz} = \frac{1}{32} \pi h \rho d^4$ Inertia plain cylinder

$J = \frac{1}{8} m * d^2$ Inertia

$m = V * \rho$ Mass

$V = \frac{1}{4} \pi d^2 * h$ Volume

The call of this function is permissible in both the asynchronous program section PLC_PRG as well as the synchronous program section FPLC_PRG.

User interface

	fbrInertiaPlainCylinder			
Diameter -	reDiameter	fbrInertiaPlainCylinder	-	Inertia
Height -	reHeight			
Density -	reDensity			

Input variables

Name	Type	Description
reDiameter	REAL	Diameter d [m]
reHeight	REAL	Height h [m]
reDensity	REAL	Spec. density ρ [kg/m^3]

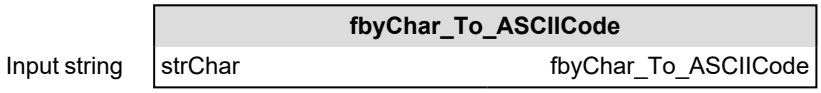
Output variable

Name	Type	Description
fbrInertiaPlainCylinder	REAL	Moment of inertia J [kg x m^2]

4.2.1.1.5.7 fbyChar_To_ASCIICode (F)

The function 'fbyChar_To_ASCIICode' generates the corresponding ASCII code for a character.

User interface



Input variable

Name	Type	Description
strChar	STRING (1)	String; e.g. 'a'

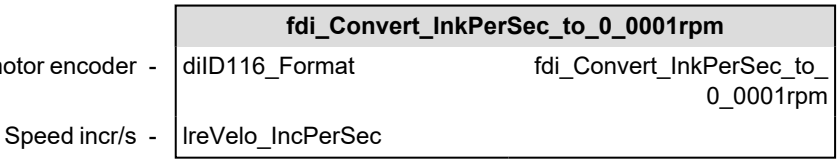
Output variable

Name	Type	Description
fbyChar_To_ASCIICode	BYTE	ASCII code, e.g 97

4.2.1.1.5.8 fdi_Convert_InkPerSec_to_0_0001rpm (F)

The function 'fdi_Convert_InkPerSec_to_0_0001rpm' converts a velocity value from [increments/s] to [0.0001 revolutions/m].

User interface



Input variables

Name	Type	Description
diID116_Format	DINT	ID116 'Resolution motor encoder' [Increments]
lreVelo_IncPerSec	LREAL	Velocity value that is to be converted [increments/s]

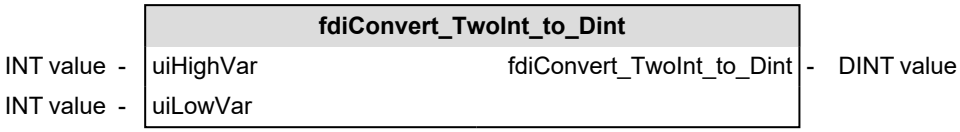
Output variable

Name	Type	Description
fdi_Convert_InkPerSec_to_0_0001rpm	DINT	Converted velocity [0.0001 rpm]

4.2.1.1.5.9 fdiConvert_TwoInt_to_Dint (F)

The function 'fdiConvert_TwoInt_to_Dint' combines two INT values to one DINT value.

User interface



Input variables

Name	Type	Description
uiHighVar	INT	INT value
uiLowVar	INT	INT value

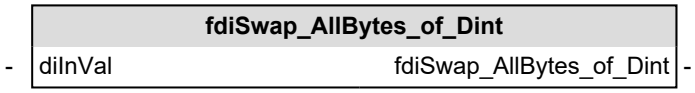
Output variable

Name	Type	Description
fdiConvert_TwoInt_to_Dint	DINT	DINT value: <uiHighVar><uiLowVar>

4.2.1.1.5.10 fdiSwap_AllBytes_of_Dint (F)

The function 'fdiSwap_AllBytes_of_Dint' swaps the order of all bytes of a DINT value.

User interface



Input variable

Name	Type	Description
diInVal	DINT	Byte3 | Byte2 | Byte1 | Byte0 HighWord | LowWord

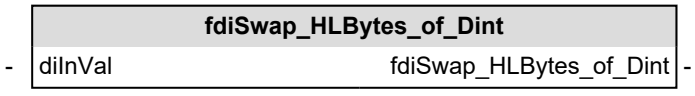
Output variable

Name	Type	Description
fdiSwap_AllBytes_of_Dint	DINT	Byte0 | Byte1 | Byte2 | Byte3 HighWord | LowWord

4.2.1.1.5.11 fdiSwap_HLBytes_of_Dint (F)

The function 'fdiSwap_HLBytes_of_Dint' swaps the high and low bytes within the words of a DINT value

User interface



Input variable

Name	Type	Description
diInVal	DINT	HighByte | LowByte | HighByte | LowByte HighWord | LowWord

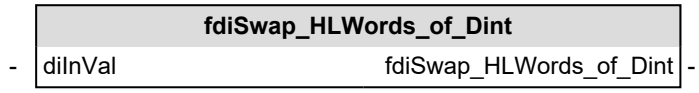
Output variable

Name	Type	Description
fdiSwap_HLBytes_of_Dint	DINT	LowByte | HighByte | LowByte | HighByte HighWord | LowWord

4.2.1.1.5.12 fdiSwap_HLWords_of_Dint (F)

The function 'fdiSwap_HLWords_of_Dint swaps' the two words of a DINT value

User interface



Input variable

Name	Type	Description
diInVal	DINT	HighWord | LowWord

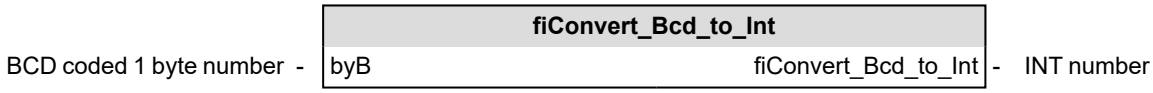
Output variable

Name	Type	Description
fdiSwap_HLWords_of_Dint	DINT	LowWord | HighWord

4.2.1.1.5.13 fiConvert_Bcd_to_Int (F)

The function 'fiConvert_Bcd_to_Int' converts a BCD coded 1-byte number into an INT number.

User interface



Input variable

Name	Type	Description
byB	BYTE	BCD-coded 1-byte number

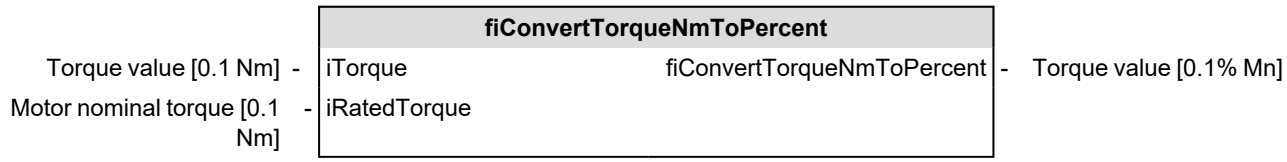
Output variable

Name	Type	Description
fiConvert_Bcd_to_Int	INT	No BCD code: fiConvert_Bcd_to_Int = -1 otherwise: fiConvert_Bcd_to_Int = 0...10

4.2.1.1.5.14 fiConvertTorqueNmToPercent (F)

The function 'fiConvertTorqueNmToPercent' converts a torque value [0.1 Nm] into an percentage output value to the nominal motor torque [% Mn].

User interface



Input variable

Name	Type	Description
iTorque	INT	Torque value [0.1 Nm]
iRatedTorque	INT	Motor nominal torque [0.1 Nm]

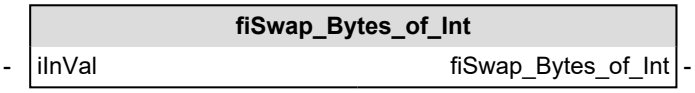
Output variable

Name	Type	Description
fiConvertTorqueNmToPercent	INT	Torque value [0.1% Nm]

4.2.1.1.5.15 fiSwap_Bytes_of_Int (F)

The function 'fiSwap_Bytes_of_Int' swaps the two bytes of an INT value

User interface



Input variable

Name	Type	Description
iInVal	INT	HighByte | LowByte

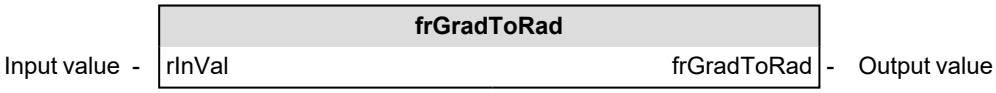
Output variable

Name	Type	Description
fiSwap_Bytes_of_Int	INT	LowByte | HighByte

4.2.1.1.5.16 frGradToRad (F)

The function 'frGradToRad' converts the degree value of an angle to radians.

User interface



Input variables

Name	Type	Description
rInVal	REAL	Input value [°]

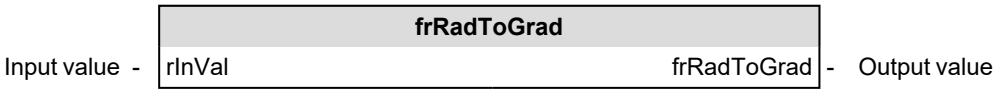
Output variable

Name	Type	Description
frGradToRad	REAL	Output value [rad]

4.2.1.1.5.17 frRadToGrad (F)

The function 'frRadToGrad' converts the radian value of an angle to degrees.

User interface



Input variables

Name	Type	Description
rInVal	REAL	Input value [rad]

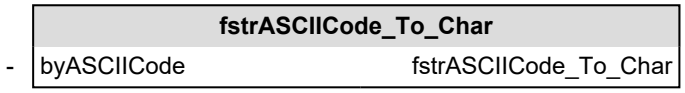
Output variable

Name	Type	Description
frRadToGrad	REAL	Output value [°]

4.2.1.1.5.18 fstrASCIICode_To_Char (F)

The function 'fstrASCIICode_To_Char' generates the corresponding character from an ASCII code

User interface



Input variable

Name	Type	Description
byASCIICode	BYTE	ASCII code, e.g 097

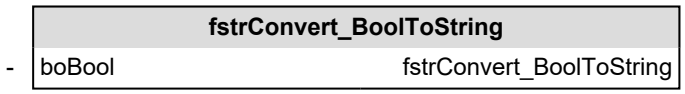
Output variable

Name	Type	Description
fstrASCIICode_To_Char	STRING (1)	String, e.g. 'a'

4.2.1.1.5.19 fstrConvert_BoolToString (F)

The function 'fstrConvert_BoolToString' converts a boolean variable into a string.

User interface



Input variable

Name	Type	Description
boBool	BOOL	e.g. TRUE

Output variable

Name	Type	Description
fstrConvert_BoolToString	STRING (1)	e.g. '1'

4.2.1.1.5.20 fstrConvert_DintToString (F)

The function 'fstrConvert_DintToString' converts a DINT variable into a string.

User interface

Input variable

Name	Type	Description
diDint	DINT	e.g. 12345

Output variable

Name	Type	Description
fstrConvert_DintToString	STRING	e.g. '12345'

4.2.1.1.5.21 fudCalc_Ramp_Acceleration (F)

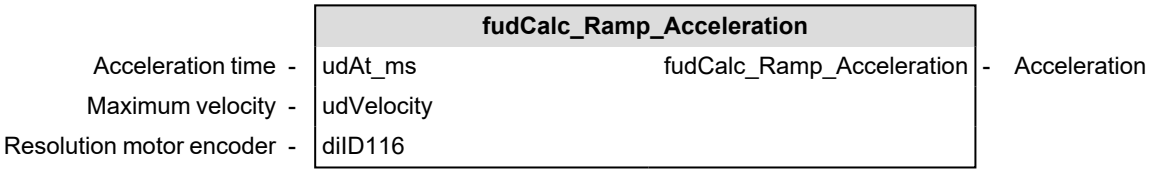
The function 'fudCalc_Ramp_Acceleration' calculates the acceleration ramp.

$$a\ [\frac{Inkr}{s^2}] = \frac{v\ [\frac{Inkr}{s}]}{t\ [s]}$$ Acceleration

$$v\ [\frac{Inkr}{s}] = \frac{udVelocity\ [\frac{U}{min}]}{10000 * 60\ [\frac{s}{min}]} * ID116\ [\frac{Inkr}{U}]$$ Velocity

$$t\ [s] = \frac{udAt\_ms\ [ms]}{1000\ [\frac{ms}{s}]}$$ Acceleration time

User interface



Input variables

Name	Type	Description
udAt_ms	UDINT	Acceleration time [ms]
udVelocity	UDINT	Maximum velocity [0.0001 rpm]
diID116	DINT	ID116 'Resolution motor encoder' [Increments]

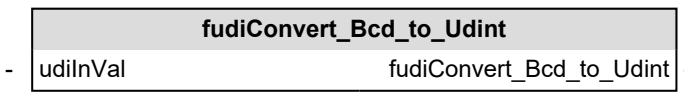
Output variable

Name	Type	Description
fudCalc_Ramp_Acceleration	UDINT	Acceleration [increments/s^2]

4.2.1.1.5.22 fudiConvert_Bcd_to_Udint (F)

The function 'fudiConvert_Bcd_to_Udint' converts an 8-digit BCD-coded positive number into a UDINT value.

User interface



Input variable

Name	Type	Description
udiInVal	UDINT	8-digit BCD-coded positive number

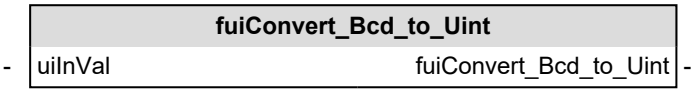
Output variable

Name	Type	Description
fudiConvert_Bcd_to_Udint	UDINT	No BCD code: fudiConvert_Bcd_to_Udint = -1 otherwise: fudiConvert_Bcd_to_Udint = binary value

4.2.1.1.5.23 fuiConvert_Bcd_to_Uint (F)

The function 'fuiConvert_Bcd_to_Uint' converts an 8-digit BCD-coded positive number into a UINT value.

User interface



Input variable

Name	Type	Description
uiInVal	UINT	8-digit BCD-coded positive number

Output variable

Name	Type	Description
fuiConvert_Bcd_to_Uint	UINT	No BCD code: fuiConvert_Bcd_to_Uint = -1 otherwise: fuiConvert_Bcd_to_Uint = binary value

4.2.1.1.5.24 udfConvert_Acceleration (F)

The function 'udfConvert_Acceleration' is replaced completely by 'fudCalc_Ramp_Acceleration'.
For new applications, only use 'fudCalc_Ramp_Acceleration'!

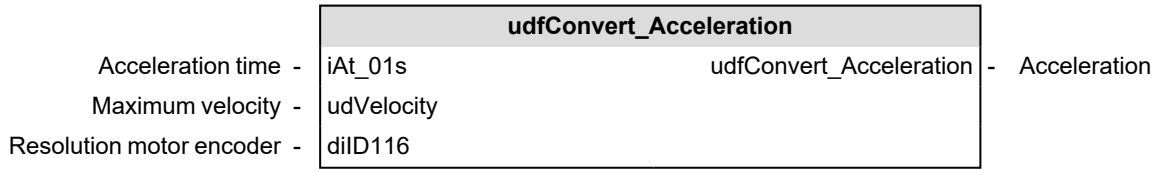
The function 'udfConvert_Acceleration' calculates the acceleration ramp.

$$a\left[\frac{Inkr}{s^2}\right] = \frac{v\left[\frac{Inkr}{s}\right]}{t\,[s]}$$ Acceleration

$$v\left[\frac{Inkr}{s}\right] = \frac{udVelocity\left[\frac{U}{min}\right]}{10000 * 60\left[\frac{s}{min}\right]} * ID116\left[\frac{Inkr}{U}\right]$$ Velocity

$$t\,[s] = \frac{udAt\_01s\,[s]}{10}$$ Acceleration time

User interface



Input variables

Name	Type	Description
iAt_01s	INT	Acceleration time [0.1 s]
diVelocity	DINT	Maximum velocity [0.0001 rpm]
diID116	DINT	ID116 'Resolution motor encoder' [Increments]

Output variable

Name	Type	Description
udfConvert_Acceleration	UDINT	Acceleration [increments/s^2]

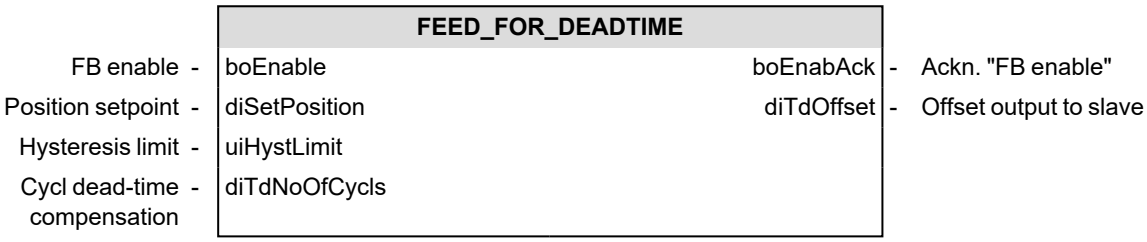
4.2.1.1.6 08_Feedforward

4.2.1.1.6.1 FEED_FOR_DEADTIME (FB)

The function block 'FEED_FOR_DEADTIME' realises a dead-time compensation. The block measures the speed at which the position setpoint changes and determines the offset of the slave as a function of the dead time.

The function block is called in the synchronous program level FPLC_PRG.

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]
uiHystLimit	UINT	Position setpoint that is output to the KW (SET_SETPOINT_POSITION block)
diTdNoOfCycls	DINT	Number of cycles ID2 for the dead-time compensation [0.1 * ID2]

Output variables

Name	Type	Description
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled
diTdOffset	DINT	Offset for output to slave [increments]

Usage note in the CoDeSys program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
	FEED_FOR_DEADTIME

Associated with this function block, a visualisation is prepared in CoDeSys.

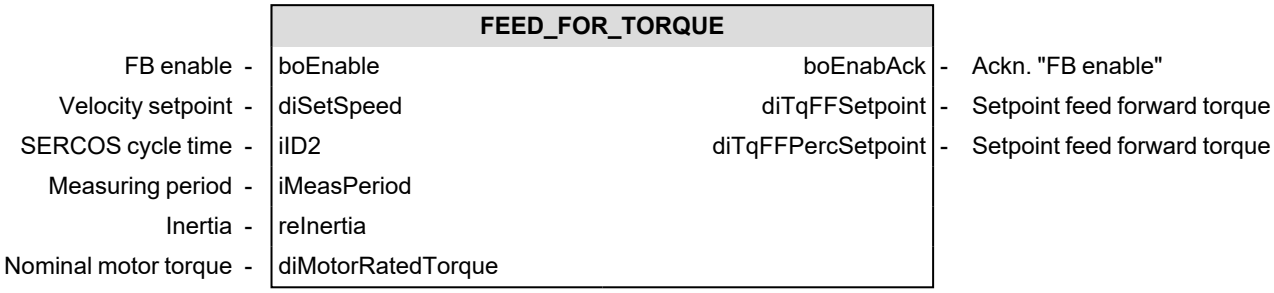
Siehe 'Visualization of AFL blocks' auf Seite 806.

4.2.1.1.6.2 FEED_FOR_TORQUE (FB)

The function block 'FEED_FORWARD_TORQUE' realises a torque control with feedforward. The block measures the velocity with which the speed setpoint changes and calculates the corresponding torque.

The function block is called in the synchronous program level FPLC_PRG.

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	Velocity setpoint [0.0001 rpm] SET_SETPOINT_SPEED block
iID2	INT	ID2 'SERCOS cycle time' [µs]
iMeasPeriod	INT	Period of the speed measurement [multiple of ID2]
reInertia	REAL	Determined moment of inertia [kg x m^2]
diMotorRatedTorque	DINT	Nominal motor torque Mn [0.1 Nm]

Output variables

Name	Type	Description
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled
diTqFFSetpoint	DINT	Torque control with feedforward; must be output with the block 'SET_PRE_SETPOINT_TORQUE' to the drive. [0.1 Nm]
diTqFFPercSetpoint	DINT	Torque control with feedforward; must be output with the block 'SET_PRE_SETPOINT_TORQUE' to the drive. [0.1% Mn]

Usage note in the CoDeSys program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
	FEED_FOR_TORQUE

Associated with this function block, a visualisation is prepared in CoDeSys.

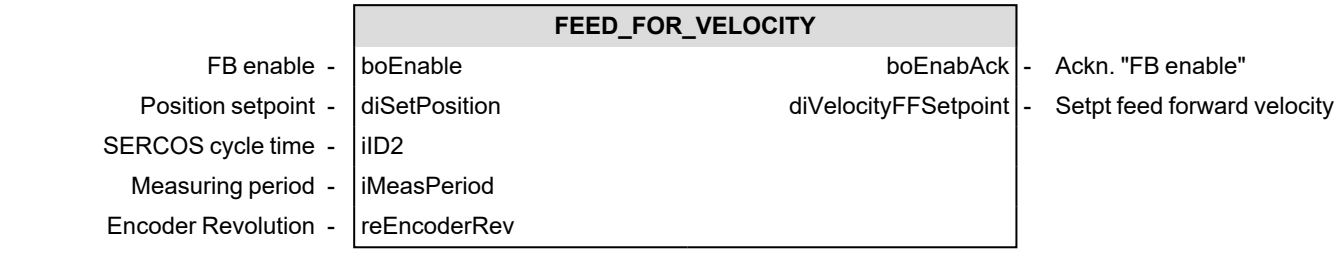
Siehe 'Visualization of AFL blocks' auf Seite 806.

4.2.1.1.6.3 FEED_FOR_VELOCITY (FB)

The function block 'FEED_FOR_VELOCITY' realises a speed control with feedforward. The block calculates the speed at which the position setpoint changes and calculates the corresponding motor speed.

The function block is called in the synchronous program level FPLC_PRG.

User interface



Input variables of the synchronous program levels (FPLC_PRG)

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] SET_SETPOINT_POSITION block
iID2	INT	ID2 'SERCOS cycle time' [µs]
iMeasPeriod	INT	Period of the speed measurement [multiple of ID2]
reEncoderRev	REAL	Motor encoder resolution

Output variables of the synchronous program levels (FPLC_PRG)

Name	Type	Description
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled
diVelocityFFSetpoint	DINT	Speed control with feedforward; must be output with the block 'SET_PRE_SETPOINT_SPEED' to the drive.

Usage note in the CoDeSys program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
	FEED_FOR_VELOCITY

Associated with this function block, a visualisation is prepared in CoDeSys.

Siehe 'Visualization of AFL blocks' auf Seite 806.

4.2.1.1.7 09_Filter

4.2.1.1.7.1 FILTER_BLOCKING_NEGATIVE_COUNT (FB)

The function block 'FILTER_BLOCKING_NEGATIVE_COUNT' blocks negative counting pulses.

At the filter output, positive counting occurs again only after the negative "count path" of a square-wave encoder, for instance, has been processed. Advantage: Position control around a position (e.g. ±1 increment) does not cause the filter output to count up. Thus the relationship to the input (e.g. master square-wave encoder) is preserved.

The call of this function block is permissible in both the asynchronous program section PLC_PRG as well as the synchronous program section FPLC_PRG.

User interface

	FILTER_BLOCKING_NEGATIVE_COUNT		
FB enable -	boEnable	boEnabAck	- Ackn. "FB enable"
Input value -	diInVal	diOutVal	- Output value

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 [Increments]

Output variables

Name	Type	Description
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled
diOutVal	DINT	Output value [Increments]

Usage note in the CoDeSys program

The call of this function block is permissible in both the asynchronous program section PLC_PRG as well as the synchronous program section FPLC_PRG.

Associated with this function block, a visualisation is prepared in CoDeSys.

Siehe 'Visualization of AFL blocks' auf Seite 806.

4.2.1.1.7.2 FILTER_BLOCKING_POSITIVE_COUNT (FB)

The function block 'FILTER_BLOCKING_POSITIVE_COUNT' blocks positive count pulses. At the filter output, negative counting occurs again only after the positive "count path" of a square-wave encoder, for instance, has been processed. Advantage: Position control around a position (e.g. +/- 1 increment) does not cause the filter output to count. Thus the relationship to the input (e.g. master square-wave encoder) is preserved.

The call of this function block is permissible in both the asynchronous program section PLC_PRG as well as the synchronous program section FPLC_PRG.

User interface

	FILTER_BLOCKING_POSITIVE_COUNT		
FB enable -	boEnable	boEnabAck	- Ackn. "FB enable"
Input value -	diInVal	diOutVal	- Output value

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 [Increments]

Output variables

Name	Type	Description
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled
diOutVal	DINT	Output value [Increments]

Usage note in the CoDeSys program

The call of this function block is permissible in both the asynchronous program section PLC_PRG as well as the synchronous program section FPLC_PRG.

Associated with this function block, a visualisation is prepared in CoDeSys.

Siehe 'Visualization of AFL blocks' auf Seite 806.

4.2.1.1.7.3 FILTER_HYSTERESIS (FB)

The function block 'FILTER_HYSTERESIS' realizes a hysteresis.

The block controls the output size as a function of the change of the input size and its limits.

The call of this function block is permissible in both the asynchronous program section PLC_PRG as well as the synchronous program section FPLC_PRG.

User interface

		FILTER_HYSTERESIS			
FB enable	-	boEnable	boEnabAck	-	Ackn. "FB enable"
Input value	-	diInVal	diOutVal	-	Output value
Limit positive	-	uiLimitPos			
Limit negative	-	uiLimitNeg			

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
uiLimitPos	UINT	Limit positive of the hysteresis
uiLimitNeg	UINT	Limit negative of the hysteresis

Output variables of the asynchronous program levels (PLC_PRG)

Name	Type	Description
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled
diOutVal	DINT	Output value with hysteresis

Usage note in the CoDeSys program

The call of this function block is permissible in both the asynchronous program section PLC_PRG as well as the synchronous program section FPLC_PRG.

Associated with this function block, a visualisation is prepared in CoDeSys.

Siehe 'Visualization of AFL blocks' auf Seite 806.

4.2.1.1.7.4 FILTER_MEAN_VALUE_DINT_MAX20 (FB)

The function block 'FILTER_MEAN_VALUE_DINT_MAX20' forms the arithmetic mean of the input values using shift registers of up to 20 values.

When averaging, the new input value is written to the variable 'diInVal'. After calling the block, the new average can be read in the variable 'diMeanVal'.

The call of this function block is permissible in both the asynchronous program section PLC_PRG as well as the synchronous program section FPLC_PRG.

User interface

	FILTER_MEAN_VALUE_DINT_MAX20			
FB enable -	boEnable	boEnabAck	-	Ackn. "FB enable"
Input value -	diInVal	diMeanVal	-	Mean value
Number of values -	iNoOfValues			

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
iNoOfValues	INT	Number of values over which the average is calculated. 1 ≤ iNoOfValues ≤ 20

Output variables

Name	Type	Description
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled
diMeanVal	DINT	Arithmetic mean

Usage note in the CoDeSys program

The call of this function block is permissible in both the asynchronous program section PLC_PRG as well as the synchronous program section FPLC_PRG.

Associated with this function block, a visualisation is prepared in CoDeSys.

Siehe 'Visualization of AFL blocks' auf Seite 806.

4.2.1.1.7.5 FILTER_MEAN_VALUE_MAX20 (FB)

The function block 'FILTER_MEAN_VALUE_MAX20' forms the arithmetic mean of the input values using shift registers of up to 20 values.

When averaging, the new input value is written to the variable 'iInVal'. After calling the block, the new average can be read in the variable 'iMeanVal'.

The call of this function block is permissible in both the asynchronous program section PLC_PRG as well as the synchronous program section FPLC_PRG.

User interface

	FILTER_MEAN_VALUE_MAX20			
FB enable -	boEnable	boEnabAck	-	Ackn. "FB enable"
Input value -	iInVal	iMeanVal	-	Mean value
Number of values -	iNoOfValues			

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
iNoOfValues	INT	Number of values over which the average is calculated. 1 ≤ iNoOfValues ≤ 20

Output variables

Name	Type	Description
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled
iMeanVal	INT	Arithmetic mean

Usage note in the CoDeSys program

The call of this function block is permissible in both the asynchronous program section PLC_PRG as well as the synchronous program section FPLC_PRG.

Associated with this function block, a visualisation is prepared in CoDeSys.

Siehe 'Visualization of AFL blocks' auf Seite 806.

4.2.1.1.7.6 FILTER_PT1 (FB)

The function block 'FILTER_PT1' realises a low-pass filter of the 1st order

The call of this function block is permissible in both the asynchronous program section PLC_PRG as well as the synchronous program section FPLC_PRG.

User interface

	FILTER_PT1			
FB enable -	boEnable	boEnabAck	-	Ackn. "FB enable"
Input value -	diInVal	diOutVal	-	Output value
Filter time -	diTime			
Device structure -	stDevice	stDevice	-	Device structure

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
diTime	DINT	Filter time constant [ms]

Output variables

Name	Type	Description
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled
diOutVal	DINT	Filtered output value

Input and output variables

Name	Type	Description
stDevice	STRUCT	ST_DEVICE The device description structure assigns the block a device. (See document Software description AmkBase Bibliothek, Part no.204986)

Usage note in the CoDeSys program

The call of this function block is permissible in both the asynchronous program section PLC_PRG as well as the synchronous program section FPLC_PRG.

Associated with this function block, a visualisation is prepared in CoDeSys.
Siehe 'Visualization of AFL blocks' auf Seite 806.

4.2.1.1.7.7 FILTER_RAMP_INCRDECR_TIMECONST (FB)

The function block 'FILTER_RAMP_INCRDECR_TIMECONST' supplies a ramped output value for abrupt input values.

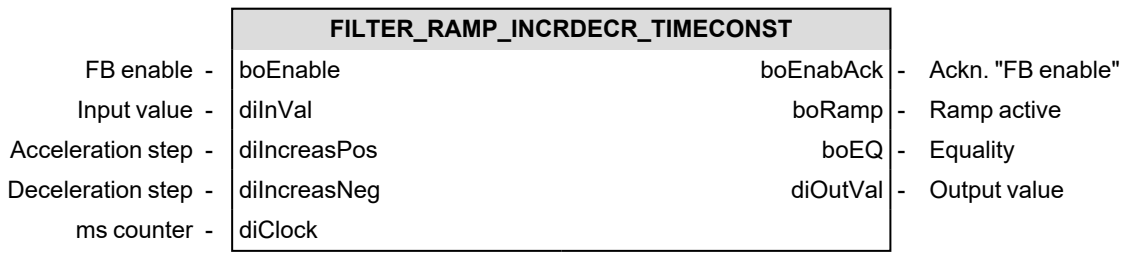
- The block limits the output value 'diOutVal' ≤ 'diInVal'.
- If the input signal 'boEnable' = FALSE, the input value is copied without ramp to the output.
- 'diIncreasPos' and 'diIncreasNeg' determine, independent of one another, the slopes of the ramp during acceleration or decelaration.

The call of this function block is permissible in both the asynchronous program section PLC_PRG as well as the synchronous program section FPLC_PRG.

Example:

Jump of the input value 'diInVal' from 0 -> 1000
Ramp increase for acceleration 'diIncreasPos' = 100 [1/s]
This results in: The output value 'diOutVal' ramps from 0 -> 1000 within 10 seconds

User interface



Input variables

Name	Type	Description
boEnable	BOOL	Ramp enable signal: With a positive edge, the ramp function of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and the output is ramped. In the state 'boEnable' = FALSE the block is no longer ramp enabled and the input value is copied directly to the output value
diInVal	DINT	Input value
diIncreasPos	DINT	Acceleration steps: Number of steps that the output value increases by per second [1/s]
diIncreasNeg	DINT	Deceleration steps: Number of steps that the output value decreases by per second [1/s]
diClock	DINT	ms counter [ms]

Output variables

Name	Type	Description
boEnabAck	BOOL	Acknowledgement: Ramp function is active
boRamp	BOOL	Ramp is run
boEQ	BOOL	Input value = output value
diOutVal	DINT	Output value

Usage note in the CoDeSys program

The call of this function block is permissible in both the asynchronous program section PLC_PRG as well as the synchronous program section FPLC_PRG.

Associated with this function block, a visualisation is prepared in CoDeSys.

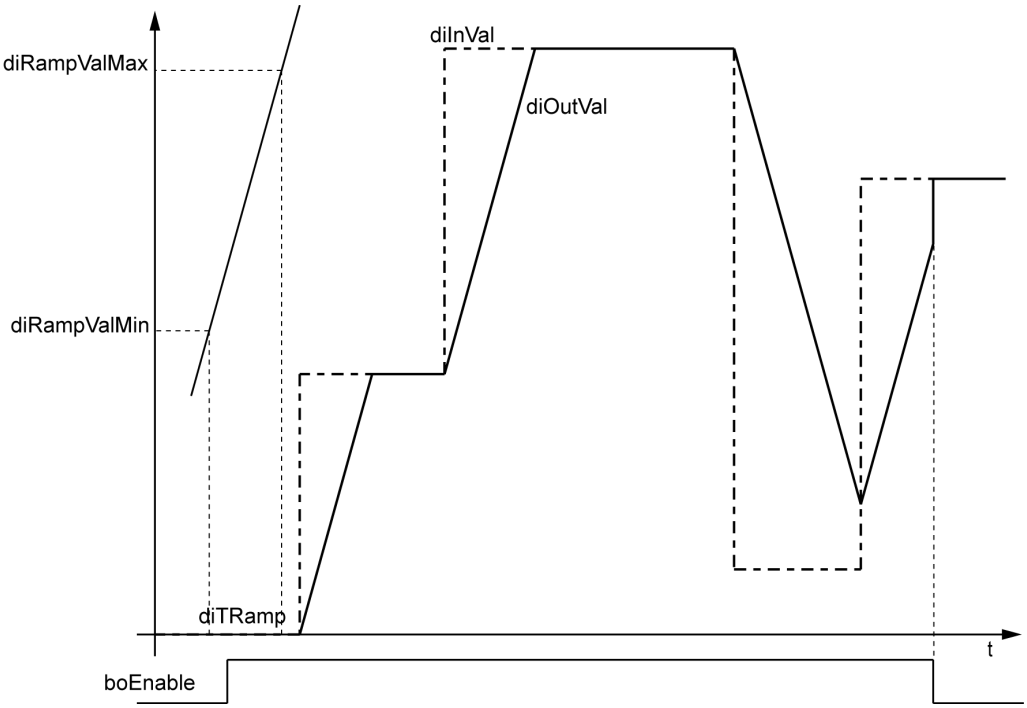
Siehe 'Visualization of AFL blocks' auf Seite 806.

4.2.1.1.7.8 FILTER_RAMP_MINMAX_TIME (FB)

The function block 'FILTER_RAMP_MINMAX_TIME' supplies a ramped output value for abrupt input values.

- The block limits the output value 'diOutVal' ≤ 'diInVal'.
- If the input signal 'boEnable' = FALSE, the input value is copied without ramp to the output.
- The ramp time specifies the time in which the output value is increased in case of a jump of the input value from 'diRampValMin' to 'diRampValMax'.

The call of this function block is permissible in both the asynchronous program section PLC_PRG as well as the synchronous program section FPLC_PRG.



User interface

	FILTER_RAMP_MINMAX_TIME			
FB enable -	boEnable	boEnabAck	-	Ackn. "FB enable"
Input value -	diInVal	boRamp	-	Ramp active
Min. value ramp -	diRampValMin	boEQ	-	Equality
Max. value ramp -	diRampValMax	diOutVal	-	Output value
Ramp time -	diTRamp			
ms counter -	diClock			

Input variables

Name	Type	Description
boEnable	BOOL	Ramp enable signal: With a positive edge, the ramp function of the block starts. As long as 'boEnable' = TRUE, the block remains enabled and the output is ramped. In the state 'boEnable' = FALSE the block is no longer ramp enabled and the input value is copied directly to the output value
diInVal	DINT	Input value
diRampValMin	DINT	Minimum value of the ramp
diRampValMax	DINT	Maximum value of the ramp
diTRamp	DINT	Ramp time [ms]
diClock	DINT	ms counter [ms]

Output variables

Name	Type	Description
boEnabAck	BOOL	Acknowledgement: Ramp function is active
boRamp	BOOL	Ramp is run
boEQ	BOOL	Input value = output value
diOutVal	DINT	Output value

Usage note in the CoDeSys program

The call of this function block is permissible in both the asynchronous program section PLC_PRG as well as the synchronous program section FPLC_PRG.

Associated with this function block, a visualisation is prepared in CoDeSys.

Siehe 'Visualization of AFL blocks' auf Seite 806.

4.2.1.1.7.9 FILTER_SETPOINT (FB)

The function block 'FILTER_SETPOINT' realises a setpoint filter

The call of this function block is permissible in both the asynchronous program section PLC_PRG as well as the synchronous program section FPLC_PRG.

User interface

		FILTER_SETPOINT			
FB enable	-	boEnable	boEnabAck	-	Ackn. "FB enable"
Filter time	-	lreFilterTime	diOutVal	-	Output value
Input value	-	diPosition	lreVelocity	-	Actual velocity
Modulo input	-	diModuloIn	lreAccel	-	Actual acceleration
Modulo output	-	diModuloOut			
Device structure	-	stDevice	stDevice	-	Device structure

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.
lreFilterTime	LREAL	Filter time [s]
diPosition	DINT	Input value [increments]
diModuloIn	DINT	Modulo for the input value
diModuloOut	DINT	Modulo for the output value

Output variables

Name	Type	Description
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled
diOutVal	DINT	Output value [increments]
lreVelocity	LREAL	Actual velocity [increments/s]
lreAccel	LREAL	Actual acceleration value [increments/s^2]

Input and output variables

Name	Type	Description
stDevice	STRUCT	ST_DEVICE The device description structure assigns the block a device. (See document Software description AmkBase Bibliothek, Part no.204986)

Usage note in the CoDeSys program

The call of this function block is permissible in both the asynchronous program section PLC_PRG as well as the synchronous program section FPLC_PRG.

Associated with this function block, a visualisation is prepared in CoDeSys.
Siehe 'Visualization of AFL blocks' auf Seite 806.

4.2.1.1.8 10_Buffer

4.2.1.1.8.1 BUFFER_FIFO_DINT_MAX_50 (FB)

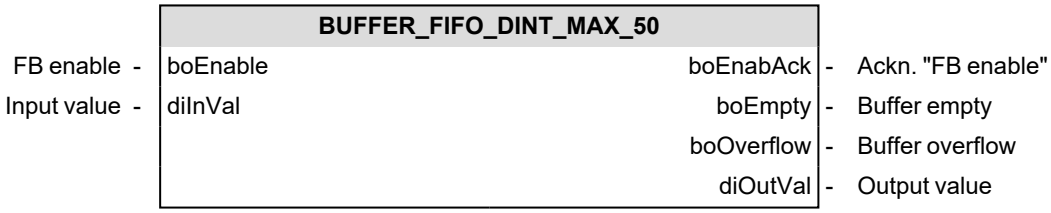
The function block 'BUFFER_FIFO_DINT_MAX_50' provides a FIFO buffer for double integer values.

There are three actions:

- actInPut - Saving a value to the FIFO
- actOutPut - Reclaiming a value from the FIFO
- actReset - resetting the FIFO

The call of this function block is permissible in both the asynchronous program section PLC_PRG as well as the synchronous program section FPLC_PRG.

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	Save value

Output variables

Name	Type	Description
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled
boEmpty	BOOL	The FiFo buffer is empty
boOverflow	BOOL	Buffer overflow: A new value is to be saved to the buffer, but no free position is available.
diOutVal	DINT	Value that was read out the last time the action 'actOutPut' was called

Actions

Name	Description
actInput	Action saves the value from 'diInVal' to the FIFO <BUFFER_FIFO_DINT_MAX_50>.actInput(diInVal:=...);
actOutPut	Action reads the next value out of the FIFO and writes the value to 'diOutVal' <BUFFER_FIFO_DINT_MAX_50>.actOutPut(diOutVal=>...);
actReset	Action resets the FIFO; all saved values are lost <BUFFER_FIFO_DINT_MAX_50>.actReset();

Usage note in the CoDeSys program

The call of this function block is permissible in both the asynchronous program section PLC_PRG as well as the synchronous program section FPLC_PRG.

Associated with this function block, a visualisation is prepared in CoDeSys.

Siehe 'Visualization of AFL blocks' auf Seite 806.

4.2.1.1.8.2 BUFFER_FIFO_INT_MAX50 (FB)

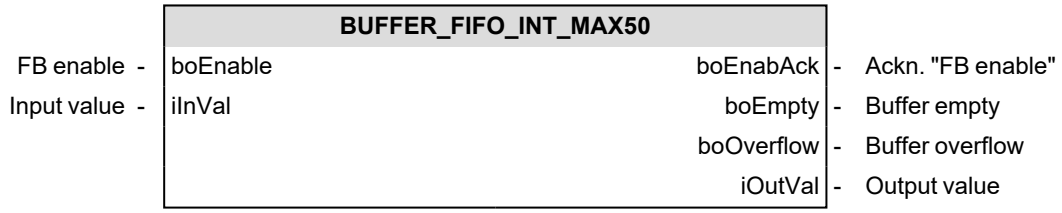
The function block 'BUFFER_FIFO_INT_MAX50' provides a FIFO buffer for integer values.

There are three actions:

- actInPut - Saving a value to the FIFO
- actOutPut - Reclaiming a value from the FIFO
- actReset - resetting the FIFO

The call of this function block is permissible in both the asynchronous program section PLC_PRG as well as the synchronous program section FPLC_PRG.

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	Save value

Output variables

Name	Type	Description
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled
boEmpty	BOOL	The FiFo buffer is empty
boOverflow	BOOL	Buffer overflow: A new value is to be saved to the buffer, but no free position is available.
iOutVal	INT	Value that was read out the last time the action 'actOutPut' was called

Actions

Name	Description
actInput	Action saves the value from 'diInVal' to the FIFO <BUFFER_FIFO_INT_MAX50>.actInput(diInVal:=...);
actOutPut	Action reads the next value out of the FIFO and writes the value to 'diOutVal' <BUFFER_FIFO_INT_MAX50>.actOutPut(diOutVal=>...);
actReset	Action resets the FIFO; all saved values are lost <BUFFER_FIFO_INT_MAX50>.actReset();

Usage note in the CoDeSys program

The call of this function block is permissible in both the asynchronous program section PLC_PRG as well as the synchronous program section FPLC_PRG.

Associated with this function block, a visualisation is prepared in CoDeSys.

Siehe 'Visualization of AFL blocks' auf Seite 806.

4.2.1.1.8.3 BUFFER_SHIFT_REG_INT_MAX10 (FB)

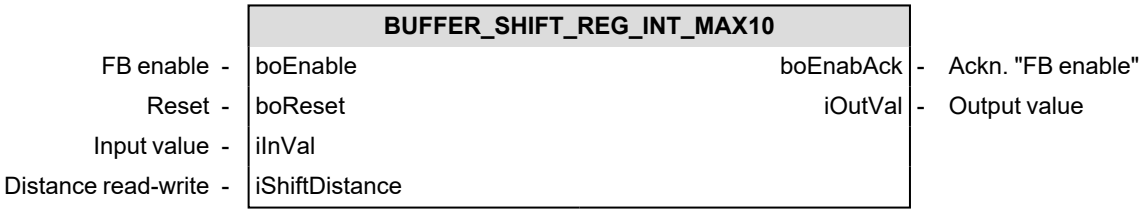
The function block 'BUFFER_SHIFT_REG_INT_MAX10' provides a shift register with 10 locations.

There are two actions:

- actClock - Continuing to write the shift register one location further
- actReset - Resetting the shift register

The call of this function block is permissible in both the asynchronous program section PLC_PRG as well as the synchronous program section FPLC_PRG.

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.
boReset	BOOL	Reset shift register
iInVal	INT	Save value
iShiftDistance	INT	Number of cycles until the save value appears at the output

Output variables

Name	Type	Description
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled
iOutVal	INT	After the action 'actClock', the variable has the value of the shift register at the read-out position

Actions

Name	Description
actClock	The action saves the value of 'iInVal' to the shift register, shifts the register and sets the output variable 'diOutVal' <BUFFER_SHIFT_REG_INT_MAX10>.actClock();
actReset	Action empties the entire shift register <BUFFER_SHIFT_REG_INT_MAX10>.actReset();

Usage note in the CoDeSys program

The call of this function block is permissible in both the asynchronous program section PLC_PRG as well as the synchronous program section FPLC_PRG.

Associated with this function block, a visualisation is prepared in CoDeSys.

Siehe 'Visualization of AFL blocks' auf Seite 806.

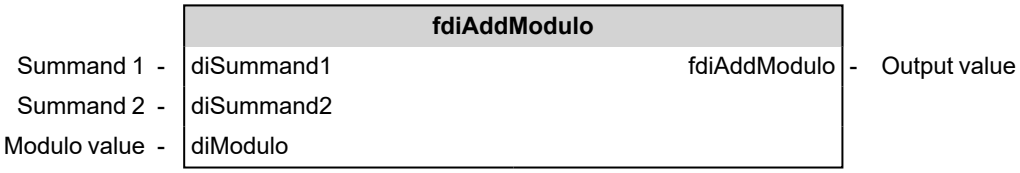
4.2.1.1.9 11_Modulo

4.2.1.1.9.1 fdiAddModulo (F)

The function 'fdiAddModulo' adds up two values with modulo

The call of this function is permissible in both the asynchronous program section PLC_PRG as well as the synchronous program section FPLC_PRG.

User Interface



Input variables

Name	Type	Description
diSummand1	DINT	Summand 1 [increments]
diSummand2	DINT	Summand 2 [increments]
diModulo	DINT	Modulowert [increments]

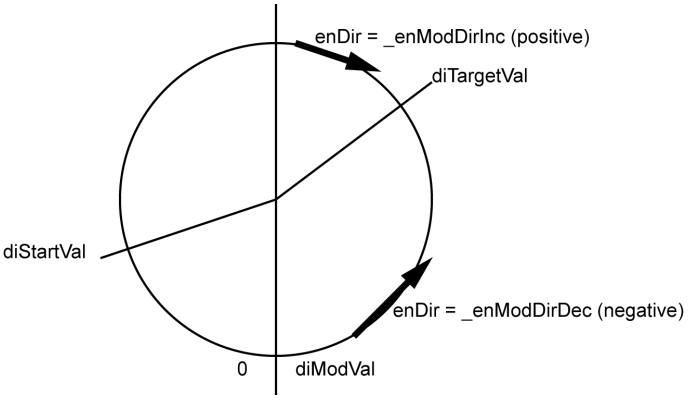
Output variable

Name	Type	Description
fdiAddModulo	DINT	Output value [increments]

4.2.1.1.9.2 fdiModulo_DiffModulo (F)

The function 'fdiModulo_DiffModulo' supplies the modulo difference between two values.

The call of this function is permissible in both the asynchronous program section PLC_PRG as well as the synchronous program section FPLC_PRG.



User interface

	fdiModulo_DiffModulo		
Start value -	diStartVal	fdiModulo_DiffModulo -	Difference
Target value -	diTargetVal		
Modulo value -	diModVal		
Direction of difference -	enDir		
Direction negative -	boDirNeg		

Input variables

Name	Type	Description
diStartVal	DINT	Start value of difference; 0 < diStartVal < diModVal [increments]
diTargetVal	DINT	Target value of difference; 0 < diTargetVal < diModVal [increments]
diModVal	DINT	Modulo end value of the modulo counter [increments] diModVal > 0
enDir	ENUM	EN_FILTER_MODULO_DIR Direction of difference: _enModDirShortDist: shortest distance _enModDirInc: Difference in direction of ascending values _enModDirDec: Difference in direction of descending values
boDirNeg	BOOL	Calculation of the difference in the negative direction

Output variable

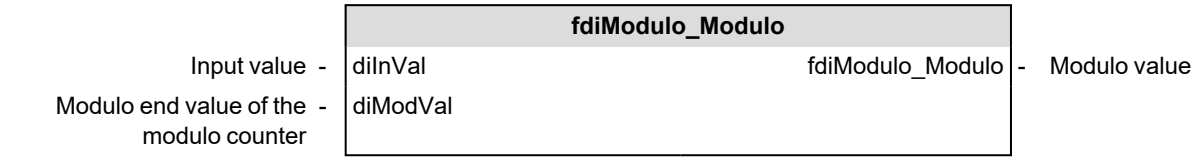
Name	Type	Description
fdiModulo_DiffModulo	DINT	Difference between start and target value [increments]

4.2.1.1.9.3 fdiModulo_Modulo (F)

The function 'fdiModulo_Modulo' calculates the modulo value of the sequential input value.

The call of this function is permissible in both the asynchronous program section PLC_PRG as well as the synchronous program section FPLC_PRG.

User interface



Input variables

Name	Type	Description
diInVal	DINT	Input value [increments]
diModVal	DINT	Modulo end value of the modulo counter [increments]

Output variable

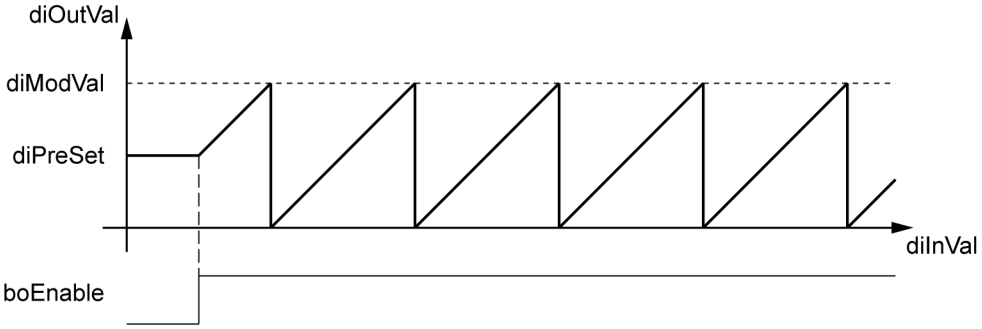
Name	Type	Description
fdiModulo_Modulo	DINT	Modulo value of 'diInVal' [increments]

4.2.1.1.9.4 MODULO_COUNT (FB)

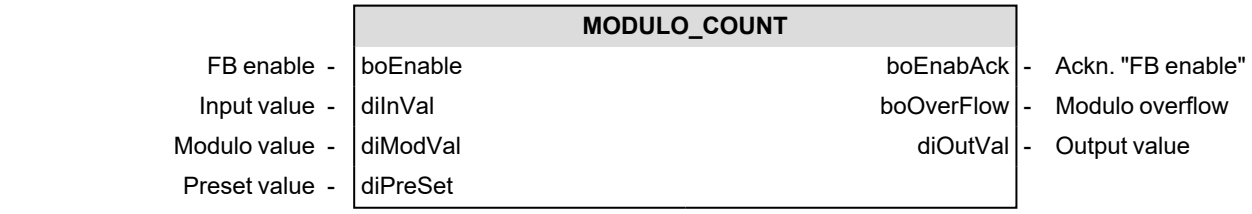
The function block 'MODULO_COUNT' realizes a modulo counter.

The counter can be preset to a value.

The call of this function block is permissible in both the asynchronous program section PLC_PRG as well as the synchronous program section FPLC_PRG.



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
diInVal	DINT	Input value [increments]
diModVal	DINT	Modulo end value of the modulo counter [increments] output 'diOutVal' counts from 0 to 'diModVal'
diPreSet	DINT	Preset value to which the output is set. The value is applied with the positive edge of 'boEnable'. [Increments]

Output variables

Name	Type	Description
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled
boOverFlow	BOOL	Marking of modulo flow; for overflow or underflow, 'boOverFLow' is set TRUE for one cycle.
diOutVal	DINT	Output value [increments]

Usage note in the CoDeSys program

The call of this function block is permissible in both the asynchronous program section PLC_PRG as well as the synchronous program section FPLC_PRG.

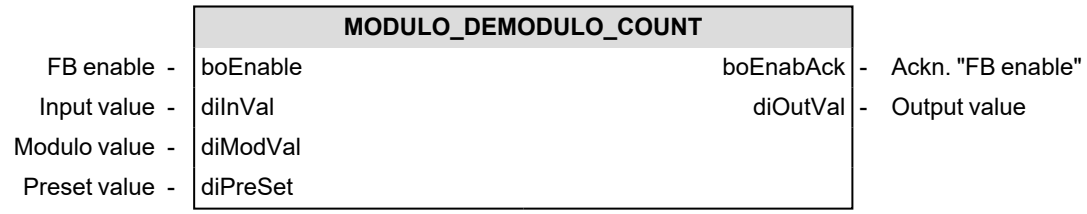
Associated with this function block, a visualisation is prepared in CoDeSys.
Siehe 'Visualization of AFL blocks' auf Seite 806.

4.2.1.1.9.5 MODULO_DEMODULO_COUNT (FB)

The function block 'MODULO_DEMODULO_COUNT' reaslises a counter that converts a modulo count value into a sequential count value.

The call of this function block is permissible in both the asynchronous program section PLC_PRG as well as the synchronous program section FPLC_PRG.

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 [increments]
diModVal	DINT	Modulo end value of the modulo counter [increments]
diPreSet	DINT	Preset value to which the output is set. The value is applied with the positive edge of 'boEnable'. [Increments]

Output variables

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

Name	Type	Description
diOutVal	DINT	Sequential Output value [increments]

Usage note in the CoDeSys program

The call of this function block is permissible in both the asynchronous program section PLC_PRG as well as the synchronous program section FPLC_PRG.

Associated with this function block, a visualisation is prepared in CoDeSys.

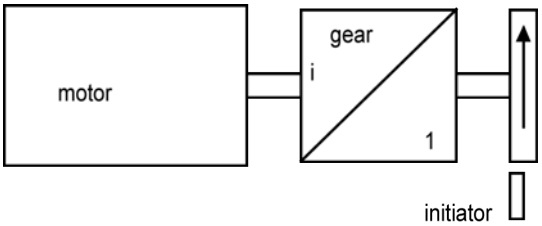
Siehe 'Visualization of AFL blocks' auf Seite 806.

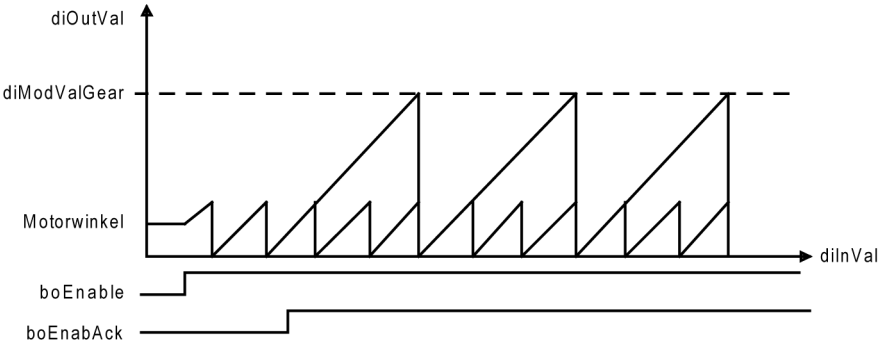
4.2.1.1.9.6 MODULO_MODCOUNT_GEAR (FB)

The function block 'MODULO_MODCOUNT_GEAR' gives the prerequisites for an exact positioning of endless rotating axes with gears beyond the switching off and on of a machine, e.g. impression cylinder.

The positioning is done modulo, i.e. within one rotation of the gear output. Therefore, the motor has to be equipped with a single turn absolute encoder (e.g. S encoder) and an initiator has to mark turn 1 of the motor within one rotation of the gear output.

The function block is called in the synchronous program level FPLC_PRG.





User interface

		MODULO_MODCOUNT_GEAR			
FB enable	-	boEnable	boEnabAck	-	Ackn. "FB enable"
Initiator Input	-	boInitiator	boOverFlow	-	Modulo overflow
Input impulse	-	diInVal	diOutVal	-	Modulo Output value
Modulo value	-	diModInVal	diIniPos	-	Initiator position
Gear ratio	-	iGearRatio	boErr	-	Error
			iErrID	-	Error ID

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.
boInitiator	BOOL	Initiator input
diInVal	DINT	Input value, counter input, actual position value motor
diModInVal	DINT	Modulo value; determines how much the output value 'diOutVal' counts (from 0 to 'diModInVal') [increments]
iGearRatio	INT	Gear ratio between input and output

Output variables

Name	Type	Description
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled
boOverFlow	BOOL	Marking of modulo flow; for overflow or underflow, 'boOverFLow' is set TRUE for one cycle.
diOutVal	DINT	Output value; the output value 'diOutVal' counts from 0 to 'diModInVal' [Inkremente]
diIniPos	DINT	Position at which the low > high flank of the initiator is detected
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 = 1: diModInVal ≤ 1 iErrID = 2: iGearRatio < 1

Usage note in the CoDeSys program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
	MODULO_MODCOUNT_GEAR

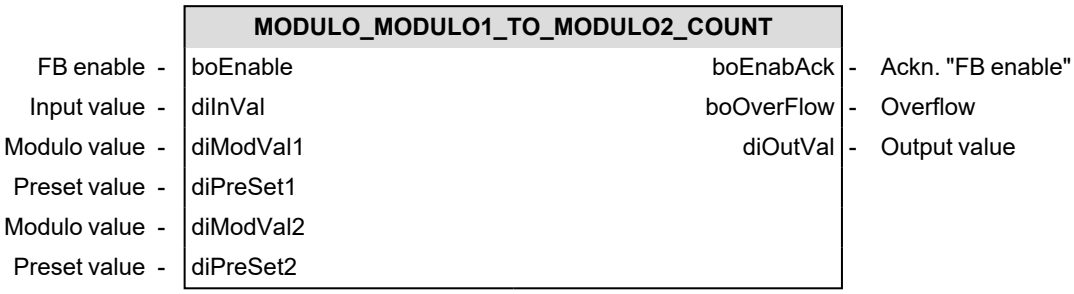
Associated with this function block, a visualisation is prepared in CoDeSys.
Siehe 'Visualization of AFL blocks' auf Seite 806.

4.2.1.1.9.7 MODULO_MODULO1_TO_MODULO2_COUNT (FB)

The function block 'MODULO_MODULO1_TO_MODULO2_COUNT' realises a conversion counter that converts one modulo count value to another modulo count value.

The call of this function block is permissible in both the asynchronous program section PLC_PRG as well as the synchronous program section FPLC_PRG.

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 [increments]
diModVal1	DINT	Input: Modulo end value of the modulo counter [increments]
diPreSet1	DINT	Preset value to which the input value is set for a positive edge at 'boEnable' [increments]
diModVal2	DINT	Output: Modulo end value of the modulo counter [increments]
diPreSet2	DINT	Preset value to which the output value is set for a positive edge at 'boEnable' [increments]

Output variables

Name	Type	Description
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled
boOverFlow	BOOL	Marking of modulo flow; for overflow or underflow, 'boOverFLow' is set TRUE for one cycle.
diOutVal	DINT	Output value [increments]

Usage note in the CoDeSys program

The call of this function block is permissible in both the asynchronous program section PLC_PRG as well as the synchronous program section FPLC_PRG.

Associated with this function block, a visualisation is prepared in CoDeSys.

Siehe 'Visualization of AFL blocks' auf Seite 806.

4.2.1.1.10 12_Increment

4.2.1.1.10.1 INCREMENT_DIFF_STAGE (FB)

The function block 'INCREMENT_DIFF_STAGE' realises a differential stage that decouples blocks and processes positions. The output of the differential stage forms a 32-bit counter.

The call of this function block is permissible in both the asynchronous program section PLC_PRG as well as the synchronous program section FPLC_PRG.

User interface

	INCREMENT_DIFF_STAGE		
FB enable -	boEnable	boEnabAck	- Ackn. "FB enable"
Reset -	boReset	diOutVal	- Output value
Input value -	diInVal		

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.
boReset	BOOL	Reset; output value = 0, then the input 'boReset' is reset by the block
diInVal	DINT	Input value [increments]

Output variables

Name	Type	Description
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled
diOutVal	DINT	Output value [increments] 32-bit counter

Usage note in the CoDeSys program

The call of this function block is permissible in both the asynchronous program section PLC_PRG as well as the synchronous program section FPLC_PRG.

Associated with this function block, a visualisation is prepared in CoDeSys.

Siehe 'Visualization of AFL blocks' auf Seite 806.

4.2.1.1.10.2 INCREMENT_DIFF_STAGE_I_TO_DI (FB)

The function block 'INCREMENT_DIFF_STAGE_I_TO_DI' realises a differential stage that decouples blocks and processes positions.

The format is converted from the INT input to a DINT output.

The call of this function block is permissible in both the asynchronous program section PLC_PRG as well as the synchronous program section FPLC_PRG.

User interface

	INCREMENT_DIFF_STAGE_I_TO_DI		
FB enable -	boEnable	boEnabAck	- Ackn. "FB enable"
Input value -	iInVal	diOutVal	- Output value
Reset -	boReset		

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 [increments]
boReset	BOOL	Reset; output value = 0, then the input 'boReset' is reset by the block

Output variables

Name	Type	Description
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled
diOutVal	DINT	Output value [increments] 32-bit counter

Usage note in the CoDeSys program

The call of this function block is permissible in both the asynchronous program section PLC_PRG as well as the synchronous program section FPLC_PRG.

Associated with this function block, a visualisation is prepared in CoDeSys.

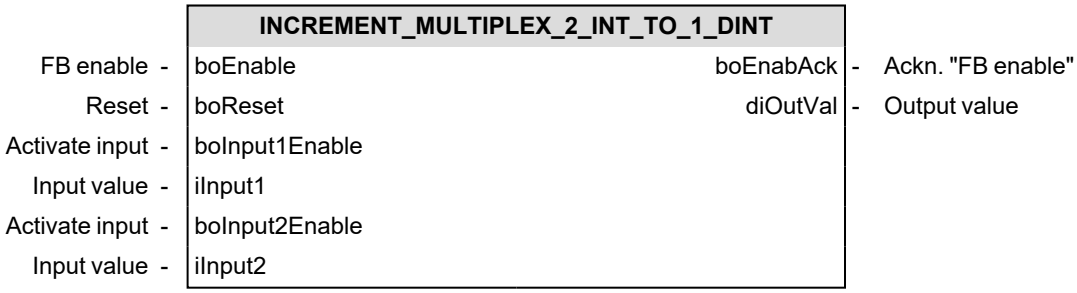
Siehe 'Visualization of AFL blocks' auf Seite 806.

4.2.1.1.10.3 INCREMENT_MULTIPLEX_2_INT_TO_1_DINT (FB)

Differential stage with 2 INT inputs. These are added together to the DINT output.

The call of this function block is permissible in both the asynchronous program section PLC_PRG as well as the synchronous program section FPLC_PRG.

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.
boReset	BOOL	Reset; output value = 0, then the input 'boReset' is reset by the block
boInput1Enable, boInput2Enable	BOOL	Activate input The difference is added to the output
iInput1, iInput2	INT	Input value [increments]

Output variables

Name	Type	Description
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled
diOutVal	DINT	Output value [increments] 32-bit counter

Usage note in the CoDeSys program

The call of this function block is permissible in both the asynchronous program section PLC_PRG as well as the synchronous program section FPLC_PRG.

Associated with this function block, a visualisation is prepared in CoDeSys.

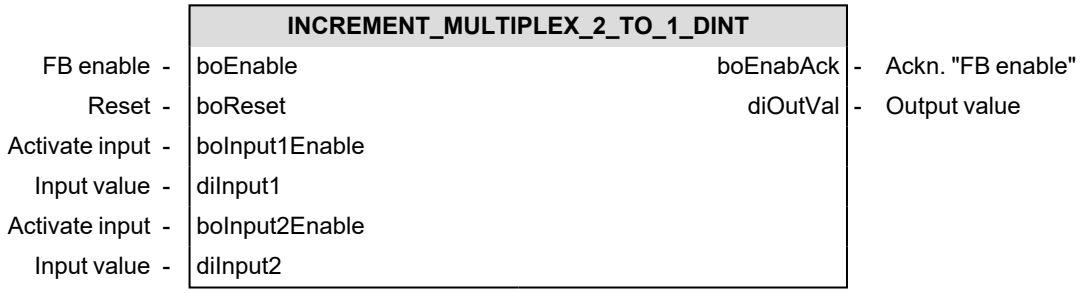
Siehe 'Visualization of AFL blocks' auf Seite 806.

4.2.1.1.10.4 INCREMENT_MULTIPLEX_2_TO_1_DINT (FB)

Multiplexer with 2 DINT inputs that are added to the output.

The call of this function block is permissible in both the asynchronous program section PLC_PRG as well as the synchronous program section FPLC_PRG.

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.
boReset	BOOL	Reset; output value = 0, then the input 'boReset' is reset by the block
boInput1Enable, boInput2Enable	BOOL	Activate input The difference is added to the output
diInput1, diInput2	DINT	Input value [increments]

Output variables

Name	Type	Description
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled
diOutVal	DINT	Output value [increments] 32-bit counter

Usage note in the CoDeSys program

The call of this function block is permissible in both the asynchronous program section PLC_PRG as well as the synchronous program section FPLC_PRG.

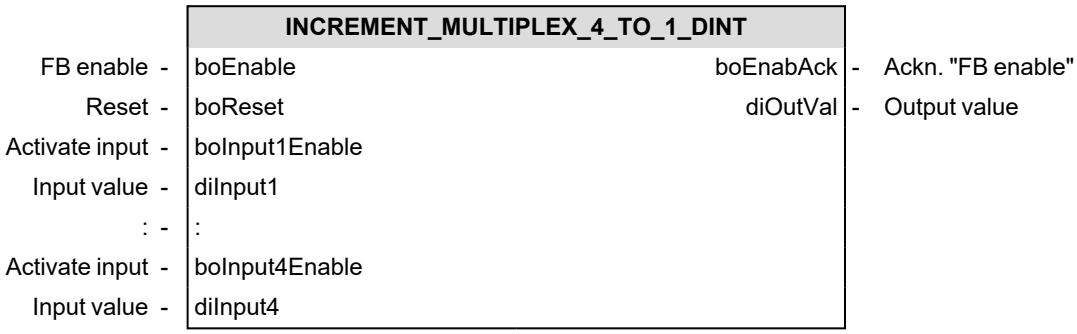
Associated with this function block, a visualisation is prepared in CoDeSys.
Siehe 'Visualization of AFL blocks' auf Seite 806.

4.2.1.1.10.5 INCREMENT_MULTIPLEX_4_TO_1_DINT (FB)

Multiplexer with 4 DINT inputs. These are added together to the output.

The call of this function block is permissible in both the asynchronous program section PLC_PRG as well as the synchronous program section FPLC_PRG.

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.
boReset	BOOL	Reset; output value = 0, then the input 'boReset' is reset by the block
boInput1Enable ... boInput4Enable	BOOL	Activate input The difference is added to the output
diInput1 ... diInput4	DINT	Input value [increments]

Output variables

Name	Type	Description
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled
diOutVal	DINT	Output value [increments] 32-bit counter

Usage note in the CoDeSys program

The call of this function block is permissible in both the asynchronous program section PLC_PRG as well as the synchronous program section FPLC_PRG.

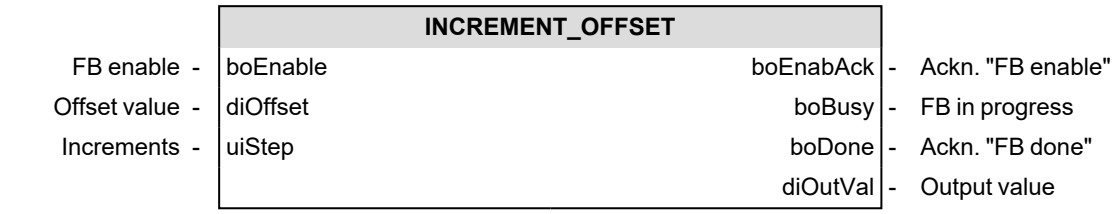
Associated with this function block, a visualisation is prepared in CoDeSys.
Siehe 'Visualization of AFL blocks' auf Seite 806.

4.2.1.1.10.6 INCREMENT_OFFSET (FB)

The function block 'INCREMENT_OFFSET' realises the output of position values for the offset. The step size and offset value can be transferred.

The call of this function block is permissible in both the asynchronous program section PLC_PRG as well as the synchronous program section FPLC_PRG.

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.
diOffset	DINT	Offset value [increments]
uiStep	UINT	Step size [increments/ID2] 'SERCOS cycle time']

Output variables

Name	Type	Description
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled
boBusy	BOOL	Execution message: This bit remains set as long as the block is being processed.
boDone	BOOL	Response that the function block has been completely executed.
diOutVal	DINT	Output value [increments]

Usage note in the CoDeSys program

The call of this function block is permissible in both the asynchronous program section PLC_PRG as well as the synchronous program section FPLC_PRG.

Associated with this function block, a visualisation is prepared in CoDeSys.

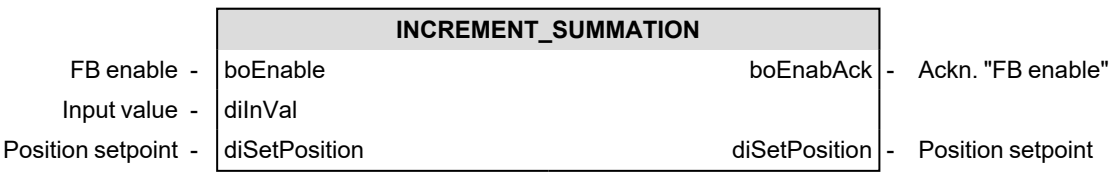
Siehe 'Visualization of AFL blocks' auf Seite 806.

4.2.1.1.10.7 INCREMENT_SUMMATION (FB)

The function block 'INCREMENT_SUMMATION' realizes a summation point for the position specification.

The function block is called in the synchronous program level FPLC_PRG.

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 [increments] Increment source that is to be written to the position setpoint system.

Output variables

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

Input and output variables

Name	Type	Description
diSetPosition	DINT	Specification of the position setpoint (position setpoint system) [increments]

Usage note in the CoDeSys program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
	INCREMENT_SUMMATION

Associated with this function block, a visualisation is prepared in CoDeSys.
Siehe 'Visualization of AFL blocks' auf Seite 806.

4.2.1.1.11 13_Timing control

4.2.1.1.11.1 TIMING_CONTROL_BLINK (FB)

The function block 'TIMING_CONTROL_BLINK' supplies a blink clock.

The times for signal on and signal off can be parametrised independently.

The call of this function block is permissible in both the asynchronous program section PLC_PRG as well as the synchronous program section FPLC_PRG.

User interface

	TIMING_CONTROL_BLINK			
FB enable -	boEnable	boEnabAck	-	Ackn. "FB enable"
Flashing signal time ON -	uiBlinkOn	boOutBlink	-	Output flashing signal
Flashing signal time OFF -	uiBlinkOff			

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.
uiBlinkOn	UINT	Time for blink signal ON [ms]
uiBlinkOff	UINT	Time for blink signal OFF [ms]

Output variables

Name	Type	Description
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled
boOutBlink	BOOL	Output flashing signal

Usage note in the CoDeSys program

The call of this function block is permissible in both the asynchronous program section PLC_PRG as well as the synchronous program section FPLC_PRG.

Associated with this function block, a visualisation is prepared in CoDeSys.

Siehe 'Visualization of AFL blocks' auf Seite 806.

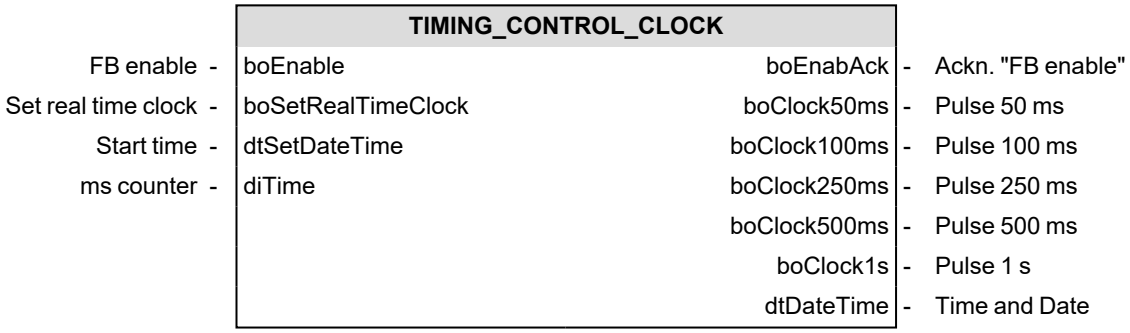
4.2.1.1.11.2 TIMING_CONTROL_CLOCK (FB)

The function block 'TIMING_CONTROL_CLOCK' provides the clock for time-controlled execution of program actions (e.g. floating point calculations and actual values outputs).

The clocks are not synchronised with one another. This means that the computational load is divided more evenly as with synchronised clocks.

The function block is called in the synchronous program level FPLC_PRG.

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.
boSetRealTimeClock	BOOL	Set real-time clock and date to start value 'dtSetDateTime'
dtSetDateTime	DATE_ AND_ TIME	Start value Time and Date
diTime	DINT	ms counter [ms]

Output variables

Name	Type	Description
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled
boClock50ms boClock100ms boClock250ms boClock500ms	BOOL	Pulse every n ms. boClocknms remains TRUE for one PLC cycle.

Name	Type	Description
boClock1s	BOOL	Pulse every 1 s. boClock1s remains TRUE for one PLC cycle.
dtDateTime	DATE_ AND_ TIME	Time and Date

Usage note in the CoDeSys program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
	TIMING_CONTROL_CLOCK

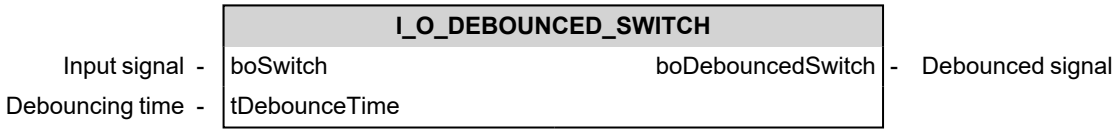
Associated with this function block, a visualisation is prepared in CoDeSys.
Siehe 'Visualization of AFL blocks' auf Seite 806.

4.2.1.1.12 14_I_O_Handling

4.2.1.1.12.1 I_O_DEBOUNCED_SWITCH (FB)

The function block 'I_O_DEBOUNCED_SWITCH' provides a debounced switch.
If the input 'boSwitch' is set to TRUE, the output 'boDebouncedSwitch' is set after 'tDebounceTime' is passed.
With 'boSwitch' = FALSE, the output 'boDebouncedSwitch' is set to FALSE, too.
The function block is called in the asynchronous program level PLC_PRG.

User interface



Input variables

Name	Type	Description
boSwitch	BOOL	Input signal to be debounced
tDebounceTime	TIME	Debouncing time [ms] (Default = 50 ms)

Output variables

Name	Type	Description
boDebouncedSwitch	BOOL	Debounced signal

Usage note in the CoDeSys program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
I_O_DEBOUNCED_SWITCH	

Associated with this function block, a visualisation is prepared in CoDeSys.
Siehe 'Visualization of AFL blocks' auf Seite 806.

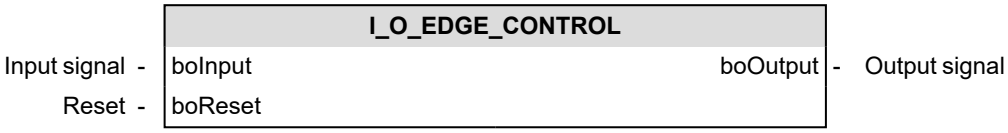
4.2.1.1.12.2 I_O_EDGE_CONTROL (FB)

The function block 'I_O_EDGE_CONTROL' controls an output 'boOutput' so that it is set by a positive edge at the input 'boInput' and reset by a negative edge.

The output is deleted by the 'boReset' input. After deletion, a 0 -> 1 edge is always needed at the 'boInput', so that the output is reset again.

The call of this function block is permissible in both the asynchronous program section PLC_PRG as well as the synchronous program section FPLC_PRG.

User interface



Input variables

Name	Type	Description
boInput	BOOL	Input signal
boReset	BOOL	Reset; The output is deleted by the 'boReset' input.

Output variables

Name	Type	Description
boOutput	BOOL	Output signal

Usage note in the CoDeSys program

The call of this function block is permissible in both the asynchronous program section PLC_PRG as well as the synchronous program section FPLC_PRG.

Associated with this function block, a visualisation is prepared in CoDeSys.

Siehe 'Visualization of AFL blocks' auf Seite 806.

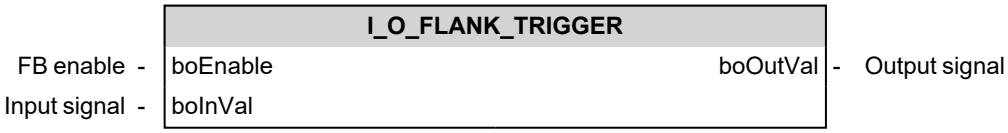
4.2.1.1.12.3 I_O_FLANK_TRIGGER (FB)

The function block 'I_O_FLANK_TRIGGER' provides a flank trigger of the input 'boInVal' to set the output 'boOutVal'.

If the flank trigger is enabled by the input 'boEnable' = TRUE, the input 'boInVal' sets the output 'boOutVal'. The output will be reset by 'boEnable' = FALSE.

The call of this function block is permissible in both the asynchronous program section PLC_PRG as well as the synchronous program section FPLC_PRG.

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 the enable signal is withdrawn, the output will be deleted.
boInVal	BOOL	Input signal, edge triggered switches the output

Output variables

Name	Type	Description
boOutVal	BOOL	Output signal

Usage note in the CoDeSys program

The call of this function block is permissible in both the asynchronous program section PLC_PRG as well as the synchronous program section FPLC_PRG.

Associated with this function block, a visualisation is prepared in CoDeSys.

Siehe 'Visualization of AFL blocks' auf Seite 806.

4.2.1.1.12.4 I_O_FLANK_TRIGGER_ACKNOWLEDGEMENT (FB)

The function block 'I_O_FLANK_TRIGGER_ACKNOWLEDGEMENT' provides a flank trigger of the input 'boInVal' to set the output 'boOutVal' when the Input 'boInValAck' = FALSE..

Is detected in 'boInValAck' the acknowledgment signal, performs a FALSE signal at the input 'boInVal' or 'boInValAck' to a reset of the output 'boOutVal'. The input 'tTimeOut' defines the time in which the acknowledgment signal at input 'boInValAck' is expected. If the acknowledgment is not in the specified time, then the output 'boErrTimeOut' is set.

The call of this function block is permissible in both the asynchronous program section PLC_PRG as well as the synchronous program section FPLC_PRG.

User interface

	I_O_FLANK_TRIGGER_ACKNOWLEDGEMENT			
Input bit -	boInVal	boOutVal	-	Output signal
Acknowledgement bit -	boInValAck	boErr	-	Error
Timeout -	tTimeOut	iErrID	-	Error ID

Input variables

Name	Type	Description
boInVal	BOOL	With a positive edge, the output 'boOutVal' is set when 'boInValAck' = FALSE
boInValAck	BOOL	Acknowledgement
tTimeOut	TIME	Timeout

Output variables

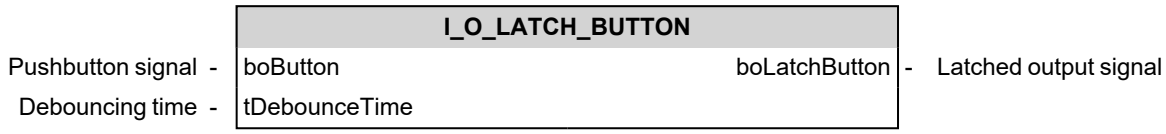
Name	Type	Description
boOutVal	BOOL	Output signal
boErr	BOOL	The function block is in an error state FALSE: No error (permitted commanding or warning) TRUE: Error
iErrID	INT	Error identity number: Diagnostic number is output iErrID = 0: No error iErrID ≠ 0, boErr = TRUE: Error iErrID ≠ 0, boErr = FALSE: Warning Value 1: Meaning: Timeout (Acknowledge signal not detected in the specified time to 'boInValAck')

4.2.1.1.12.5 I_O_LATCH_BUTTON (FB)

The function block 'I_O_LATCH_BUTTON' provides a latched pushbutton with debouncing.
On a 0 -> 1 edge of the input 'boButton', the output 'boLatchButton' is set 0 -> 1 after the 'tDebounceTime' is passed. With a new 0 -> 1 edge of the input, the output is reset 1 -> 0.

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

User interface



Input variables

Name	Type	Description
boButton	BOOL	Button to be latched and debounced
tDebounceTime	TIME	Debouncing time [ms] (Default = 50 ms)

Output variables

Name	Type	Description
boLatchButton	BOOL	Latched button

Usage note in the CoDeSys program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
I_O_LATCH_BUTTON	

Associated with this function block, a visualisation is prepared in CoDeSys.
Siehe 'Visualization of AFL blocks' auf Seite 806.

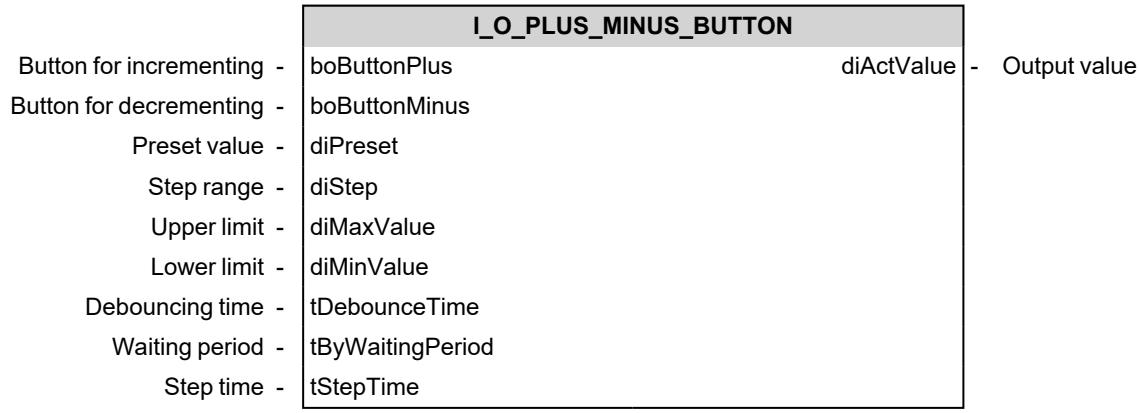
4.2.1.1.12.6 I_O_PLUS_MINUS_BUTTON (FB)

By means of the function block ' I_O_PLUS_MINUS_BUTTON', the output value is incremented resp. decremented.

If e.g. the 'boButtonPlus' is activated once, the output value 'diActValue' is incremented with 'diStep'.
If the button is activated longer than 'tByWaitingPeriod', the output is incremented automatically. The counting rapidity can be set by the input 'tStepTime'. The inputs 'diMinValue' and 'diMaxValue' limit the output value.

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

User interface



Input variables

Name	Type	Description
boButtonPlus	BOOL	Button for incrementing
boButtonMinus	BOOL	Button for decrementing
diPreset	DINT	Preset value for 'diActValue' when calling the action (actPreset)
diStep	DINT	Step width: Maximum distance that is travelled when holding down the Jog button
diMaxValue	DINT	Upper limit for output at 'diActValue'
diMinValue	DINT	Lower limit for output at 'diActValue'
tDebounceTime	TIME	Debouncing time [ms] (Default 200 ms)
tByWaitingPeriod	TIME	Waiting period until permanent counting starts [ms] (Default 1000 ms)
tStepTime	TIME	Step time for permanent counting [ms] (Default 400 ms)

Output variables

Name	Type	Description
diActValue	DINT	Output value

Usage note in the CoDeSys program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
I_O_PLUS_MINUS_BUTTON	

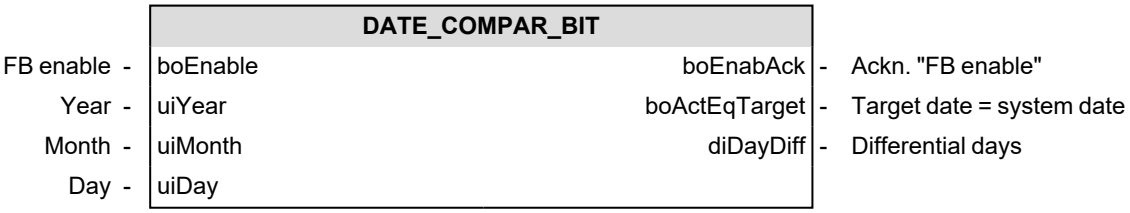
Associated with this function block, a visualisation is prepared in CoDeSys.
Siehe 'Visualization of AFL blocks' auf Seite 806.

4.2.1.1.13 17_Date

4.2.1.1.13.1 DATE_COMPAR_BIT (FB)

The function block 'DATE_COMPAR_BIT' sends a messages when the current system date matches the specified target date.
The function block is called in the asynchronous program level PLC_PRG.

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.
uiYear	UINT	Year of the date

Name	Type	Description
uiMonth	UINT	Month of the date
uiDay	UINT	Day of the date

Output variables

Name	Type	Description
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled
boActEqTarget	BOOL	Target date = system date
diDayDiff	DINT	Number of days that are between the target date and the current system date

Usage note in the CoDeSys program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
DATE_COMPAR_BIT	

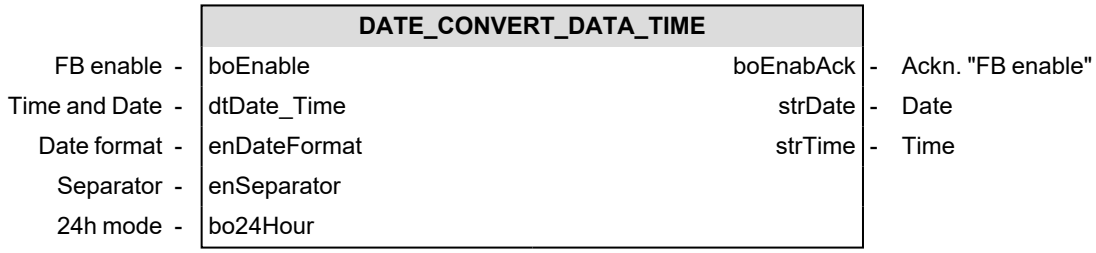
Associated with this function block, a visualisation is prepared in CoDeSys.
Siehe 'Visualization of AFL blocks' auf Seite 806.

4.2.1.1.13.2 DATE_CONVERT_DATA_TIME (FB)

The function block 'DATE_CONVERT_DATA_TIME' converts a date entry of type DT into a formatted date and time display. (Example: DT#2010-03-30-10:00:52)

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

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.
dtDate_Time	DATE_AND_TIME	Date and time that are to be converted.
enDateFormat	ENUM	Set date format: enDateFormat_YYYY_MM_DD – Order year - month - day enDateFormat_DD_MM_YYYY – Order day - month - year enDateFormat_MM_DD_YYYY – Order month - day - year

Name	Type	Description
enSeparator	ENUM	Select separator enSeparator_Point: Point separator e.g. DD.MM.YYYY: 30.03.2010 enSeparator_Minus: Minus separator e.g. YYYY-MM-DD: 2010-03-30 enSeparator_Slash: Slash separator e.g. MM/DD/YYYY: 03/30/2010
bo24Hour	BOOL	Switches between 24 and 12 hour mode bo24Hour = TRUE => 24-h mode: 23:24:53 bo24Hour = FALSE => 12-h mode: 11:24:53 pm

Output variables

Name	Type	Description
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled
strDate	STRING	Formatted date display
strTime	STRING	Formatted time display

Usage note in the CoDeSys program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
DATE_CONVERT_DATA_TIME	

Associated with this function block, a visualisation is prepared in CoDeSys.
Siehe 'Visualization of AFL blocks' auf Seite 806.

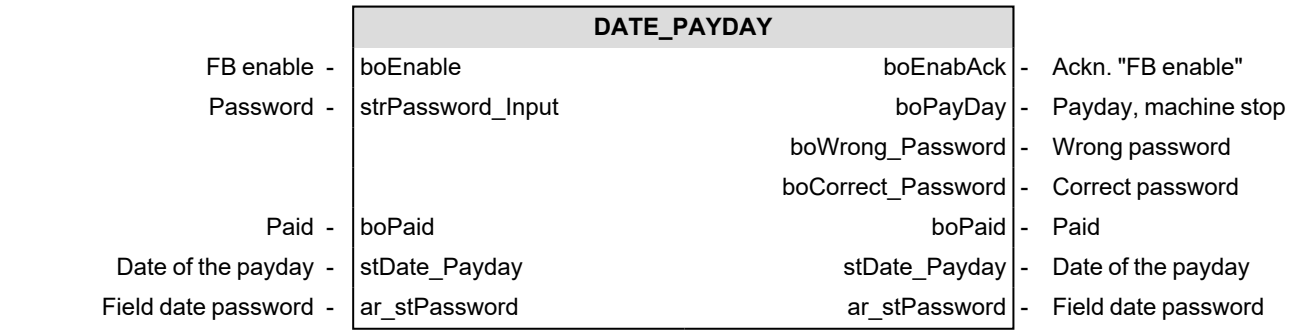
4.2.1.1.13.3 DATE_PAYDAY (FB)

The function block 'DATE_PAYDAY' compares the date from 'stDate_Payday' with the system date.

If the date has been achieved or exceeded, then the output 'boPayDay' is set to TRUE. The user must enter a password to enable the machine again.

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

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.
strPassword_Input	STRING	Input of the password with which the machine will be enabled again.

Output variables

Name	Type	Description
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled
boPayDay	BOOL	'boPayDay' = TRUE: Payday date reached, stop machine and require password.
boWrong_Password	BOOL	TRUE for one cycle when an incorrect password has been entered.
boCorrect_Password	BOOL	TRUE for one cycle when the correct password has been entered.

Input and output variables

Name	Type	Description
boPaid	BOOL	Specifies that payment has already been made. This means that a password has been entered that is stored with a payday date "0".
stDate_Payday	STRUCT	ST_DATE_PAYDAY Structure with date of the next payday.
ar_stPassword	ARRAY	ARRAY [1.. MAX_LEN_ST_PAYDAY] OF ST_PAYDAY Field, in which the date and passwords are stored. At least one field element should be transferred with a password and date 0. This will then enable the machine permanently.

The input/output variables

- boPaid
- stDate_Payday
- ar_stPassword

should be declared as remanent global

Usage note in the CoDeSys program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
DATE_PAYDAY	

Usage note in the CoDeSys program

The call of this function block is permissible in both the asynchronous program section PLC_PRG as well as the synchronous program section FPLC_PRG.

Associated with this function block, a visualisation is prepared in CoDeSys.

Siehe 'Visualization of AFL blocks' auf Seite 806.

4.2.1.1.13.4 fboDate_Leap_Year (F)

The function 'fboDate_Leap_Year' checks whether it is a leap year.

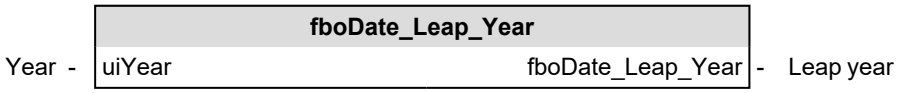
The rules is as follows:

- If it is divisible by 400, it is a leap year;
- Otherwise: If it is divisible by 100, it is not a leap year;
- Otherwise: If it is divisible by 4, it is a leap year.

(Example: 1996 = leap year; 2000 = leap year; 2100 = not a leap year)

The function is called in the asynchronous program level PLC_PRG.

User interface



Input variables

Name	Type	Description
uiYear	UINT	Year of the date that is to be checked

Output variable

Name	Type	Description
fboDate_Leap_Year	BOOL	TRUE: Leap year FALSE: Not a leap year

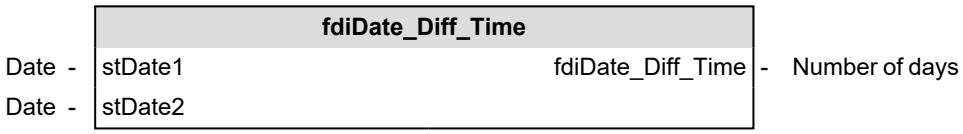
Usage note in the CoDeSys program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
fboDate_Leap_Year	

4.2.1.1.13.5 fdiDate_Diff_Time (F)

The function 'fdiDate_Diff_Time' calculates the difference between two dates in days.
The function is called in the asynchronous program level PLC_PRG.

User interface



Input variables

Name	Type	Description
stDate1	STRUCT	ST_DATE Later date
stDate2	STRUCT	ST_DATE Earlier date

Output variable

Name	Type	Description
fdiDate_Diff_Time	DINT	Difference of both days [day] "fdiDate_Diff_Time" < 0, if "stDate1" < "stDate2"

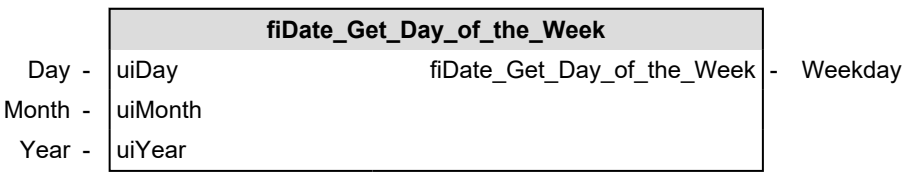
Usage note in the CoDeSys program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
fdiDate_Diff_Time	

4.2.1.1.13.6 fiDate_Get_Day_of_the_Week (F)

The function 'fiDate_Get_Day_of_the_Week' calculates the weekday of the transferred date.
The weekday is calculated according to Gaussian weekday formula. The formula applies since the introduction of the Gregorian calendar on 15.10.1582 and in principle for as long as these rules apply, i.e. until the next calendar reform.
The function is called in the asynchronous program level PLC_PRG.

User interface



Input variables

Name	Type	Description
uiDay	UINT	Day of the date for which the weekday is sought
uiMonth	UINT	Month of the date for which the weekday is sought
uiYear	UINT	Year of the date for which the weekday is sought

Output variable

Name	Type	Description
fiDate_Get_Day_of_the_Week	INT	Determined weekday: 0 = Sunday 1 = Monday 2 = Tuesday 3 = Wednesday 4 = Thursday 5 = Friday 6 = Saturday

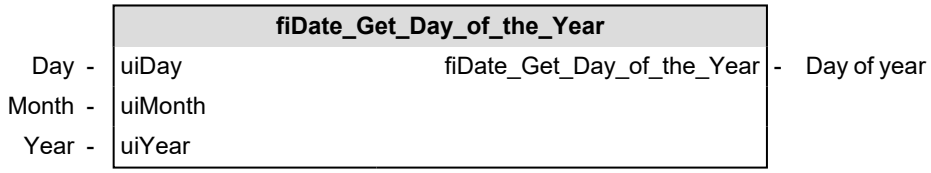
Usage note in the CoDeSys program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
fiDate_Get_Day_of_the_Week	

4.2.1.1.13.7 fiDate_Get_Day_of_the_Year (F)

The function 'fiDate_Get_Day_of_the_Year' determines the day of the year.
The function is called in the asynchronous program level PLC_PRG.

User interface



Input variables

Name	Type	Description
uiDay	UINT	Day of the date for which the day of the year is sought
uiMonth	UINT	Month of the date for which the day of the year is sought
uiYear	UINT	Year of the date for which the day of the year is sought

Output variable

Name	Type	Description
fiDate_Get_Day_of_the_Year	INT	Day of year

Usage note in the CoDeSys program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
fiDate_Get_Day_of_the_Year	

4.2.1.1.13.8 fusiDate_Get_Calendar_Week_EU (F)

The function 'fusiDate_Get_Calendar_Week_EU' determines the calendar week for the specified date.

The calculation is performed according to DIN 1355, which says:

- Each Monday and only Mondays start a new calendar week.
- The 1st calendar week is the one that has at least 4 days of the new year.

The following characteristics result from these rules:

- There are no incomplete calendar weeks; without exception each calendar week contains exactly 7 days.
- Each year has either 52 or 53 calendar weeks.
- A year has exactly 53 calendar weeks when it starts or ends with a Thursday:
  - A normal year with 53 weeks starts and ends on a Thursday.
  - A leap year with 53 weeks starts either on a Wednesday and ends thus on a Thursday or starts on a Thursday and ends on a Friday.
- The 4 January is always in calendar week 1.
- The 29, 30 and 31 December may belong to the first calendar week of the next year.
- 1, 2 and 3 January can belong to the last calendar week of the last year.
- Thursday is decisive in terms of determining which year the week belongs to. If it is in the new year, it is calendar week 1.

(from Wikipedia.org)

Standard case:

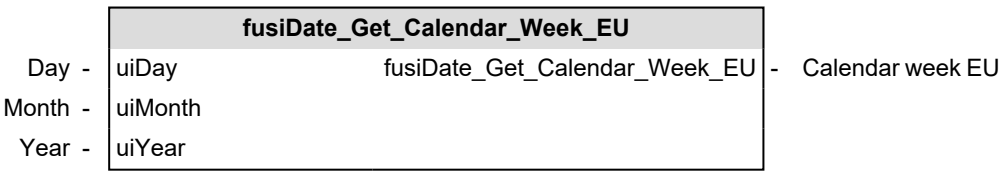
$$KW = \frac{DayOfTheYear - 1 + DayOfTheWeek(January, 1st)}{7} + 1$$

Special cases:

- 1 January of the corresponding year is a Friday or Saturday:
  $$KW = \frac{DayOfTheYear - 1 + DayOfTheWeek(January, 1st) - 7}{7} + 1$$
- Year begins 1 - 3 January:
  DayOfTheYear -1 + Weekday (1 January) - 7 ≤ 0:
  Calendar week is the last calendar week of the previous year
- Year end 29 - 31 December:
  If the above calendar week calculation results in calendar week = 53, then 1 January of this year is a Thursday or, if it is a leap year, a Wednesday.
  Otherwise it is calendar week 1 of the next year.

The function is called in the asynchronous program level PLC_PRG.

User interface



Input variables

Name	Type	Description
uiDay	UINT	Day of the date for which the calendar week is sought
uiMonth	UINT	Month of the date for which the calendar week is sought
uiYear	UINT	Year of the date for which the calendar week is sought

Output variable

Name	Type	Description
fusiDate_Get_Calendar_ Week_EU	USINT	Calendar week according to DIN 1355

Usage note in the CoDeSys program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
fusiDate_Get_Calendar_Week_EU	

4.2.1.1.13.9 fusiDate_Get_Calendar_Week_USA (F)

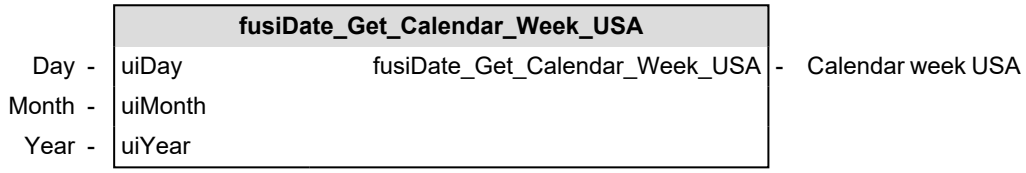
The function 'fusiDate_Get_Calendar_Week_USA' determines the calendar week of the date according to American rules.

- Week = Sun Mon .... Sat
- 1 January is in calendar week 1

$$KW = \frac{DayOfTheYear + DayOfTheWeek(January, 1st)}{7} + 1$$

The function is called in the asynchronous program level PLC_PRG.

User interface



Input variables

Name	Type	Description
uiDay	UINT	Day of the date for which the calendar week is sought
uiMonth	UINT	Month of the date for which the calendar week is sought
uiYear	UINT	Year of the date for which the calendar week is sought

Output variable

Name	Type	Description
fusiDate_Get_Calendar_ Week_USA	USINT	Calendar week according to American rules

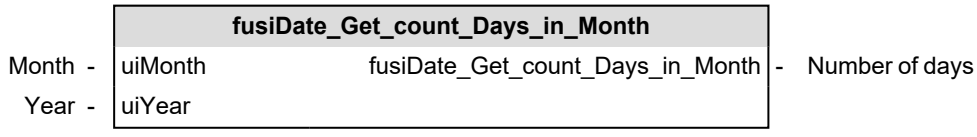
Usage note in the CoDeSys program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
fusiDate_Get_Calendar_Week_USA	

4.2.1.1.13.10 fusiDate_Get_count_Days_in_Month (F)

The function 'fusiDate_Get_count_Days_in_Month' calculates the total number of days in a month.
The function is called in the asynchronous program level PLC_PRG.

User interface



Input variables

Name	Type	Description
uiMonth	UINT	Month for which the number of days should be determined.
uiYear	UINT	Year of the month for which the number of days should be determined.

Output variable

Name	Type	Description
fusiDate_Get_count_Days_ in_Month	USINT	Number of days

Usage note in the CoDeSys program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
fusiDate_Get_count_Days_in_Month	

4.2.1.1.13.11 fusiDate_Get_Eastern (F)

The function 'fusiDate_Get_Eastern' calculates the day in the year on which Easter falls.

Easter is a variable holiday. Since the Council of Nicaea in 325 A.D., Easter is celebrated on the first Sunday (Easter Sunday) after the first full month after the start of spring. This makes 22 March the earliest date and 25 April the latest date on which Easter may fall.

The date is calculated according to the Gaussian Easter formula with the correction of 1816.

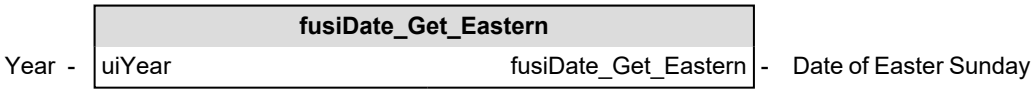
A number of other variable Christian holidays depend on this date, such as Pentecost.

Offset for other days (holidays)

Day	Offset [days]
"Fat Thursday" (Thursday before Ash Wednesday)	-52
Shrove Monday	-48
Shrove Tuesday	-47
Ash Wednesday	-46
Maundy Thursday	-3
Good Friday	-2
Easter Sunday	0
Easter Monday	+1
Ascension Day	+39
Pentecost Sunday	+49
Pentecost Monday	+50
Corpus Christi	+60

The function is called in the asynchronous program level PLC_PRG.

User interface



Input variables

Name	Type	Description
uiYear	UINT	Year for which the Easter date should be determined

Output variable

Name	Type	Description
fusiDate_Get_Eastern	USINT	Day of the Easter festival (format: 2#M00D DDDD) • Bit 0..4: day of the date • Bit 7 = 0: March Bit 7 = 1: April Examples: 2007: 1000 1000 => 08.04. 2008: 0001 0111 => 23.03.

Usage note in the CoDeSys program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
fusiDate_Get_Eastern	

4.2.1.1.14 18_File

4.2.1.1.14.1 Support

FILE_SORT_LIST_ALPHABETIC (FB)

The function block 'FILE_SORT_LIST_ALPHABETIC' sorts a file list. This list is filled by the function block 'FILE_FIND'.
The function block is called in the asynchronous program level PLC_PRG.

'Support' functions and function blocks will not be used directly. They are called as subroutines by other functions or function blocks.

User interface

	FILE_SORT_LIST_ALPHABETIC			
FB execution -	boExec	boDone	-	Ackn. "FB done"
Number of files -	iFilesFound			
File list -	arystFileName	arystFileName	-	File list

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.
iFilesFound	INT	Number of files found in 'arystFileName'

Output variables

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

Input and output variables

Name	Type	Description
arystFileName	ARRAY	ARRAY[0..MAX_LEN_ST_FILE] OF ST_FILE List of file names that are to be sorted

Usage note in the CoDeSys program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
FILE_SORT_LIST_ALPHABETIC	

It is recommended that you call blocks that work with file access in a separate task in the asynchronous program level PLC_PRG (type "asynchronous").

FILE_STRING_FILTER (FB)

The function block 'FILE_STRING_FILTER' removes or replaces forbidden characters from a string.
The function block is called in the asynchronous program level PLC_PRG.

'Support' functions and function blocks will not be used directly. They are called as subroutines by other functions or function blocks.

User interface

		FILE_STRING_FILTER			
FB execution	-	boExec	strFilterString	-	Filtered string
Characters to be filtered out	-	strForbiddenChar	iForbiddenCharCount	-	Number of forbidden characters
Replacement characters	-	strReplaceChar	boDone	-	Ackn. "FB done"
Replace/delete	-	boReplace	boErr	-	Error
String that is checked	-	strCheckString	iErrID	-	Error ID
			strErr	-	String with filt. characters

Input variables

Name	Type	Description
boExec	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.
strForbiddenChar	STRING (15)	The string can transfer up to 15 characters that are to be sorted out. Default \ : * ? " < > |
strReplaceChar	STRING (1)	Replacement characters Default _
boReplace	BOOL	boReplace = TRUE: Characters found are replaced by the replacement character. boReplace = FALSE: Characters found are deleted from the string
strCheckString	STRING (64)	String that will be checked

Output variables

Name	Type	Description
strFilterString	STRING (64)	Output string without the forbidden characters
iForbiddenCharCount	INT	Number of forbidden characters found in 'strCheckString'
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 1 — 'strForbiddenChar' is not defined or has the length 0 2 — 'strReplaceChar' is not defined or has the length 0 3 — 'strCheckString' is not defined or has the length 0
strErr	STRING (64)	Return message of the forbidden characters that were found.

Usage note in the CoDeSys program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
FILE_STRING_FILTER	

It is recommended that you call blocks that work with file access in a separate task in the asynchronous program level PLC_PRG (type "asynchronous").

4.2.1.1.14.2 FILE_COPY (FB)

The function block 'FILE_COPY' copies files.

If copies should be made between external storage media, such as USB storage media or network drives, access must be organised first with the support function 'FiFileConnect' (AmkFile.lib).

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

User interface

		FILE_COPY			
FB execution	-	boExec	udSize	-	File size
File name Source	-	strSourceFile	boErr	-	Error
File name Target	-	strDestinationFile	iErrID	-	Error ID
Pointer to Buffer	-	ptbyWorkBuffer	enErrName	-	Name of faulty FB
Buffer size	-	udWorkBufferSize	boDone	-	Ackn. "FB done"

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.
strSourceFile	STRING	STRING(64) File name, including extension, total maximum 64 characters
strDestinationFile	STRING	STRING(64) File name, including extension, total maximum 64 characters
ptbyWorkBuffer	POINTER	POINTER TO BYTE Pointer for the buffer in which the date of the source can be buffered.
udWorkBufferSize	UDINT	Size of the buffer [byte]

Output variables

Name	Type	Description
udSize	UDINT	Size of the source file [byte]

<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><table><tr><td>1</td><td>In 'strSourceFile', no file name is specified</td></tr><tr><td>2</td><td>LEN(strSourceFile) > 64: file name is too long</td></tr><tr><td>3</td><td>In 'strDestinationFile', no file name is specified</td></tr><tr><td>4</td><td>LEN(strDestinationFile) > 64: file name is too long</td></tr><tr><td>5</td><td>ptbyWorkBuffer: Pointer to the buffer area is invalid</td></tr></table></td></tr>
<tr><td>enErrName</td><td>ENUM</td><td>EN_FB_NAME
Name of faulty block for which the diagnostic number is output.
The diagnostic number is explained in the description of the corresponding library.
<table><tr><th>Block</th><th>Library</th></tr><tr><td>READ_FILE_1</td><td>AmkFile.lib</td></tr><tr><td>WRITE_FILE_1</td><td>AmkFile.lib</td></tr><tr><td>SIZE_FILE_1</td><td>AmkFile.lib</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>

Usage note in the CoDeSys program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
FILE_COPY	

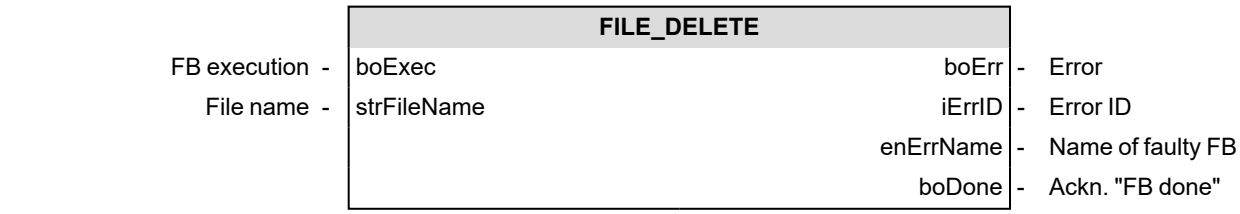
It is recommended that you call blocks that work with file access in a separate task in the asynchronous program level PLC_PRG (type "asynchronous").

Associated with this function block, a visualisation is prepared in CoDeSys.
Siehe 'Visualization of AFL blocks' auf Seite 806.

4.2.1.1.14.3 FILE_DELETE (FB)

The function block 'FILE_DELETE' deletes the specified file.
The function block is called in the asynchronous program level PLC_PRG.

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	File name, including extension

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 1: In 'strFileName', no file name is specified 2: LEN(strFileName) > 64: file name is too long
enErrName	ENUM	EN_FB_NAME Name of faulty block for which the diagnostic number is output. The diagnostic number is explained in the description of the corresponding library. Block: REMOVE_FILE_1 — Library: AmkFile.lib

Usage note in the CoDeSys program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
FILE_DELETE	

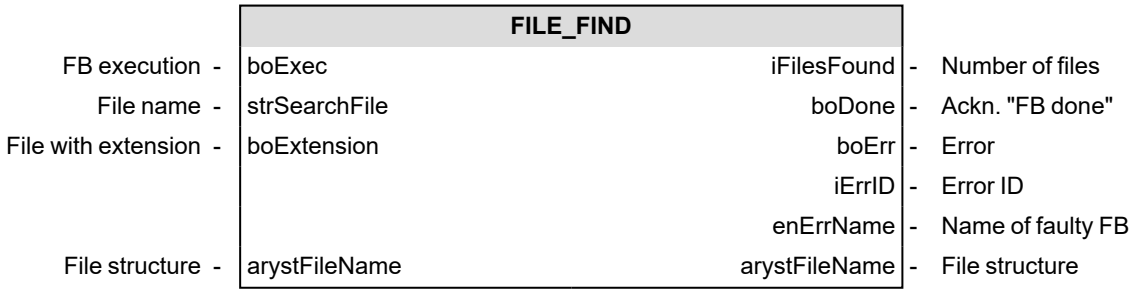
It is recommended that you call blocks that work with file access in a separate task in the asynchronous program level PLC_PRG (type "asynchronous").

Associated with this function block, a visualisation is prepared in CoDeSys.
Siehe 'Visualization of AFL blocks' auf Seite 806.

4.2.1.1.14.4 FILE_FIND (FB)

The function block 'FILE_FIND' finds one or more files and stores their names and size in the an array.
The function block is called in the asynchronous program level PLC_PRG.

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.
strSearchFile	STRING	STRING(64) File name that will be searched for. Wildcards can also be used, such as '*.*'

Name	Type	Description
boExtension	BOOL	Files with extension

Output variables

Name	Type	Description
iFilesFound	INT	Number of files found
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 1: In 'strSearchFile', no search criteria are specified 2: LEN(strSearchFile) > 64: search string is too long
enErrName	ENUM	EN_FB_NAME Name of faulty block for which the diagnostic number is output. The diagnostic number is explained in the description of the corresponding library. Block: FIND_FILE_1 — Library: AmkFile.lib

Input and output variables

Name	Type	Description
arystFileName	ARRAY	ARRAY[0..MAX_LEN_ST_FILE] OF ST_FILE File structure

Usage note in the CoDeSys program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
FILE_FIND	

It is recommended that you call blocks that work with file access in a separate task in the asynchronous program level PLC_PRG (type "asynchronous").

Associated with this function block, a visualisation is prepared in CoDeSys.
Siehe 'Visualization of AFL blocks' auf Seite 806.

4.2.1.1.14.5 FILE_READ (FB)

The function block 'FILE_READ' is used to read a file and write the content to a memory area.
The function block is called in the asynchronous program level PLC_PRG.

User interface

	FILE_READ		
FB execution -	boExec	udNoByte	- Number of bytes
File name -	strFileName	udSizeOfFile	- File size
Buffer size -	udSize	boDone	- Ackn. "FB done"
Pointer to Buffer -	pbyFileBuff	boErr	- Error
Read mode -	enReadMode	iErrID	- Error ID
		enErrName	- Name of faulty FB

Input variables

Name	Type	Description
boExec	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.
strFileName	STRING (64)	STRING(64) File name, including extension, total maximum 64 characters
udSize	UDINT	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.
pbyFileBuff	POINTER	POINTER TO BYTE Pointer to the data memory to which the data from the file are to be written
enReadMode	ENUM	EN_READ_MODE Selection of read mode READ_BEGIN: The memory area is overwritten with the date from the file from the start. READ_CURRENT: The memory area is overwritten with the date from the file from the current position.

Output variables

Name	Type	Description
udNoByte	UDINT	Number of read bytes from the file [byte]
udSizeOfFile	UDINT	Size of the read file [byte] udSize = 0 => udSizeOfFile = size of the file otherwise: udSizeOfFile = udSize
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 1: strFileName: File name not defined 2: LEN(strFileName) > 64: file name too long 3: pbyFileBuff: Pointer to the buffer area is not defined 4: udSizeOfFile = 0: Dateilänge = 0
enErrName	ENUM	EN_FB_NAME Name of faulty block for which the diagnostic number is output. The diagnostic number is explained in the description of the corresponding library. READ_FILE_1: AmkFile.lib SIZE_FILE_1: AmkFile.lib

Usage note in the CoDeSys program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
FILE_READ	

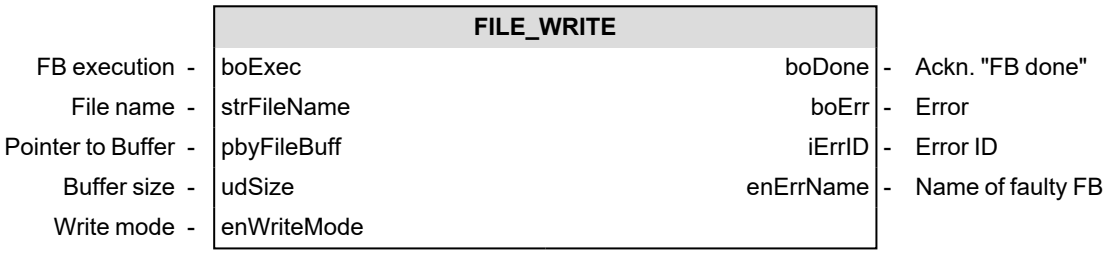
It is recommended that you call blocks that work with file access in a separate task in the asynchronous program level PLC_PRG (type "asynchronous").

Associated with this function block, a visualisation is prepared in CoDeSys.
Siehe 'Visualization of AFL blocks' auf Seite 806.

4.2.1.1.14.6 FILE_WRITE (FB)

The function block 'FILE_WRITE' is used to write memory data to a file.
After the write procedure has been completed, a check is performed whether the sizes of the file and the memory match.
The function block is called in the asynchronous program level PLC_PRG.

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	STRING(64) File name, including extension, total maximum 64 characters
pbyFileBuff	POINTER	POINTER TO BYTE Pointer to the data memory that contains the data that is to be written to the file
udSize	UDINT	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.
enWriteMode	ENUM	EN_WRITE_MODE Selection of write mode WRITE_NEW: The content of the memory overwrites the previous content of the file. WRITE_APPEND: The content of the memory is appended to the previous content of the file.

Output variables

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

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 1 — strFileName: File name not defined 2 — LEN(strFileName) > 64: file name too long 3 — pbyFileBuff: Pointer to the buffer area is undefined 4 — udSize: Number of bytes to be written is invalid
boErr	BOOL	The function block is in an error state FALSE — No error (permitted commanding or warning) TRUE — Error
enErrName	ENUM	EN_FB_NAME Name of faulty block for which the diagnostic number is output. The diagnostic number is explained in the description of the corresponding library. WRITE_FILE_1 — AmkFile.lib SIZE_FILE_1 — AmkFile.lib

Conversion in the CoDeSys program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
FILE_WRITE	

It is recommended that you call blocks that work with file access in a separate task in the asynchronous program level PLC_PRG (type "asynchronous").

Associated with this function block, a visualisation is prepared in CoDeSys.
Siehe 'Visualization of AFL blocks' auf Seite 806.

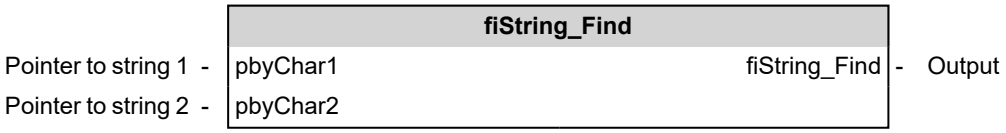
4.2.1.1.15 19_Stringfunctions

4.2.1.1.15.1 fiString_Find (F)

The function 'fiString_Find' locates a part of a string.

The call of this function is permissible in both the asynchronous program section PLC_PRG as well as the synchronous program section FPLC_PRG.

User interface



Input variables

Name	Type	Description
pbyChar1	POINTER	POINTER TO BYTE Pointer to string 1
pbyChar2	POINTER	POINTER TO BYTE Pointer to string 2

Output variable

Name	Type	Description
fiString_Find	INT	Output Position of the first character of the first occurence of 'pbyChar2' in 'pbyChar1'. If 'pbyChar2' is not part of 'pbyChar1', fiString_Find := 0.

Usage not in the CoDeSys program

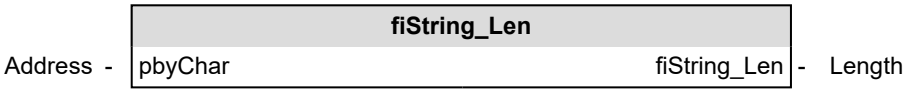
The call of this function is permissible in both the asynchronous program section PLC_PRG as well as the synchronous program section FPLC_PRG.

4.2.1.1.15.2 fiString_Len (F)

The function 'fiString_Len' determines the length of a string (number of characters). The string can also contain more than 255 characters.

The call of this function is permissible in both the asynchronous program section PLC_PRG as well as the synchronous program section FPLC_PRG.

User interface



Input variables

Name	Type	Description
pbyChar	POINTER	POINTER TO BYTE Address of the string

Output variable

Name	Type	Description
fiString_Len	INT	Length of the string, number of characters

Usage note in the CoDeSys program

The call of this function is permissible in both the asynchronous program section PLC_PRG as well as the synchronous program section FPLC_PRG.

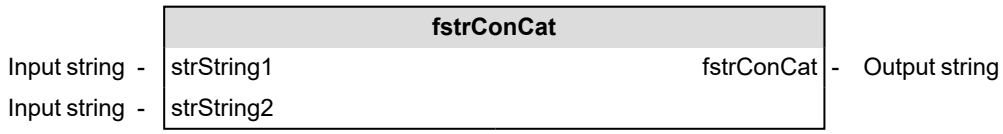
4.2.1.1.15.3 fstrConCat (F)

The function 'fstrConCat' merges two STRING variables.

The function is called in the asynchronous program level PLC_PRG.

In the AMKAMAC A4 control unit, use this function instead of CONCAT from the Standard.lib library.

User interface



Input variables

Name	Type	Description
strString1	STRING	Input string 1, maximum length 80 characters
strString2	STRING	Input string 2, maximum length 80 characters

Output variable

Name	Type	Description
fstrConCat	STRING	Output string, maximum length 80 characters

Usage note in the CoDeSys program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
fstrConCat	

4.2.1.1.15.4 fstrString_Append

The function 'fstrString_Append' combines two strings. String 2 is appended to string 1 repeatedly until the character length of 'iMaxLenStr' has been reached.

Examples:

iMaxLenStr	String 1	String 2	String 1 new
0	Hello	world	
3	Hello	world	Hel
7	Hello	world	HelloWo
9	Hello	world	HelloWorl
15	Hello	world	HelloWorldWorld

The call of this function is permissible in both the asynchronous program section PLC_PRG as well as the synchronous program section FPLC_PRG.

User interface

	fstrString_Append		
Address -	pbyString1	fstrString_Append	-
Address -	pbyString2		
Max. length -	iMaxLenStr		

Input variables

Name	Type	Description
pbyString1	POINTER	POINTER TO BYTE Address of string 1
pbyString2	POINTER	POINTER TO BYTE Address of string 2
iMaxLenStr	INT	Maximum string length for new string 1. Determines through the length of string 1 the number of times that string 2 can be appended to string 1.

Output variable

Name	Type	Description
fstrString_Append	STRING	No response

Usage note in the CoDeSys program

The call of this function is permissible in both the asynchronous program section PLC_PRG as well as the synchronous program section FPLC_PRG.

4.2.1.1.15.5 fstrString_Append1 (F)

The function fstrString_Append1 combines two strings in a new string. String 2 is appended to string 1 until the character length of iMaxLenStr has been reached.

Examples:

iMaxLenStr	String 1	String 2	String
0	Hello	world	
3	Hello	world	Hel
7	Hello	world	HelloWo
9	Hello	world	HelloWorl
15	Hello	world	HelloWorldWorld

The call of this function is permissible in both the asynchronous program section PLC_PRG as well as the synchronous program section FPLC_PRG.

User interface

	fstrString_Append1		
Address -	pbyString	fstrString_Append1	-
Address -	pbyString1		
Address -	pbyString2		
Max. length -	iMaxLenStr		

Input variables

Name	Type	Description
pbyString	POINTER	POINTER TO BYTE Address of string
pbyString1	POINTER	POINTER TO BYTE Address of string 1
pbyString2	POINTER	POINTER TO BYTE Address of string 2
iMaxLenStr	INT	Maximum string length

Output variable

Name	Type	Description
fstrString_Append1	STRING	Composite string

Usage note in the CoDeSys program

The call of this function is permissible in both the asynchronous program section PLC_PRG as well as the synchronous program section FPLC_PRG.

4.2.1.1.15.6 fstrString_Compare (F)

The function 'fstrString_Compare' compares the length of two strings.

The call of this function is permissible in both the asynchronous program section PLC_PRG as well as the synchronous program section FPLC_PRG.

User Interface

	fstrString_Compare			
Pointer to string 1 -	pbyChar1	fstrString_Compare	-	Output value
Pointer to string 2 -	pbyChar2			

Input variables

Name	Type	Description
pbyChar1	POINTER	POINTER TO BYTE Pointer to string 1
pbyChar2	POINTER	POINTER TO BYTE Pointer to string 2

Output variable

Name	Type	Description
fstrString_Compare	STRING	Output value 0 length (pbyChar1) = length (pbyChar2) 1 length (pbyChar1) > length (pbyChar2) -1 length (pbyChar1) < length (pbyChar2)

Usage note in the CoDeSys program

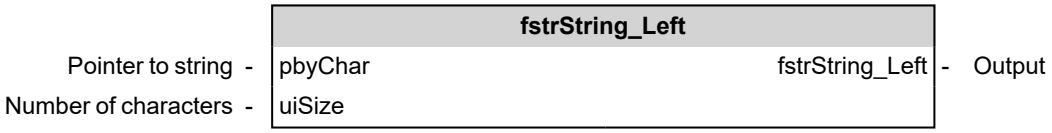
The call of this function is permissible in both the asynchronous program section PLC_PRG as well as the synchronous program section FPLC_PRG.

4.2.1.1.15.7 fstrString_Left (F)

The function 'fstrString_Left' outputs the first 'uiSize' characters of a string, beginning left.

The call of this function is permissible in both the asynchronous program section PLC_PRG as well as the synchronous program section FPLC_PRG.

User Interface



Input variables

Name	Type	Description
pbyChar	POINTER	POINTER TO BYTE Pointer to string
uiSize	UINT	Number of characters

Output variable

Name	Type	Description
fstrString_Left	STRING	Output fstrString_Left(pbyChar, uiSize) means: take the first 'uiSize' characters of 'pbyChar', beginning left Example: fstrString_Left(SUSI,3) = SUS

Usage note in the CoDeSys program

The call of this function is permissible in both the asynchronous program section PLC_PRG as well as the synchronous program section FPLC_PRG.

4.2.1.1.15.8 fstrString_Mid (F)

The function 'fstrString_Mid' extracts a part of a string.

The call of this function is permissible in both the asynchronous program section PLC_PRG as well as the synchronous program section FPLC_PRG.

Input variables

Name	Type	Description
pbyChar	POINTER	POINTER TO BYTE Pointer to string
uiPos	UINT	Start position
uiSize	UINT	Number of characters

Output variable

Name	Type	Description
fstrString_Mid	STRING	Output fstrString_Mid(pbyChar, uiPos, uiSize) means: take 'uiSize' characters of the string 'pbyChar', beginning with the 'uiPos'th. Example: fstrString_Mid(SUSI,2,2) = US

Usage note in the CoDeSys program

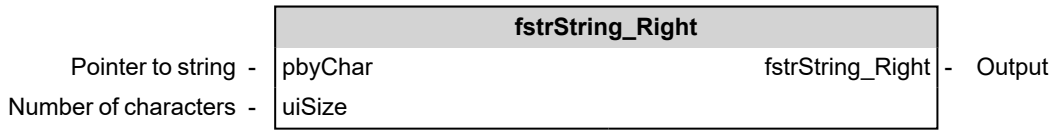
The call of this function is permissible in both the asynchronous program section PLC_PRG as well as the synchronous program section FPLC_PRG.

4.2.1.1.15.9 fstrString_Right (F)

The function 'fstrString_Right' extracts the first 'uiSize' characters of a string, beginning right.

The call of this function is permissible in both the asynchronous program section PLC_PRG as well as the synchronous program section FPLC_PRG.

User Interface



Input variablew

Name	Type	Description
pbyChar	POINTER	POINTER TO BYTE Pointer to string
uiSize	UINT	Number of characters

Output variable

Name	Type	Description
fstrString_Right	STRING	Output fstrString_Right(pbyChar, uiSize) means: take the first 'uiSize' characters of the string 'pbyChar', beginning right. Example: fstrString_Right(SUSI,3) = USI

Usage note in the CoDeSys program

The call of this function is permissible in both the asynchronous program section PLC_PRG as well as the synchronous program section FPLC_PRG.

4.2.1.1.16 20_Virtual Master

4.2.1.1.16.1 VIRTUAL_RFGEN (FB)

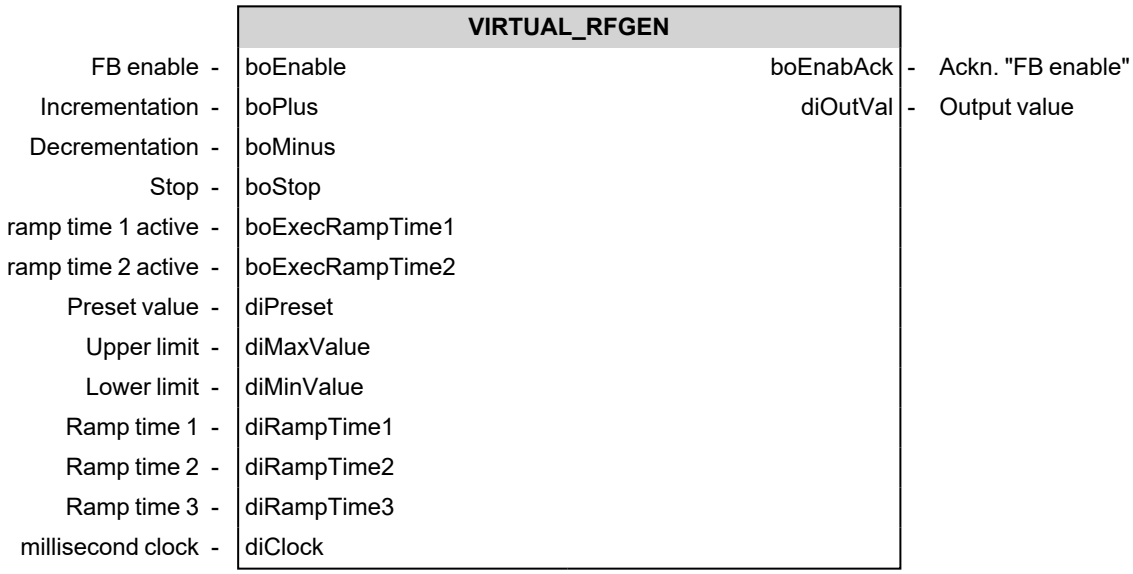
The function block 'VIRTUAL_RFGEN' (Ramp Function Generator) implements an acceleration ramp generator. The output 'diOutVal' is adapted by ramp times in order to avoid sudden changes of the setpoint values.

Properties of VIRTUAL_RFGEN

- When the block is enabled by 'boEnable', the output is set to 'diMinVal'.
- If 'diPreset' > 'diMinVal', the output is incremented during the selected ramp time up to 'diPreset'.
- By means of 'boStop' = TRUE, the output is decremented to 'diMinVal' during the selected ramp time.
- It depends on the inputs 'boExecRampTime1' and 'boExecRampTime2', which ramp time will become active.
  If none of these inputs is set TRUE, 'diRampTime3' is valid.
- By means of the inputs 'boPlus' and 'boMinus', the output can be incremented and decremented with the active ramp.
- The output is limited by 'diMaxVal' and 'diMinVal'. They both have to be positive values.

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

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.
boPlus	BOOL	Activate increment counting
boMinus	BOOL	Activate decrement counting
boStop	BOOL	Stop (decrement counting)
boExecRampTime1	BOOL	Ramp time 'diRampTime1' is active
boExecRampTime2	BOOL	Ramp time 'diRampTime2' is active
diPreset	DINT	Preset value

Name	Type	Description
diMaxValue	DINT	Upper limit
diMinValue	DINT	Lower limit
diRampTime1	DINT	Ramp time 1 [s] (Default: 5s)
diRampTime2	DINT	Ramp time 2 [s] (Default: 10s)
diRampTime3	DINT	Ramp time 3 [s] (Default: 20s)
diClock	DINT	ms counter [ms]

Output variables

Name	Type	Description
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled
diOutVal	DINT	Output value

Usage note in the CoDeSys program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
VIRTUAL_RFGEN	

Associated with this function block, a visualisation is prepared in CoDeSys.
Siehe 'Visualization of AFL blocks' auf Seite 806.

4.2.1.1.16.2 VIRTUAL_VGEN (FB)

The function block 'VIRTUAL_VGEN' realises a virtual velocity generator with the specification of an emergency-stop ramp.
The function block is called in the synchronous program level FPLC_PRG.

User interface

		VIRTUAL_VGEN			
FB execution	-	boExec	boBusy	-	FB in progress
Emergency stop	-	boEmergency_Stop	boErr	-	Error
Emergency stop ramp	-	diEmergency_StopRamp	iErrID	-	Error ID
Velocity setpoint	-	diVelocity	enErrName	-	Name of faulty FB
Velocity override %	-	siOverride			
Position setpoint	-	diSetPosition	diSetPosition	-	Position setpoint
Device structure	-	stDevice	stDevice	-	Device structure

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.
boEmergency_Stop	BOOL	EMERGENCY STOP: The setpoint of the velocity is decreased to zero along the emergency-stop ramp. Once initiated, an emergency stop cannot be aborted.
diEmergency_StopRamp	DINT	EMERGENCY-STOP ramp: Ramp time in which the drive is slowed down from the current velocity to zero [ms].
diVelocity	DINT	Velocity setpoint [increments/s]
siOverride	SINT	Velocity output factor

Output variables

Name	Type	Description
boBusy	BOOL	Execution message: This bit remains set as long as the block is being processed.
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
enErrName	ENUM	EN_FB_NAME Name of faulty block for which the diagnostic number is output. The diagnostic number is explained in the description of the corresponding library. Block: VGEN_AJ — Library: AmkBase.lib

Input and output variables

Name	Type	Description
diSetPosition	DINT	Specification of the position setpoint (position setpoint system) [increments]
stDevice	STRUCT	ST_DEVICE The device description structure assigns the block a device. (See document Software description AmkBase Bibliothek, Part no.204986)

Usage note in the CoDeSys program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
	VIRTUAL_VGEN

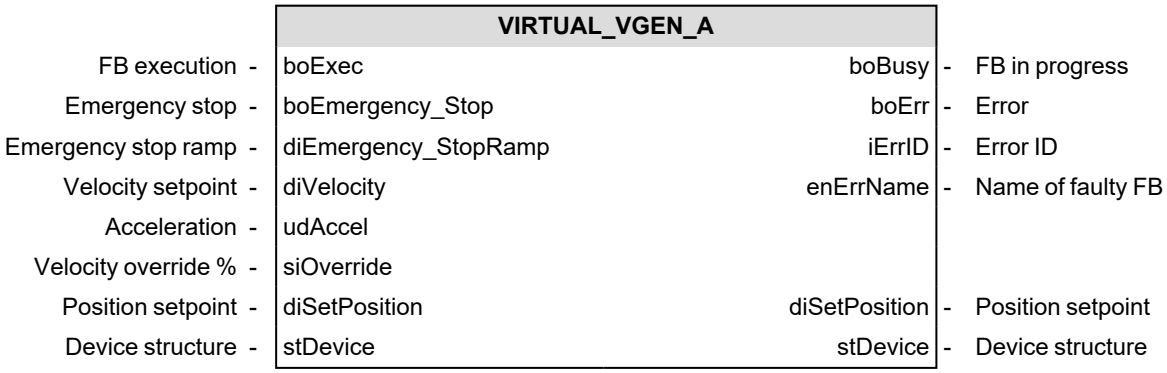
Associated with this function block, a visualisation is prepared in CoDeSys.
Siehe 'Visualization of AFL blocks' auf Seite 806.

4.2.1.1.16.3 VIRTUAL_VGEN_A (FB)

The function block 'VIRTUAL_VGEN_A' realises a virtual velocity generator with the specification
- Of an acceleration ramp
- Of an emergency-stop ramp.

The function block is called in the synchronous program level FPLC_PRG.

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.
boEmergency_Stop	BOOL	EMERGENCY STOP: The setpoint of the velocity is decreased to zero along the emergency-stop ramp. Once initiated, an emergency stop cannot be aborted.
diEmergency_StopRamp	DINT	EMERGENCY-STOP ramp: Ramp time in which the drive is slowed down from the current velocity to zero [ms].
diVelocity	DINT	Velocity setpoint [increments/s]
udAccel	UDINT	Acceleration with which the target velocity is run
siOverride	SINT	Velocity output factor

Output variables

Name	Type	Description
boBusy	BOOL	Execution message: This bit remains set as long as the block is being processed.
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
enErrName	ENUM	EN_FB_NAME Name of faulty block for which the diagnostic number is output. The diagnostic number is explained in the description of the corresponding library. Block: VGEN_AJ — Library: AmkBase.lib

Input and output variables

Name	Type	Description
diSetPosition	DINT	Specification of the position setpoint (position setpoint system) [increments]
stDevice	STRUCT	ST_DEVICE The device description structure assigns the block a device. (See document Software description AmkBase Bibliothek, Part no.204986)

Usage note in the CoDeSys program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
	VIRTUAL_VGEN_A

Associated with this function block, a visualisation is prepared in CoDeSys.
Siehe 'Visualization of AFL blocks' auf Seite 806.

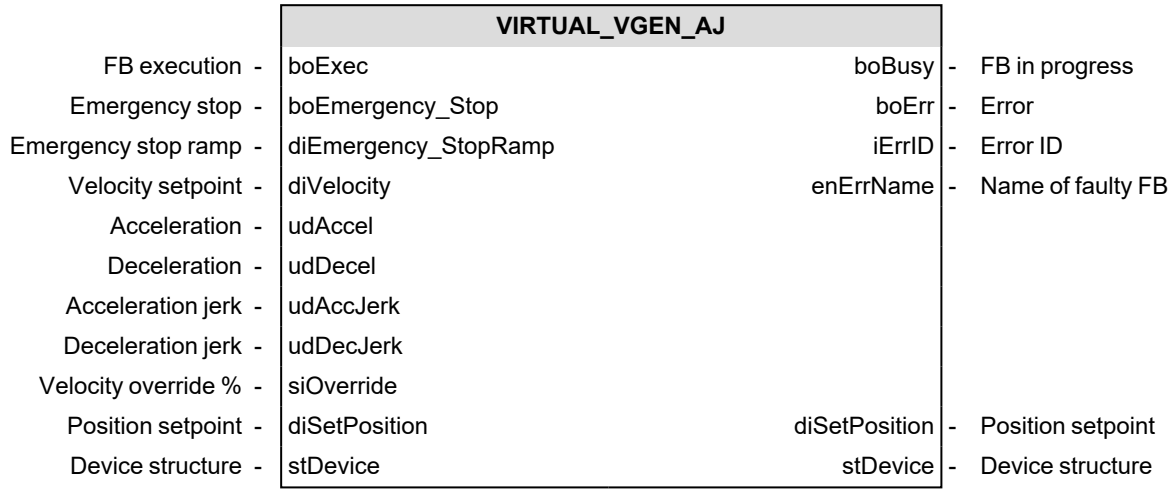
4.2.1.1.16.4 VIRTUAL_VGEN_AJ (FB)

The function block 'VIRTUAL_VGEN_AJ' realises a virtual velocity generator with the specification

- Of an acceleration ramp with jerk limitation
- Of a deceleration with jerk limitation
- Of an emergency-stop ramp.

The function block is called in the synchronous program level FPLC_PRG.

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.
boEmergency_Stop	BOOL	EMERGENCY STOP: The setpoint of the velocity is decreased to zero along the emergency-stop ramp. Once initiated, an emergency stop cannot be aborted.
diEmergency_StopRamp	DINT	EMERGENCY-STOP ramp: Ramp time in which the drive is slowed down from the current velocity to zero [ms].
diVelocity	DINT	Velocity setpoint [increments/s]
udAccel	UDINT	Acceleration with which the target velocity is run
udDecel	UDINT	Deceleration with which a lower target velocity is achieved
udAccJerk / udDecJerk	UDINT	Jerk for the acceleration/deceleration [increments/s^3]
siOverride	SINT	Velocity output factor

Output variables

Name	Type	Description
boBusy	BOOL	Execution message: This bit remains set as long as the block is being processed.
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

Name	Type	Description
enErrName	ENUM	EN_FB_NAME Name of faulty block for which the diagnostic number is output. The diagnostic number is explained in the description of the corresponding library. **Block** / **Library** VGEN_AJ / AmkBase.lib

Input and output variables

Name	Type	Description
diSetPosition	DINT	Specification of the position setpoint (position setpoint system) [increments]
stDevice	STRUCT	ST_DEVICE The device description structure assigns the block a device. (See document Software description AmkBase Bibliothek, Part no.204986)

Usage note in the CoDeSys program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
	VIRTUAL_VGEN_AJ

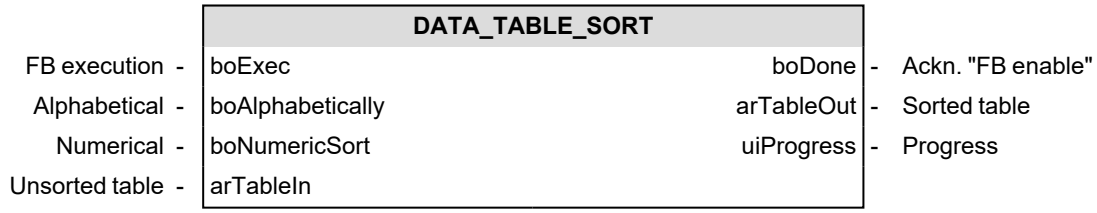
Associated with this function block, a visualisation is prepared in CoDeSys.
Siehe 'Visualization of AFL blocks' auf Seite 806.

4.2.1.1.17 21_Data

4.2.1.1.17.1 DATA_TABLE_SORT (FB)

The function block 'DATA_TABLE_SORT' sorts the transferred data field alphabetically or numerically.
The function block is called in the asynchronous program level PLC_PRG.

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.
boAlphabetically	BOOL	Alphabetical sorting
boNumericSort	BOOL	Numerical sorting
arTableIn	ARRAY	ARRAY [1..g_cuiDataTableLength] OF DATA_TABLE Table that is to be sorted

Output variables

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

Name	Type	Description
arTableOut	ARRAY	ARRAY [1..g_cuiDataTableLength] OF DATA_TABLE Sorted table
uiProgress	UINT	Progress: Shows the progress as per cent

Usage note in the CoDeSys program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
DATA_TABLE_SORT	

Associated with this function block, a visualisation is prepared in CoDeSys.
Siehe 'Visualization of AFL blocks' auf Seite 806.

4.2.1.2 06_TechnologyLevel

4.2.1.2.1 01_Emergency stop

4.2.1.2.1.1 EMERGENCY_BREAK_POSITION (FB)

In case of an emergency stop, the function block 'EMERGENCY_BREAK_POSITION' decouples the position specification and slows it down to a standstill in the time specified at the input 'diTRamp'.

The function block is called in the synchronous program level FPLC_PRG.

User interface

		EMERGENCY_BREAK_POSITION			
FB enable	-	boEnable	boEnabAck	-	Ackn. "FB enable"
Emergency stop	-	boEmergency_Stop	boBusy	-	FB in progress
Input value	-	diInVal	boDone	-	Ackn. "FB done"
Ramp time	-	diTRamp	diOutVal	-	Output value
SERCOS cycle time	-	diID2_SERCOS_cycle			

Input variables of the synchronous program levels (FPLC_PRG)

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.
boEmergency_Stop	BOOL	EMERGENCY STOP: The setpoint of the velocity is decreased to zero along the emergency-stop ramp. Once initiated, an emergency stop cannot be aborted.
diInVal	DINT	Input value [increments]
diTRamp	DINT	Ramp time [ms]
diID2_SERCOS_cycle	DINT	ID2 'SERCOS cycle time' [µs]

Output variables of the synchronous program levels (FPLC_PRG)

Name	Type	Description
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled
boBusy	BOOL	Execution message: This bit remains set as long as the block is being processed.

Name	Type	Description
boDone	BOOL	Response that the function block has been completely executed.
diOutVal	DINT	Output value [increments]

Usage note in the CoDeSys program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
	EMERGENCY_BREAK_POSITION

Associated with this function block, a visualisation is prepared in CoDeSys.
Siehe 'Visualization of AFL blocks' auf Seite 806.

4.2.1.2.1.2 EMERGENCY_BREAK_VELOCITY (FB)

In case of an emergency stop, the function block 'EMERGENCY_BREAK_VELOCITY' decouples the velocity setpoint and slows it down to a zero in the time specified at the input 'diTRamp'.

The function block is called in the synchronous program level FPLC_PRG.

User interface

	EMERGENCY_BREAK_VELOCITY			
FB enable -	boEnable	boEnabAck	-	Ackn. "FB enable"
Emergency stop -	boEmergency_Stop	boBusy	-	FB in progress
Input value -	diInVal	boDone	-	Ackn. "FB done"
Ramp time -	diTRamp	diOutVal	-	Output value
SERCOS cycle time -	diID2_SERCOS_cycle			

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.
boEmergency_Stop	BOOL	EMERGENCY STOP: The setpoint of the velocity is decreased to zero along the emergency-stop ramp. Once initiated, an emergency stop cannot be aborted.
diInVal	DINT	Input value [increments]
diTRamp	DINT	Ramp time [ms]
diID2_SERCOS_cycle	DINT	ID2 'SERCOS cycle time' [µs]

Output variables

Name	Type	Description
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled
boBusy	BOOL	Execution message: This bit remains set as long as the block is being processed.
boDone	BOOL	Response that the function block has been completely executed.
diOutVal	DINT	Output value [increments]

Usage note in the CoDeSys program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
	EMERGENCY_BREAK_VELOCITY

Associated with this function block, a visualisation is prepared in CoDeSys.
Siehe 'Visualization of AFL blocks' auf Seite 806.

4.2.1.2.2 02_Print marks

4.2.1.2.2.1 PRINT_MARK_CONTROL (FB)

The function block 'PRINT_MARK_CONTROL' is a basic block for realising a print mark control. It is not to be used directly but is a subroutine of the following function blocks:

- PRINT_MARK_INSETTER_CONT
- PRINT_MARK_INSETTER_INTERVAL
- PRINT_MARK_REGCONT_CONTINUOUS
- PRINT_MARK_REGISTER_CONTROL_DISCONT

The following functionality is covered by this function block:

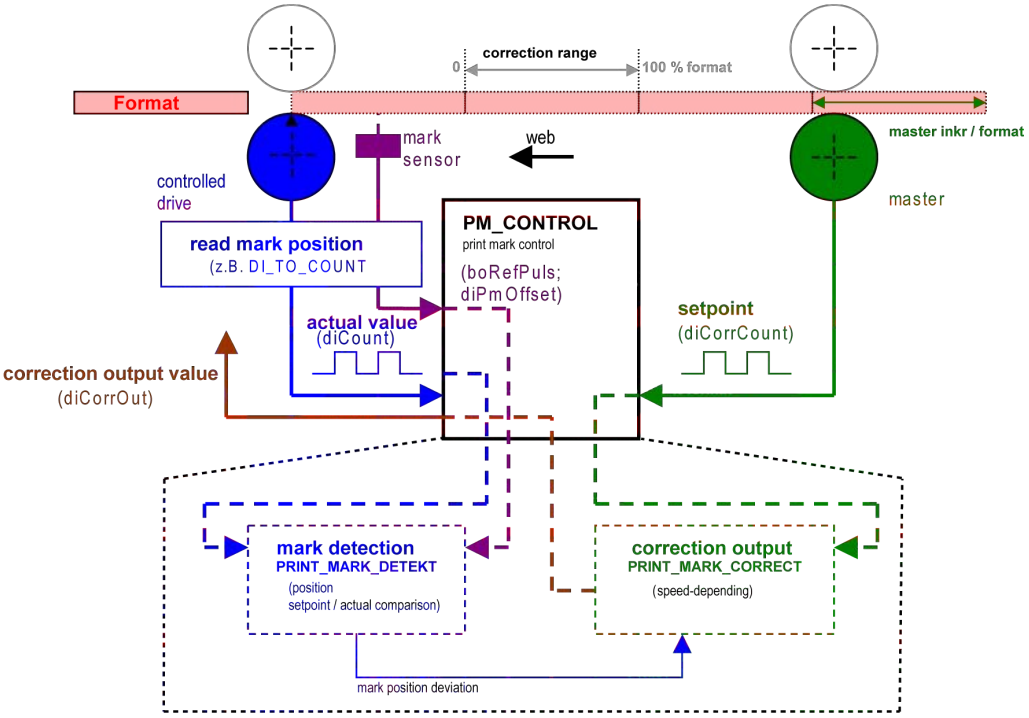
- The difference between setpoint and actual position is calculated and output as a correction value
- The function block implements two kinds of collecting the setpoint position of the print mark:
  - The setpoint position can be set to an absolute position within the print format.
  - The first detected print mark sets the setpoint position
- Control of the print mark sensor depending on a window
- Monitoring of the print marks
- Simulation of missing print marks
- Output of the correction path as even distribution of the increments on a preset correction path
- Limitation of the correction velocity
- Output an offset for shifting the print mark setpoint position

The function block operates with positive counting direction of the input pulses and sets positive output pulses.

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

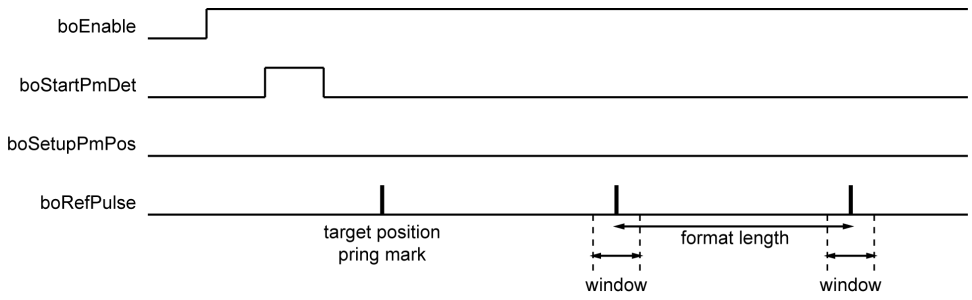
Setpoints and current values are transferred in a synchronous action. The synchronous action must be called in the synchronous program level FPLC_PRG.

Schematic diagram: PRINT_MARK_CONTROL used as register controller

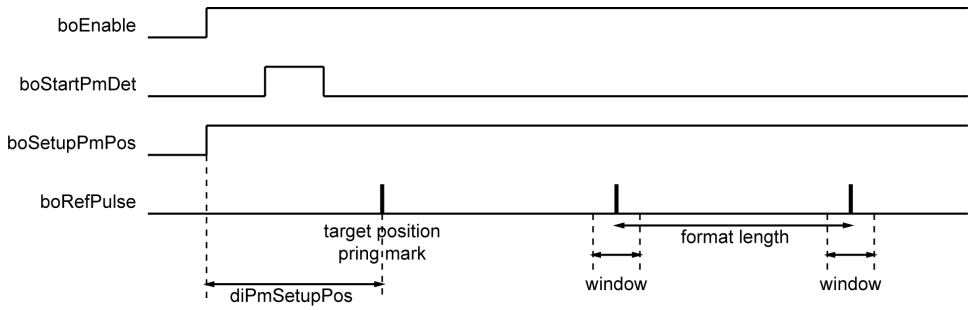


Specification of the print mark setpoint position

- **Start of the print mark control with search for the setpoint postion:**
  After a positive edge of 'boStartPmDet', the first pulse is set as setpoint position of the print mark. All following print marks are related to this position.



- **Start of the print mark control with preset of the setpoint position**:
  On a positive edge of 'boStartPmDet', the value of 'diPmSetupPos' is added to the actual master position. This value specifies the setpoint position of the next print mark. All following print marks are related to this position.



Print mark simulation

If no print mark pulse is detected up to the end of the print mark window, one print mark will be simulated at the setpoint position. The output 'iActPmLost' will be increased by 1.
'iNoPmLost' specifies how many print marks may be simulated until 'boPmLost' = TRUE. The output 'iActPmLost' shows how many print marks are already simulated since the last print mark has been detected.
The output 'boPmLost' has no effect on the control process.

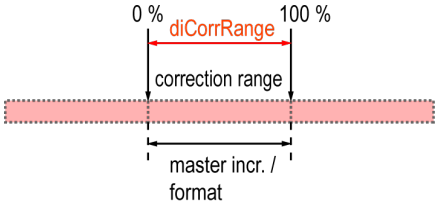
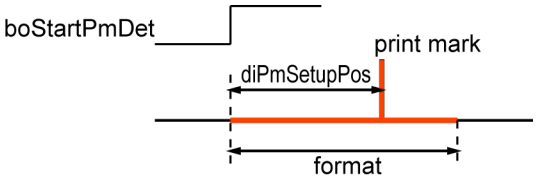
User interface

		PRINT_MARK_CONTROL			
FB enable	-	boEnable	boEnabAck	-	Ackn. "FB enable"
Format length	-	diFormat			
Window length	-	diWndLen			
Number of marks until message 'mark lost'	-	iNoPmLost			
Correction range	-	diCorrRange			
Print mark position setpoint	-	diPmSetupPos			
Max. correction per format	-	diCorrectionLimit			
Modus insetter / register controller	-	enPmcMode			

		actSync			
Reset of output *boPmLost*	-	boResetPmLost	boPmDetected	-	PM detected
Start mark detection	-	boStartPmDet	boPmLost	-	PM lost
Enable print mark correction	-	boCorrEnable	boPmWnd	-	PM window active
Start output operator offset	-	boSetupOffset	diPmDeviation	-	Deviation betw. setpoint and actual position
Start with PM setpoint position	-	boSetupPmPos	diOpOffsetOut	-	Output operator offset
PM detected	-	boRefPulse	diCorrOut	-	Output value
Impulse of the controlled aggregate	-	diCount	iActPmLost	-	Number of lost marks
PM offset	-	diPmOffset			
Operator offset	-	diOpOffset			
Impulse of the speed master	-	diCorrCount		-	

Input variables of the asynchronous part of the program (PLC_PRG)

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.
diFormat	DINT	Format length Nominal distance between two print marks [increments]
diWndLen	DINT	Window length Length in front and after the nominal print mark position. In this range, the print mark sensor is activated and will accept a print mark. 

Name	Type	Description
iNoPmLost	INT	Number of print marks to be missed in succession, before the output 'boPmLost' is set.
diCorrRange	DINT	Correction range The print mark control tries to output the correction in the specified range [% diFormatLength] 
diPmSetupPos	DINT	The parameter specifies the nominal position of the print mark relatively to the actual master position on a positive edge of 'boStartPmDet'. The detected print marks are controlled to this position [increments] 
diCorrectionLimit	DINT	Maximum correction output per format [increments] 'diCorrectionLimit' = 0: no limitation 'diCorrectionLimit' > 0: maximal value
enPmcMode	ENUM	EN_PMC_MODE Mode of print mark control: _enPmcInsetter: control of an insetter _enPmcRegisterController: control of a register controller

Input variables of the synchronous part of the program (FPLC_PRG)

Name	Type	Description
boResetPmLost	BOOL	Resetting the output 'boPmLost'
boStartPmDet	BOOL	Start of print mark control 'boStartPmDet' = TRUE: print mark control activated By means of the input 'boSetupPmPos', it is specified which kind of print mark position will be taken into account when starting the print mark control
boCorrEnable	BOOL	Release of the correction output (Can be controlled with block 'PRINT_MARK_CORROUT_CONTROL' to allow correction of outputs in certain areas only)
boSetupOffset	BOOL	Output start of the specified value of 'diOpOffset'. This value is used as an offset to shift the printing mark set position
boSetupPmPos	BOOL	Mode how to determine the nominal position of the print mark on a positive edge of 'boStartPmDet' 'boPmSetupPos' = FALSE: The position of the next print mark is set as nominal position. All following print marks are adjusted to this position. 'boPmSetupPos' = TRUE: The parameter 'diPmSetupPos' specifies the print mark relatively to the actual master position. The following print marks are adjusted to this position

Name	Type	Description
boRefPulse	BOOL	Homing pulse (See document Software description AmkBase Bibliothek, Part no. 204986) BasicSupport DI_TO_COUNT
diCount	DINT	Counter value Pulses of the controlled unit for determining the print mark deviation
diPmOffset	DINT	Print mark offset of the actual position (See document Software description AmkBase Bibliothek, Part no. 204986) BasicSupport DI_TO_COUNT
diOpOffset	DINT	Operator offset. Shifts the set position of the print mark. The shift takes place in the correction range and starts with a 0 → 1 edge at 'boSetupOffset'. It is independent of changes in the value of 'diCount'. The offset is calculated as
diCorrCount	DINT	Impulses of the speed masters (path speed master)

Output variables of the asynchronous part of the program (PLC_PRG)

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

Output variables of the synchronous part of the program (FPLC_PRG)

Name	Type	Description
boPmDetected	BOOL	Print mark detected, control activated
boPmLost	BOOL	Print mark lost activated when the number of print marks set in 'iNoPmLost' is not detected successively.
boPmWnd	BOOL	Mark window activates the print mark sensor
diPmDeviation	DINT	Deviation between setpoint and actual position of the actual print mark
diOpOffsetOut	DINT	Output value contains the actually output operator offset
diCorrOut	DINT	Correction value
iActPmLost	INT	Number of lost print marks

Usage note in the CoDeSys program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
PRINT_MARK_CONTROL	PRINT_MARK_CONTROL.actSync

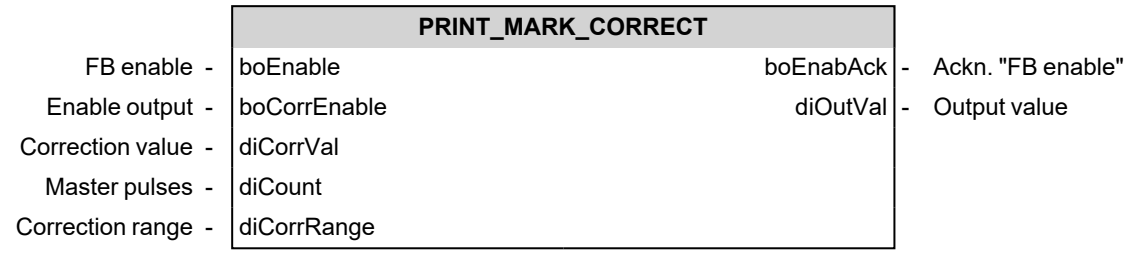
Associated with this function block, a visualisation is prepared in CoDeSys.
Siehe 'Visualization of AFL blocks' auf Seite 806.

4.2.1.2.2.2 PRINT_MARK_CORRECT (FB)

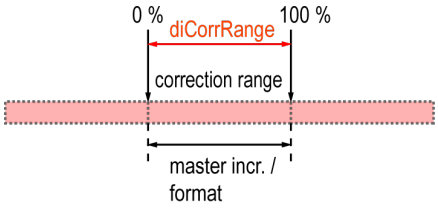
The function block 'PRINT_MARK_CORRECT' outputs a path depending on the velocity of a master.
The correction velocity depends on the correction value and the range in which the correction has to be done.
The output value can be sent directly to the controlled drive.

The function block is called in the synchronous program level FPLC_PRG.

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.
boCorrEnable	BOOL	Enable the output; by means of this input, the output is controlled. 'boCorrEnable' = TRUE, the output at 'diOutVal' is enabled. 'boCorrEnable' = FALSE the output at 'diOutVal' is disabled So it is possible to output the correction only if useful, e.g. in a special range.
diCorrVal	DINT	Length of the path (correction value) [increments]
diCount	DINT	Master pulses [increments]
diCorrRange	DINT	Correction range The print mark control tries to output the correction in the specified range [% diFormatLength]  Correction range, should be less than one format length [increments]

Output variables

Name	Type	Description
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled
diOutVal	DINT	Output value [increments] The former correction value at the input 'diCorrVal' will be charged against the actual correction value. Example: Former value = 500 increments, actual value = 700 increments => at the output 'diOutVal', 200 increments will be output, depending on the master velocity ('diCount') and the correction range ('diCorrRange').

Usage note in the CoDeSys program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
	PRINT_MARK_CORRECT

Associated with this function block, a visualisation is prepared in CoDeSys.
Siehe 'Visualization of AFL blocks' auf Seite 806.

4.2.1.2.2.3 PRINT_MARK_CORROUT_CONTROL (FB)

The function block 'PRINT_MARK_CORROUT_CONTROL' implements the following functionality:

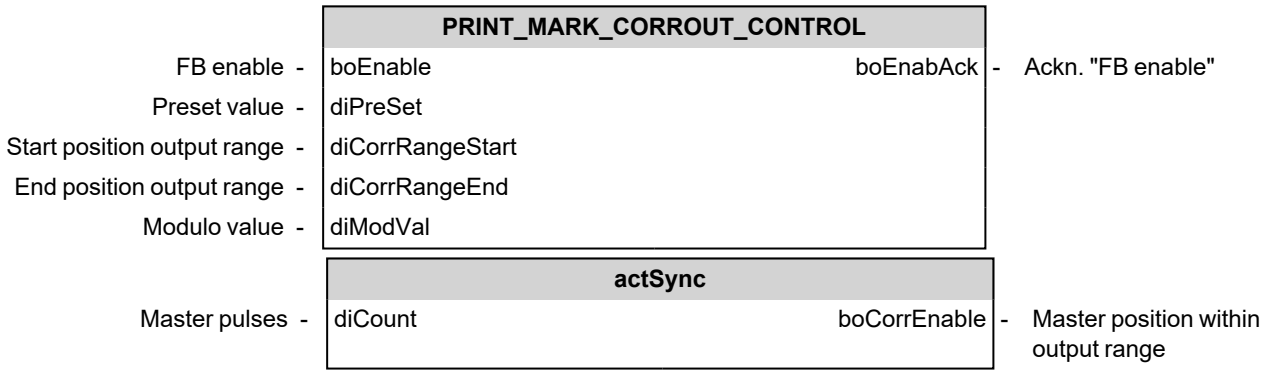
- Monitoring whether the input signal is in the specified range
  This range is between 0 and a modulo value
- The output range is specified by upper and lower limit
- The function block can be set to a defined start position by a preset value
- The input of the function block is decoupled by a differential stage

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

Setpoints and current values are transferred in a synchronous action. The synchronous action must be called in the synchronous program level FPLC_PRG.

If the function block is used as a print mark control, it must be completely implemented in the synchronous program level.

User interface



Input variables of the asynchronous part of the program (PLC_PRG)

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.
diPreSet	DINT	Preset value to which the output is set. The value is applied with the positive edge of 'boEnable'. [Increments]
diCorrRangeStart	DINT	Start position of the correction range [increments] (value can be changed with actParameterSetup at 'boEnable' = TRUE)
diCorrRangeEnd	DINT	End position of the correction range [increments] (value can be changed with actParameterSetup at 'boEnable' = TRUE)
diModVal	DINT	Modulo value (format length) [increments]

Input variables of the synchronous part of the program (FPLC_PRG)

Name	Type	Description
diCount	DINT	Master pulses [increments]

Output variables of the asynchronous part of the program (PLC_PRG)

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

Output variables of the synchronous part of the program (FPLC_PRG)

Name	Type	Description
boCorrEnable	BOOL	Master position within output range

Usage notes in the CoDeSys program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
PRINT_MARK_CORROUT_CONTROL	PRINT_MARK_CORROUT_CONTROL.actSync

Associated with this function block, a visualisation is prepared in CoDeSys.
Siehe 'Visualization of AFL blocks' auf Seite 806.

4.2.1.2.2.4 PRINT_MARK_DETECT (FB)

The function block 'PRINT_MARK_DETECT' implements the following functionality:

- Detects and measures the print marks, calculates the deviation of setpoint and actual position of the mark
- Controls the print mark sensor depending on a window around the print mark
- Monitoring of missing print marks
- Simulation of missing print marks
- Limits the correction output per format
- The difference between actual and setpoint print mark position is provided at the output. It can be used for readjusment. The differences are added
- An offset is processed to shift the print mark setpoint position

The function block operates with positive counting direction of the input pulses and sets positive output pulses.

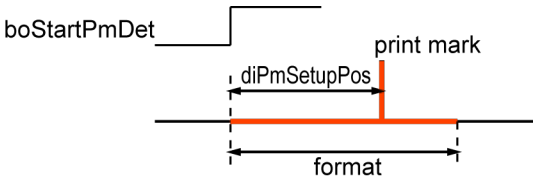
If the function block is used as a print mark control, it must be implemented in the synchronous program level.

User interface

		PRINT_MARK_DETECT			
FB enable	-	boEnable	boEnabAck	-	Ackn. "FB enable"
Start	-	boStartPmDet	boPmDetected	-	Print mark detected
Setup print mark position	-	boSetupPmPos	boPmLost	-	Print mark lost
Reset output	-	boResetPmLost	boPmWnd	-	Print mark window active
Reference pulse	-	boRefPulse	diPmDeviation	-	Deviation setpoint - actual
Master pulses	-	diCount	diCorrVal	-	Correction value
Print mark offset	-	diPmOffset	iPmLostCnt	-	Number of lost printmarks
Operator offset	-	diOpOffset			
Setpoint format	-	diFormat			
Window length	-	diWndLen			
Number of maximal lost print marks	-	iNoPmLost			
Relative setpoint position	-	diPmSetupPos			
Maximum correction value	-	diCorrectionLimit			
Control mode	-	enPmcMode			

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.
boStartPmDet	BOOL	Start of the print mark control and specification of the print mark setpoint position by parameter 'diPmSetupPos' if 'boSetupPmPos' = TRUE
boSetupPmPos	BOOL	Mode how to determine the nominal position of the print mark on a positive edge of 'boStartPmDet' 'boPmSetupPos' = FALSE: The position of the next print mark is set as nominal position. All following print marks are adjusted to this position. 'boPmSetupPos' = TRUE: The parameter 'diPmSetupPos' specifies the print mark relatively to the actual master position. The following print marks are adjusted to this position
boResetPmLost	BOOL	Resetting the output 'boPmLost'
boRefPulse	BOOL	Homing pulse recognised (See document Software description AmkBase Bibliothek, Part no. 204986) BasicSupport DI_TO_COUNT
diCount	DINT	Master pulses [increments] (See document Software description AmkBase Bibliothek, Part no. 204986) BasicSupport DI_TO_COUNT
diPmOffset	DINT	Print mark offset of the actual position (See document Software description AmkBase Bibliothek, Part no. 204986) BasicSupport DI_TO_COUNT
diOpOffset	DINT	Operator offset. Shifts the set position of the print mark. The shift takes place in the correction range and starts with a 0 → 1 edge at 'boSetupOffset'. It is independent of changes in the value of 'diCount'. The offset is calculated as
diFormat	DINT	Format length Nominal distance between two print marks [increments]
diWndLen	DINT	Window length Length in front and after the nominal print mark position. In this range, the print mark sensor is activated and will accept a print mark. 
iNoPmLost	INT	Number of print marks to be missed in succession, before the output 'boPmLost' is set.

Name	Type	Description
diPmSetupPos	DINT	The parameter specifies the nominal position of the print mark relatively to the actual master position on a positive edge of 'boStartPmDet'. The detected print marks are controlled to this position [increments] 
diCorrectionLimit	DINT	Maximum correction output per format [increments] 'diCorrectionLimit' = 0: no limitation 'diCorrectionLimit' > 0: maximal value
enPmcMode	ENUM	EN_PMC_MODE Mode of print mark control: Insetting controller: enPmcInsetter Register Controller: enPmcRegisterController

Output variables

Name	Type	Description
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled
boPmDetected	BOOL	Print mark detected, control activated
boPmLost	BOOL	Print mark lost activated when the number of print marks set in 'iNoPmLost' is not detected successively.
boPmWnd	BOOL	Mark window activates the print mark sensor
diPmDeviation	DINT	Deviation between setpoint and actual position of the actual print mark
diCorrVal	DINT	Correction value Sum of all former deviations
iPmLostCnt	INT	Counter of lost print marks

Usage note in the CoDeSys program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
	PRINT_MARK_DETECT

Associated with this function block, a visualisation is prepared in CoDeSys.
Siehe 'Visualization of AFL blocks' auf Seite 806.

4.2.1.2.2.5 PRINT_MARK_INSETTER_CONT (FB)

The function block 'PRINT_MARK_INSETTER_CONT' implements a print mark control. The position of the web to be treated is controlled in relation to a master format (insetting)

- The controlled web moves constantly
- A trend deviation between actual and setpoint print mark position will be controlled
- The function block implements two kinds of collecting the setpoint position of the print mark:
  - The setpoint position can be set to an absolute position within the print format.
  - The first detected print mark sets the setpoint position (Siehe 'PRINT_MARK_CONTROL (FB)' auf Seite 480.)
- Control of the print mark sensor depending on a window
- Monitoring of the print marks
- Simulation of missing print marks
- Output of the correction path as even distribution of the increments on a preset correction path
- Limitation of the correction velocity
- Output an offset for shifting the print mark setpoint position

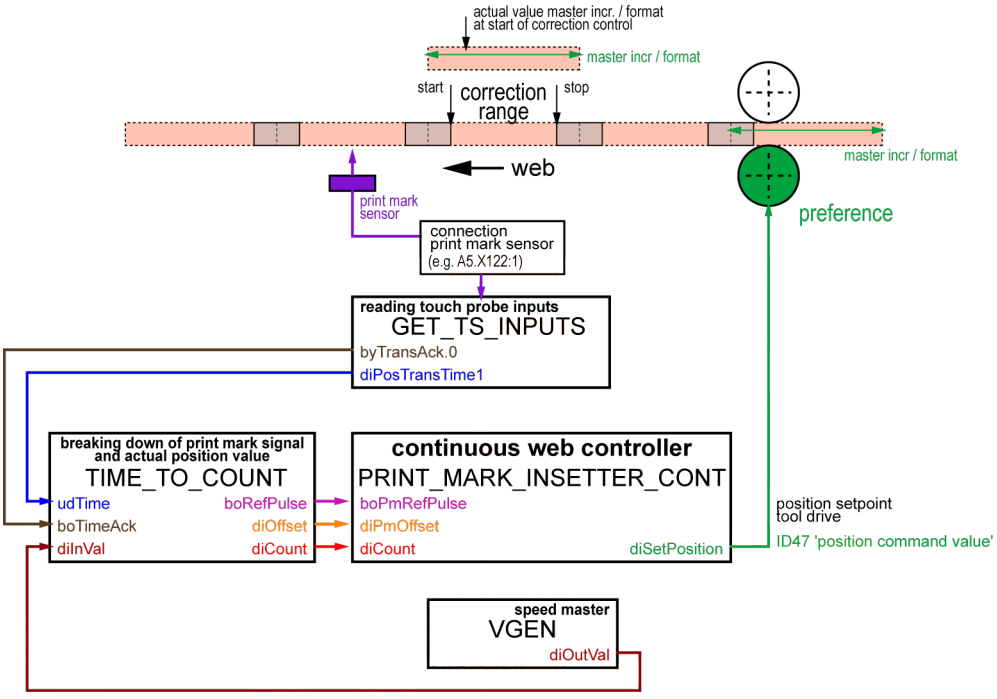
The function block operates with positive counting direction of the input pulses and sets positive output pulses.

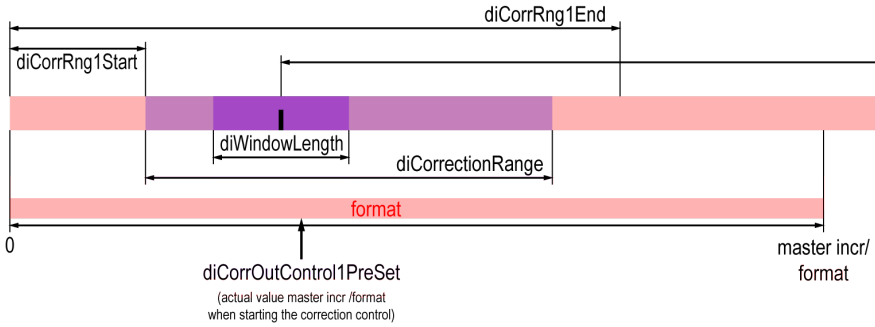
By means of 'boMasterNegDir' and 'boMotorNegDir', counting and motor direction can be inverted. There is no influence to the other output values.

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

Setpoints and current values are transferred in a synchronous action. The synchronous action must be called in the synchronous program level FPLC_PRG.

Schematic diagram





Example values

diFormatLength	10000 Incr
diWindowLength	2500 Incr
diCorrectionRange	70 %
diCorrRng1Start	1000 Incr
diCorrRng1End	9000 Incr
diCorrOutControl1PreSet	0 Incr (no shift)

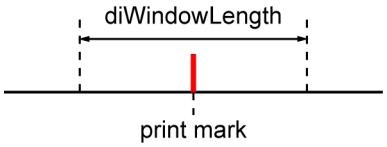
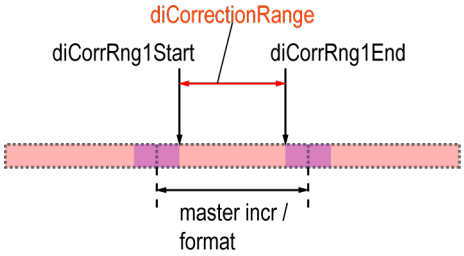
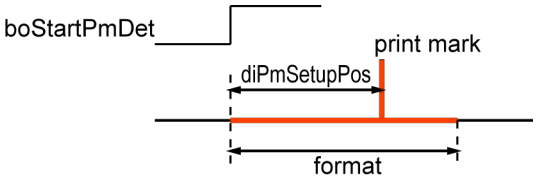
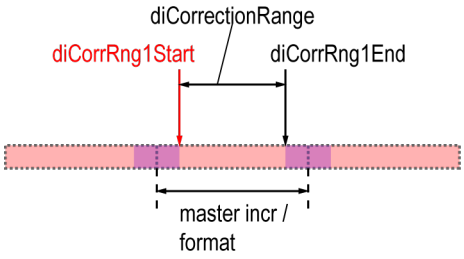
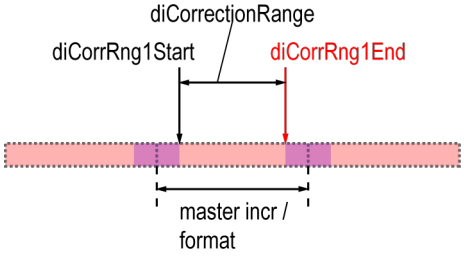
User interface

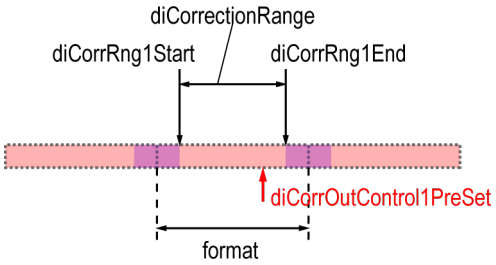
		PRINT_MARK_INSETTER_CONT			
FB enable	-	boEnable	boEnabAck	-	Ackn. "FB enable"
Reset output boPmLost	-	boResetPmLost	boParameterSetupDone	-	Parameterisation finished
Start output operator offset	-	boSetupOffset	boPmDetected	-	Print mark detected
Start with print mark setpoint position	-	boSetupPmPos	boPmLost	-	Print mark lost
Master counts negative	-	boMasterNegDir	boPmWnd	-	Print mark window active
Motor direction negative	-	boMotorNegDir	diPmDeviation	-	Deviation between setpoint and actual position
Enable correction output	-	boCorrOutControlOn	diOpOffsetOut	-	Output operator offset
Operator offset	-	diOpOffset	iActPmLost	-	Number of lost print marks
Format length	-	diFormatLength	boErr	-	Error
Window length	-	diWindowLength	iErrID	-	Error ID
Number of marks until message 'mark lost'	-	iNoPmLost			
Correction range	-	diCorrectionRange			
Print mark position setpoint	-	diPmSetupPos			
Max. correction per format	-	diCorrectionLimit			
Start correction range 1	-	diCorrRng1Start			
End correction range 1	-	diCorrRng1End			
Offset correction control 1	-	diCorrOutControl1PreSet			
Start correction range 2	-	diCorrRng2Start			
End correction range 2	-	diCorrRng2End			
Offset correction control 2	-	diCorrOutControl2PreSet			
Position setpoint	-	diSetPosition	diSetPosition	-	Position setpoint

		actSync			
Start print mark detection	-	boStartPmDet			
Print mark detected	-	boPmRefPulse			
Master pulses	-	diCount			
Print mark offset	-	diPmOffset			
Position setpoint	-	diSetPosition	diSetPosition	-	Position setpoint

Input variables of the asynchronous program level (PLC_PRG)

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.
boResetPmLost	BOOL	Resetting the output 'boPmLost'
boSetupOffset	BOOL	Output start of the specified value of 'diOpOffset'. This value is used as an offset to shift the printing mark set position
boSetupPmPos	BOOL	Mode how to determine the nominal position of the print mark on a positive edge of 'boStartPmDet' 'boPmSetupPos' = FALSE: The position of the next print mark is set as nominal position. All following print marks are adjusted to this position. 'boPmSetupPos' = TRUE: The parameter 'diPmSetupPos' specifies the print mark relatively to the actual master position. The following print marks are adjusted to this position
boMasterNegDir	BOOL	Direction of master pulse negated 'boMasterNegDir' = FALSE: no negation 'boMasterNegDir' = TRUE: negation  This input must only be set on a program start.
boMotorNegDir	BOOL	Direction of motor rotation negated 'boMotorNegDir' = FALSE: no negation 'boMotorNegDir' = TRUE: negation  This input must only be set on a program start.
boCorrOutControlOn	BOOL	Correction output enabled 'boCorrOutControlOn' = FALSE: always correction output 'boCorrOutControlOn' = TRUE: correction output only in the ranges specified by 'diCorrRng1Start' and 'diCorrRng1End' resp. 'diCorrRng2Start' and 'diCorrRng2End'. The ranges are OR operated  The modulo counting is started with a 0 -> 1 edge at 'boCorrOutControlOn' and can be preset by 'diCorrOutControl1PreSet' or 'diCorrOutControl2PreSet' Example: If the controlled drive is positioned in the middle of the format when starting the correction control, >format/2< must be set to 'diCorrOutControl1PreSet' on a 0 -> 1 edge of 'boCorrOutControlOn'
diOpOffset	DINT	Operator offset. Shifts the set position of the print mark. The shift takes place in the correction range and starts with a 0 → 1 edge at 'boSetupOffset'. It is independent of changes in the value of 'diCount'. The offset is calculated as
diFormatLength	DINT	Format length Nominal distance between two print marks [increments]

Name	Type	Description
diWindowLength	DINT	Window length Length in front and after the nominal print mark position. In this range, the print mark sensor is activated and will accept a print mark. 
iNoPmLost	INT	Number of print marks to be missed in succession, before the output 'boPmLost' is set.
diCorrectionRange	DINT	Correction range The print mark control tries to output the correction in the specified range [% diFormatLength] 
diPmSetupPos	DINT	The parameter specifies the nominal position of the print mark relatively to the actual master position on a positive edge of 'boStartPmDet'. The detected print marks are controlled to this position [increments] 
diCorrectionLimit	DINT	Maximum correction output per format [increments] 'diCorrectionLimit' = 0: no limitation 'diCorrectionLimit' > 0: maximal value
diCorrRng1Start diCorrRng2Start	DINT	Correction range 1 / 2 Start value related to master increments / format [increments] 
diCorrRng1End diCorrRng2End	DINT	Correction range 1 / 2 End value related to master increments / format [increments] 

Name	Type	Description
diCorrOutControl1PreSet diCorrOutControl2PreSet	DINT	Actual position value (zero offset) at start of the correction control, related to the master increments / format [increments] Example: If the controlled drive is positioned in the middle of the format when starting the correction control, >format/2< must be set on a 0 -> 1 edge of 'boCorrOutControlOn' 

Input variables of the synchronous program level (FPLC_PRG)

Name	Type	Description
boStartPmDet	BOOL	Start of the print mark control and specification of the print mark setpoint position by parameter 'diPmSetupPos' if 'boSetupPmPos' = TRUE
boPmRefPulse	BOOL	Print mark pulse detected (reference pulse) (See document Software description AmkBase Bibliothek, Part no. 204986) BasicSupport TIME_TO_COUNT
diCount	DINT	Master pulses [increments] (See documentSoftware descriptionAmkBase Bibliothek, Part no. 204986) BasicSupport TIME_TO_COUNT
diPmOffset	DINT	Print mark offset of the actual master position (See documentSoftware descriptionAmkBase Bibliothek, Part no. 204986) BasicSupport TIME_TO_COUNT

Output variables of the asynchronous program level (PLC_PRG)

Name	Type	Description
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled
boParameterSetupDone	BOOL	Parameterisation finished
boPmDetected	BOOL	Print mark detected, control activated
boPmLost	BOOL	Print mark lost activated when the number of print marks set in 'iNoPmLost' is not detected successively.
boPmWnd	BOOL	Mark window activates the print mark sensor
diPmDeviation	DINT	Deviation between setpoint and actual position of the actual print mark
diOpOffsetOut	DINT	Output value contains the actually output operator offset
iActPmLost	INT	Number of lost print marks
boErr	DINT	The function block is in an error state FALSE: No error (permitted commanding or warning) TRUE: Error iErrID = 0: No error iErrID ≠ 0, boErr = TRUE: Error iErrID ≠ 0, boErr = FALSE: Warning

Name	Type	Description
iErrID	INT	Error identity number: Diagnostic number is output 101 diFormatLength ≤ 0 104 diCorrectionRange ≤ 0

Input and output variables

Name	Type	Description
diSetPosition	DINT	Specification of the position setpoint (position setpoint system) [increments]

Usage note in the CoDeSys program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
PRINT_MARK_INSETTER_CONT	PRINT_MARK_INSETTER_CONT.actSync

Associated with this function block, a visualisation is prepared in CoDeSys.
Siehe 'Visualization of AFL blocks' auf Seite 806.

4.2.1.2.2.6 PRINT_MARK_INSETTER_INTERVAL (FB)

The function block 'PRINT_MARK_INSETTER_INTERVAL' implements a print mark control. The position of the web to be treated is controlled in relation to a master format (insetting)

- The controlled web moves incrementally
- The kind of movement can be selected:
  - 45° straight line
  - Sinus curve
  - Sinus^2 curve
- The length of the standstill range can be parameterised
- A trend deviation between actual and setpoint print mark position will be controlled
- The function block implements two kinds of collecting the setpoint position of the print mark:
  - The setpoint position can be set to an absolute position within the print format.
  - The first detected print mark sets the setpoint position (Siehe 'PRINT_MARK_CONTROL (FB)' auf Seite 480.)
- Control of the print mark sensor depending on a window
- Monitoring of the print marks
- Simulation of missing print marks
- Output of the correction path as even distribution of the increments on a preset correction path
- Limitation of the correction velocity
- Output an offset for shifting the print mark setpoint position

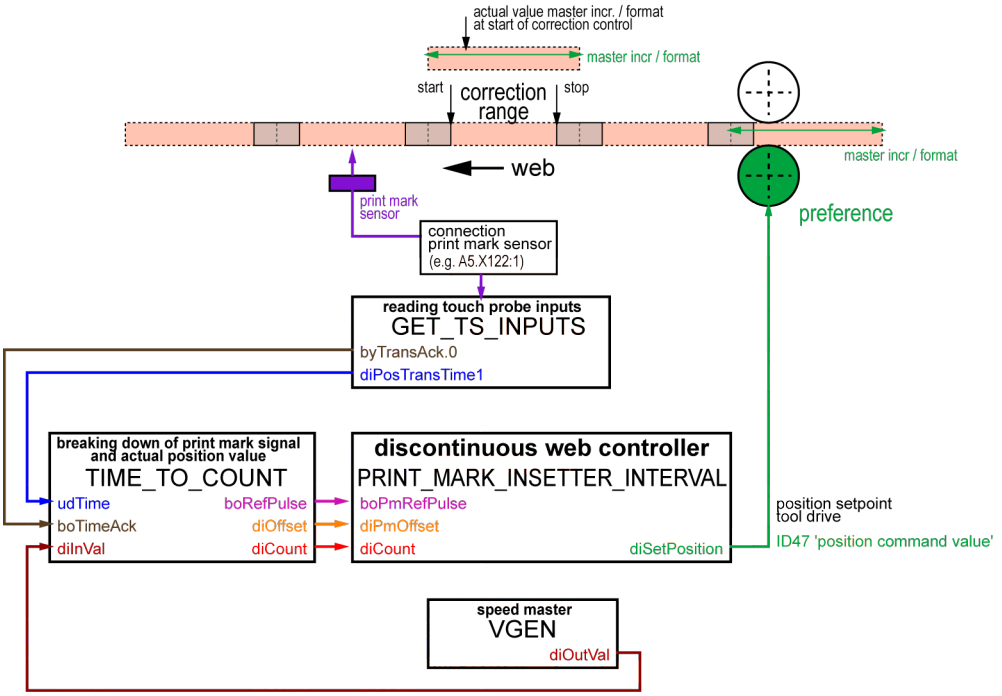
The function block operates with positive counting direction of the input pulses and sets positive output pulses.

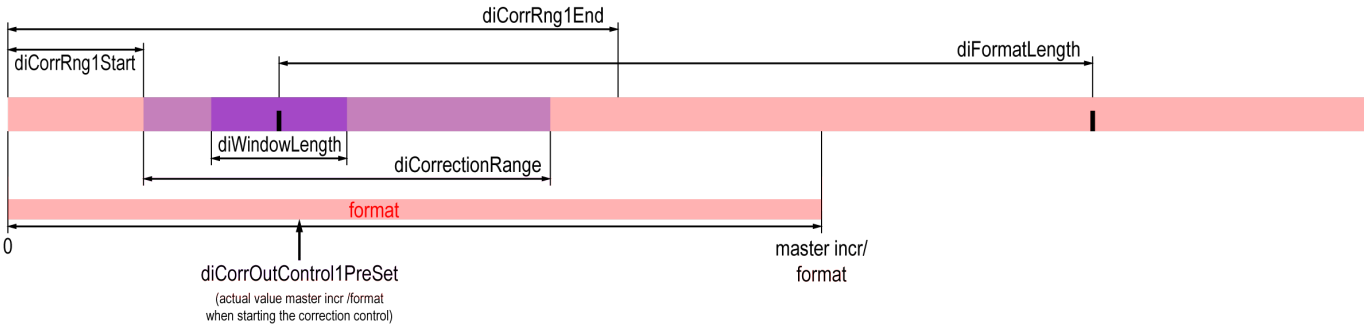
By means of 'boMasterNegDir' and 'boMotorNegDir', counting and motor direction can be inverted. There is no influence to the other output values.

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

Setpoints and current values are transferred in a synchronous action. The synchronous action must be called in the synchronous program level FPLC_PRG.

Schematic diagram

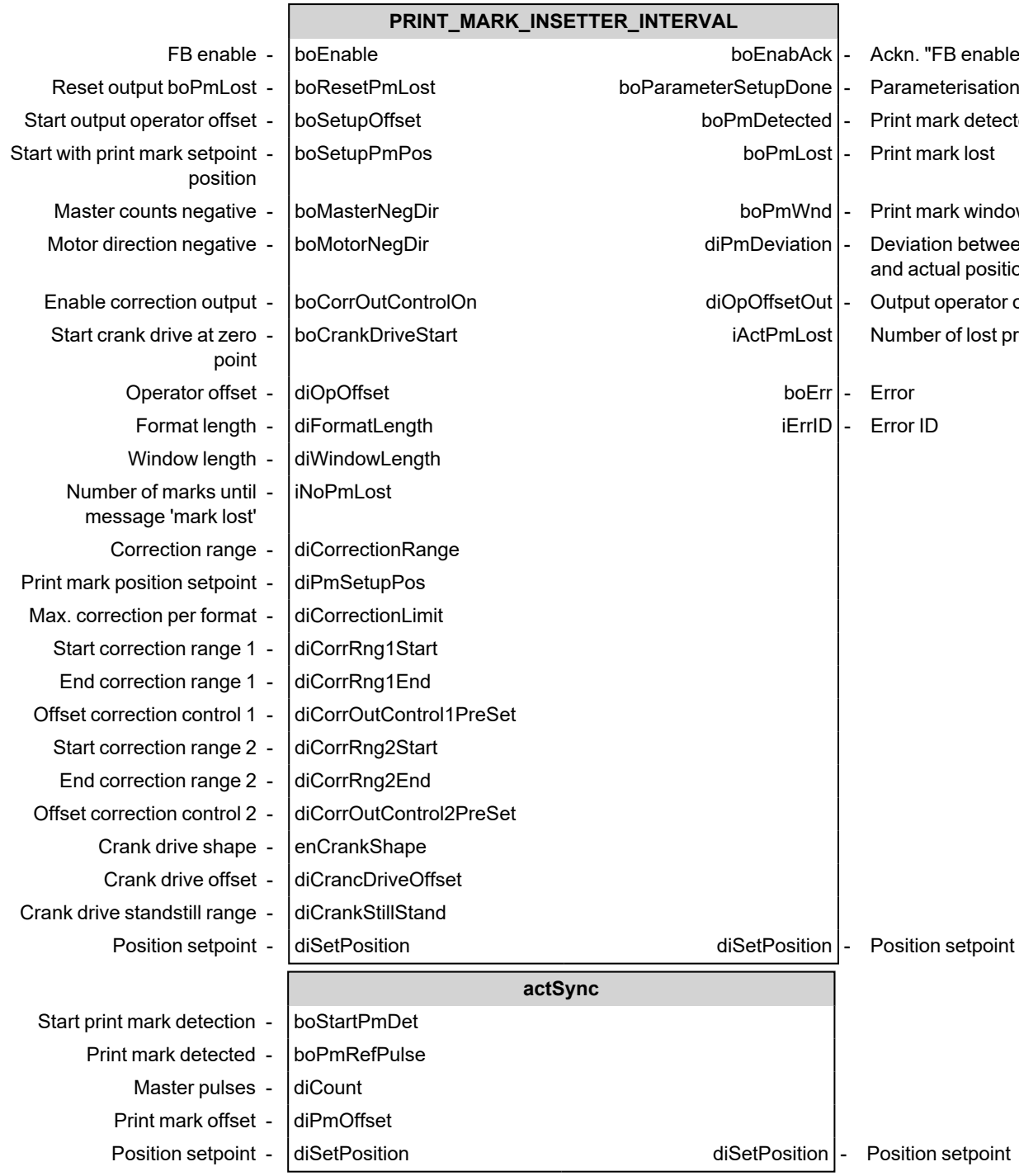




Example values

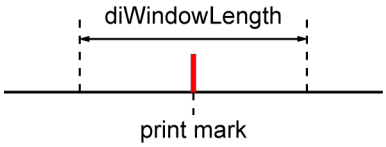
diFormatLength	10000 Incr
diWindowLength	2500 Incr
diCorrectionRange	70 %
diCorrRng1Start	1000 Incr
diCorrRng1End	9000 Incr
diCorrOutControl1PreSet	0 Incr (no shifting)

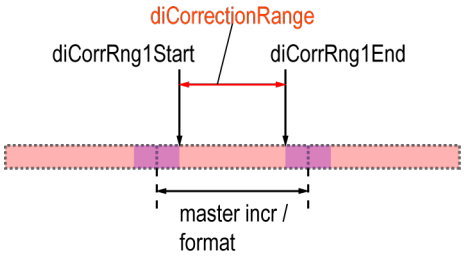
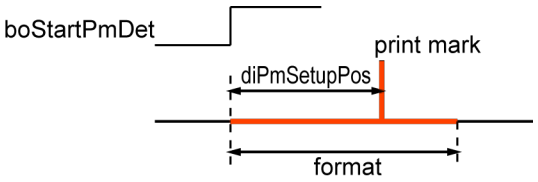
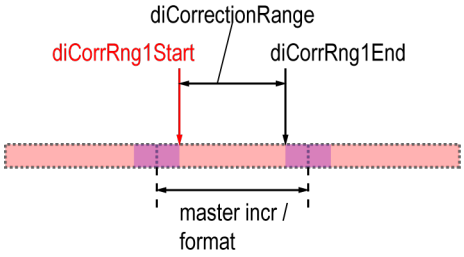
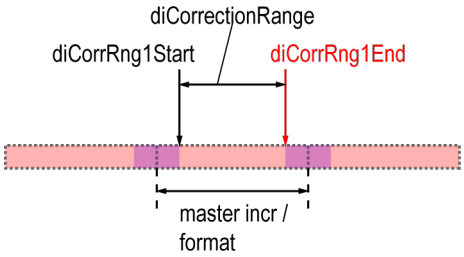
User interface

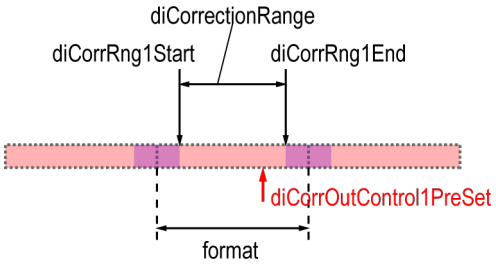
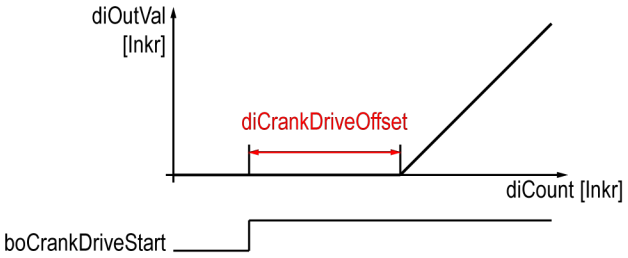
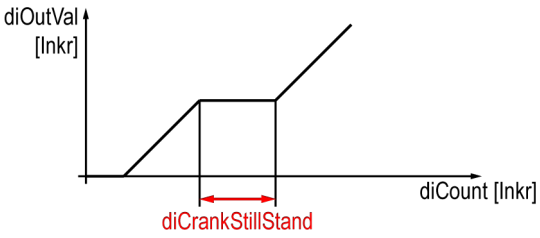


Input variables of the asynchronous program level (PLC_PRG)

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.
boResetPmLost	BOOL	Resetting the output 'boPmLost'
boSetupOffset	BOOL	Output start of the specified value of 'diOpOffset'. This value is used as an offset to shift the printing mark set position

Name	Type	Description
boSetupPmPos	BOOL	Mode how to determine the nominal position of the print mark on a positive edge of 'boStartPmDet' 'boPmSetupPos' = FALSE: The position of the next print mark is set as nominal position. All following print marks are adjusted to this position. 'boPmSetupPos' = TRUE: The parameter 'diPmSetupPos' specifies the print mark relatively to the actual master position. The following print marks are adjusted to this position
boMasterNegDir	BOOL	Direction of master pulse negated 'boMasterNegDir' = FALSE: no negation 'boMasterNegDir' = TRUE: negation This input must only be set on a program start.
boMotorNegDir	BOOL	Direction of motor rotation negated 'boMotorNegDir' = FALSE: no negation 'boMotorNegDir' = TRUE: negation This input must only be set on a program start.
boCorrOutControlOn	BOOL	Correction output enabled 'boCorrOutControlOn' = FALSE: always correction output 'boCorrOutControlOn' = TRUE: correction output only in the ranges specified by 'diCorrRng1Start' and 'diCorrRng1End' resp. 'diCorrRng2Start' and 'diCorrRng2End'. The ranges are OR operated The modulo counting is started with a 0 -> 1 edge at 'boCorrOutControlOn' and can be preset by 'diCorrOutControl1PreSet' or 'diCorrOutControl2PreSet' Example: If the controlled drive is positioned in the middle of the format when starting the correction control, >format/2< must be set to 'diCorrOutControl1PreSet' on a 0 -> 1 edge of 'boCorrOutControlOn'
diOpOffset	DINT	Operator offset. Shifts the set position of the print mark. The shift takes place in the correction range and starts with a 0 → 1 edge at 'boSetupOffset'. It is independent of changes in the value of 'diCount'. The offset is calculated as
diFormatLength	DINT	Format length Nominal distance between two print marks [increments]
diWindowLength	DINT	Window length Length in front and after the nominal print mark position. In this range, the print mark sensor is activated and will accept a print mark. 
iNoPmLost	INT	Number of print marks to be missed in succession, before the output 'boPmLost' is set.

Name	Type	Description
diCorrectionRange	DINT	Correction range The print mark control tries to output the correction in the specified range [% diFormatLength] 
diPmSetupPos	DINT	The parameter specifies the nominal position of the print mark relatively to the actual master position on a positive edge of 'boStartPmDet'. The detected print marks are controlled to this position [increments] 
diCorrectionLimit	DINT	Maximum correction output per format [increments] 'diCorrectionLimit' = 0: no limitation 'diCorrectionLimit' > 0: maximal value
diCorrRng1Start diCorrRng2Start	DINT	Correction range 1 / 2 Start value related to master increments / format [increments] 
diCorrRng1End diCorrRng2End	DINT	Correction range 1 / 2 End value related to master increments / format [increments] 

Name	Type	Description
diCorrOutControl1PreSet diCorrOutControl2PreSet	DINT	Actual position value (zero offset) at start of the correction control, related to the master increments / format [increments] Example: If the controlled drive is positioned in the middle of the format when starting the correction control, >format/2< must be set on a 0 -> 1 edge of 'boCorrOutControlOn' 
enCrankShape	ENUM	EN_CRANK_DRIVE_SHAPE Shape of crank drive curve enCrankShapeLinear: linear 45° straight line enCrankShapeSine: SINUS - function enCrankShapeSquareSine: SINUS^2 - function
diCrancDriveOffset	DINT	Crank drive start offset. After the crank drive is started (boCrankDriveStart = TRUE), the function waits until the master has passed the offset 
diCrankStillStand	DINT	Crank drive standstill range 

Input variables of the synchronous program level (FPLC_PRG)

Name	Type	Description
boStartPmDet	BOOL	Start of the print mark control and specification of the print mark setpoint position by parameter 'diPmSetupPos' if 'boSetupPmPos' = TRUE
boCrankDriveStart	BOOL	Start of the crank drive function, Output at 'diOutVal' starts according to the selected crank shape. The output is only done if 'diCount' changes
boPmRefPulse	BOOL	Print mark pulse detected (reference pulse) (See document Software description AmkBase Bibliothek, Part no. 204986) BasicSupport TIME_TO_COUNT

Name	Type	Description
diCount	DINT	Master pulses [increments] (See documentSoftware descriptionAmkBase Bibliothek, Part no. 204986) BasicSupport TIME_TO_COUNT
diPmOffset	DINT	Print mark offset of the actual master position (See documentSoftware descriptionAmkBase Bibliothek, Part no. 204986) BasicSupport TIME_TO_COUNT

Output variables of the asynchronous program level (PLC_PRG)

Name	Type	Description
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled
boParameterSetupDone	BOOL	Parameterisation finished
boPmDetected	BOOL	Print mark detected, control activated
boPmLost	BOOL	Print mark lost activated when the number of print marks set in 'iNoPmLost' is not detected successively.
boPmWnd	BOOL	Mark window activates the print mark sensor
diPmDeviation	DINT	Deviation between setpoint and actual position of the actual print mark
diOpOffsetOut	DINT	Output value contains the actually output operator offset
iActPmLost	INT	Number of lost print marks
boErr	BOOL	The function block is in an error state FALSE: No error (permitted commanding or warning) TRUE: Error iErrID = 0: No error iErrID ≠ 0, boErr = TRUE: Error iErrID ≠ 0, boErr = FALSE: Warning
iErrID	INT	Error identity number: Diagnostic number is output 1 ... 99 Error see block 'CAM_PROF' (AmkBase.lib) 100 Wrong mode in parameter 'enCrankShape' 101 diFormatLength ≤ 0 102 diCrankStillStand zu groß 103 diCrancDriveOffset < 0 104 diCorrectionRange ≤ 0

Input and output variables

Name	Type	Description
diSetPosition	DINT	Specification of the position setpoint (position setpoint system) [increments]

Usage note in the CoDeSys program

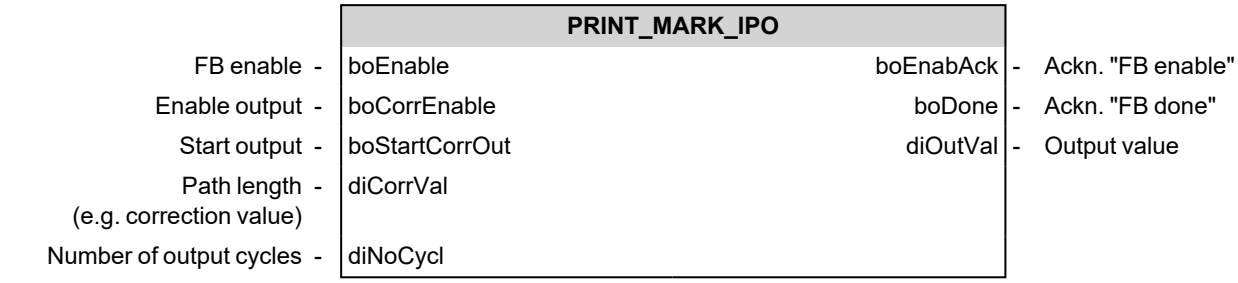
PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
PRINT_MARK_INSETTER_INTERVAL	PRINT_MARK_INSETTER_INTERVAL.actSync

Associated with this function block, a visualisation is prepared in CoDeSys.
Siehe 'Visualization of AFL blocks' auf Seite 806.

4.2.1.2.2.7 PRINT_MARK_IPO (FB)

The function block 'PRINT_MARK_IPO' implements the output of a path step-by-step. The step width per cycle can be preset.
The function block is called in the synchronous program level FPLC_PRG.

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.
boCorrEnable	BOOL	Enable the output; by means of this input, the output is controlled. 'boCorrEnable' = TRUE, the output at 'diOutVal' is enabled. 'boCorrEnable' = FALSE the output at 'diOutVal' is disabled So it is possible to output the correction only if useful, e.g. in a special range.
boStartCorrOut	BOOL	Start correction output
diCorrVal	DINT	Length of the path (correction value) [increments]
diNoCycl	DINT	Number of output cycles

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. complete output
diOutVal	DINT	Output value [increments]

Usage note in the CoDeSys program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
	PRINT_MARK_IPO

Associated with this function block, a visualisation is prepared in CoDeSys.
Siehe 'Visualization of AFL blocks' auf Seite 806.

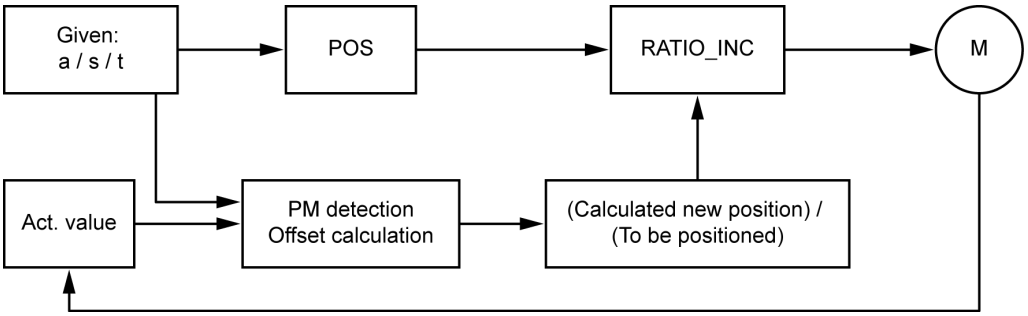
4.2.1.2.2.8 PRINT_MARK_POSITION_TO_THE_MARK (FB)

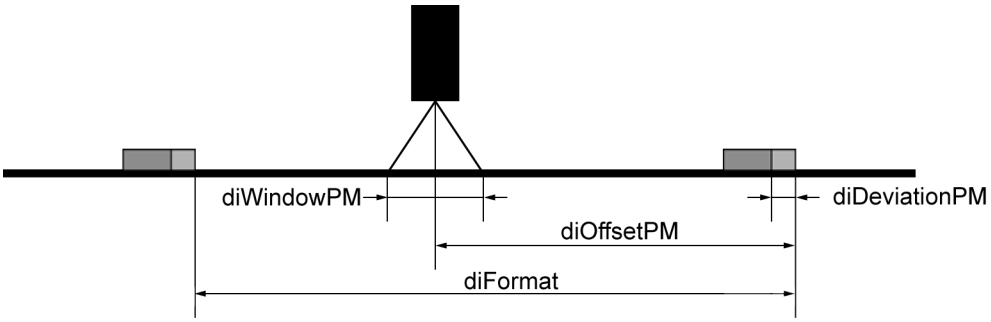
The function block 'PRINT_MARK_POSITION_TO_THE_MARK' positions an AMK axis relatively.

During positioning, a printing mark within the window is sought. If the printing mark is found, the positioning is oriented according to the mark.

The function block is called in the synchronous program level FPLC_PRG.

Scheme





- The function block 'RATIO_INC' streteches or compresses the actual positioning according to the PM position
- Thus, nothing changes for the function block POS.
- The positioning is completed in the specified time
- If the PM is missing, the normal format length is used

User interface

		PRINT_MARK_POSITION_TO_THE_MARK			
FB enable	-	boEnable	boDone	-	Ackn. "FB done"
Emergency stop	-	boEmergency_Stop	boBusy	-	FB in progress
Emergency stop ramp	-	diEmergency_StopRamp	boErr	-	Error
Execution started	-	boStart	iErrID	-	Error ID
Start homing cycle	-	boHome	enErrName	-	Name of faulty FB
Distance PM	-	diFormat	boEnabAck	-	Ackn. "FB enable"
Offset PM	-	diOffsetPM	boRefPulse	-	Homing pulse
Number of lost PMs	-	usiNoLostPM	diDeviationPM	-	PM offset
Velocity homing drive	-	diSpeedHome	usiNoPmLost	-	Number of lost PMs
PM window	-	diWindowPM			
Max. correction	-	diMaxCorrection			
Acceleration	-	udAccel			
Deceleration	-	udDecel			
Time for positioning	-	udTimeForPos			
Actual position value	-	diActPosition			
Cur. latch value	-	diLatchValue			
Device structure	-	stDevice	stDevice	-	Device structure
Position setpoint	-	diSetPosition	diSetPosition	-	Position setpoint

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.
boEmergency_Stop	BOOL	EMERGENCY STOP: The setpoint of the velocity is decreased to zero along the emergency-stop ramp. Once initiated, an emergency stop cannot be aborted.
diEmergency_StopRamp	DINT	EMERGENCY-STOP ramp: Ramp time in which the drive is slowed down from the current velocity to zero [ms].
boStart	BOOL	With a positive edge, the execution of the block starts.
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. Axis drive to PM + offset
diFormat	DINT	Format length Nominal distance between two print marks [increments]
diOffsetPM	DINT	Distance to PM (distance between PM sensor and stop position) Unit: Increments Default value: 10240
usiNoLostPM	USINT	Once the preset number of marks is lost, an error message is generated. Default 0 = Error message is never generated Unit: Increments Default value: 10
diSpeedHome	DINT	Velocity of the homing drive [increments/s]
diWindowPM	DINT	Window in which the marks must be located for detection (+/- target position) Unit: Increments Default value: 20480
diMaxCorrection	DINT	Max. correction value Default 0 = The calculated correction value is always computed without limit. Unit: Increments Default value: 20480
udAccel	UDINT	Acceleration with which the target velocity is run
udDecel	UDINT	Deceleration with which a lower target velocity is achieved
udTimeForPos	UDINT	Time in which the positioning must be completed. Unit: ms Default value: 2000
diActPosition	DINT	Actual position feedback value [increments] ID51 'Position feedback value'
diLatchValue	DINT	Current latch value [increments] ID130 'Probe value 1 positive edge'

Output variables

Name	Type	Description
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled
boBusy	BOOL	Execution message: This bit remains set as long as the block is being processed.
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 enErrName = 0: 0: No error 1: Zero entry, required values have "0" has transfer value 2: Ramp (motion profile) mathematically impossible (e.g. a,d too small for time) 3: Printing mark lost 4: diOffsetPM> diFormat 5: No PM found during homing, search aborted enErrName ≠ 0: Error number see block (enErrName)
enErrName	ENUM	EN_FB_NAME Name of faulty block for which the diagnostic number is output. The diagnostic number is explained in the description of the corresponding library. Block: RATIO_INC_1 — Library: AmkBase.lib Block: POS_1 — Library: AmkBase.lib
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled
boRefPulse	BOOL	For one cycle boRefPuls = TRUE when the latch value is changed.
diDeviationPM	DINT	Deviation between actual and target position of the last found PM. Unit: Increments
usiNoPmLost	USINT	Number of lost printing marks

Input and output variables

Name	Type	Description
diSetPosition	DINT	Specification of the position setpoint (position setpoint system) [increments]
stDevice	STRUCT	ST_DEVICE The device description structure assigns the block a device. (See document Software description AmkBase Bibliothek, Part no.204986)

Usage note in the CoDeSys program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
	PRINT_MARK_POSITION_TO_THE_MARK

Associated with this function block, a visualisation is prepared in CoDeSys.

Siehe 'Visualization of AFL blocks' auf Seite 806.

4.2.1.2.2.9 PRINT_MARK_REGISTER_CONTROL_CONTINUOUS (FB)

The function block 'PRINT_MARK_REGISTER_CONTROL_CONTINUOUS' implements a print mark control. The position of the treating tool (e.g. cutter, punch) is controlled in relation to a master format (register control)

- The controlled treating tool moves constantly
- Master format and circumference of the treating tool must be equal
- A trend deviation between actual and setpoint print mark position will be controlled
- The function block implements two kinds of collecting the setpoint position of the print mark:
  - The setpoint position can be set to an absolute position within the print format.
  - The first detected print mark sets the setpoint position (Siehe 'PRINT_MARK_CONTROL (FB)' auf Seite 480.)
- Control of the print mark sensor depending on a window
- Monitoring of the print marks
- Simulation of missing print marks
- Output of the correction path as even distribution of the increments on a preset correction path
- Limitation of the correction velocity
- Output an offset for shifting the print mark setpoint position

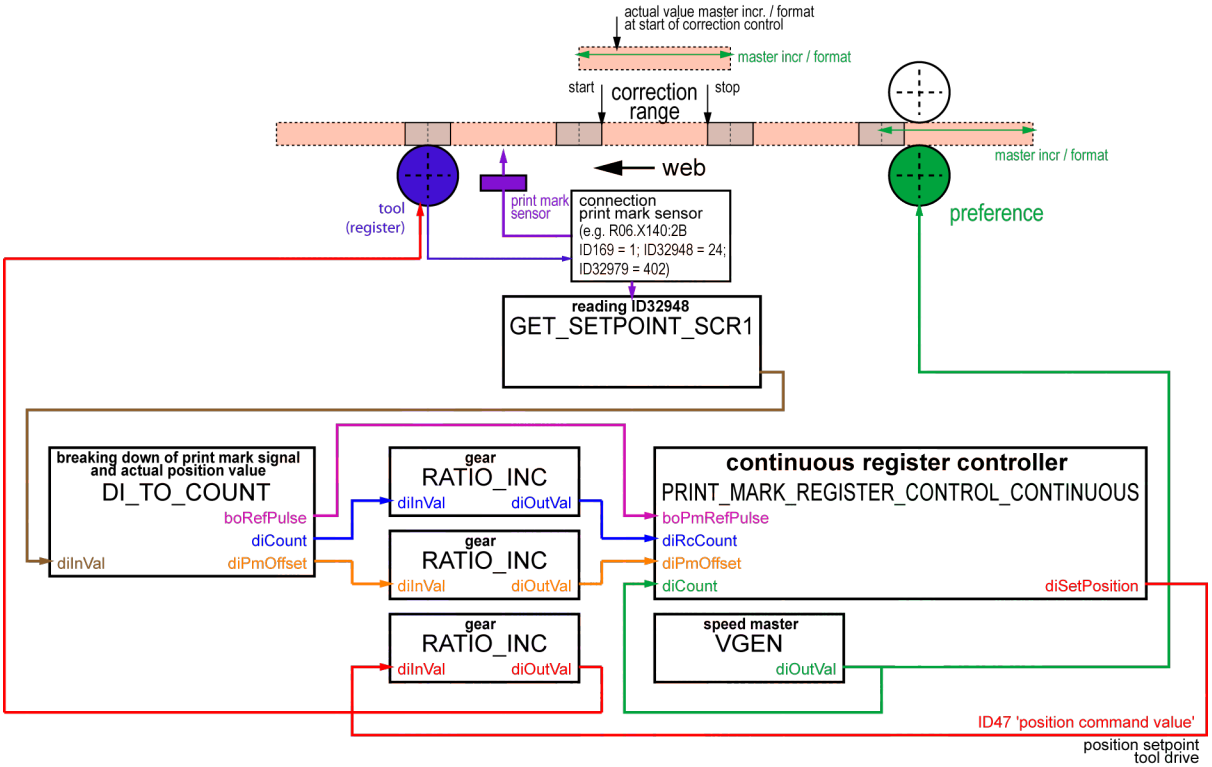
The function block operates with positive counting direction of the input pulses and sets positive output pulses.

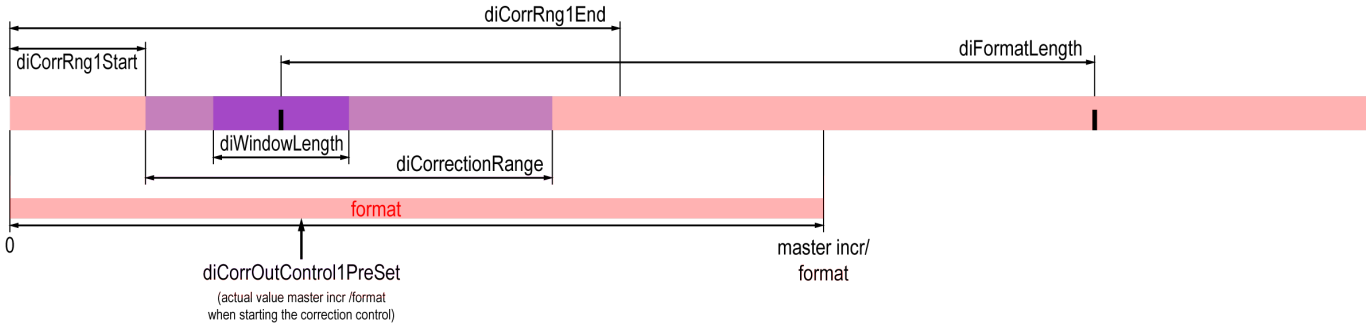
By means of 'boMasterNegDir' and 'boMotorNegDir', counting and motor direction can be inverted. There is no influence to the other output values.

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

Setpoints and current values are transferred in a synchronous action. The synchronous action must be called in the synchronous program level FPLC_PRG.

Schematic diagram





Example values

diFormatLength	10000 Inkr
diWindowLength	2500 Inkr
diCorrectionRange	70 %
diCorrRng1Start	1000 Inkr
diCorrRng1End	9000 Inkr
diCorrOutControl1PreSet	0 Inkr (keine Verschiebung)

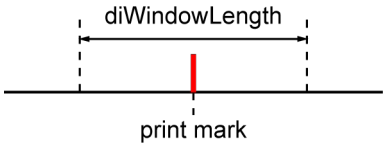
User interface

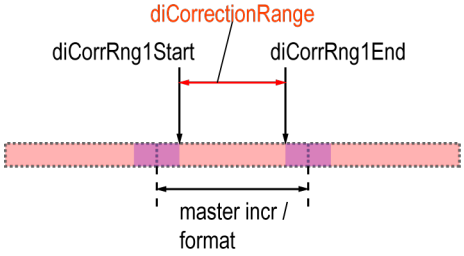
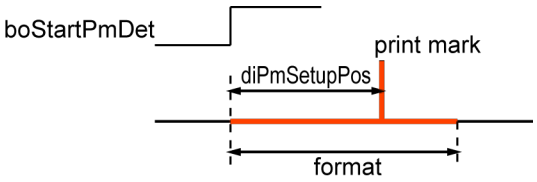
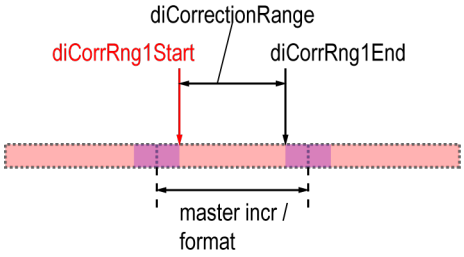
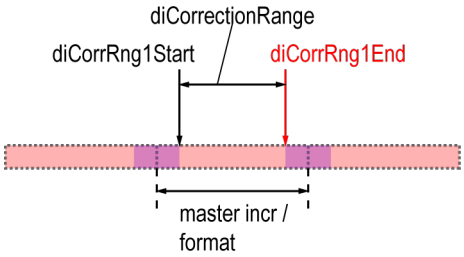
		PRINT_MARK_REGISTER_CONTROL_CONTINUOUS			
FB enable	-	boEnable	boEnabAck	-	Ackn. "FB enable"
Reset output boPmLost	-	boResetPmLost	boParameterSetupDone	-	Parameterisation finished
Start output operator offset	-	boSetupOffset	boPmDetected	-	Print mark detected
Start with print mark setpoint position	-	boSetupPmPos	boPmLost	-	Print mark lost
Master counts negative	-	boMasterNegDir	boPmWnd	-	Print mark window active
Motor direction negative	-	boMotorNegDir	diPmDeviation	-	Deviation between setpoint and actual position
Enable correction output	-	boCorrOutControlOn	diOpOffsetOut	-	Output operator offset
Operator offset	-	diOpOffset	iActPmLost	-	Number of lost print marks
Format length	-	diFormatLength	boErr	-	Error
Window length	-	diWindowLength	iErrID	-	Error ID
Number of marks until message 'mark lost'	-	iNoPmLost			
Correction range	-	diCorrectionRange			
Print mark position setpoint	-	diPmSetupPos			
Max. correction per format	-	diCorrectionLimit			
Start correction range 1	-	diCorrRng1Start			
End correction range 1	-	diCorrRng1End			
Offset correction control 1	-	diCorrOutControl1PreSet			
Start correction range 2	-	diCorrRng2Start			
End correction range 2	-	diCorrRng2End			
Offset correction control 2	-	diCorrOutControl2PreSet			
Position setpoint	-	diSetPosition	diSetPosition	-	Position setpoint

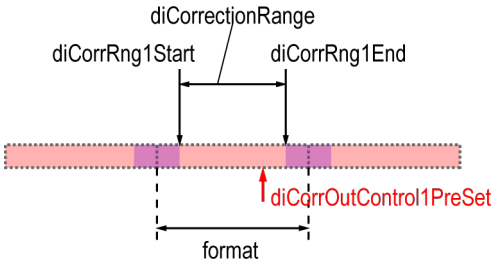
		actSync			
Start print mark detection	-	boStartPmDet			
Print mark detected	-	boPmRefPulse			
Master pulses	-	diCount			
Print mark offset	-	diPmOffset			
Register controller pulses	-	diRcCount			
Position setpoint	-	diSetPosition	diSetPosition	-	Position setpoint

Input variables of the asynchronous program level (PLC_PRG)

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.
boResetPmLost	BOOL	Resetting the output 'boPmLost'
boSetupOffset	BOOL	Output start of the specified value of 'diOpOffset'. This value is used as an offset to shift the printing mark set position

Name	Type	Description
boSetupPmPos	BOOL	Mode how to determine the nominal position of the print mark on a positive edge of 'boStartPmDet' 'boPmSetupPos' = FALSE: The position of the next print mark is set as nominal position. All following print marks are adjusted to this position. 'boPmSetupPos' = TRUE: The parameter 'diPmSetupPos' specifies the print mark relatively to the actual master position. The following print marks are adjusted to this position
boMasterNegDir	BOOL	Direction of master pulse negated 'boMasterNegDir' = FALSE: no negation 'boMasterNegDir' = TRUE: negation This input must only be set on a program start.
boMotorNegDir	BOOL	Direction of motor rotation negated 'boMotorNegDir' = FALSE: no negation 'boMotorNegDir' = TRUE: negation This input must only be set on a program start.
boCorrOutControlOn	BOOL	Correction output enabled 'boCorrOutControlOn' = FALSE: always correction output 'boCorrOutControlOn' = TRUE: correction output only in the ranges specified by 'diCorrRng1Start' and 'diCorrRng1End' resp. 'diCorrRng2Start' and 'diCorrRng2End'. The ranges are OR operated The modulo counting is started with a 0 -> 1 edge at 'boCorrOutControlOn' and can be preset by 'diCorrOutControl1PreSet' or 'diCorrOutControl2PreSet' Example: If the controlled drive is positioned in the middle of the format when starting the correction control, >format/2< must be set to 'diCorrOutControl1PreSet' on a 0 -> 1 edge of 'boCorrOutControlOn'
diOpOffset	DINT	Operator offset. Shifts the set position of the print mark. The shift takes place in the correction range and starts with a 0 → 1 edge at 'boSetupOffset'. It is independent of changes in the value of 'diCount'. The offset is calculated as
diFormatLength	DINT	Format length Nominal distance between two print marks [increments]
diWindowLength	DINT	Window length Length in front and after the nominal print mark position. In this range, the print mark sensor is activated and will accept a print mark. 
iNoPmLost	INT	Number of print marks to be missed in succession, before the output 'boPmLost' is set.

Name	Type	Description
diCorrectionRange	DINT	Correction range The print mark control tries to output the correction in the specified range [% diFormatLength] 
diPmSetupPos	DINT	The parameter specifies the nominal position of the print mark relatively to the actual master position on a positive edge of 'boStartPmDet'. The detected print marks are controlled to this position [increments] 
diCorrectionLimit	DINT	Maximum correction output per format [increments] 'diCorrectionLimit' = 0: no limitation 'diCorrectionLimit' > 0: maximal value
diCorrRng1Start diCorrRng2Start	DINT	Correction range 1 / 2 Start value related to master increments / format [increments] 
diCorrRng1End diCorrRng2End	DINT	Correction range 1 / 2 End value related to master increments / format [increments] 

Name	Type	Description
diCorrOutControl1PreSet diCorrOutControl2PreSet	DINT	Actual position value (zero offset) at start of the correction control, related to the master increments / format [increments] Example: If the controlled drive is positioned in the middle of the format when starting the correction control, >format/2< must be set on a 0 -> 1 edge of 'boCorrOutControlOn' 

Input variables of the synchronous program level (FPLC_PRG)

Name	Type	Description
boStartPmDet	BOOL	Start of the print mark control and specification of the print mark setpoint position by parameter 'diPmSetupPos' if 'boSetupPmPos' = TRUE
boPmRefPulse	BOOL	Print mark pulse detected (reference pulse) (See document Software description AmkBase Bibliothek, Part no. 204986) BasicSupport TIME_TO_COUNT
diCount	DINT	Master pulses [increments] (See document Software description AmkBase Bibliothek, Part no. 204986) BasicSupport DI_TO_COUNT
diPmOffset	DINT	Print mark offset of the actual master position (See documentSoftware descriptionAmkBase Bibliothek, Part no. 204986) BasicSupport TIME_TO_COUNT
diRcCount	DINT	Register controller pulses Pulses of the tool [Incr]

Output variables of the asynchronous program level (PLC_PRG)

Name	Type	Description
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled
boParameterSetupDone	BOOL	Parameterisation finished
boPmDetected	BOOL	Print mark detected, control activated
boPmLost	BOOL	Print mark lost activated when the number of print marks set in 'iNoPmLost' is not detected successively.
boPmWnd	BOOL	Mark window activates the print mark sensor
diPmDeviation	DINT	Deviation between setpoint and actual position of the actual print mark
diOpOffsetOut	DINT	Output value contains the actually output operator offset
iActPmLost	INT	Number of lost print marks

<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>
<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>iErrID</td><td>INT</td><td>Error identity number: Diagnostic number is output
101 diFormatLength ≤ 0
104 diCorrectionRange ≤ 0</td></tr>
</table>

Input and output variables

Name	Type	Description
diSetPosition	DINT	Specification of the position setpoint (position setpoint system) [increments]

Usage note in the CoDeSys program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
PRINT_MARK_REGISTER_CONTROL_CONTINUOUS	PRINT_MARK_REGISTER_CONTROL_CONTINUOUS.actSync

Associated with this function block, a visualisation is prepared in CoDeSys.
Siehe 'Visualization of AFL blocks' auf Seite 806.

4.2.1.2.2.10 PRINT_MARK_REGISTER_CONTROL_DISCONT (FB)

The function block 'PRINT_MARK_REGISTER_CONTROL_DISCONT' implements a print mark control. The position of the treating tool (e.g. cutter, punch) is controlled in relation to a master format (register control)

- Master format and circumference of the treating tool do not have to be equal
- While the treating tool is in mesh with the web (e.g. cut), it moves synchronously.
  After the treatment, an adequate compensation movement is done.
  The movement is discontinuous
- A trend deviation between actual and setpoint print mark position will be controlled
- The function block implements two kinds of collecting the setpoint position of the print mark:
  - The setpoint position can be set to an absolute position within the print format.
  - The first detected print mark sets the setpoint position (Siehe 'PRINT_MARK_CONTROL (FB)' auf Seite 480.)
- Control of the print mark sensor depending on a window
- Monitoring of the print marks
- Simulation of missing print marks
- Output of the correction path as even distribution of the increments on a preset correction path
- Limitation of the correction velocity
- Output an offset for shifting the print mark setpoint position

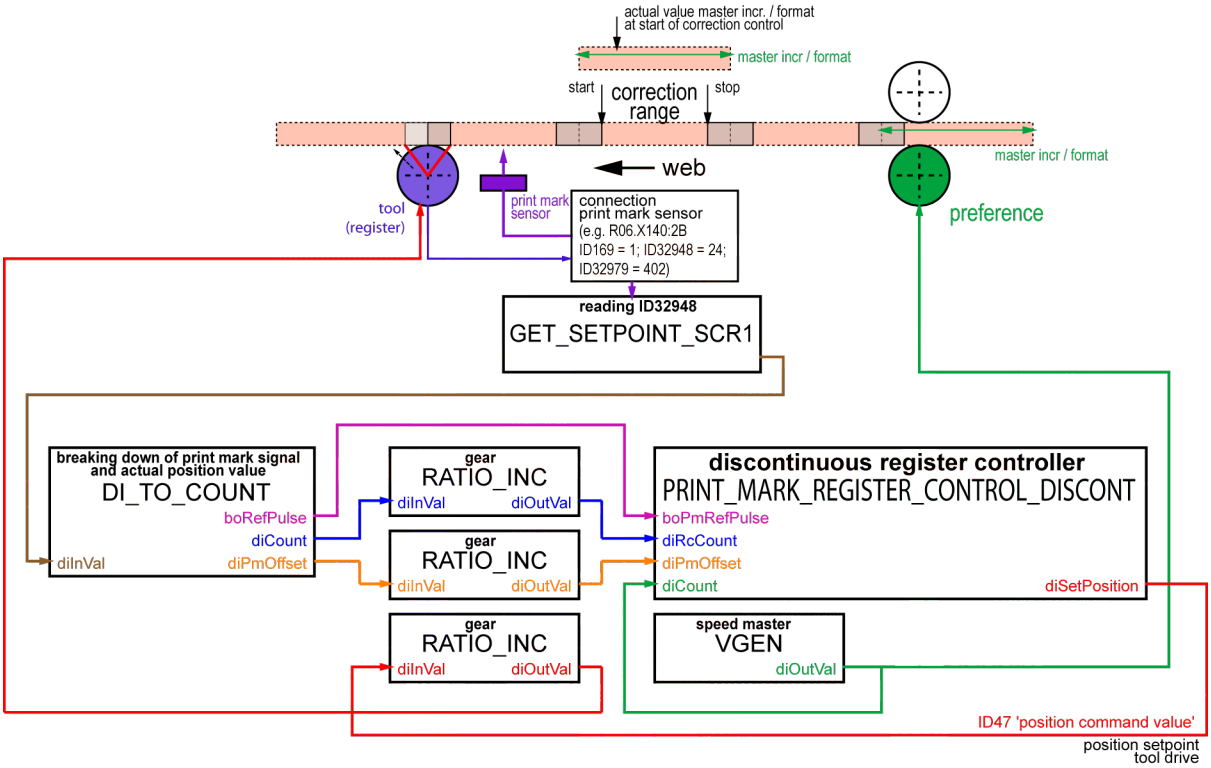
The function block operates with positive counting direction of the input pulses and sets positive output pulses.

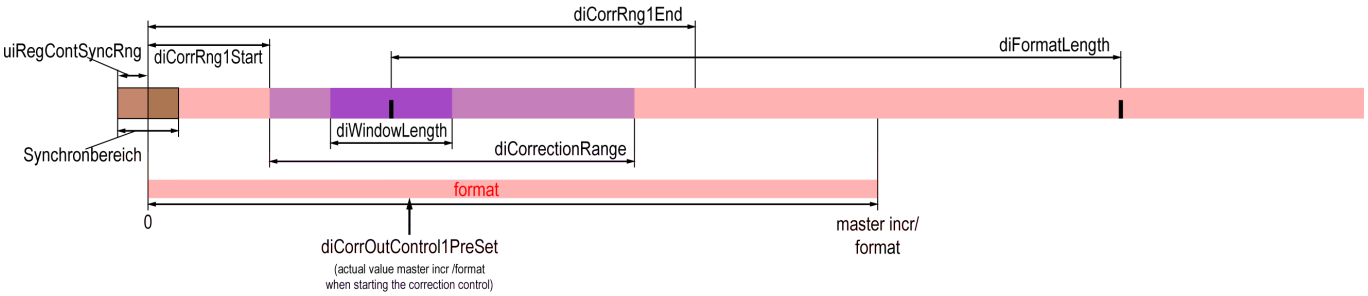
By means of 'boMasterNegDir' and 'boMotorNegDir', counting and motor direction can be inverted. There is no influence to the other output values.

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

Setpoints and current values are transferred in a synchronous action. The synchronous action must be called in the synchronous program level FPLC_PRG.

Schematic diagram





Example values

diFormatLength	10000 Inkr
diWindowLength	2500 Inkr
diCorrectionRange	70 %
diCorrRng1Start	1000 Inkr
diCorrRng1End	9000 Inkr
diCorrOutControl1PreSet	0 Inkr (keine Verschiebung)
uiRegContSyncRng	18 °

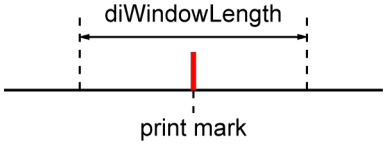
User interface

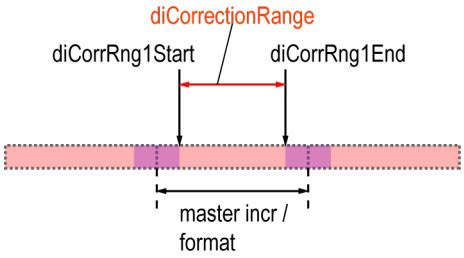
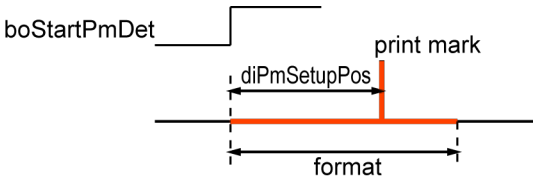
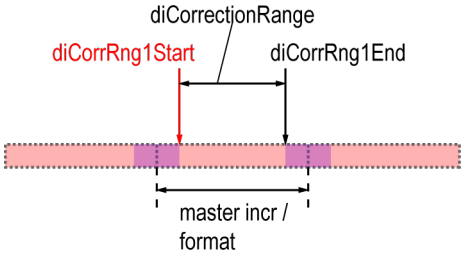
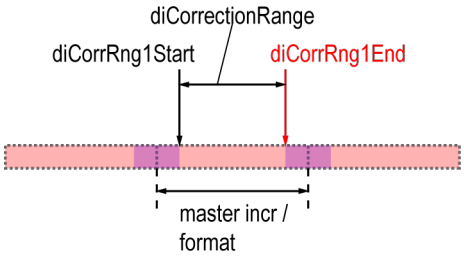
PRINT_MARK_REGISTER_CONTROL_DISCONT					
FB enable	-	boEnable	boEnabAck	-	Ackn. "FB enable"
Reset output boPmLost	-	boResetPmLost	boParameterSetupDone	-	Parameterisation finished
Start output operator offset	-	boSetupOffset	boPmDetected	-	Print mark detected
Start with print mark setpoint position	-	boSetupPmPos	boPmLost	-	Print mark lost
Master counts negative	-	boMasterNegDir	boPmWnd	-	Print mark window active
Motor direction negative	-	boMotorNegDir	diPmDeviation	-	Deviation between setpoint and actual position
Enable correction output	-	boCorrOutControlOn	diOpOffsetOut	-	Output operator offset
Operator offset	-	diOpOffset	iActPmLost	-	Number of lost print marks
Format length	-	diFormatLength	boErr	-	Error
Window length	-	diWindowLength	iErrID	-	Error ID
Number of marks until message 'mark lost'	-	iNoPmLost			
Correction range	-	diCorrectionRange			
Print mark position setpoint	-	diPmSetupPos			
Max. correction per format	-	diCorrectionLimit			
Start correction range 1	-	diCorrRng1Start			
End correction range 1	-	diCorrRng1End			
Offset correction control 1	-	diCorrOutControl1PreSet			
Start correction range 2	-	diCorrRng2Start			
End correction range 2	-	diCorrRng2End			
Offset correction control 2	-	diCorrOutControl2PreSet			
Register controller format	-	diRegContFormat			
Ratio master increments	-	diRegControllerRatio			
Register controller synchronous range	-	uiRegContSyncRng			
Position setpoint	-	diSetPosition	diSetPosition	-	Position setpoint

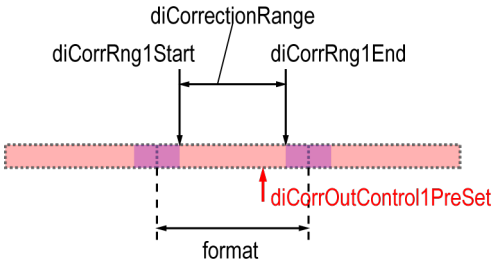
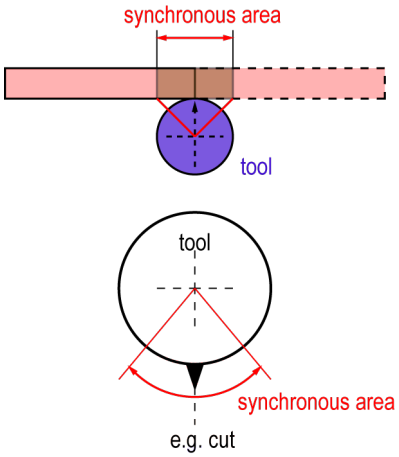
actSync					
Start print mark detection	-	boStartPmDet			
Print mark detected	-	boPmRefPulse			
Master pulses	-	diCount			
Print mark offset	-	diPmOffset			
Register controller pulses	-	diRcCount			
Position setpoint	-	diSetPosition	diSetPosition	-	Position setpoint

Input variables of the asynchronous program level (PLC_PRG)

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.
boResetPmLost	BOOL	Resetting the output 'boPmLost'
boSetupOffset	BOOL	Output start of the specified value of 'diOpOffset'. This value is used as an offset to shift the printing mark set position

Name	Type	Description
boSetupPmPos	BOOL	Mode how to determine the nominal position of the print mark on a positive edge of 'boStartPmDet' 'boPmSetupPos' = FALSE: The position of the next print mark is set as nominal position. All following print marks are adjusted to this position. 'boPmSetupPos' = TRUE: The parameter 'diPmSetupPos' specifies the print mark relatively to the actual master position. The following print marks are adjusted to this position
boMasterNegDir	BOOL	Direction of master pulse negated 'boMasterNegDir' = FALSE: no negation 'boMasterNegDir' = TRUE: negation This input must only be set on a program start.
boMotorNegDir	BOOL	Direction of motor rotation negated 'boMotorNegDir' = FALSE: no negation 'boMotorNegDir' = TRUE: negation This input must only be set on a program start.
boCorrOutControlOn	BOOL	Correction output enabled 'boCorrOutControlOn' = FALSE: always correction output 'boCorrOutControlOn' = TRUE: correction output only in the ranges specified by 'diCorrRng1Start' and 'diCorrRng1End' resp. 'diCorrRng2Start' and 'diCorrRng2End'. The ranges are OR operated The modulo counting is started with a 0 -> 1 edge at 'boCorrOutControlOn' and can be preset by 'diCorrOutControl1PreSet' or 'diCorrOutControl2PreSet' Example: If the controlled drive is positioned in the middle of the format when starting the correction control, >format/2< must be set to 'diCorrOutControl1PreSet' on a 0 -> 1 edge of 'boCorrOutControlOn'
diOpOffset	DINT	Operator offset. Shifts the set position of the print mark. The shift takes place in the correction range and starts with a 0 → 1 edge at 'boSetupOffset'. It is independent of changes in the value of 'diCount'. The offset is calculated as
diFormatLength	DINT	Format length Nominal distance between two print marks [increments]
diWindowLength	DINT	Window length Length in front and after the nominal print mark position. In this range, the print mark sensor is activated and will accept a print mark. 
iNoPmLost	INT	Number of print marks to be missed in succession, before the output 'boPmLost' is set.

Name	Type	Description
diCorrectionRange	DINT	Correction range The print mark control tries to output the correction in the specified range [% diFormatLength] 
diPmSetupPos	DINT	The parameter specifies the nominal position of the print mark relatively to the actual master position on a positive edge of 'boStartPmDet'. The detected print marks are controlled to this position [increments] 
diCorrectionLimit	DINT	Maximum correction output per format [increments] 'diCorrectionLimit' = 0: no limitation 'diCorrectionLimit' > 0: maximal value
diCorrRng1Start diCorrRng2Start	DINT	Correction range 1 / 2 Start value related to master increments / format [increments] 
diCorrRng1End diCorrRng2End	DINT	Correction range 1 / 2 End value related to master increments / format [increments] 

Name	Type	Description
diCorrOutControl1PreSet diCorrOutControl2PreSet	DINT	Actual position value (zero offset) at start of the correction control, related to the master increments / format [increments] Example: If the controlled drive is positioned in the middle of the format when starting the correction control, >format/2< must be set on a 0 -> 1 edge of 'boCorrOutControlOn' 
diRegContFormat	DINT	Number of increments per register rotation [increments]
diRegControllerRatio	DINT	Synchron ratio [%]:
uiRegContSyncRng	UINT	Range where the register controller must run synchronously with the web [degree] 

Input variables of the synchronous program level (FPLC_PRG)

Name	Type	Description
boStartPmDet	BOOL	Start of the print mark control and specification of the print mark setpoint position by parameter 'diPmSetupPos' if 'boSetupPmPos' = TRUE
boPmRefPulse	BOOL	Print mark pulse detected (reference pulse) (See document Software description AmkBase Bibliothek, Part no. 204986) BasicSupport TIME_TO_COUNT
diCount	DINT	Master pulses [increments] (See document Software description AmkBase Bibliothek, Part no. 204986) BasicSupport DI_TO_COUNT
diPmOffset	DINT	Print mark offset of the actual position (See document Software description AmkBase Bibliothek, Part no. 204986) BasicSupport DI_TO_COUNT
diRcCount	DINT	Register controller pulses Pulses of the tool [Incr]

Output variables of the asynchronous program level (PLC_PRG)

Name	Type	Description
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled
boParameterSetupDone	BOOL	Parameterisation finished
boPmDetected	BOOL	Print mark detected, control activated
boPmLost	BOOL	Print mark lost activated when the number of print marks set in 'iNoPmLost' is not detected successively.
boPmWnd	BOOL	Mark window activates the print mark sensor
diPmDeviation	DINT	Deviation between setpoint and actual position of the actual print mark
diOpOffsetOut	DINT	Output value contains the actually output operator offset
iActPmLost	INT	Number of lost print marks
boErr	DINT	The function block is in an error state FALSE: No error (permitted commanding or warning) TRUE: Error iErrID = 0: No error iErrID ≠ 0, boErr = TRUE: Error iErrID ≠ 0, boErr = FALSE: Warning
iErrID	INT	Error identity number: Diagnostic number is output 101 diFormatLength ≤ 0 104 diCorrectionRange ≤ 0 105 diRegContFormat ≤ 0 106 AND 107 uiRegContSyncRng < 0 OR uiRegContSyncRng > 180 > 1000 Error code see AmkBase.lib block 'TAB_CALC' iErrID – 1000

Input and output variables

Name	Type	Description
diSetPosition	DINT	Specification of the position setpoint (position setpoint system) [increments]

Usage note in the CoDeSys program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
PRINT_MARK_REGISTER_CONTROL_DISCONT	PRINT_MARK_REGISTER_CONTROL_DISCONT.actSync

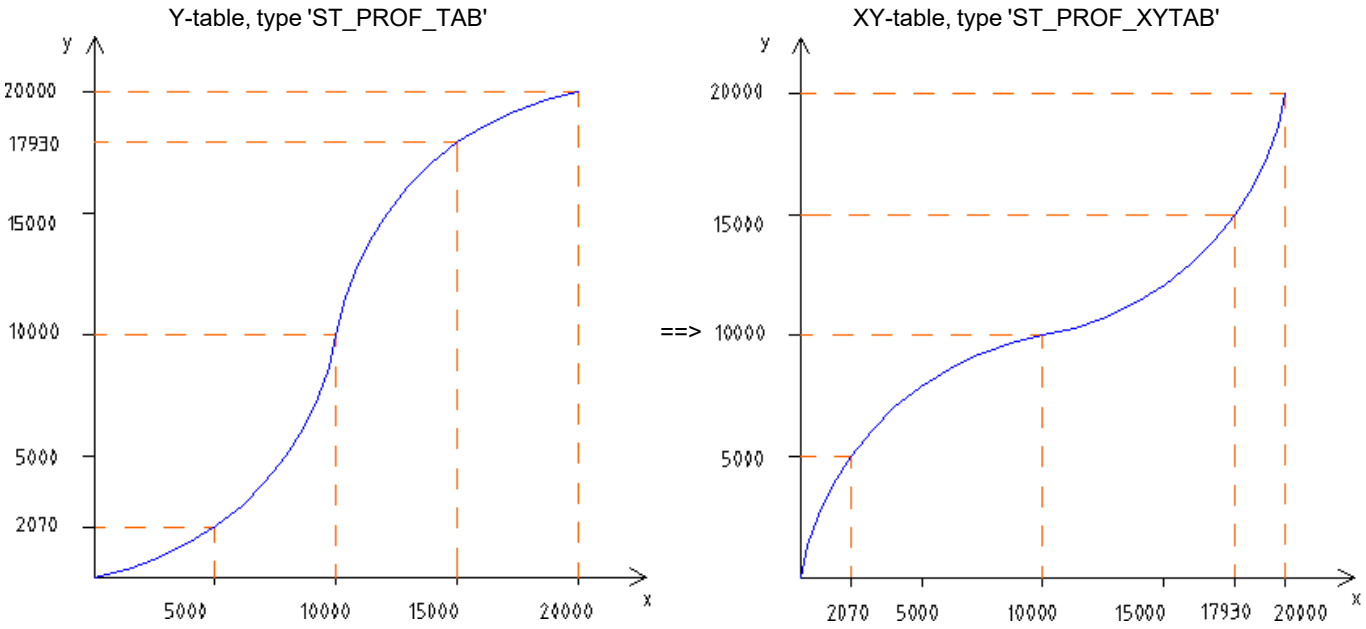
Associated with this function block, a visualisation is prepared in CoDeSys.
Siehe 'Visualization of AFL blocks' auf Seite 806.

4.2.1.2.2.11 PRINT_MARK_TABLE_INVERTER (FB)

The function block 'PRINT_MARK_TABLE_INVERTER' transforms and inverts a Y-table of type 'ST_PROF_TAB' to a XY-table of type 'ST_PROF_XYTAB'.

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

Mode of action of the function block PRINT_MARK_TABLE_INVERTER



User interface

	PRINT_MARK_TABLE_INVERTER			
FB enable -	boEnable	boEnabAck	-	Ackn. "FB enable"
FB execution -	boExec	boDone	-	Ackn. "FB done"
Y-table -	stYTab	stYTab	-	Y-table
XY-table -	stXYInversTab	stXYInversTab	-	XY-table

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. Execute transformation

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. Output finished

Input and output variables

Name	Type	Description
stYTab	STRUCT	ST_PROF_TAB Table structure (See document Software description AmkBase Bibliothek, Part no.204986)
stXYInversTab	STRUCT	ST_PROF_XYTAB Table structure XY-table (See document Software description AmkBase Bibliothek, Part no.204986)

Usage note in the CoDeSys program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
PRINT_MARK_TABLE_INVERTER	

- If the calculation of the table is done with the function block 'TAB_CALC' (AmkBase.lib) or a block from the AmkTabC.lib library, the table structure can be used without changes at the input 'stYTab'.
- If the table is created with the editor, the following adaptation is necessary:

Declaration:

```
VAR
    pYTab:   POINTER TO ST_PROF_TAB;
END_VAR
```

Program call:

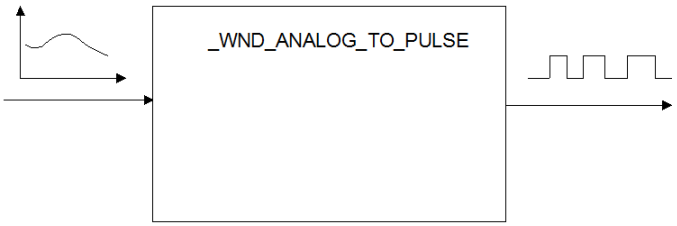
```
pYTab:=ADR(g_TestCamY);

fbPrintMarkTableInvert(
            boEnable:= ,
            boExec:= ,
            stYTab:=          pYTab^,
            stXYInversTab:= g_TestXY,
            boEnabAck=> ,
            boDone=> );
```

Associated with this function block, a visualisation is prepared in CoDeSys.
Siehe 'Visualization of AFL blocks' auf Seite 806.

4.2.1.2.3 03_Winder

4.2.1.2.3.1 WINDER_ANALOG_TO_PULSE (FB)

The function block 'WINDER_ANALOG_TO_PULSE' allows the conversion of an analog value into pulses.
The function block is called in the synchronous program level FPLC_PRG.



User interface

	WINDER_ANALOG_TO_PULSE			
FB enable -	boEnable	boEnabAck	-	Ackn. "FB enable"
Reset output value -	boResetOutput	diPulsOutVal	-	Puls output
Analog input -	iAnalogInVal	boErr	-	Error
Number of pulses at 10 V -	diPulsesAt10V	iErrID	-	Error ID
Device structure -	stDevice	stDevice	-	Device structure

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.
boResetOutput	BOOL	Resets the output value
iAnalogInVal	INT	Analog input, current analog value [mV]
diPulsesAt10V	DINT	Number of pulses per second at 10 V [increments/s] Conversion to U / min of the drive:

Output variables

Name	Type	Description
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled
diPulsOutVal	DINT	Pulse output, the pulses can be copied to the realization of an analog velocity input to module 'WINDER_WINDER' input 'diActWebPos'.
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	ST_DEVICE The device description structure assigns the block a device. (See document Software description AmkBase Bibliothek, Part no.204986)

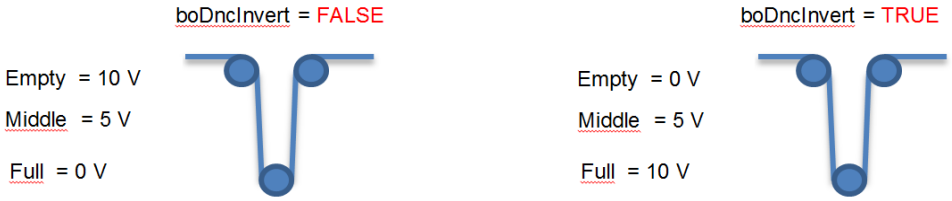
Usage note in the CoDeSys program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
	WINDER_ANALOG_TO_PULSE

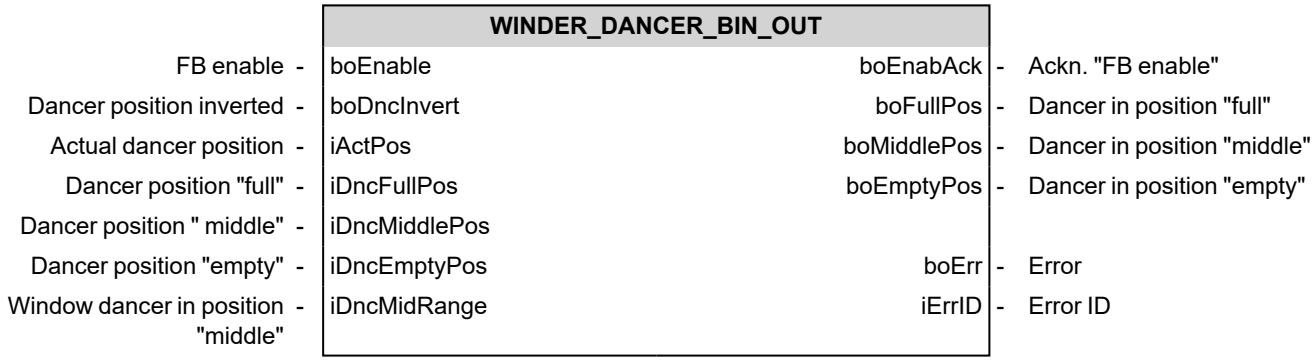
4.2.1.2.3.2 WINDER_DANCER_BIN_OUT (FB)

The function block 'WINDER_DANCER_BIN_OUT' allows an assessment of the dancer position. The assessment is output via digital outputs and can be applied for monitoring or status display further in further program.

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



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.
boDncInvert	BOOL	FALSE: No inversion dancer, position "full" of dancer at least voltage TRUE: Inversion dancer active, position "full" of dancer at largest voltage
iActPos	INT	Actual dancer position [mV]
iDncFullPos	INT	Dancer in position "full" [mV] Default: 1500 mV
iDncMiddlePos	INT	Dancer in position "middle" [mV] Default: 5000 mV
iDncEmptyPos	INT	Dancer in position "empty" [mV] Default: 9500 mV

Name	Type	Description
iDncMidRange	INT	Range around 'iDNC Middelpos' for dancers in middle position [mV] Default: 1000 mV

Output variables

Name	Type	Description
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled
boFullPos	BOOL	Dancer in position "full"
boMiddlePos	BOOL	Dancer in position "middle"
boEmptyPos	BOOL	Dancer in position "empty"
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

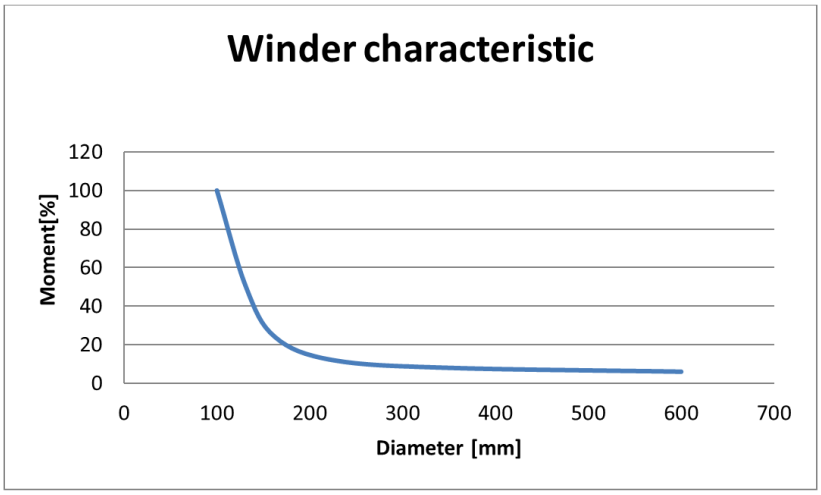
Usage note in the CoDeSys program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
WINDER_DANCER_BIN_OUT	

4.2.1.2.3.3 WINDER_CHARACTERISTICS (FB)

The function block 'WINDER CHARACTERISTICS' realizes the adjustment of the web tension as a function of the reel diameter by means of a winding characteristic. The course of the characteristic is specified in a value table. A maximum of 6 pairs of values are given.

The function block is called in the synchronous program level FPLC_PRG.



User interface

	WINDER_CHARACTERISTICS			
FB enable -	boEnable	boEnabAck	-	Ackn. "FB enable"
Actual Diameter -	iActDm	iForce	-	Winding force
Value table -	arstFunction	boErr	-	Error
		iErrID	-	Error ID

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.
iActDm	INT	Actual diameter [mm]
arstFunction	ARRAY	ARRAY OF ST_WINDER_CHARACTERISTICS_XY_POINT Value table of the characteristic as point array iDiameter - diameter sizes iTorque - winding force in %

Output variables

Name	Type	Description
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled
iForce	INT	winding force [%]
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

Usage note in the CoDeSys program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
	WINDER_CHARACTERISTICS

Program Example for winding characteristics

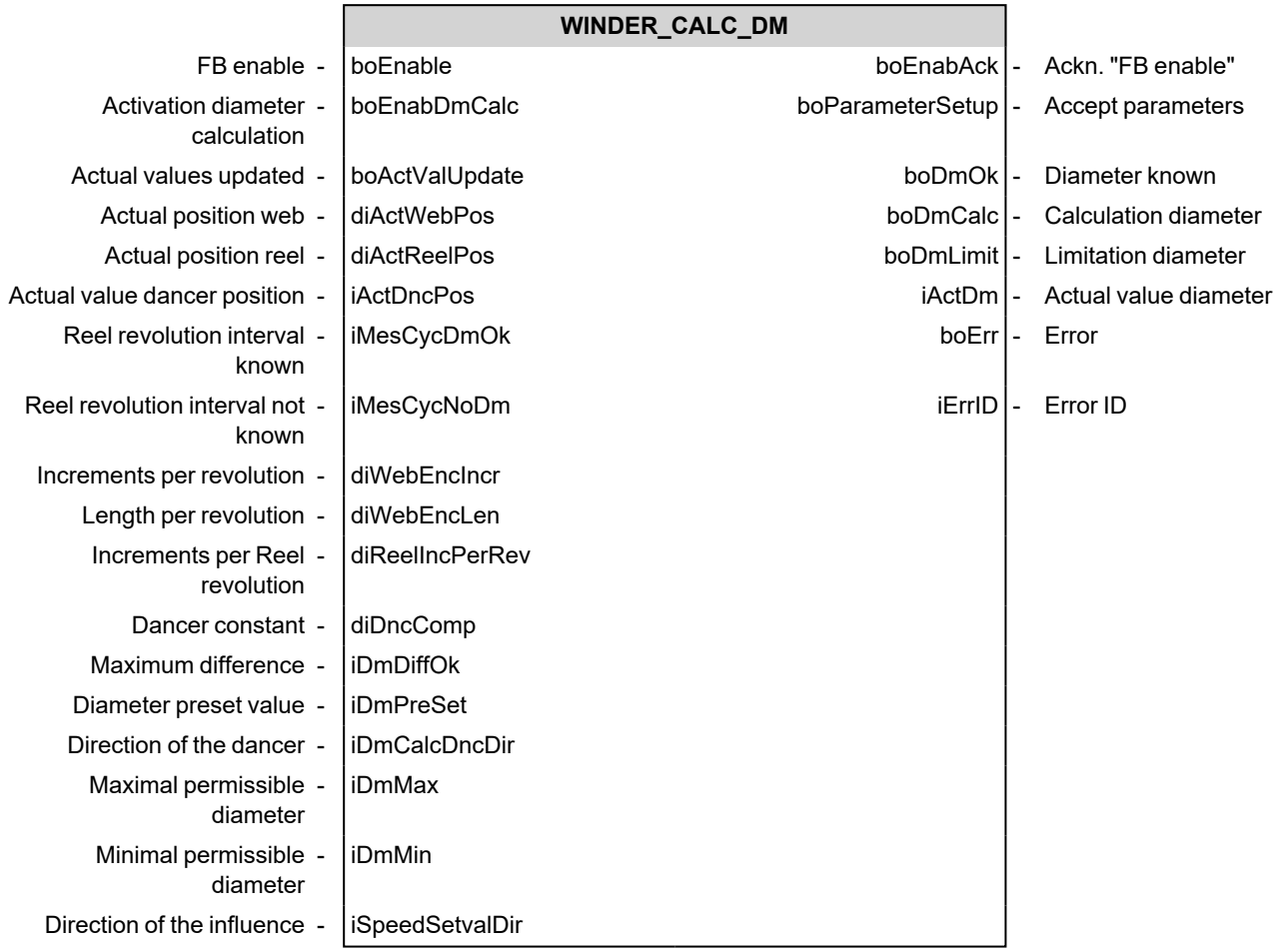
```
iDmDiff:=iDmMax - iDmMin;
fbWinderChar.arstFunction[0].iDiameter:= iDmMin;
fbWinderChar.arstFunction[0].iTorque:= 100;

fbWinderChar.arstFunction[1].iDiameter:= iDmMin+iDmDiff/16;
fbWinderChar.arstFunction[1].iTorque:= 50;

fbWinderChar.arstFunction[2].iDiameter:= iDmMin+iDmDiff/8;
fbWinderChar.arstFunction[2].iTorque:= 24;

fbWinderChar.arstFunction[3].iDiameter:= iDmMin+iDmDiff/4;
fbWinderChar.arstFunction[3].iTorque:= 12;

fbWinderChar.arstFunction[4].iDiameter:= iDmMin+iDmDiff/2;
fbWinderChar.arstFunction[4].iTorque:= 8;

fbWinderChar.arstFunction[5].iDiameter:= iDmMin+iDmDiff;
fbWinderChar.arstFunction[5].iTorque:= 6;
```

4.2.1.2.3.4 WINDER_CALC_DM (FB)

The function block 'WINDER_CALC_DM' realizes the diameter calculation. The dancers movement is compensated in the diameter calculation.

The function block is called in the synchronous program level FPLC_PRG.

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.
boEnabDmCalc	BOOL	Activation of the diameter calculation
boActValUpdate	BOOL	Actual values have been updated by fast level
diActWebPos	DINT	Actual web position [increments]
diActReelPos	DINT	Actual reel position [increments]
iActDncPos	INT	Actual value dancer position [mV]
iMesCycDmOk	INT	Reel revolution interval for cycle Calculating diameter if diameter known [0.1 revolutions]
iMesCycNoDm	INT	Reel revolution interval for cycle Calculating diameter if diameter not known [0.1 revolutions]

Name	Type	Description
diWebEncIncr	DINT	Increments per revolution web - position encoder [increments]
diWebEncLen	DINT	Length per revolution web - position encoder [increments]
diReelIncPerRev	DINT	Increments per reel revolution [increments]
diDncComp	DINT	Dancers constant for calculating the diameter
iDmDiffOk	INT	Maximum difference between two successively calculated diameters, so that the review "diameter valid" is accepted [mm]
iDmPreSet	INT	Diameter preset value
iDmCalcDncDir	INT	Direction of the dancer's influence on the diameter calculation [1, -1]
iDmMax	INT	Maximum permissible diameter value [mm]
iDmMin	INT	Minimum permissible diameter value [mm]
iSpeedSetvalDir	INT	Direction of the influence of web speed on the diameter calculation [1, -1]

Output variables

Name	Type	Description
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled
boParameterSetup	BOOL	Accept parameters
boDmOk	BOOL	Known diameter
boDmCalc	BOOL	Diameter was calculated, and the values were out of 'iDmMin' and 'iDmMax', the value was limited accordingly
boDmLimit	BOOL	New diameter - actual value calculated
iActDm	INT	Actual diameter [mm]
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

Actions

Name	Description
actParameterSetup	Parametrized the block < WINDER_CALC_DM >. actParameterSetup(iMesCycDmOk:=..., iMesCycNoDm:=..., diWebEncIncr:=..., diWebEncLen:=..., diReelIncPerRev:=..., iSpeedSetvalDir:=..., iMesCycDmOk:=..., iMesCycNoDm:=..., iDmDiffOk:=..., iDmMax:=..., iDmMin:=...); Called when enable the block.
actPresetDm	Sets the diameter to the value 'iDmPreSet' <_WND_CALC_DM >. actPresetDm (iDmPreSet:=...));
actResetDmOk	Sets "diameter known" to FALSE <_WND_CALC_DM >. actResetDmOk();
actSetDmOk	Sets the diameter to "known" <_WND_CALC_DM >. actPresetDmOk();

Usage note in the CoDeSys program

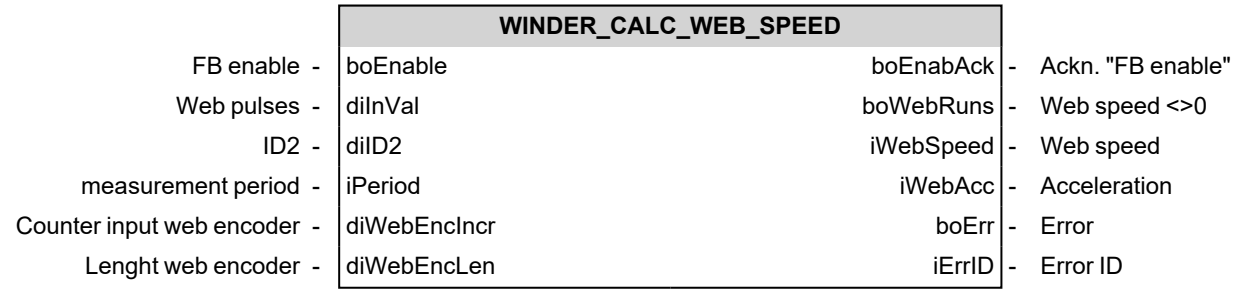
PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
	WINDER_CALC_DM

4.2.1.2.3.5 WINDER_CALC_WEB_SPEED (FB)

The function block 'WINDER_CALC_WEB_SPEED' calculates the actual web speed value and the actual value of the web acceleration.

The function block is called in the synchronous program level FPLC_PRG.

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	Counter input web encoder, puls input [increments]

Name	Type	Description
diID2	INT	PGT clock: ID2 'SERCOS cycle time' [µs]
iPeriod	DINT	Measurement period in multiples of ID2
diWebEncIncr	DINT	Number of counted increments at input 'diInVal' with move on the track 'diWebEncLen' [increments]
diWebEncLen	DINT	Distance covered web track when changing the input 'diInVal' to 'diWebEncIncr' increments [mm]

Output variables

Name	Type	Description
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled
boWebRuns	BOOL	Web speed <>0
iWebSpeed	INT	Web speed [mm/s]
iWebAcc	INT	Web acceleration [mm/s^2]
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

Usage note in the CoDeSys program

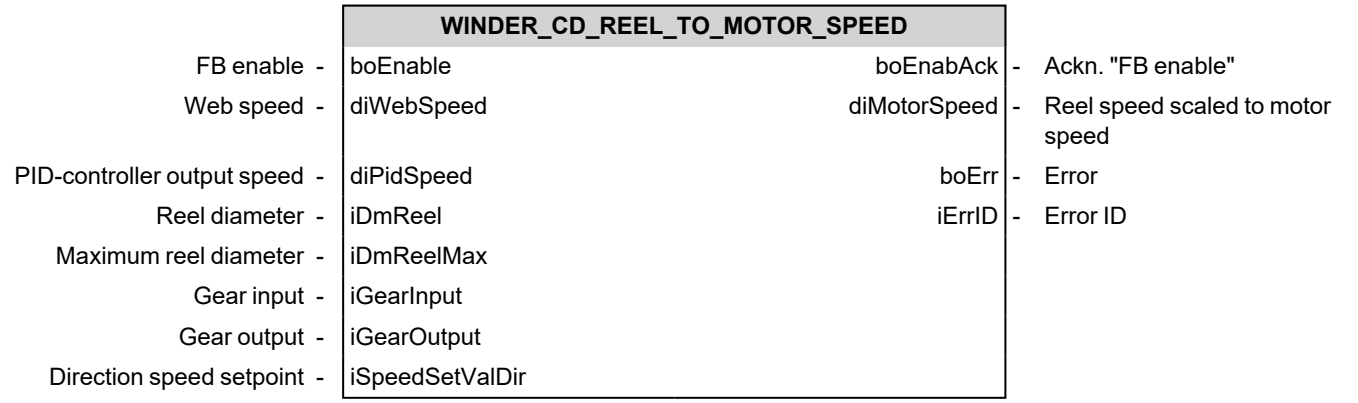
PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
	WINDER_CALC_WEB_SPEED

4.2.1.2.3.6 WINDER_CD_REEL_TO_MOTOR_SPEED (FB)

The function block 'WINDER_CD_REEL_TO_MOTOR_SPEED' calculates the motor speed for the reel that reported in 'iDmReel' diameter from the web speed and the output speed of the PID controller, based on the maximum roll diameter 'iDmReelMax'.

The function block is called in the synchronous program level FPLC_PRG.

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.
diWebSpeed	DINT	Web speed [mm/s]
diPidSpeed	DINT	Output PID-controller [0.1 r/min]
iDmReel	INT	Reel diameter [mm]
iDmReelMax	INT	Maximum reel diameter [mm]
iGearInput	INT	Gear input Motor → Reel
iGearOutput	INT	Gear output Motor → Reel
iSpeedSetValDir	INT	Direction speed setpoint [1,-1]

Output variables

Name	Type	Description
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled
diMotorSpeed	DINT	Scaled reel speed, setpoint speed motor [0.0001 r/min]
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

Usage note in the CoDeSys program

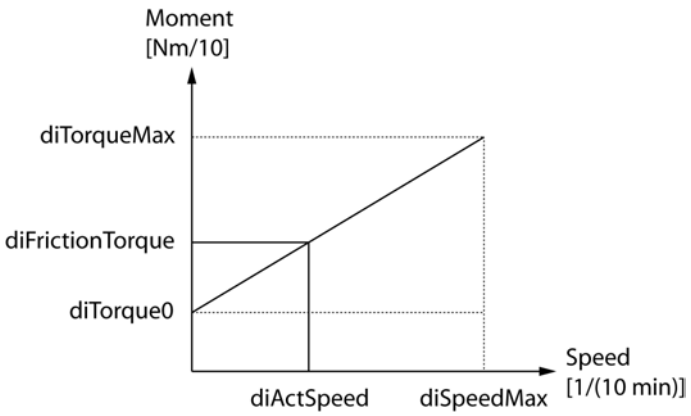
PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
	WINDER_CD_REEL_TO_MOTOR_SPEED

4.2.1.2.3.7 WINDER_CALC_FRICTION (FB)

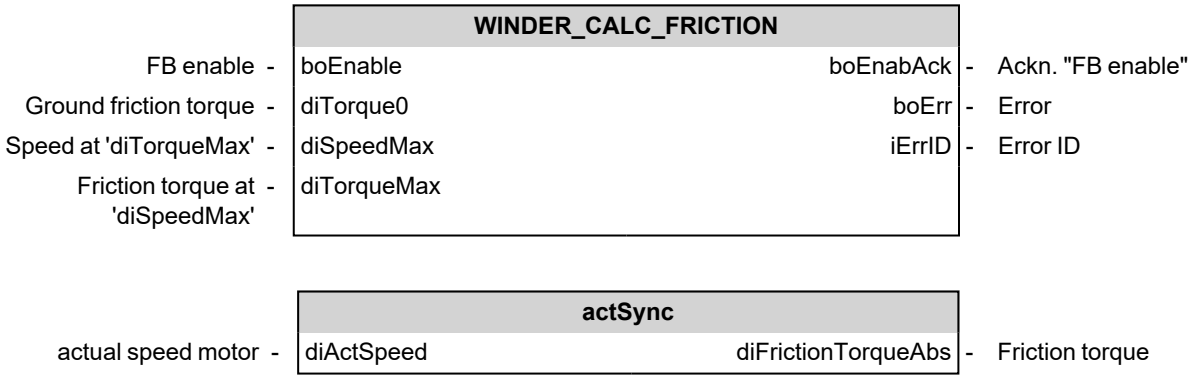
The function block 'WINDER_CALC_FRICTION' calculates the friction torque of a winder in depending on the actual speed.

Setpoints and current values are transferred in a synchronous action. The synchronous action must be called in the synchronous program level FPLC_PRG.

Principle calculation: Function of the friction torque



User interface



Input variables of the asynchronous part of the program (PLC_PRG)

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.
diTorque0	DINT	Ground friction torque [0.1 Nm]
diSpeedMax	DINT	Speed at the 'diTorqueMax' affects [0.1Nm]
diTorqueMax	DINT	Friction torque at speed 'diSpeedMax' [0.1 Nm]

Input variables of the synchronous part of the program (FPLC_PRG)

Name	Type	Description
diActSpeed	DINT	Actual speed [0.1 r/min]

Output variables of the asynchronous part of the program (PLC_PRG)

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

Output variables of the synchronous part of the program (FPLC_PRG)

Name	Type	Description
diFrictionTorqueAbs	DINT	Effective friction torque absolute [0.1 Nm]

Actions

Name	Description
actParameterSetup	Parametrized the block < WINDER_CALC_DM >. actParameterSetup(diTorque0:=..., diSpeedMax:=..., diTorqueMax:=...,); Called when enable the block.

Usage note in the CoDeSys program

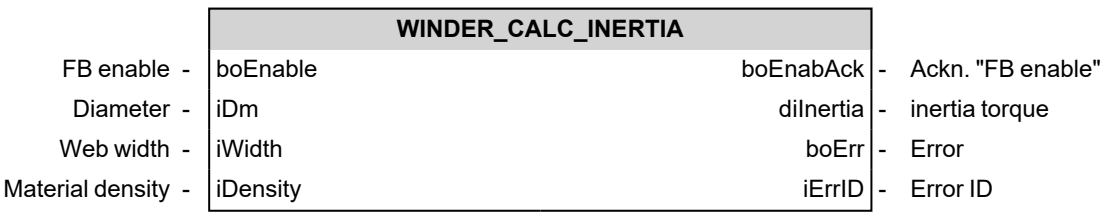
PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
WINDER_CALC_FRICTION	WINDER_CALC_FRICTION.actSync

4.2.1.2.3.8 WINDER_CALC_INERTIA (FB)

The function module 'WINDER_CALC_INERTIA' calculates the moment of inertia of a reel. Winders are assumed to be homogeneous solid cylinder ($J_{Winder} = 0.5 \times m \times r^2$) for the calculation. In order to determine the moment of inertia, the calculation parameters diameter, density and web width are needed.

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

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
iDm	INT	Reel diameter [mm]
iWidth	INT	Reel width [mm]
iDensity	INT	Density of the web material [kg/m^3]

Output variables

Name	Type	Description
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled
diInertia	DINT	Inertia torque of winder [kgm^{2}x1000]
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

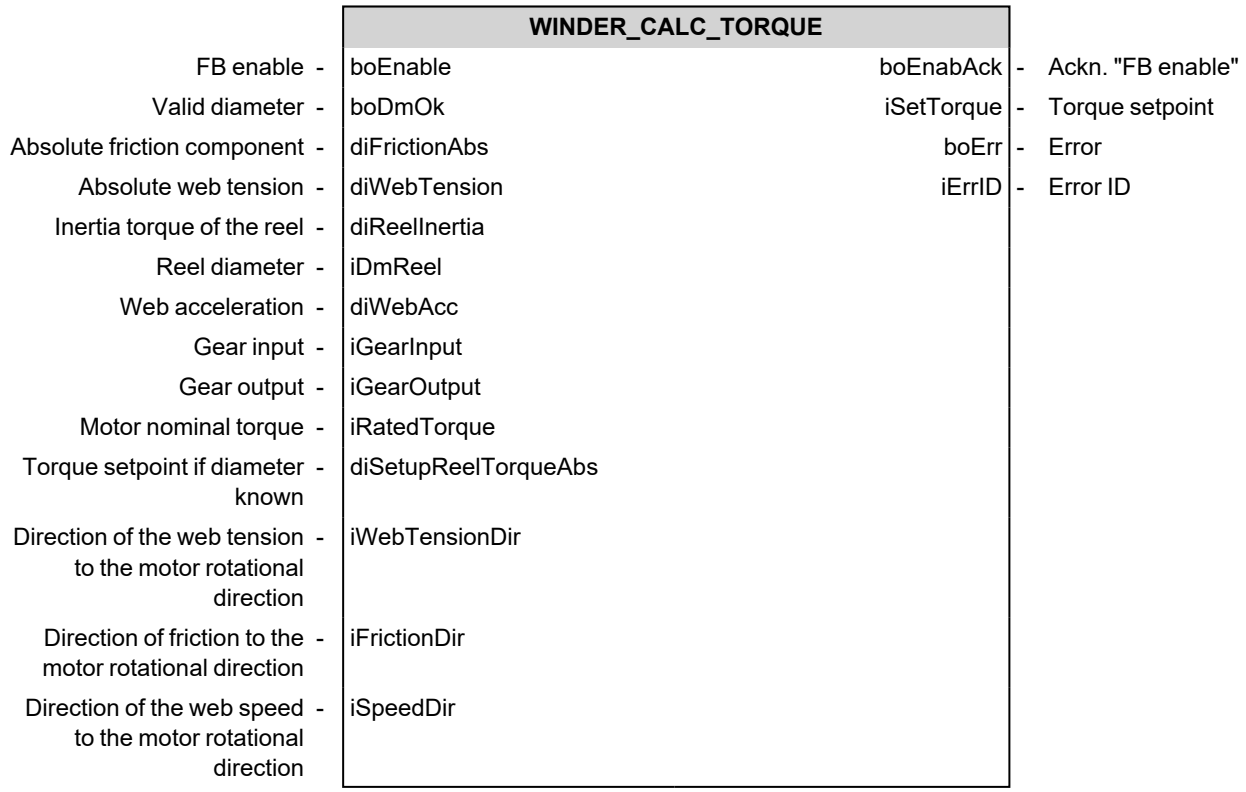
Usage note in the CoDeSys program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
WINDER_CALC_INERTIA	

4.2.1.2.3.9 WINDER_CALC_TORQUE (FB)

The function block 'WINDER_CALC_TORQUE' calculates the friction torque of the reel.
The function block is called in the synchronous program level FPLC_PRG.

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.
boDmOk	BOOL	Known diameter[mm] Valid web diameter 'iDmReel'
diFrictionAbs	DINT	Absolute friction moment [0.1% Nm]
diWebTension	DINT	Absolute web tension moment [0.1 Nm]
diReelInertia	DINT	Inertia torque of the reel [0.001 kgm^2]
iDmReel	INT	Reel diameter [mm]
diWebAcc	DINT	Web acceleration [mm/s^2]
iGearInput	INT	Gear input speed
iGearOutput	INT	Gear output speed
iRatedTorque	INT	Motor nominal torque [0.1 Nm]
diSetupReelTorqueAbs	DINT	Torque setpoint of the reel when the diameter 'iDmReel' is unknown [0.1% Mn]
iWebTensionDir	INT	Direction of the web tension to the positive motor rotation direction [1,-1]
iFrictionDir	INT	Direction of friction in relation to the positive motor rotation direction [1,-1]
iSpeedDir	INT	Direction of the web speed in relation to the positive motor rotation direction [1,-1]

Output variables

Name	Type	Description
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled
iSetTorque	INT	Torque setpoint [0.1% Mn]
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

Usage note in the CoDeSys program

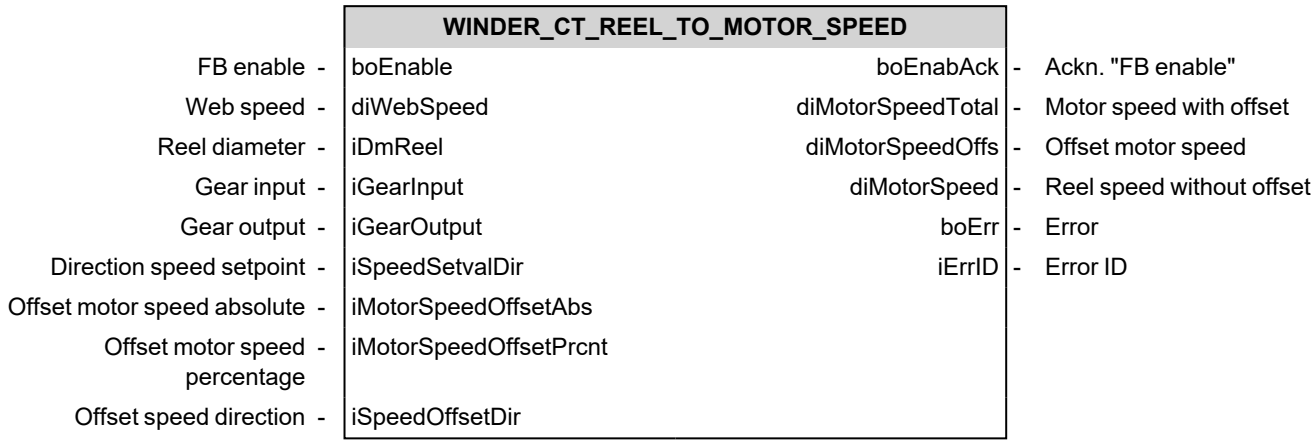
PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
	WINDER_CALC_TORQUE

4.2.1.2.3.10 WINDER_CT_REEL_TO_MOTOR_SPEED (FB)

The function block 'WINDER_CT_REEL_TO_MOTOR_SPEED' is calculated from the reel speed the speed of the motor. It is considered the direction of the speed due to the web speed, as well as the direction of the speed offsets due to the direction of the web turn.

The function block is called in the synchronous program level FPLC_PRG.

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.
diWebSpeed	DINT	Web speed [mm/s]
iDmReel	INT	Reel diameter [mm]
iGearInput	INT	Gear input Motor → Reel
iGearOutput	INT	Gear output Motor → Reel
iSpeedSetvalDir	INT	Direction speed setpoint [1,-1]
iMotorSpeedOffsetAbs	INT	Offset speed absolute [0.1 r/min]
iMotorSpeedOffsetPrcnt	INT	Offset speed percentage [%]
iSpeedOffsetDir	INT	Offset speed direction [1,-1]

Output variables

Name	Type	Description
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled
boParamSetupDone	BOOL	Parameterisation finished
diMotorSpeedTotal	DINT	Motor speed setpoint [0.0001 r/min]
diMotorSpeedOffs	DINT	Offset motor speed [0.1 r/min]
diMotorSpeed	DINT	Reel speed setpoint [0.1 r/min]

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

Usage note in the CoDeSys program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
	WINDER_CT_REEL_TO_MOTOR_SPEED

4.2.1.2.3.11 WINDER_PID_CTRL (FB)

The function block 'WINDER_PID_CTRL' realizes the dancer control of the reel.

Setpoints and current values are transferred in a synchronous action. The synchronous action must be called in the synchronous program level FPLC_PRG.

User interface

	WINDER_PID_CTRL			
FB enable -	boEnable	boEnabAck	-	Ackn. "FB enable"
permit negative speed -	boNegSpeedPermit	boParamSetupDone	-	Parameters accept
Dancers gain by characteristics curve active -	boRaiseGain	boErr	-	Error
I component of the PID controller is active -	boActivateKi	iErrID	-	Error ID
Delete integral sum -	boClearI			
Dancer position setpoint -	iSetVal			
milliseconds counter -	diClock			
Configuration structure -	stConfig			
Device structure -	stDevice	stDevice	-	Device structure

	actSync			
Actual value dancer position -	iActVal	diOutSpeed	-	actual speed value

Input variables of the asynchronous part of the program (PLC_PRG)

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.
boNegSpeedPermit	BOOL	permit negative speed
boRaiseGain	BOOL	FALSE: Dancers gain equal parameter 'iKp' TRUE: Dancers gain by characteristics curve active
boActivateKi	BOOL	I component of the PID controller is active
boClearI	BOOL	Delete integral sum of the PID controller

Name	Type	Description
iSetVal	INT	Setpoint dancer position [mV]
stConfig	STRUCT	ST_WINDER_PID_CONFIG Configuration structure

Input variables of the synchronous part of the program (FPLC_PRG)

Name	Type	Description
iActVal	INT	Actual value dancer position [ms]
diClock	DINT	ms counter [ms]

Output variables of the asynchronous part of the program (PLC_PRG)

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

Output variables of the synchronous part of the program (FPLC_PRG)

Name	Type	Description
diOutSpeed	DINT	Speed setpoint (controlled variable) The unit depends on 'ST_WINDER_PID_CONFIG' → 'diOutputScale' diOutputScale = 1 → [0.0001 r/min] diOutputScale = 10 → [0.001 r/min] diOutputScale = 100 → [0.01 r/min] diOutputScale = 1000 → [0.1 r/min]

Input and output variables

Name	Type	Description
stDevice	STRUCT	ST_DEVICE The device description structure assigns the block a device. (See document Software description AmkBase Bibliothek, Part no.204986)

Actions

Name	Description
actParameterSetup	Parametrized the block < WINDER_PID_CTRL >. actParameterSetup(stConfig...,); Called when enable the block.

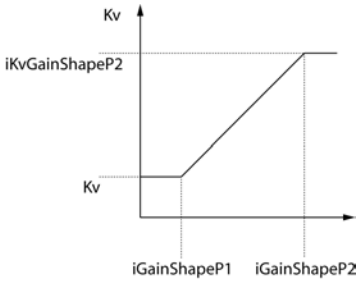
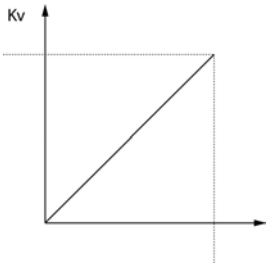
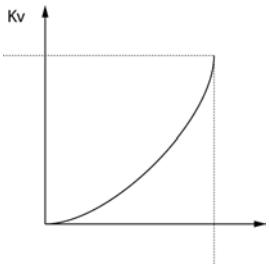
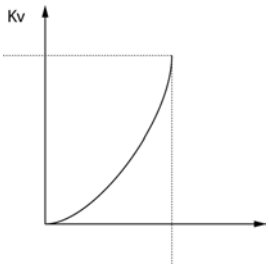
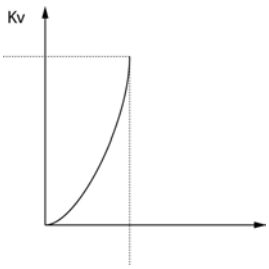
Usage note in the CoDeSys program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
WINDER_PID_CTRL	WINDER_PID_CTRL.actSync

ST_WINDER_PID_CONFIG (ST)

Configuration structure PID-controller

Name	Type	Ex. value	Description
iPidTick	INT	10	Call cycle of the PID controller [iPidTick * ID2]
iKp	INT	5000	Kp for PID-controller [-]
iKi	INT	10	Ki for PID-controller [-]
iKd	INT	0	Kd for PID-controller [-]
iKv	INT	25	Total gain of the PID controller: When enabled, dancer position-dependent gain Kv is the gain at (Dancers setpoint - actual value dancer) < iGainShapeP1 [-]
diOutputScale	DINT	1	Scale speed output [-]
iPosKiActivate	INT	2500	Dancer position for activating Ki [mV]
diImax	DINT	20000	Maximum permissible integral action [r/10000 min]
iDmPidSet	INT	600	Reel diameter, in which the parameters 'iKp', 'iKi', 'iKd' and 'iKv' of the PID controller are set [mm]

Name	Type	Ex. value	Description
iGainShapeType	INT		Type of the gain adjustment Value = 0:  Voltage difference dancer [mV] = (dancer actual position - dancer setpoint position) Value = 1: linear gain characteristic  Voltage difference dancer [mV] = (dancer actual position - dancer setpoint position) Value = 2: square gain characteristic  Voltage difference dancer [mV] = (dancer actual position - dancer setpoint position) Value =3: gain characteristic[3]  Voltage difference dancer [mV] = (dancer actual position - dancer setpoint position) Value = 4: gain characteristic[4]  Voltage difference dancer [mV] = (dancer actual position - dancer setpoint position)

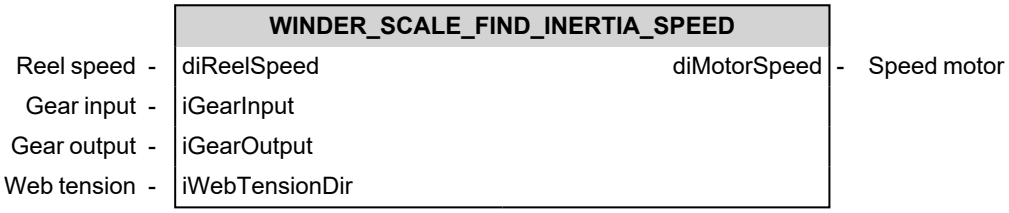
Name	Type	Ex. value	Description
iGainShapeP1	INT	1000	(Dancers setpoint - actual value dancer) = iGainShapeP1 Startpoint of the gain ramp [mV]
iGainShapeP2	INT	3000	(Dancers setpoint - actual value dancer) = iGainShapeP1 Endpoint of the gain ramp [mV]
iKvGainShapeP2	INT	50	Gain at (dancers setpoint - actual value dancer) > iGainShapeP2

4.2.1.2.3.12 WINDER_SCALE_FIND_INERTIA_SPEED (FB)

The function block 'WINDER_SCALE_FIND_INERTIA_SPEED' calculates the speed to determine the moment of inertia of the reel.

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

User interface



Input variables

Name	Type	Description
diReelSpeed	DINT	Reel speed [0.1 r/min]
iGearInput	INT	Gear input
iGearOutput	INT	Gear output
iWebTensionDir	INT	Direction of the web tension to the positive motor rotation direction [1,-1]

Output variables

Name	Type	Description
diMotorSpeed	DINT	Motor speed

Usage note in the CoDeSys program

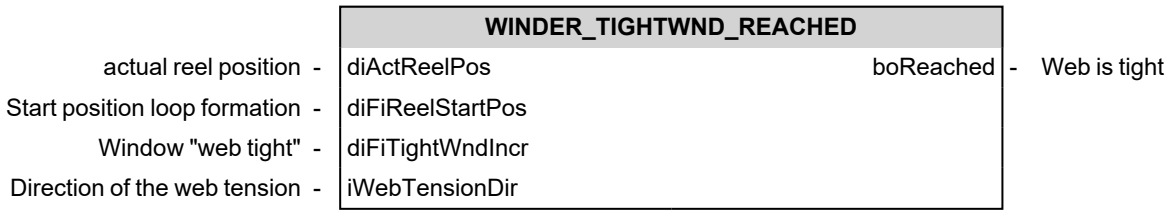
PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
WINDER_SCALE_FIND_INERTIA_SPEED	

4.2.1.2.3.13 WINDER_TIGHTWND_REACHED (FB)

The function block 'WINDER_TIGHTWND_REACHED' monitored the web for tightness

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

User interface



Input variables

Name	Type	Description
diActReelPos	DINT	Actual reel position [increments]
diFiReelStartPos	DINT	Actual position value of the reel on which the web loop was formed [increments]
diFiTightWndIncr	DINT	Window in which the web is monitored for tightness [increments]
iWebTensionDir	INT	Direction of the web tension to the positive motor rotation direction [1,-1]

Output variables

Name	Type	Description
boReached	BOOL	Web is tight

Usage note in the CoDeSys program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
WINDER_TIGHTWND_REACHED	

4.2.1.2.3.14 WINDER_DANCER_CONTROL (FB)

The function block 'WINDER_DANCER_CONTROL' realizes a dancer winder with the following characteristics:

- PID regulated dancer winder
- Operation as unwinder and rewinder
- Diameter calculation
- Search diameter at the start of the winder

Setpoints and current values are transferred in a synchronous action. The synchronous action must be called in the synchronous program level FPLC_PRG.

User interface

		WINDER_DANCER_CONTROL			
FB enable	-	boEnable	boEnabAck	-	Ackn. "FB enable"
Activation of the diameter calculation	-	boDmCalc	boDmOk	-	Diameter ok, status "known"
Dancer in middle position	-	boDncMidPos	boDmLimit	-	Diameter was limited
Dancer mode "Follow actual value"	-	boDncTraceMode	iActDm	-	Actual value diameter
Activate gain ramp	-	boRaiseGain	boErr	-	Error
Activate output web speed	-	boOutputSpeed	iErrID	-	Error ID
Set diameter	-	boDmPreset			
Set diameter, status "known"	-	boDmOkPreset			
Set diameter, status "unknown"	-	boDmNokPreset			
Set diameter status of "known" → "Unknown"	-	boDmReset			
Set output 'boDmOk'	-	boSetDmOk			
Dancer controller delete i component	-	boClearI			
Diameter preset value	-	iDmPreset			
PGT clock	-	uiID2			
Device structure	-	stDevice	stDevice	-	Device structure

actSync

Actual reel position	-	diActReelPos	diVWebSpeed	-	Actual value web speed
Actual web position	-	diActWebPos	diSetValMotorSpeed	-	Speed setpoint motor
Actual value dancer position	-	iActDncPos			

Input variables of the asynchronous part of the program (PLC_PRG)

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.
boDmCalc	BOOL	Activation of the diameter calculation
boDncMidPos	BOOL	Dancers move to the middle position
boDncTraceMode	BOOL	Dancer mode Follow actual value, the actual value of the dancer is accepted as setpoint at the PID controller. Only enabled when 'boDncMidPos' = FALSE
boRaiseGain	BOOL	Position dependent dancers gain ramp active
boOutputSpeed	BOOL	Output web speed as the speed setpoint for motor
boDmPreset	BOOL	Set diameter value of the input 'iDmPreset'. The input bit is automatically reset after set.
boDmOkPreset	BOOL	Diameter value of input variable 'iDmPreset' set. The input bit 'boDmPreset' is automatically reset after set. Diameter status "known": The diameter value is interpreted as valid diameter, that is, acceleration and friction torques are calculated.
boDmNokPreset	BOOL	Diameter value of input variable 'iDmPreset' set. The input bit 'boDmPreset' is automatically reset after set. Diameter status "unknown": The diameter value is interpreted as invalid diameter, that is, acceleration and friction torques are calculated.
boDmReset	BOOL	Reset output 'boDmOk', → diameter status "unknown"
boSetDmOk	BOOL	Set output 'boDmOk', → diameter status "known"
boClearI	BOOL	Dancer controller Delete i component
iDmPreset	INT	Diameter preset value Value to which the diameter with 'boDmNokPreset' = TRUE or 'boDmOkPreset' =TRUE is set.
uiID2	UINT	PGT clock [ms]

Input variables of the synchronous part of the program (FPLC_PRG)

Name	Type	Description
diActReelPos	DINT	Actual reel position [increments]
diActWebPos	DINT	Actual web position [increments]
iActDncPos	INT	Actual value dancer position [mV]

Output variables of the asynchronous part of the program (PLC_PRG)

Name	Type	Description
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled
boDmOk	BOOL	Known diameter
boDmLimit	BOOL	Diameter was calculated, and the result was outside of 'ST_WINDER_DANCER_CONTROL_CONFIG.iDmMin' and 'ST_WINDER_DANCER_CONTROL_CONFIG.iDmMax', the value was limited accordingly
iActDm	INT	Actual diameter [mm]
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>
<table>
<tr><th>Value</th><th>Meaning</th></tr>
<tr><td>1</td><td>The value at input uiID2 = 0</td></tr>
<tr><td>2</td><td>One of the following parameters has a value <=0:
ST_WINDER_DANCER_CONTROL_CONFIG.iGearInput,
ST_WINDER_DANCER_CONTROL_
CONFIG.iGearOutput</td></tr>
<tr><td>3</td><td>One of the following parameters has a value <=0:
ST_WINDER_DANCER_CONTROL_CONFIG.iDmMin,
ST_WINDER_DANCER_CONTROL_CONFIG.iDmMax
oder
ST_WINDER_DANCER_CONTROL_CONFIG.iDmMin >=
ST_WINDER_DANCER_CONTROL_CONFIG.iDmMax</td></tr>
<tr><td>4</td><td>The parameter has a Value <=0:
ST_WINDER_DANCER_CONTROL_
CONFIG.diMotIncPerRev</td></tr>
<tr><td>5</td><td>One of the following parameters has a value <=0:
ST_WINDER_DANCER_CONTROL_
CONFIG.iMesCycDmOk,
ST_WINDER_DANCER_CONTROL_
CONFIG.iMesCycNoDm,
ST_WINDER_DANCER_CONTROL_CONFIG.iDmDiffOk</td></tr>
<tr><td>9</td><td>One of the following parameters has a value <=0:
ST_WINDER_DANCER_CONTROL_
CONFIG.diWebSpeedMax,
ST_WINDER_DANCER_CONTROL_
CONFIG.diWebEncLen,
ST_WINDER_DANCER_CONTROL_
CONFIG.diWebEncIncr,
ST_WINDER_DANCER_CONTROL_
CONFIG.iWebSpdPeriod,
ST_WINDER_DANCER_CONTROL_
CONFIG.iMotorSpeedOffsetAbs,
ST_WINDER_DANCER_CONTROL_
CONFIG.iMotorSpeedOffsetPrcnt</td></tr>
<tr><td>16</td><td>One of the following parameters has a value <=0:
ST_WINDER_DANCER_CONTROL_
CONFIG.stPidConfig.iPidTick,
ST_WINDER_DANCER_CONTROL_
CONFIG.stPidConfig.iKp,
ST_WINDER_DANCER_CONTROL_
CONFIG.stPidConfig.iKv,
ST_WINDER_DANCER_CONTROL_
CONFIG.stPidConfig.diOutputScale,
ST_WINDER_DANCER_CONTROL_
CONFIG.stPidConfig.diImax,
ST_WINDER_DANCER_CONTROL_
CONFIG.stPidConfig.iDmPidSet</td></tr>
</table></td></tr>
</table>

Name	Type	Description
		Value / Meaning **17** — One of the following parameters has a value <=0: ST_WINDER_DANCER_CONTROL_CONFIG.stDancerConfig.iDancerFullPosition, ST_WINDER_DANCER_CONTROL_CONFIG.stDancerConfig.iDancerMidPosition, ST_WINDER_DANCER_CONTROL_CONFIG.stDancerConfig.iDancerEmptyPosition, ST_WINDER_DANCER_CONTROL_CONFIG.stDancerConfig.diTUpRamp, ST_WINDER_DANCER_CONTROL_CONFIG.stDancerconfig.diTDownRamp

Output variables Input variables of the synchronous part of the program (FPLC_PRG)

Name	Type	Description
diVWebSpeed	DINT	Actual value web speed [mm/s]
diSetValMotorSpeed	DINT	Speed setpoint motor [0.0001 r/min]

Input and output variables

Name	Type	Description
stDevice	STRUCT	ST_DEVICE The device description structure assigns the block a device. (See document Software description AmkBase Bibliothek, Part no.204986)
stConfig	STRUCT	ST_WINDER_DANCER_CONTROL_CONFIG Configuration structure of the winder (See document Software description AFL - AMK Function Library, part 2, Part no.203694)

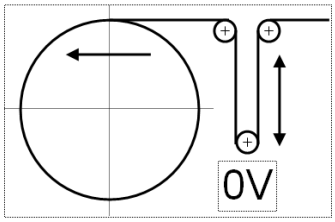
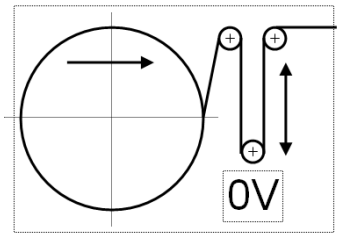
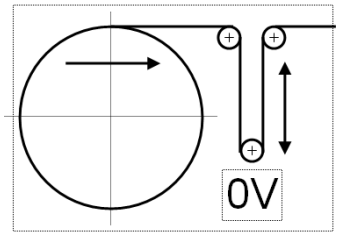
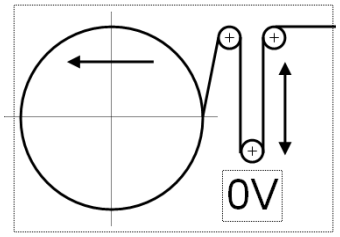
Usage note in the CoDeSys program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
WINDER_DANCER_CONTROL	WINDER_DANCER_CONTROL.actSync

ST_WINDER_DANCER_CONTROL_CONFIG (ST)

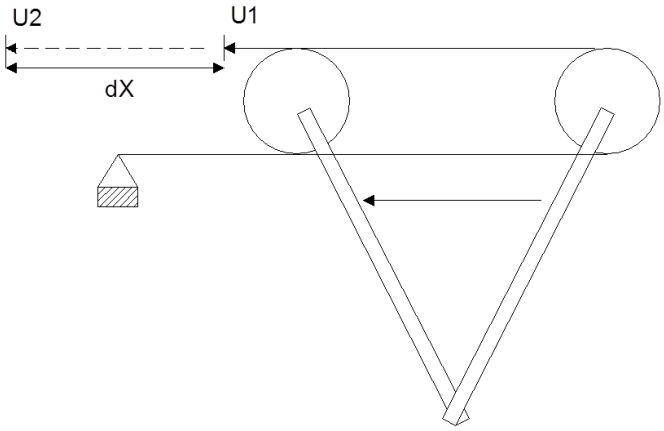
Configuration structure

Name	Type	Ex. value	Description
boWindCWise	BOOL	FALSE	FALSE: reel is wound up when the motor rotation direction counterclockwise. TRUE: reel is wound up when the motor rotation direction clockwise.
boUnwinder	BOOL	FALSE	FALSE: winder runs as a rewinder. With positive web speed is rewinded. TRUE: winder runs as unwinder. With a positive web velocity is unwinded.

Arrangement	Type	boWindCWise	boUnwinder
	Rewinder	FALSE	FALSE
	Rewinder	TRUE	FALSE
	Unwinder	FALSE	TRUE
	Unwinder	TRUE	TRUE

Name	Type	Ex. value	Description
boMotCcWise	BOOL	FALSE	Mounting position of the motor opposite to the reel
iGearInput	INT	1	Gear input (viewed from the motor)
iGearOutput	INT	1	Gear output (viewed from the motor)
iDmMin	INT	100	Minimum permissible diameter value [mm]
iDmMax	INT	600	Maximum permissible diameter value [mm]
diMotIncPerRev	DINT	20000	Increments per motor revolution (ID 116 'Resolution motor encoder')
diDncComp	DINT		Dancers constant → tension difference at 1 m way dancers [mV / m]

The dancers constant 'diDncComp' can be determined or verified by measurement as follows:



By the dancer a rope or a wire is pulled.

The dancer is provided at a lower position. Now the position of the cable end is marked and noted the tension value of the dancer as U1.

Then the dancer is provided at an upper position. Again, the position of the cable end is marked and noted the tension value of the dancer as U2 (U2 > U1).

The distance between the rope ends of the two dancer positions is measured and gives .ΔX. The value .ΔX, used together with the two tension values U1 and U2 in the formula and 'diDncComp' is calculated:

Parameter	Unit	Meaning
diDncComp	[mV/1000mm]	Dancers compensation constant
U1	[mV]	Tension U1 on a dancer position X1
U2	[mV]	Tension U2 on a dancer position X2
ΔX	[mm]	Difference between the positions X2 - X1

Example:

Parameter	measured value	Unit	Meaning
U1	3625	[mV]	Tension U1 on a dancer position X1
U2	7453	[mV]	Tension U2 on a dancer position X2
ΔX	800	[mm]	Difference between the positions X2 - X1

In the measurement, make sure that the dancer is not moved to the end positions, because there usually is a non-linear relationship between the recorded web and tension difference.

Name	Type	Ex. value	Description
iMesCycDmOk	INT	10	Reel revolution interval for cycle Calculating diameter if diameter known [0.1 revolutions]
iMesCycNoDm	INT	1	Reel revolution interval for cycle Calculating diameter if diameter not known [0.1 revolutions]
iDmDiffOk	INT	10	Maximum difference between two successively calculated diameters, so that the review "diameter valid" is accepted [mm]

Speed parameter

Name	Type	Ex. value	Description
diWebEncIncr	DINT	20000	Number of counted increments at input 'diInVal' with move on the track 'diWebEncLen' [increments]
diWebEncLen	DINT	1000	Distance covered web track when changing the input 'diInVal' to 'diWebEncIncr' increments [mm]

Name	Type	Ex. value	Description
iWebSpdPeriod	INT	5	Number of calls cycles ID2 'SERCOS cycle time' for measurement of the web speed

Dancer- and controller parameter

Name	Type	Ex. value	Description
stDancerConfig	STRUCT		ST_WINDER_DANCER_CONFIG
stPidConfig	STRUCT		ST_WINDER_PID_CONFIG

Tuning parameter

Name	Type	Ex. value	Description
diTimeDmRamp	DINT	2000	Ramp time for diameter output [ms] There is a sharp change in diameter at the input (eg setting a preset value), the diameter is adjusted linearly in the set ramp time.
diWebSpeedMax	DINT	20000	Maximum web speed [mm/s]
diTimeWebSpeedRamp	DINT	5000	Web speed ramp [ms] The path velocity is ramped passed to the speed calculation of the reel

ST_WINDER_DANCER_CONFIG (ST)

Configuration structure dancer

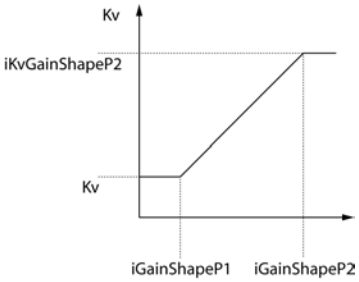
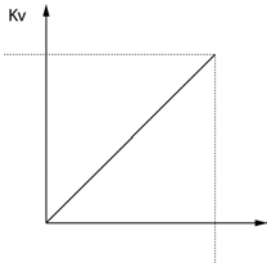
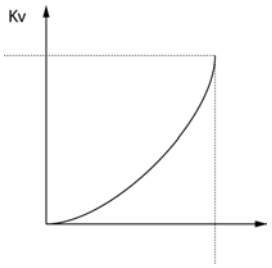
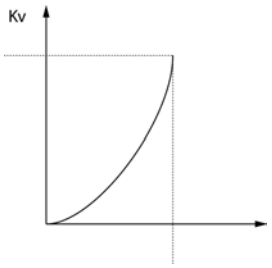
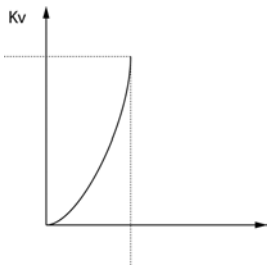
Name	Type	Ex. value	Description
iDancerFullPosition	INT	1000	Dancers tension when the dancer is filled with material [mV]
iDancerMidPosition	INT	5000	Dancers tension in the middle position of the dancer (operating point) [mV]
iDancerEmptyPosition	INT	9000	Dancers tension when the dancer is empty [mV]
diTUpRamp	DINT	1000	Ramp time for the movement of the dancer in the operating point position [ms]
diTDownRamp	DINT	3000	Ramp time for the movement of the dancer in deposition position (dancers filled) [ms]

ST_WINDER_PID_CONFIG (ST)

Configuration structure PID-controller

Name	Type	Ex. value	Description
iPidTick	INT	10	Call cycle of the PID controller [iPidTick * ID2]
iKp	INT	5000	Kp for PID-controller [-]
iKi	INT	10	Ki for PID-controller [-]
iKd	INT	0	Kd for PID-controller [-]
iKv	INT	25	Total gain of the PID controller: When enabled, dancer position-dependent gain Kv is the gain at (Dancers setpoint - actual value dancer) < iGainShapeP1 [-]
diOutputScale	DINT	1	Scale speed output [-]
iPosKiActivate	INT	2500	Dancer position for activating Ki [mV]

Name	Type	Ex. value	Description
diImax	DINT	20000	Maximum permissible integral action [r/10000 min]
iDmPidSet	INT	600	Reel diameter, in which the parameters 'iKp', 'iKi', 'iKd' and 'iKv' of the PID controller are set [mm]

Name	Type	Ex. value	Description
iGainShapeType	INT		Type of the gain adjustment Value = 0:  Voltage difference dancer [mV] = (dancer actual position - dancer setpoint position) Value = 1: linear gain characteristic  Voltage difference dancer [mV] = (dancer actual position - dancer setpoint position) Value = 2: square gain characteristic  Voltage difference dancer [mV] = (dancer actual position - dancer setpoint position) Value =3: gain characteristic[3]  Voltage difference dancer [mV] = (dancer actual position - dancer setpoint position) Value = 4: gain characteristic[4]  Voltage difference dancer [mV] = (dancer actual position - dancer setpoint position)

Name	Type	Ex. value	Description
iGainShapeP1	INT	1000	(Dancers setpoint - actual value dancer) = iGainShapeP1 Startpoint of the gain ramp [mV]
iGainShapeP2	INT	3000	(Dancers setpoint - actual value dancer) = iGainShapeP1 Endpoint of the gain ramp [mV]
iKvGainShapeP2	INT	50	Gain at (dancers setpoint - actual value dancer) > iGainShapeP2

4.2.1.2.3.15 WINDER_TORQUE_CONTROL (FB)

The function block 'WINDER_TORQUE_CONTROL' realizes a sensorless rewinder and unwinder.

- Torque controlled center winder
- Operation as rewinder and unwinder
- Diameter calculation
- Search diameter at the start of the winder
- Determine the reel of inertia of unknown diameter
- Synchronize onto a running web

Setpoints and current values are transferred in a synchronous action. The synchronous action must be called in the synchronous program level FPLC_PRG.

User interface

		WINDER_TORQUE_CONTROL			
FB enable	-	boEnable	boEnabAck	-	Ackn. "FB enable"
Activation of the diameter calculation	-	boDmCalc	boDmOk	-	Diameter ok, status "known"
Determination the inertia moment of the reel	-	boFindInertia	boFindInertiaDone	-	Inertia moment determined
Wind on	-	boWind	boWindRuns	-	Wind runs
Synchronize on	-	boSync	boSyncRuns	-	Synchronize runs
Set diameter	-	boDmPreset	boDmLimit	-	Diameter was limited
Set diameter, status "known"	-	boDmOkPreset	iActDm	-	Actual value diameter
Set diameter, status "unknown"	-	boDmNOkPreset	diActValWebSpeed	-	Actual value web speed
Set diameter status of "known" → "Unknown"	-	boDmReset	boErr	-	Error
Set output 'boDmOk'	-	boSetDmOk	iErrID	-	Error ID
Setpoint web tension	-	diWebTension			
Diameter preset value	-	iDmPreset			
PGT clock	-	uiID2			
Device structure	-	stDevice	stDevice	-	Device structure

		actSync			
Actual speed motor	-	diActMotorSpeed	diSetValMotorSpeed	-	Speed setpoint motor
Actual reel position	-	diActReelPos	iSetValTorqueAbs	-	Limit torque setpoint
Actual web position	-	diActWebPos	diActMotorSpeedAbs	-	Actual motor speed absolute
	-		diActReelPosAbs	-	Actual reel position absolute

Input variables of the asynchronous part of the program (PLC_PRG)

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.
boDmCalc	BOOL	Activation of the diameter calculation
boFindInertia	BOOL	Determination the inertia moment of the reel and tightening web on
boWind	BOOL	Wind on
boSync	BOOL	Synchronize on
boDmPreset	BOOL	Set diameter value of the input 'iDmPreset'. The input bit is automatically reset after set.
boDmOkPreset	BOOL	Diameter value of input variable 'iDmPreset' set. The input bit 'boDmPreset' is automatically reset after set. Diameter status "known": The diameter value is interpreted as valid diameter, that is, acceleration and friction torques are calculated.
boDmNOkPreset	BOOL	Diameter value of input variable 'iDmPreset' set. The input bit 'boDmPreset' is automatically reset after set. Diameter status "unknown": The diameter value is interpreted as invalid diameter, that is, acceleration and friction torques are calculated.
boDmReset	BOOL	Reset output 'boDmOk'
boSetDmOk	BOOL	Set output 'boDmOk'
diWebTension	DINT	Absolute web tension moment [0.1 Nm]
iDmPreset	INT	Diameter preset value Value to which the diameter with 'boDmNokPreset' = TRUE or 'boDmOkPreset' =TRUE is set [mm]
uiID2	UINT	PGT clock [ms]

Input variables of the synchronous part of the program (FPLC_PRG)

Name	Type	Description
diActMotorSpeed	DINT	Actual speed value motor [0.0001 r/min]
diActReelPos	DINT	Actual reel position [increments]
diActWebPos	DINT	Actual web position [increments]

Output variables of the asynchronous part of the program (PLC_PRG)

Name	Type	Description
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled
boDmOk	BOOL	Known diameter
boFindInertiaDone	BOOL	Winder moment determined, tightened web
boWindRuns	BOOL	Wind runs
boSyncRuns	BOOL	Synchronize runs
boDmLimit	BOOL	Diameter was calculated 'ST_WINDER_TORQUE_CONTROL_CONFIG.iDmMin' and 'ST_WINDER_TORQUE_CONTROL_CONFIG.iDmMax', the value was limited

Name	Type	Description
iActDm	INT	Actual diameter [mm]
diActValWebSpeed	DINT	Actual value web speed [mm/s]
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>
<table>
<tr><th>Value</th><th>Meaning</th></tr>
<tr><td>1</td><td>The value at input uiID2 = 0</td></tr>
<tr><td>2</td><td>One of the following parameters has a value <=0:
ST_WINDER_TORQUE_CONTROL_CONFIG.iGearInput,
ST_WINDER_TORQUE_CONTROL_
CONFIG.iGearOutput</td></tr>
<tr><td>3</td><td>One of the following parameters has a value <=0:
ST_WINDER_TORQUE_CONTROL_CONFIG.iDmMin,
ST_WINDER_TORQUE_CONTROL_CONFIG.iDmMax
oder
ST_WINDER_TORQUE_CONTROL_CONFIG.iDmMin >=
ST_WINDER_TORQUE_CONTROL_CONFIG.iDmMax</td></tr>
<tr><td>4</td><td>The parameter has a Value <=0:
ST_WINDER_TORQUE_CONTROL_
CONFIG.diMotIncPerRev</td></tr>
<tr><td>5</td><td>One of the following parameters has a value <=0:
ST_WINDER_TORQUE_CONTROL_
CONFIG.iMesCycDmOk,
ST_WINDER_TORQUE_CONTROL_
CONFIG.iMesCycNoDm,
ST_WINDER_TORQUE_CONTROL_CONFIG.iDmDiffOk</td></tr>
<tr><td>6</td><td>The parameter has a Value <=0:
ST_WINDER_TORQUE_CONTROL_
CONFIG.iSyncTorque</td></tr>
<tr><td>7</td><td>One of the following parameters has a value <=0:
ST_WINDER_TORQUE_CONTROL_CONFIG.iWidth,
ST_WINDER_TORQUE_CONTROL_CONFIG.iDensity</td></tr>
<tr><td>8</td><td>The parameter has a Value <=0:
ST_WINDER_TORQUE_CONTROL_
CONFIG.diFrictionSpeedMax</td></tr>
<tr><td>9</td><td>One of the following parameters has a value <=0:
ST_WINDER_TORQUE_CONTROL_
CONFIG.diWebSpeedMax,
ST_WINDER_TORQUE_CONTROL_
CONFIG.diWebEncLen,
ST_WINDER_TORQUE_CONTROL_
CONFIG.diWebEncIncr,
ST_WINDER_TORQUE_CONTROL_
CONFIG.iWebSpdPeriod,
ST_WINDER_TORQUE_CONTROL_
CONFIG.iMotorSpeedOffsetAbs,
ST_WINDER_TORQUE_CONTROL_
CONFIG.iMotorSpeedOffsetPrcnt</td></tr>
<tr><td>10</td><td>The parameter has a Value <=0:
ST_WINDER_TORQUE_CONTROL_
CONFIG.iRatedTorque</td></tr>
<tr><td>11</td><td>Block WINDER_CALC_INERTIA:
Error in calculatiing of the inertia moment of reels. One of the variables 'iDensity' or 'iWidth' has a value <=0</td></tr>
</table></td></tr>
</table>

Name	Type	Description
		<table><tr><th>Value</th><th>Meaning</th></tr><tr><td>12</td><td>Block WINDER_FRICTION_CALC: One of the input variables 'iRatedTorque' or 'diSpeedMax' has a value <=0</td></tr><tr><td>13</td><td>Block WINDER_WINDER_CT: The maximum permissible torque with the function "Determination the inertia moment of the reel and tightening web" on parameter 'ST_WINDER_TORQUE_CONTROL_CONFIG.iFiMaxTorque' has been achieved.</td></tr><tr><td>14</td><td>Block WINDER_TORQUE_CONTROL: The maximum permissible time with the function "Determination the inertia moment of the reel and tightening web" on parameter 'ST_WINDER_TORQUE_CONTROL_CONFIG.tFiTightTOut' has been exceeded. The reel could not tight the web. Probably the web is ripped.</td></tr><tr><td>15</td><td>One of the following parameters has a value <=0: ST_WINDER_TORQUE_CONTROL_CONFIG.udFiSlackTorque, ST_WINDER_TORQUE_CONTROL_CONFIG.stConfig.udFiSlackVelocity, ST_WINDER_TORQUE_CONTROL_CONFIG.stConfig.udFiTramp, ST_WINDER_TORQUE_CONTROL_CONFIG.stConfig.iFiMaxTorque, ST_WINDER_TORQUE_CONTROL_CONFIG.stConfig.iFiTightSpeed, ST_WINDER_TORQUE_CONTROL_CONFIG.stConfig.iFiSpeedThresholdPerc ST_WINDER_TORQUE_CONTROL_CONFIG.stConfig.iFiSpeed0Perc ST_WINDER_TORQUE_CONTROL_CONFIG.stConfig.iFiTightWnd</td></tr><tr><td>18</td><td>Block WINDER_CHARACTERISTICS: One of the diameter values is smaller than the previous value</td></tr></table>

Output variables of the synchronous part of the program (FPLC_PRG)

Name	Type	Description
diSetValMotorSpeed	DINT	Speed setpoint motor [0.0001 r/min]
iSetValTorqueAbs	INT	Limit torque setpoint [0.1% Mn]
diActMotorSpeedAbs	DINT	Actual motor speed absolute [0.0001 r/min]
diActReelPosAbs	DINT	Actual reel position absolute [increments]

Input and output variables

Name	Type	Description
stDevice	STRUCT	ST_DEVICE The device description structure assigns the block a device. (See document Software description AmkBase Bibliothek, Part no.204986)

Name	Type	Description
stConfig	STRUCT	ST_WINDER_TORQUE_CONTROL_CONFIG Configuration structure of the winder (See document Software description AFL - AMK Function Library, part 2, Part no.203694)

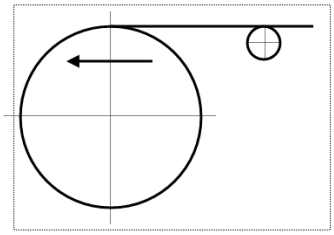
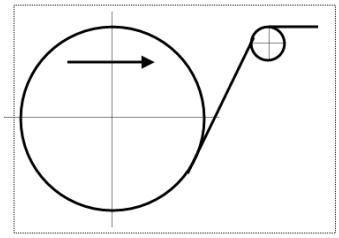
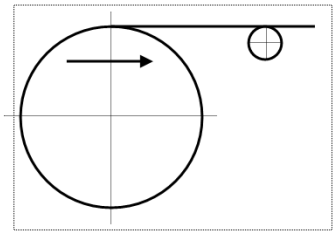
Usage note in the CoDeSys program

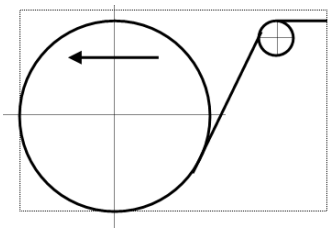
PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
WINDER_TORQUE_CONTROL	WINDER_TORQUE_CONTROL.actSync

ST_WINDER_TORQUE_CONTROL_CONFIG (ST)

Configuration structure

Name	Type	Ex. value	Description
boWndCWise	BOOL	FALSE	FALSE: reel is wound up when the motor rotation direction counterclockwise. TRUE: reel is wound up when the motor rotation direction clockwise.
boUnwinder	BOOL	FALSE	FALSE: winder runs as a rewinder. With positive web speed is rewinded. TRUE: winder runs as unwinder. With a positive web velocity is unwinded.

Arrangement	Type	boWndCWise	boUnwinder
	Rewinder	FALSE	FALSE
	Rewinder	TRUE	FALSE
	Unwinder	FALSE	TRUE

Arrangement	Type	boWndCWise	boUnwinder
	Unwinder	TRUE	TRUE

Name	Type	Ex. value	Description
boMotCcWise	BOOL	FALSE	Mounting position of the motor counterclockwise
iGearInput	INT	1	Gear input multiplier winder drive
iGearOutput	INT	1	Gear output divider winder drive
iDmMin	INT	100	Minimum permissible diameter value [mm]
iDmMax	INT	600	Maximum permissible diameter value [mm]
diMotIncPerRev	DINT		Increments per motor revolution (ID 116 'Resolution motor encoder')
iMesCycDmOk	INT	10	Reel revolution interval for cycle Calculating diameter if diameter known [0.1 revolutions]
iMesCycNoDm	INT	1	Reel revolution interval for cycle Calculating diameter if diameter not known [0.1 revolutions]
iDmDiffOk	INT	20	Maximum difference between two successively calculated diameters, so that the review "diameter valid" is accepted [mm]

Reel synchronization

Name	Type	Ex. value	Description
udSyncTramp	UDINT	40000	Slope of the synchronization ramp [ms]
iSyncTorque	INT	1000	Torque during synchronization [0.1% Mn]

Parameters for calculating the inertia moment of the reel

Name	Type	Ex. value	Description
iWidth	INT		web width [mm]
iDensity	INT	800	web density [kg/m^3]

Parameters for the friction compensation

Name	Type	Ex. value	Description
diFrictionTorque0	DINT		Friction torque at speed 0 [0.1 Nm]
diFrictionSpeedMax	DINT		Speed at the friction torque 'diFrictionTorqueMax' was determined [0.1 r/min]
diFrictionTorqueMax	DINT		Max. friction torque at speed 'diFrictionSpeedMax' [0.1 Nm]
uiFrcSetSpeedLimitPos	UINT	50	Positive hysteresis limit for the calculation of the friction torque [0.1 r/min]
uiFrcSetSpeedLimitNeg	UINT	50	Negative hysteresis limit for the calculation of the friction torque [0.1 r/min]

Speed parameter

Name	Type	Ex. value	Description
diWebSpeedMax	DINT	20000	Maximum web speed [mm/s]
diWebEncLen	DINT	1000	Distance covered web track when changing the input 'diInVal' to 'diWebEncIncr' increments [mm]
diWebEncIncr	DINT		Number of counted increments at input 'diInVal' with move on the track 'diWebEncLen' [increments]
iWebSpdPeriod	INT	5	Number of calls cycles ID2 'SERCOS cycle time' for measurement of the web speed
iMotorSpeedOffsetAbs	INT	200	Offset speed absolute [0.1 r/min]
iMotorSpeedOffsetPrcnt	INT	20	Offset speed percentage [%]
iRatedTorque	INT		Motor nominal torque [0.1 Nm] ID32771 'Nominal torque'

Friction moment parameter

Name	Type	Ex. value	Description
udFiSlackTorque	UDINT	1000	Setpoint motor torque during the determination of the reels inertia torque [0.1% Mn]
udFiSlackVelocity	UDINT	50	Speed setpoint to push back the reel to determine the reels inertia torque [0.1 U / min]
udFiTramp	UDINT	1000	Acceleration ramp for the return rotation of the reel [ms]
tFiTimeStep	TIME	500	Time interval to increase torque while determining the reels inertia torque [ms]
iFiTorqueStep	INT	50	Torque step increase during the determination of the reels inertia torque [0.1% Mn]
iFiMaxTorque	INT	1000	Maximum torque for the determination of the reels inertia torque [0.1% Mn]
iFiTightSpeed	INT	50	Speed setpoint for tightening the web [0.1 r/min]
iFiSpeedThresholdPerc	INT	90	Speed threshold for tightening the web [% iFiTightSpeed]
iFiSpeed0Perc	INT	60	Threshold speed = 0 for tightening the web [% iFiTightSpeed]
tFiTightTout	TIME	120	Timeout when tightening the web [s]
iFiTightWnd	INT	20	Position window for tightening the web when reel stands and reel position in the window, the web is considered tightened [1/r]

Tuning parameter

Name	Type	Ex. value	Description
diTimeDmRamp	DINT	2000	Ramp time for diameter output [ms] There is a sharp change in diameter at the input (eg setting a preset value), the diameter is adjusted linearly in the set ramp time.

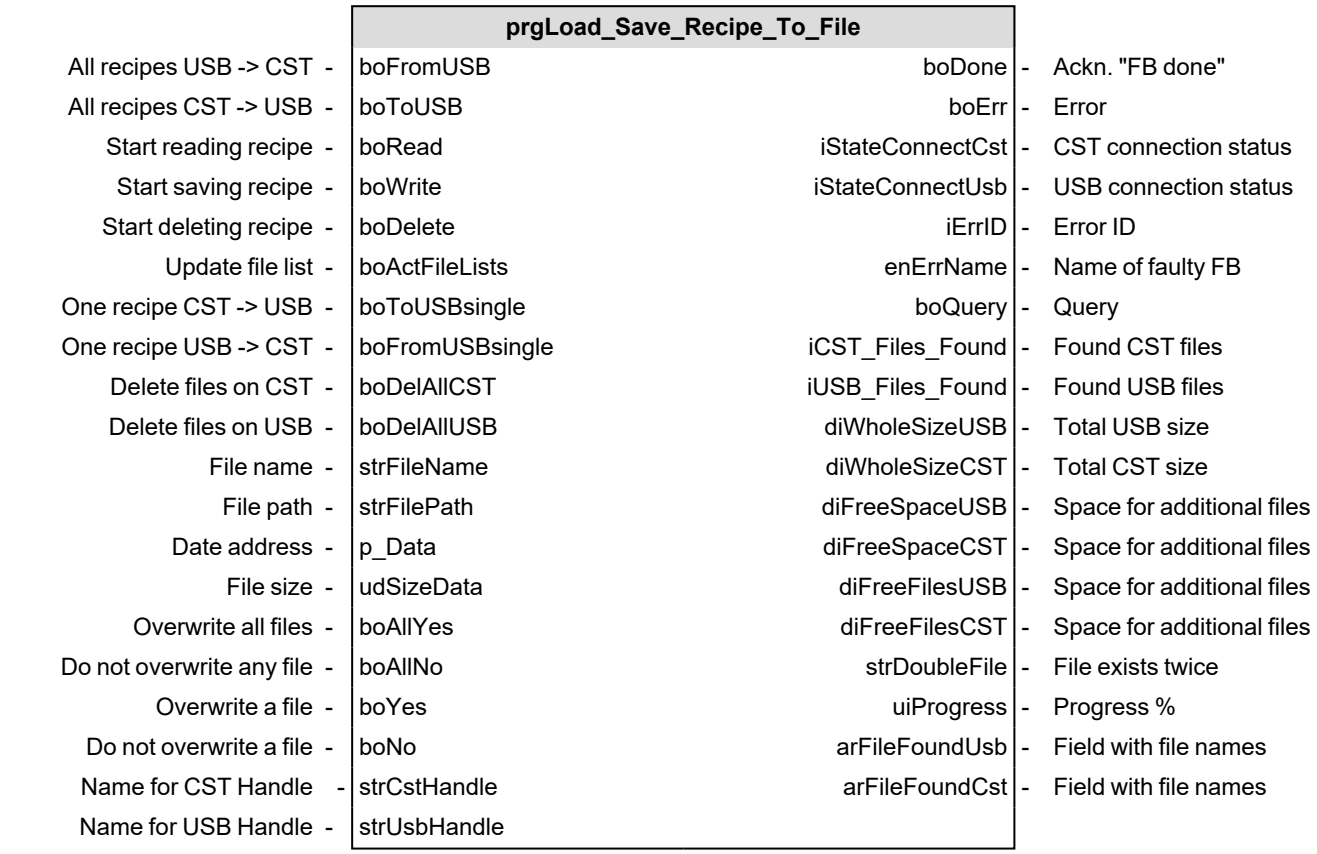
4.2.1.2.4 04_Recipe

4.2.1.2.4.1 prgLoad_Save_Recipe_To_File (PRG)

Via the inputs of the program 'prgLoad_Save_Recipe_To_File', recipe data can be stored, read and/or copied to USB. In addition, the saved recipe data can be deleted again.

The program is called in the asynchronous program level PLC_PRG.

User interface



Input variables

Name	Type	Description
boFromUSB	BOOL	Copy all recipe files from USB to CST
boToUSB	BOOL	Copy all recipe files from CST to USB
boRead	BOOL	Read out selected file ('strFileName') and recipe (only CST)
boWrite	BOOL	Save recipe in selected file ('strFileName') (only CST)
boDelete	BOOL	Delete selected file ('strFileName') (only CST)
boActFileLists	BOOL	Update file lists ('arFileFoundUsb' and 'arFileFoundCst')
boToUSBsingle	BOOL	Copy selected ('strFileName') recipe file to USB
boFromUSBsingle	BOOL	Copy selected ('strFileName') recipe file to CST
boDelAllCST	BOOL	Delete all recipe files from CST
boDelAllUSB	BOOL	Delete all recipe files from USB
strFileName	STRING	File name, excluding extension
strFilePath	STRING	Path (USB and CST), to which the recipe files will be stored (‘/’ as separator)

Name	Type	Description
p_Data	POINTER	POINTER TO BYTE Pointer to the recipe structure; data that will be stored or loaded
udSizeData	UDINT	Data size (p_Data) [byte]
boAllYes	BOOL	If query is made via 'boQuery': Overwrite all double files
boAllNo	BOOL	If query is made via 'boQuery': Do not overwrite all double files
boYes	BOOL	If query is made via 'boQuery': Overwrite the double files ('strDoubleFile')
boNo	BOOL	If query is made via 'boQuery': Do not overwrite the double files ('strDoubleFile')
strCstHandle	STRING	Name for CST-Handle Default: strCstHandle = 'CST_1'
strUsbHandle	STRING	Name for USB-Handle **Controller** / **SW-version** / **strUsbHandle** A4 / ≥ A4_406_1151_203984 / 'USB11_1' A5 / < A5_410_ / 'USB21_2' A5 / ≥ A5_410_ / 'USB11_1' ASC / ≥ AS_CP_200_0812_202053 / 'USB11_2' Default: strUsbHandle = 'USB21_2'

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
iStateConnectCst	INT	Connection status to CST 0: Connection successfully established 1: Incorrect logical drive 2: Drive missing (e.g. USB stick in not plugged in) 3: Incorrect physical address (e.g. for EXT drive without physical address) 4: Unable to successfully establish connection 5: Connection already exists; when attempted again, unable to re-establish the connection
iStateConnectUsb	INT	Connection status to USB stick 0: Connection successfully established 1: Incorrect logical drive 2: Drive missing (e.g. USB stick in not plugged in) 3: Incorrect physical address (e.g. for EXT drive without physical address) 4: Unable to successfully establish connection 5: Connection already exists; when attempted again, unable to re-establish the connection

<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>enErrName = NONE:<table><tr><td>1</td><td>Not enough storage space</td></tr></table>enErrName ≠ NONE:
Error number, s. block</td></tr>
<tr><td>enErrName</td><td>ENUM</td><td>EN_FB_NAME
Name of faulty block for which the diagnostic number is output.
The diagnostic number is explained in the description of the corresponding library.<table><tr><th>Block</th><th>Library</th></tr><tr><td>WRITE_FILE_1</td><td>AmkFile.lib</td></tr><tr><td>READ_FILE_1</td><td>AmkFile.lib</td></tr><tr><td>FIND_FILE_1</td><td>AmkFile.lib</td></tr><tr><td>REMOVE_FILE_1</td><td>AmkFile.lib</td></tr><tr><td>CREATE_DIR_1</td><td>AmkFile.lib</td></tr><tr><td>FILE_FIND</td><td>AmkAfl.Lib</td></tr></table></td></tr>
<tr><td>boQuery</td><td>BOOL</td><td>boQUERY = TRUE: Query, whether the file ('strDoubleFile') should be overwritten.
Must be answered with 'boYes', 'boNo', 'boAllYes' or 'boAllNo'.</td></tr>
<tr><td>iCST_Files_Found</td><td>INT</td><td>Number of recipe files found on CST</td></tr>
<tr><td>iUSB_Files_Found</td><td>INT</td><td>Number of recipe files found on USB</td></tr>
<tr><td>diWholeSizeUSB</td><td>DINT</td><td>Total size of recipe files on USB [byte]</td></tr>
<tr><td>diWholeSizeCST</td><td>DINT</td><td>Total size of recipe files on CST [byte]</td></tr>
<tr><td>diFreeSpaceUSB</td><td>DINT</td><td>Free space on USB [kByte]</td></tr>
<tr><td>diFreeSpaceCST</td><td>DINT</td><td>Free space on CST [kByte]</td></tr>
<tr><td>diFreeFilesUSB</td><td>DINT</td><td>Number of files, that still fit on USB (with 1 kByte clusters)</td></tr>
<tr><td>diFreeFilesCST</td><td>DINT</td><td>Number of files, that still fit on CST (with 1 kByte clusters)</td></tr>
<tr><td>strDoubleFile</td><td>STRING</td><td>Name of the recipe file that already exists and should be transferred</td></tr>
<tr><td>uiProgress</td><td>UINT</td><td>Progress: Shows the progress as per cent</td></tr>
<tr><td>arFileFoundUsb</td><td>ARRAY</td><td>ARRAY[0..MAX_LEN_ST_FILE] OF ST_FILE
Field with all found recipe files on USB</td></tr>
<tr><td>arFileFoundCst</td><td>ARRAY</td><td>ARRAY[0..MAX_LEN_ST_FILE] OF ST_FILE
Field with all found recipe files on CST</td></tr>
</table>

Usage note in the CoDeSys program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
prgLoad_Save_Recipe_To_File	

It is recommended that you call blocks that work with file access in a separate task in the asynchronous program level PLC_PRG (type "asynchronous").

History

Library	AmkAfl	
Target	**Functionality**	**Firmware (PLC controller)**
A4 V3.01/1128	Basic functionality (1st version)	≥ A4_406_1151_203984
A5 V4.01/1026	Basic functionality (1st version)	≥ A5_V301_1018

4.2.1.2.5 05_Alarm

4.2.1.2.5.1 Alarm table

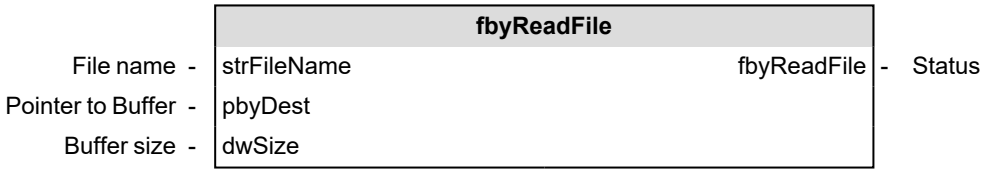
The blocks and visualisations are used as alarm processing on AMK controllers.
To realise the functionality, the program 'PRG_ERR_HANDLER' must be called in a separate asynchronous task.
The function block 'FB_READ_ERR_HISTORIE' is used to read out history files.
The visualisation 'VI_ALARMTAB' shows the current errors that have been added by 'fboSetErr'.
The previous errors can be viewed in the visualisation 'VI_HISTORIE'.

4.2.1.2.5.2 FILE_MANAGER

fbyReadFile (F)

The function 'fbyReadFile' reads the content of a file. It then returns the status.
Different from the two function blocks 'READ_FILE' and 'READ_FILE_1' of the AmkFile.lib, here the SysLibFile.lib is used.
The function is called in the asynchronous program level PLC_PRG.

User interface



Input variables

Name	Type	Description
strFileName	STRING	File name, including extension to be read
pbyDest	POINTER	POINTER TO BYTE Pointer to the buffer in which the read data is to be stored
dwSize	DWORD	Buffer size [byte]

Output variable

Name	Type	Description
fbyReadFile	BYTE	Status 0 Undefined state 1 File was read successfully 10 Incorrect input parameters 11 Error while opening file 14 Error while closing the file 15 Error while reading

Usage note in the CoDeSys program

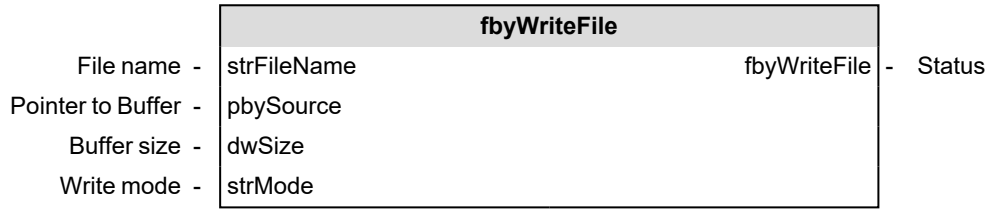
PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
fbyReadFile	

It is recommended that you call blocks that work with file access in a separate task in the asynchronous program level PLC_PRG (type "asynchronous").

fbyWriteFile (F)

The function 'fbyWriteFile' writes to a file and returns the state.
Different from the two function blocks 'WRITE_FILE' and 'WRITE_FILE_1' of the AmkFile.lib, here the SysLibFile.lib is used.
The function is called in the asynchronous program level PLC_PRG.

User interface



Input variables

Name	Type	Description
strFileName	STRING	File name, including extension to be written
pbySource	POINTER	POINTER TO BYTE Pointer to the buffer whose contents will be written to the file
dwSize	DWORD	Specifies the size of the data memory [byte]
strMode	STRING(2)	Access mode of the file Mode in which the file should be processed: w – write – File is overwritten or created new r – read – File is only opened for reading; if the file does not exist, an error is returned. rw – read and write – File is overwritten; if the file does not exist, an error is returned a – append – File is opened as for 'w', but writing is appended at the end of the file

Output variable

Name	Type	Description
fbyWriteFile	BYTE	Status 0 – Undefined state 1 – File was written successfully 10 – Incorrect input parameters 11 – Error while opening file 13 – Error while writing the file 14 – Error while closing the file

Usage note in the CoDeSys program

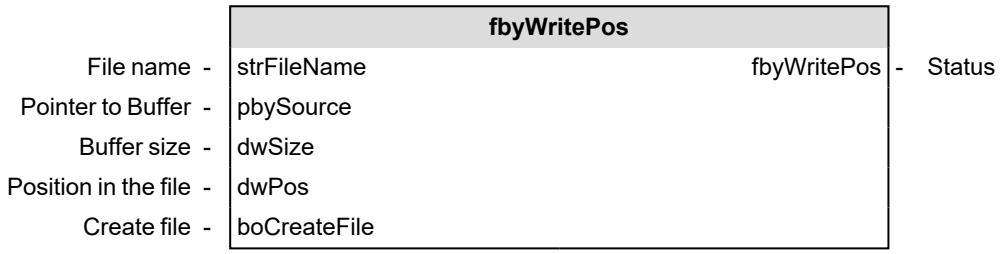
PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
fbyWriteFile	

It is recommended that you call blocks that work with file access in a separate task in the asynchronous program level PLC_PRG (type "asynchronous").

fbyWritePos (F)

This function writes to a file at the desired position and returns the state.
Different from the two function blocks 'WRITE_FILE' and 'WRITE_FILE_1' of the AmkFile.lib, here the SysLibFile.lib is used.
The function is called in the asynchronous program level PLC_PRG.

User interface



Input variables

Name	Type	Description
strFileName	STRING	File name, including extension to be written
pbySource	POINTER	POINTER TO BYTE Pointer to the buffer whose contents will be written to the file
dwSize	DWORD	Specifies the size of the data memory [byte]
dwPos	DWORD	Position in the file where writing should start
boCreateFile	BOOL	TRUE: File is created new FALSE: It is written in the existing file

Output variable

Name	Type	Description
fbyWritePos	BYTE	Status 0 – Undefined state 1 – File was written successfully 10 – Incorrect input parameters 11 – Error while opening file 12 – Error during changing the offset for the file access 13 – Error while writing the file 14 – Error while closing the file 16 – Error when setting the position: Current value ≠ specification

Usage note in the CoDeSys program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
fbyWritePos	

4.2.1.2.5.3 Support

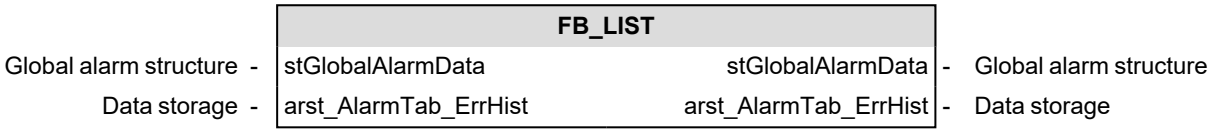
FB_LIST (FB)

The function block 'FB_LIST' is called in the program 'PRG_ERR_HANDLER'. The block is required for the internally created tables of this library. It realises index switching and the corresponding copying of the data.

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

'Support' functions and function blocks will not be used directly. They are called as subroutines by other functions or function blocks.

User interface



Input and output variables

Name	Type	Description
stGlobalAlarmData	STRUCT	ST_GLOBAL_ALARM_DATA Global alarm structure
arst_AlarmTab_ErrHist	ARRAY	ST_ERR_ANZ Structure in which the file content is written (g_arst_AlarmTab_ErrHist)

Usage note in the CoDeSys program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
FB_LIST	

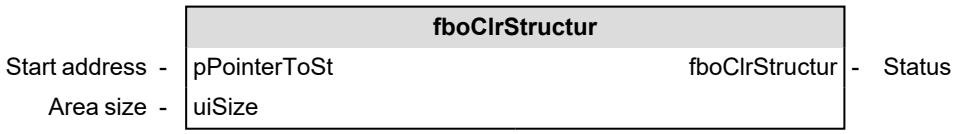
fboClrStructur (F)

The function 'fboClrStructur' overwrites the specified memory area completely with 0, i.e. the area is deleted.

The function is called in the asynchronous program level PLC_PRG.

'Support' functions and function blocks will not be used directly. They are called as subroutines by other functions or function blocks.

User interface



Input variables

Name	Type	Description
pPointerToSt	POINTER	POINTER TO BYTE Start address of the area to be deleted.
uiSize	UINT	Size of the area to be deleted [byte]

Output variable

Name	Type	Description
fboClrStructur	BOOL	Status

Usage note in the CoDeSys program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
fboClrStructur	

fstrGetDateStr (F)

The function 'fstrGetDateStr' convert an integer to a string variable

The function is called in the asynchronous program level PLC_PRG.

'Support' functions and function blocks will not be used directly. They are called as subroutines by other functions or function blocks.

User interface

	fstrGetDateStr		
Input value -	siZahl	fstrGetDateStr	- Output value

Input variables

Name	Type	Description
siZahl	SINT	SINT(-99..99) Input variable, integer, e.g. 37

Output variable

Name	Type	Description
fstrGetDateStr	STRING (2)	Output variable string, e.g. '37'

Usage note in the CoDeSys program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
fstrGetDateStr	

4.2.1.2.5.4 FB_READ_ERR_HISTORIE (FB)

With the function block 'FB_READ_ERR_HISTORIE', the saved history date can also be read out. They are stored n the global structure 'g_arst_AlarmTab_ErrHist'.

With the visualisation 'VI_HISTORIE', the data are displayed accordingly.

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

User interface

	FB_READ_ERR_HISTORIE		
FB execution -	boExec	boErr	- Error
Day -	usiDay	iErrID	- Error ID
Month -	usiMonth	enErrName	- Name of faulty FB
Year -	uiYear	boDone	- Ackn. "FB done"
File number -	usiFileNr	uiCount	- Number of errors read
File name -	strFileNameIn		
Data storage -	arstErr	arstErr	- Data storage
Global alarm structure -	stGlobalAlarmData	stGlobalAlarmData	- Global alarm structure

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.
usiDay	USINT	Day, used for file selection (strFileNameIn='0' and PRG_ERR_HANDLER.enSaveMode ≠ enSaveErrRingBuffer)
usiMonth	USINT	Month, used for file selection (strFileNameIn='0' and PRG_ERR_HANDLER.enSaveMode ≠ enSaveErrRingBuffer)
uiYear	UINT	Year, used for file selection (strFileNameIn='0' and PRG_ERR_HANDLER.enSaveMode ≠ enSaveErrRingBuffer)
usiFileNr	USINT	File number, used for data selection (strFileNameIn='0' and PRG_ERR_HANDLER.enSaveMode ≠ enSaveErrRingBuffer and PRG_ERR_HANDLER.enSaveMode= enSaveErrSequential)
strFileNameIn	STRING (40)	String that contains the file names

- Either you transfer to the function block the corresponding data (month, year, day and file number) to read out a history file (*.err). In this case you should transfer the input variables strFileNameIn:='0'. Or you transfer the entire file names of the variable 'strFileNameIn' (e.e. '07042008_0.err'). Here, the transferred date values are ignored.
- If PRG_ERR_HANDLER.enSaveMode = enSaveErrRingBuffer, then no value or '0' should be transferred to 'strFileNameIn'.

Output variables

<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></td></tr>
<tr><td>enErrName</td><td>ENUM</td><td>EN_FB_NAME
Name of faulty block for which the diagnostic number is output.
The diagnostic number is explained in the description of the corresponding library.
<table><tr><th>Block</th><th>Library</th></tr><tr><td>FIND_FILE_1</td><td>AmkFile.lib</td></tr><tr><td>READ_FILE_1</td><td>AmkFile.lib</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>uiCount</td><td>UINT</td><td>Number of errors read</td></tr>
</table>

Input and output variables

Name	Type	Description
arstErr	ARRAY	ST_ERR_ANZ Structure in which the file content is written (g_arst_AlarmTab_ErrHist)
stGlobalAlarmData	STRUCT	ST_GLOBAL_ALARM_DATA Global alarm structure

Usage note in the CoDeSys program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
FB_READ_ERR_HISTORIE	

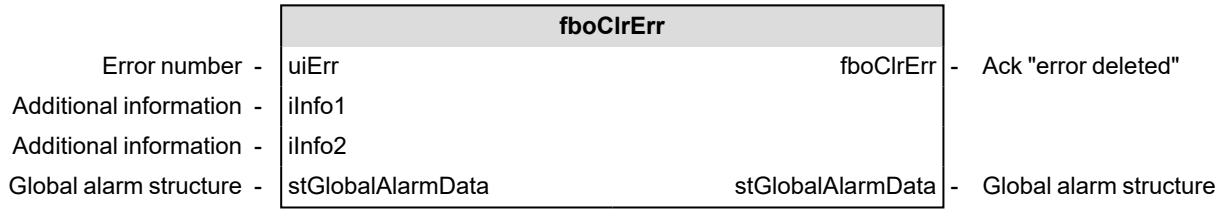
4.2.1.2.5.5 fboClrErr (F)

The error that is transferred to the function 'fboClrErr' with its additional info is deleted from the current alarm table, but not from the history.

To do so, the entire field is searched for the error. If the error does not exist, nothing is deleted, but TRUE is responded nonetheless. If the error to be deleted is found, it is deleted from the table and subsequent errors in the table move up. The view is then updated.

The function is called in the asynchronous program level PLC_PRG.

User interface



Input variables

Name	Type	Description
uiErr	UINT	Error number
iInfo1	INT	Error additional information 1
iInfo1	INT	Error additional information 2

Output variable

Name	Type	Description
fboClrErr	BOOL	Acknowledgement: Error deleted TRUE: Error deleted from the table FALSE: Buffer blocked

Input and output variable

Name	Type	Description
stGlobalAlarmData	STRUCT	ST_GLOBAL_ALARM_DATA Global alarm structure

Usage note in the CoDeSys program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
fboClrErr	

4.2.1.2.5.6 fboSetErr (F)

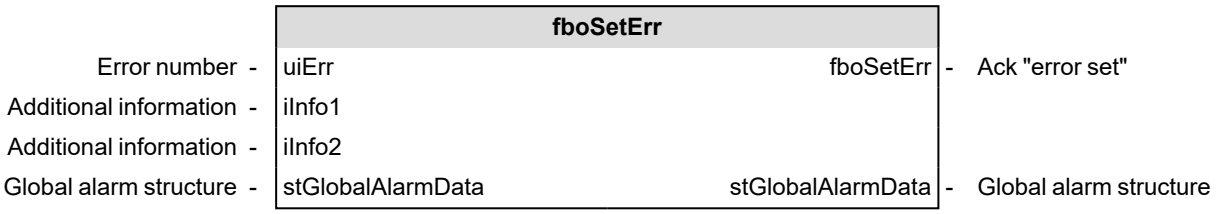
The error that is transferred to the function 'fboSetErr' is set and written in the global structure g_stGlobalAlarmData at the next free position.

If the error could not be set, the function returns FALSE. In this case, the function should be called repeatedly until it returns TRUE.

The function returns TRUE when the error is set and has been written to the FIFO. If the alarm table already contains the current error (with additional info 1+2), the error is not set again, but the function returns the value TRUE.

The function is called in the asynchronous program level PLC_PRG.

User interface



Input variables

Name	Type	Description
uiErr	UINT	Error number
iInfo1	INT	Error additional information 1
iInfo1	INT	Error additional information 2

Output variable

Name	Type	Description
fboSetErr	BOOL	Acknowledgement: Error set TRUE: Error written in the table FALSE: Buffer blocked or full

Input and output variable

Name	Type	Description
stGlobalAlarmData	STRUCT	ST_GLOBAL_ALARM_DATA Global alarm structure

Usage note in the CoDeSys program

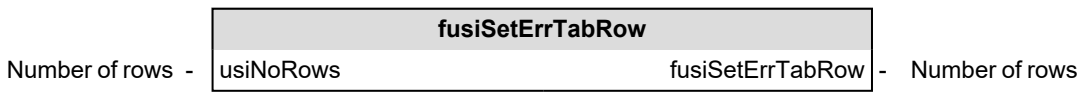
PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
fboSetErr	

4.2.1.2.5.7 fusiSetErrTabRow (F)

The function 'fusiSetErrTabRow' specifies how many rows the alarm tables in the visualisation contains.

The function is called in the asynchronous program level PLC_PRG.

User interface



Input variables

Name	Type	Description
usiNoRows	USINT	USINT(3..15)Number of lines in 'VI_ALARMTAB'

Output variable

Name	Type	Description
fusiSetErrTabRow	USINT	Number of lines in 'VI_ALARMTAB'

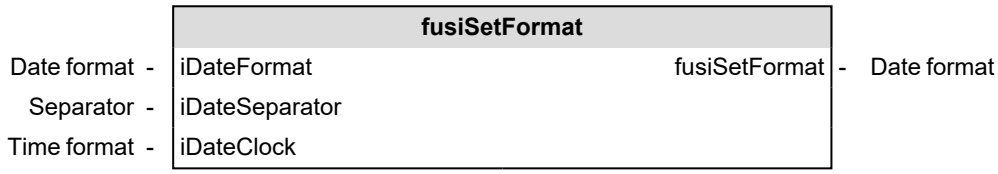
Usage note in the CoDeSys program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
fusiSetErrTabRow	

4.2.1.2.5.8 fusiSetFormat (F)

The function 'fusiSetFormat' sets the time and date format.
The function is called in the asynchronous program level PLC_PRG.

User interface



Input variables

Name	Type	Description
iDateFormat	INT	INT(0..2) 0 enDD_MM_YYYY 1 enYYYY_MM_DD 2 enMM_DD_YYYY
iDateSeparator	INT	INT(3..5) 3 enSeperatot_Point [.] 4 enSeperatot_Hyphen [-] 5 enSeperatot_Slash [/])
iDateClock	INT	INT(6..7) 6 enClock_24_Hours 7 enClock_12_Hours

Output variable

Name	Type	Description
fusiSetFormat	USINT	Date format

Usage note in the CoDeSys program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
fusiSetFormat	

4.2.1.2.5.9 fusiSetHistTabRow (F)

The function 'fusiSetHistTabRow' specifies how many rows the history table in the visualisation contains.
The function is called in the asynchronous program level PLC_PRG.

User interface

Input variables

Name	Type	Description
usiNoRows	USINT	USINT(3..15)Number of lines in 'VI_HISTORIE'

Output variable

Name	Type	Description
fusiSetHistTabRow	USINT	Number of lines in 'VI_HISTORIE'

Usage note in the CoDeSys program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
fusiSetHistTabRow	

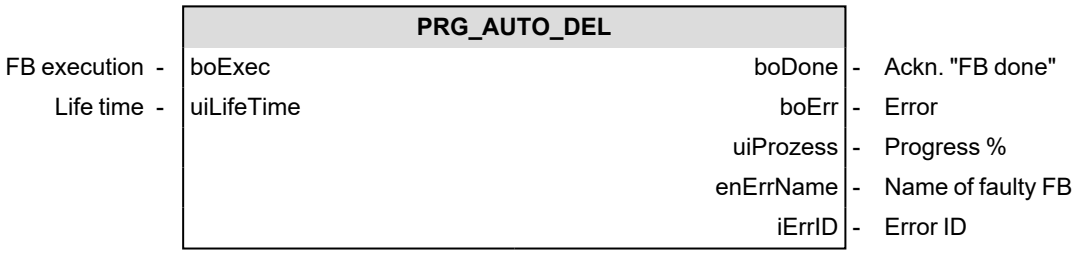
4.2.1.2.5.10 PRG_AUTO_DEL (PRG)

The program 'PRG_AUTO_DEL' is called in the 'FB_ERR_HANDLER'. It is used for automatic deletion of history files that are older than the set time. For this purpose the current date is compared with the date of the existing history files (except for the ring buffer file).

The user does not need to explicitly call or control this function block, as it occurs in the 'FB_ERR_HANDLER'. The life time of the history files can be set using the input variable 'uiLifeTimeHist' of the 'PRG_ERR_HANDLER'.

The program is called in the asynchronous program level PLC_PRG.

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.
uiLifeTime	UINT	Life time of the history files [days]

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
uiProzess	UINT	Progress (relative to the number of found *.err files) [%]

Name	Type	Description
enErrName	ENUM	EN_FB_NAME Name of faulty block for which the diagnostic number is output. The diagnostic number is explained in the description of the corresponding library. **Block** / **Library** FILE_FIND / AmkAfl.lib FILE_DELETE / AmkAfl.lib
iErrID	INT	Error identity number: Diagnostic number is output iErrID = 0 — No error iErrID ≠ 0 — boErr = TRUE — Error iErrID ≠ 0 — boErr = FALSE — Warning

Usage note in the CoDeSys program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
PRG_AUTO_DEL	

4.2.1.2.5.11 PRG_ERR_HANDLER (PRG)

The program process everything required for the alarm table and its history.

This program monitors the global buffer that is filled by the function 'fboSetErr'. If the first position of the FIFO contains an error element, it is appended to the current history file. In addition, the program calls the program 'PRG_AUTO_DEL'.

The program is called in the asynchronous program level PLC_PRG.
The program 'PRG_ERR_HANDLER' must be called in a separate task.

User interface

	PRG_ERR_HANDLER			
Delete alarm table -	bo_AlarmTab_ClrAns	boErr	-	Error
Scrollbar in visu -	boScrollbar	iErrID	-	Error ID
"?" docu. button -	boDoku	enErrName	-	Name of faulty FB
Life time history -	uiLifeTimeHist	boDone	-	Ackn. "FB done"
History storage mode -	enSaveMode	boClrAnsAckn	-	Ack "Alarm tab. deleted"

Input variables

Name	Type	Description
bo_AlarmTab_ClrAns	BOOL	With a positive edge, the alarm table is deleted
boScrollbar	BOOL	With this variable, the scrollbar can be turned on and off (also during the run time)
boDoku	BOOL	Using this input, the documentation button "?" can be turned on or off
uiLifeTimeHist	UINT	Life time of the history files (if enSaveMode ≠ enSaveErrRingBuffer)

Name	Type	Description
enSaveMode	ENUM	EN_SAVE_MODE Specifies how the errors should be saved 0: enSaveErrSequential • Errors are stored sequentially in the file. If this file is full, a new file is created. • For each day, at least one file is created (if an error occurs) 1: enSaveErrDayRingBuffer • Errors are stored sequentially in the file. If this file is full, it is overwritten from the beginning. -> Ring buffer • For each day, one file is created (if an error occurs) 2: enSaveErrRingBuffer • Errors are stored sequentially in the file. If this file is full, it is overwritten from the beginning. -> Ring buffer • Only this one file exists, which is not deleted.

Output variables

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
enErrName	ENUM	EN_FB_NAME Name of faulty block for which the diagnostic number is output. The diagnostic number is explained in the description of the corresponding library. FIND_FILE_1 (AmkFile.lib) WRITE_FILE_1 (AmkFile.lib)
boDone	BOOL	Response that the function block has been completely executed.
boClrAnsAckn	BOOL	Response that the alarm table has been deleted

Usage note in the CoDeSys program

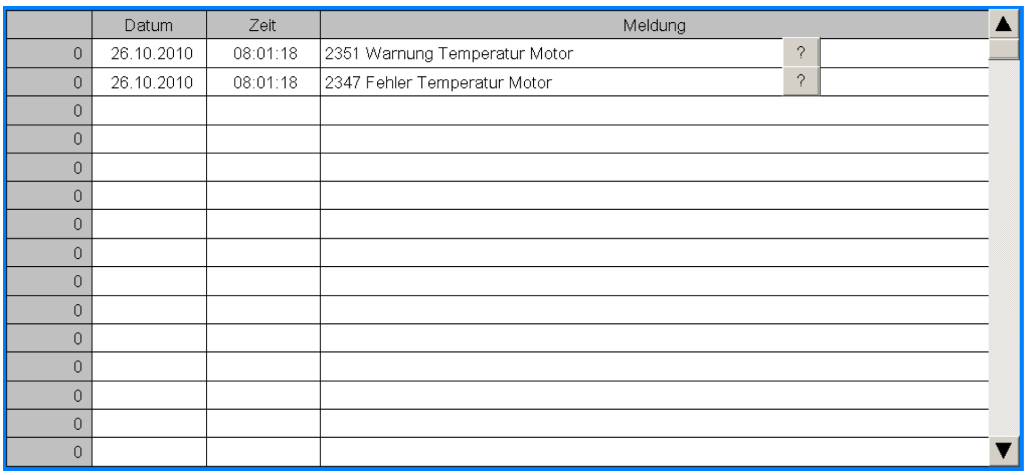
PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
PRG_ERR_HANDLER	

4.2.1.2.5.12 Visualization

VI_ALARMTAB (VI)

This visualisation can be integrated in any mask. It displays all current errors.

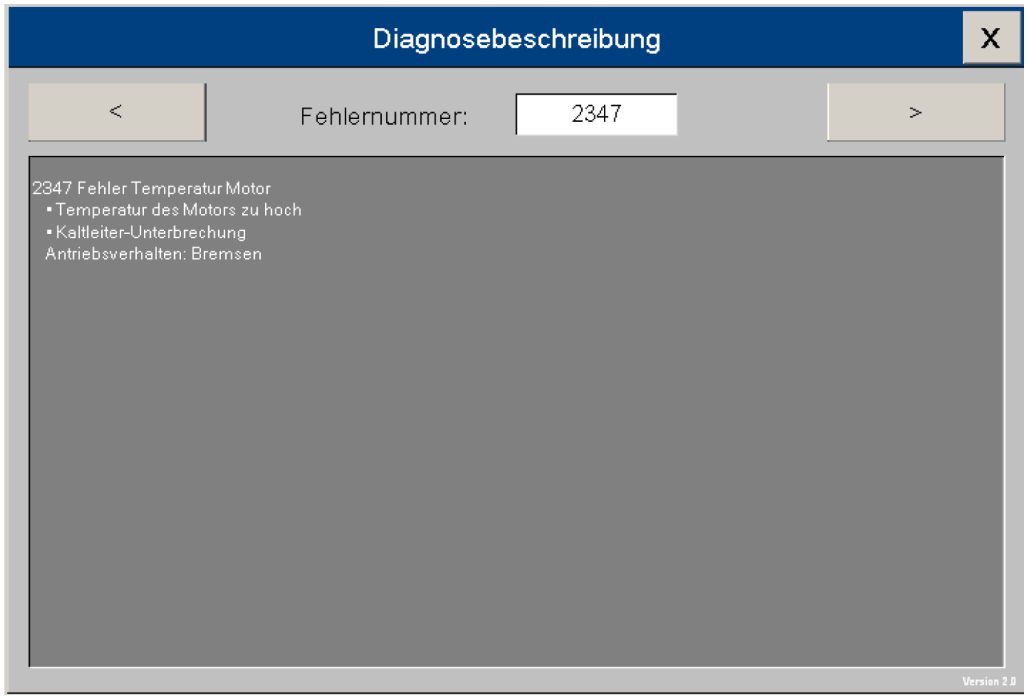
- The number of its rows (3.. 15) can be modified using 'fusiSetErrTabRow'.
- Using the function 'fusiSetFormat' the time and date format can be modified.
- Using the global variable 'g_stVisuAlarmTab' from the structure 'VisualObjectType', the size of the table can be modified.
- Using the variable 'g_stGlobalAlarmData.iAlarmFontHeight', the font height of the table can be changed.



- Using the ? button the user can access the mask VI_DOKU_ERR.
  Using PRG_ERR_HANDLER.boDoku the settings is made whether the button should appear for an error
- The scrollbar can be turned on and off using PRG_ERR_HANDLER.boScrollbar
- Using dynamic language switching, error texts in the table can be displayed
- The prefix is "Error", through which ID is specified
  Example:
  <text prefix="Error" id="2347">
  <en><![CDATA[2347 Error Temperature Motor]] ></en><text>
- With the constant gc_uiElementeAlarmTab, the table length can be set
  The value of this constant should only overshadowed between 15 and 4900

VI_DOKU_ERR (VI)

This visualisation should be seen as its own mask, which is displayed when the user presses the ? button in the alarm or history table.



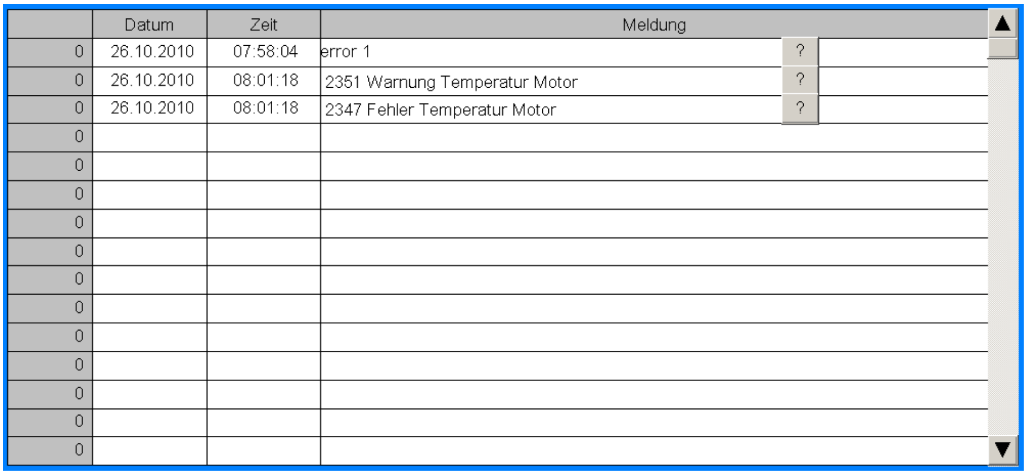
- Using the X button, the user returns to the previous mask.
- Using dynamic language switching, error texts in the table can be displayed

- The prefix is DriveErr, through which ID is specified
  Example:
  <text prefix="DriveErr" id="2347">
  <en><![CDATA[2347 Error Temperature Motor
  - Temperature of the motor is too high
  - PTC thermistor interruption
  - Drive behaviour: Brake]] ></en></text>

VI_HISTORIE (VI)

This visualisation can be integrated in any mask. Using 'FB_READ_ERR_HISTORIE', it shows the read errors of the history.

- Its number of rows (3.. 15) can be modified using 'fusiSetHistTabRow'.
- Using the function 'fusiSetFormat' the time and date format can be modified.
- Using the global variable 'g_stVisuHistoryTab' from the structure 'VisualObjectType', the size of the table can be modified.
- Using the variable 'g_stGlobalAlarmData.iHistoryFontHeight', the font height of the table can be changed.



- Using the ? button the user can access the mask 'VI_DOKU_ERR'
  Using 'PRG_ERR_HANDLER.boDoku' the settings is made whether the button should appear for an error
- The scrollbar can be turned on and off using 'PRG_ERR_HANDLER.boScrollbar'
- Using dynamic language switching, error texts in the table can be displayed
- The prefix is "Error", through which ID is specified
  Example:
  <text prefix="Error" id="2347">
  <en><![CDATA[2347 Error Temperature Motor]] ></en><text>

4.2.1.2.6 06_Commands

4.2.1.2.6.1 CMD_HOME (FB)

The function block 'CMD_HOME' commands a homing drive of an AMK drive.

The parameters from the parameter set of the axis are valid. The homing parameters are explained in the parameter description, Positioning Parameter chapter.

The homing drive can be aborted by withdrawing the execution bit.

After a successful homing cycle, the position feedback value system of the drive and the position setpoint system of the controller are set to the same value.

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

User interface

	CMD_HOME			
FB execution -	boExec	boDone	-	Ackn. "FB done"
Actual position value -	diActPosition	boBusy	-	FB in progress
		boErr	-	Error
		iErrID	-	Error ID
		enErrName	-	Name of faulty FB
Position setpoint -	diSetPosition	diSetPosition	-	Position setpoint
Device structure -	stDevice	stDevice	-	Device structure

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.
diActPosition	DINT	Actual position feedback value [increments]

Output variables

Name	Type	Description
boDone	BOOL	Response that the function block has been completely executed.
boBusy	BOOL	Execution message: This bit remains set as long as the block is being processed.
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
enErrName	ENUM	EN_FB_NAME Name of faulty block for which the diagnostic number is output. The diagnostic number is explained in the description of the corresponding library. Block: DO_CMD_ONCE — Library: AmkDevAccess.lib

Input and output variables

Name	Type	Description
diSetPosition	DINT	Specification of the position setpoint (position setpoint system) [increments]
stDevice	STRUCT	ST_DEVICE The device description structure assigns the block a device. (See document Software description AmkBase Bibliothek, Part no.204986)

Usage note in the CoDeSys program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
CMD_HOME	

Associated with this function block, a visualisation is prepared in CoDeSys.
Siehe 'Visualization of AFL blocks' auf Seite 806.

4.2.1.2.7 07_NC Motion

4.2.1.2.7.1 NC_MOTION

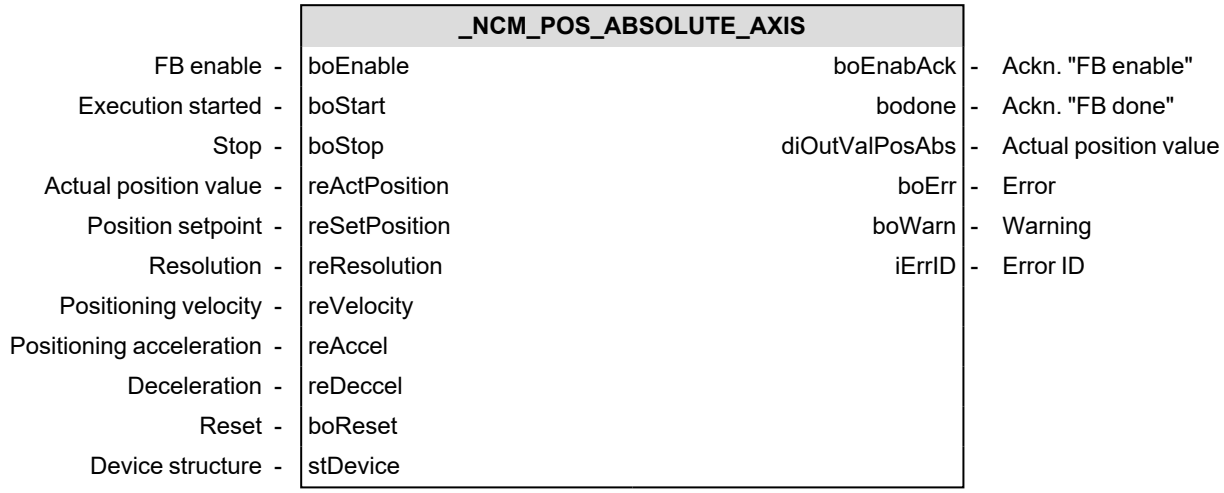
- Line interpolation with up to 16 axes
- Circle interpolation cw/ccw with axes 1/2 ,1/3, 2/3, programmable in 3 planes
- Possibility of circle interpolation (axes 1-3) and simultaneous line interpolation
- Programmable delay time between two NC blocks
- Programmable stop (after NC block)
- Immediate stop (during NC block)
- Single block mode
- 32 M functions programmable as synchronised and unsynchronised
- Up to 3 M functions programmable in one NC block
- Programmable web speed
- Programmable acceleration and deceleration ramps
- Correction of the web speed due to the maximum speed of the motor
- Override 0-200%
- Tangential transition between NC blocks
  (alternatively automatic calculation or manual entry of the tangential transition)
- Up to 16 programmable process data per NC block
- Exact stop using position window possible
- Display of current NC block number
- Each axis can be configured as "coupled axis"
- Each axis can be configured as round axis with associated modulo position
- Plausibility check of the input data
- Output of warning and error messages

4.2.1.2.7.2 FAST Modul

_NCM_POS_ABSOLUTE_AXIS (FB)

With the function block '_NCM_POS_ABSOLUTE_AXIS', an individual axis is positioned absolutely.
The function block is called in the synchronous program level FPLC_PRG.

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.
reActPosition	REAL	Actual position value [mm]
reSetPosition	REAL	Specification of the position setpoint [mm]
reResolution	REAL	Feed constant: Conversion of the rotation movement to linear movement [increments/mm]
reVelocity	REAL	Velocity setpoint [increments/s]
reAccel / reDeccel	REAL	Acceleration/deceleration [mm/s^2]
boReset	BOOL	Reset; output value = 0, then the input 'boReset' is reset by the block
stDevice	STRUCT	ST_DEVICE The device description structure assigns the block a device. (See document Software description AmkBase Bibliothek, Part no.204986)

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.
diOutValPosAbs	DINT	Actual position value [mm]

Name	Type	Description
boErr	BOOL	The function block is in an error state FALSE: No error (permitted commanding or warning) TRUE: Error
boWarn	BOOL	The function block is in a warning state
iErrID	INT	Error identity number: Diagnostic number is output

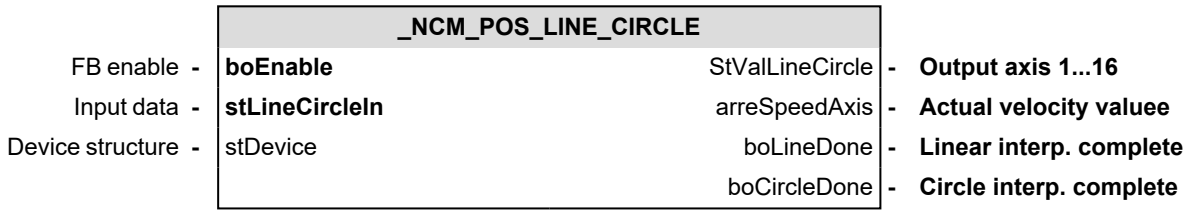
Usage note in the CoDeSys program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
	_NCM_POS_ABSOLUTE_AXIS

_NCM_POS_LINE_CIRCLE (FB)

The function block '_NCM_POS_LINE_CIRCLE' positions the axes relatively with jerk limitation.
The function block is called in the synchronous program level FPLC_PRG.

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.
stLineCircleIn	STRUCT	_NCM_ST_LINE_CIRCLE_IN Input data
stDevice	STRUCT	ST_DEVICE The device description structure assigns the block a device. (See document Software description AmkBase Bibliothek, Part no.204986)

Output variables

Name	Type	Description
StValLineCircle	STRUCT	_NCM_ST_VALOUT_LINE_CIRCLE Output data of axes 1 ... 16
arreSpeedAxis	ARRAY	ARRAY[1..16]OF REAL Velocity values of the axes
boLineDone	BOOL	Linear interpolation completed
boCircleDone	BOOL	Circle interpolation completed

Usage note in the CoDeSys program

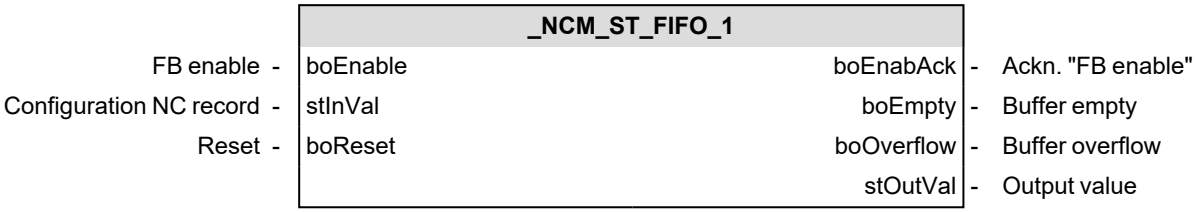
PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
	_NCM_POS_LINE_CIRCLE

4.2.1.2.7.3 FIFO

_NCM_ST_FIFO_1 (FB)

The function block '_NCM_ST_FIFO_1' manages the FIFO storage in the asynchronous program level PLC_PRG.

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.
stInVal	STRUCT	_NCM_ST_GEO_IN Data structure with data of the NC record that is to be saved.
boReset	BOOL	Reset FiFo buffer

Output variables

Name	Type	Description
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled
boEmpty	BOOL	The FiFo buffer is empty
boOverflow	BOOL	Buffer overflow: A new value is to be saved to the buffer, but no free position is available.
stOutVal	STRUCT	_NCM_ST_GEO_IN Data structure with data of the NC record that is to be saved.

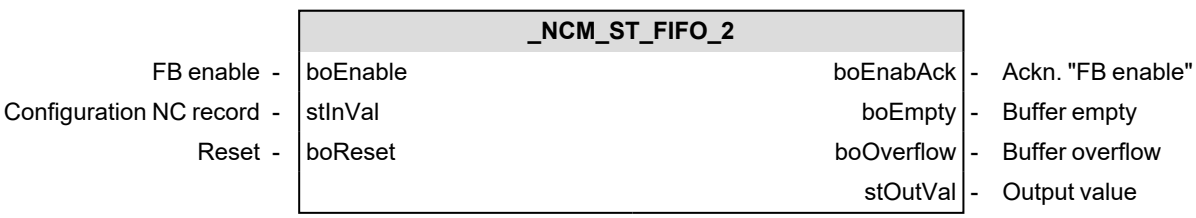
Usage note in the CoDeSys program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
_NCM_ST_FIFO_1	

_NCM_ST_FIFO_2 (FB)

The function block _NCM_ST_FIFO_2 manages the FIFO storage in the synchronous program level FPLC_PRG.

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.
stInVal	STRUCT	_NCM_ST_GEO Data structure with data of the current NC record that is to be modified.
boReset	BOOL	Reset FiFo buffer

Output variables

Name	Type	Description
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled
boEmpty	BOOL	The FiFo buffer is empty
boOverflow	BOOL	Buffer overflow: A new value is to be saved to the buffer, but no free position is available.
stOutVal	STRUCT	_NCM_ST_GEO Data structure with data of the current NC record that is to be modified.

Usage note in the CoDeSys program

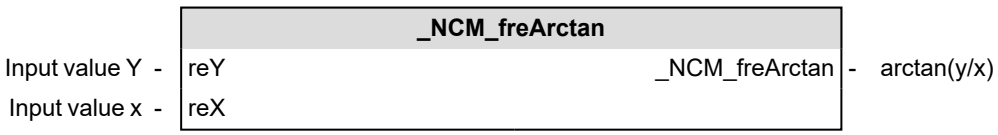
PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
	_NCM_ST_FIFO_2

4.2.1.2.7.4 Functions

_NCM_freArctan (F)

The function '_NCM_freArctan' calculates the arctan(y/x).
The function is called in the asynchronous program level PLC_PRG.

User interface



Input variables

Name	Type	Description
reY	REAL	Input value Y: Opposite side
reX	REAL	Input value X: Adjacent side

Output variable

Name	Type	Description
_NCM_freArctan	REAL	arctan (y/x) [rad]

Usage note in the CoDeSys program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
_NCM_freArctan	

4.2.1.2.7.5 Module

_NCM_CALC_LINE_CIRCLE (FB)

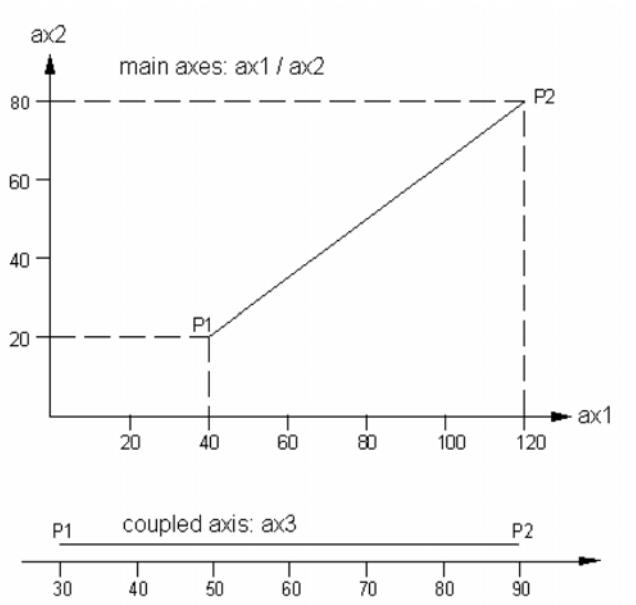
The function block '_NCM_CALC_LINE_CIRCLE' calcualtes the linear or circle interpolation for an NC data record.
The function block is called in the asynchronous program level PLC_PRG.

User interface

		_NCM_CALC_LINE_CIRCLE			
Centre X-axis	-	rePmX	reLengthCircle	-	Circle length
Centre Y-axis	-	rePmY	diLengthLineCircleInkr	-	Circle or linear length
Centre Z-axis	-	rePmZ	reAlpha	-	Angle
Definition circle plane	-	iCharPlane	reAlphaS	-	Start angle
Circle interp. Direction of rotation	-	bocw	reRS	-	Radius start point
Resolution	-	diResol	reRE	-	Radius end point
Start point axis 1-16	-	arreP1Ax	rePart1	-	Difference start to center
End point axis 1-16	-	arreP2Ax	rePart1End	-	Difference end to center
Round axis	-	arboRoundAxis	rePart2	-	Difference start to center
Modulo position	-	arreModulopos	rePart2End	-	Difference end to center
Relative positioning	-	boRelativ	rePart3	-	Difference start to center
Coupled axis	-	wCoupledAxis	rePart3End	-	Difference end to center
Code of NC dataset	-	iCharElem	uiError	-	Error ID
			uiWarn	-	Warning ID
			arreRatioAx	-	Distance ratio
			ardiEndposAx	-	End positions

Input variables

Name	Type	Description
rePmX, rePmY, rePmZ	REAL	Position of the circle centre, axis X / Y / Z [mm]
iCharPlane	INT	Code to define a circle plane **iCharPlane** / **Circle plane** 17 / X/Y (axis 1/axis 2) 18 / X/Z (axis 1/axis 3) 19 / Y/Z (axis 2/axis 3)
bocw	BOOL	Circle interpolation TRUE: clockwise (cw) FALSE: counter clockwise (ccw)
diResol	DINT	Feed constant: Conversion of the rotation movement to linear movement [increments/mm]
arreP1Ax, arreP2Ax	ARRAY	ARRAY[1..16]OF REAL Start or end point of axes 1 ... 16
arboRoundAxis	ARRAY	ARRAY[1..16]OF BOOL Definition of the axis as round axis TRUE: Round axis
arreModulopos	ARRAY	ARRAY[1..16]OF REAL Modoulo position

Name	Type	Description
boRelativ	BOOL	Relative positioning If an NC motion block should not be executed but rather relatively, 'boRelativ' must be set to true. Example: Relative line interpolation of axis 1 by 100 mm.
wCoupledAxis	WORD	With the bit string 'wCoupledAxis', each axis can be configured as a coupled axis. The coordinate values of a coupled axis are not used in the calculation of the web speed of all other axes. Example: The web speed is given for the path from P1 to P2 for axis 1 and axis 2. The web speed for axis 3 is calculated as a function of the distance that it travels so that it arrives at the target point at the same time as axis 1 and axis 2. Axis 3 as coupled axis: wCoupledAxis := 16#0004 
iCharElem	INT	Code for the NC record 1 – Line interpolation 2 – Circle interpolation clockwise (cw) 3 – Circle interpolation counter clockwise (ccw) 4 – Deceleration time 9 – Stop 30 – Program end

Output variables

Name	Type	Description
reLengthCircle	REAL	Circle length [mm]
diLengthLineCircleInkr	DINT	Circle or linear length [increments]
reAlpha	REAL	Angle of the arc [rad]
reAlphaS	REAL	Start angle [rad]
reRS, reRE	REAL	Length of the radius at the start/end point [mm]
rePart1, rePart2, rePart3	REAL	Difference start point - centre X / Y / Z [mm]
rePart1End, rePart2End, rePart3End	REAL	Difference end point - centre X / Y / Z [mm]

Name	Type	Description
uiError	UINT	Error identity number: Diagnostic number is output 0 – No error 5 – Circle radius is zero 6 – Selection of coupled axes is defective
uiWarn	UINT	Warning identity number: Diagnostic number is output 0 – No warning 1 – Length of the line is zero 6 – Radius at the start point of the circle is different to the radius at the end point of the circle.
arreRatioAx	ARRAY	ARRAY[1..16]OF REAL Field with ratio of distances of axes 1 ... 16 to total distance
ardiEndposAx	ARRAY	ARRAY[1..16]OF DINT Field with end positions of axes 1 ... 16 [increments]

Usage note in the CoDeSys program

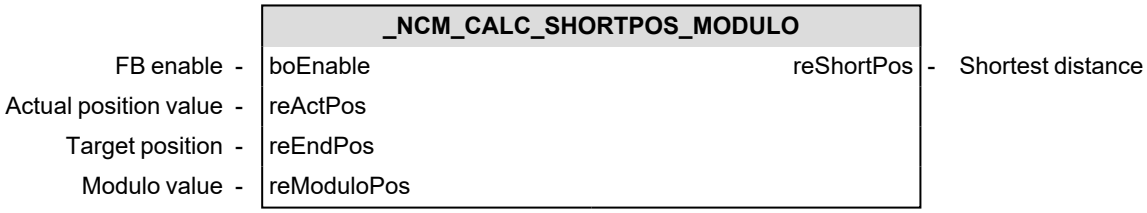
PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
_NCM_CALC_LINE_CIRCLE	

_NCM_CALC_SHORTPOS_MODULO (FB)

The function block '_NCM_CALC_SHORTPOS_MODULO' calculates the shortest path to the modulo positioning of a rotating axis.

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

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.
reActPos	REAL	Actual position of the modulo axis [°]
reEndPos	REAL	End position of the modulo axis [°]
reModuloPos	REAL	Modulo value of the axis [°]

Output variables

Name	Type	Description
reShortPos	REAL	Shortest distance between 'reActPos' and 'reEndPos' [°]

Usage note in the CoDeSys program

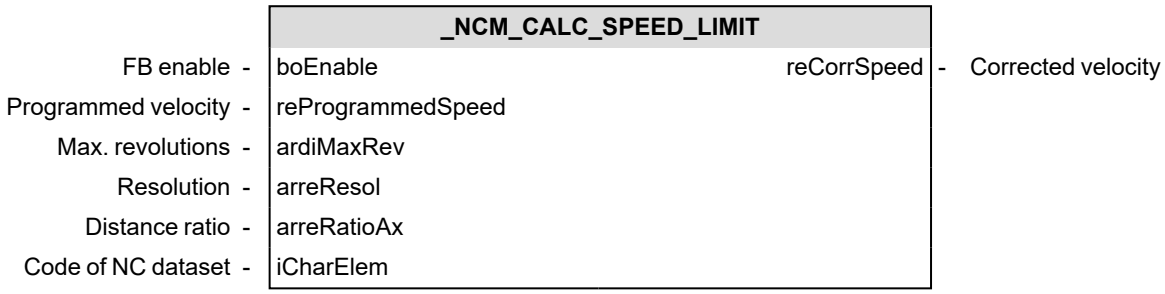
PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
_NCM_CALC_SHORTPOS_MODULO	

_NCM_CALC_SPEED_LIMIT (FB)

The function block '_NCM_CALC_SPEED_LIMIT' calculates the maximum velocity of the axis, based on the maximum permissible revolutions.

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

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.
reProgrammedSpeed	REAL	Programmed velocity [mm/min]
ardiMaxRev	ARRAY	ARRAY[1..16] OF DINT Maximum revolutions of the axis [revolutions]
arreResol	ARRAY	ARRAY[1..16] OF REAL Feed constant: Conversion of the rotation movement to linear movement [mm/revolution]
arreRatioAx	ARRAY	ARRAY[1..16]OF REAL Field with ratio of distances of axes 1 ... 16 to total distance
iCharElem	INT	Code for the NC record 1 – Line interpolation 2 – Circle interpolation clockwise (cw) 3 – Circle interpolation counter clockwise (ccw) 4 – Deceleration time 9 – Stop 30 – Program end

Output variables

Name	Type	Description
reCorrSpeed	REAL	Corrected velocity [mm/min]

Usage note in the CoDeSys program

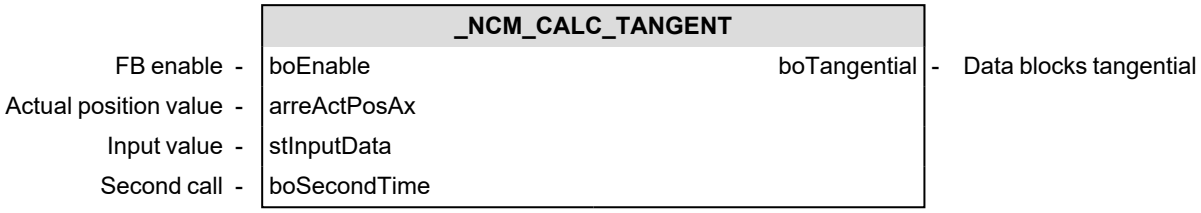
PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
_NCM_CALC_SPEED_LIMIT	

_NCM_CALC_TANGENT (FB)

The function block '_NCM_CALC_TANGENT' determines whether the transition between two NC data records is tangential so that the web speed during the transition from NC motion block (n) to NC motion block (n+1) does not need to be braked to zero.

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

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.
arreActPosAx	ARRAY	ARRAY[1..16]OF REAL Field of the actual positions
stInputData	STRUCT	_NCM_ST_GEO_IN Data structure with data of the NC record that is to be modified.
boSecondTime	BOOL	The function block starts only when it is called a second time

Output variables

Name	Type	Description
boTangential	BOOL	Data blocks are tangential, angle = 0

Usage note in the CoDeSys program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
_NCM_CALC_TANGENT	

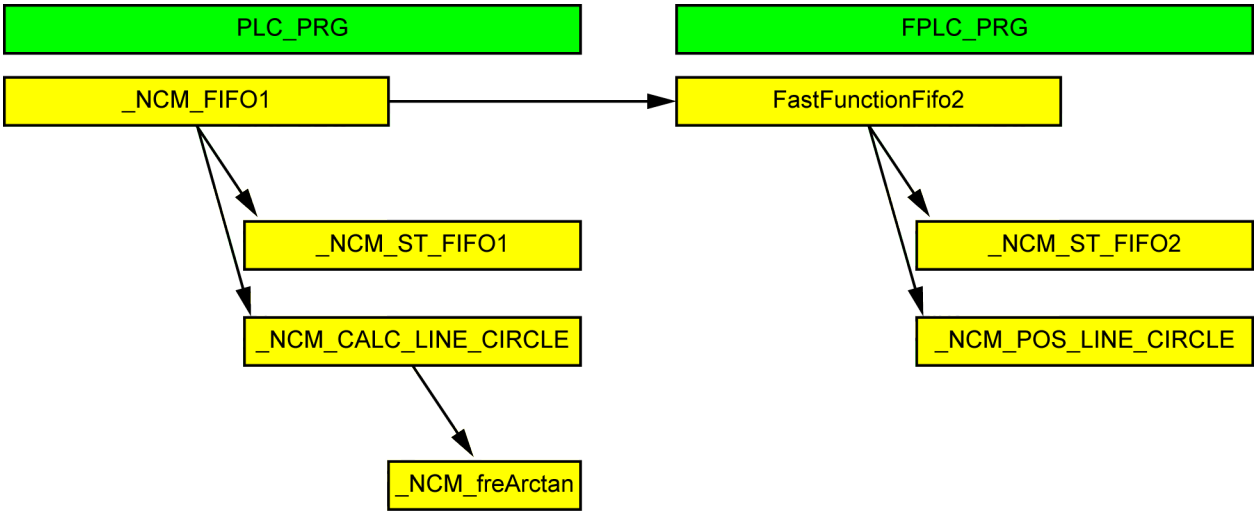
_NCM_FIFO1 (FB)

The function block is the parent block of the '_NCM_FIFO1' motion functions.

- The necessary parameters and boundary conditions are entered and all results and information are returned.
- The input data re written to the FIFO memory.
- Linear or circle interpolation is pre-calculated.
- Using the synchronous action 'FastFunctionFifo2', the pre-calculated data are

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

_NCM_FIFO1 calls the subordinate function blocks of the asynchronous and synchronous program level.



User interface

	_NCM_FIFO1			
FB enable -	boEnable	boEnabAck	-	Ackn. "FB enable"
Configuration NC record -	stInVal	boStoreDataDone	-	Ack "NC record saved"
Maximum speed 1-16 -	ardiConfigMaxRev	boLineDone	-	Ack "NC record with line"
Resolution 1-16 -	arreConfigResol	bocircleDone	-	Ack "NC record with circle"
Actual position 1-16 -	arreActPosAx	boStop	-	Ack "NC record with stop"
Round axis 1-16 -	arboRoundAxis	boMessageFin	-	Ack "NC record with end"
Modulo position 1-16 -	arreModulopos	dwMFunctOut	-	Bit string M functions
Data in FIFO -	boStoreData	boFifoOverflow	-	Buffer overflow
Reset -	boReset	arreprocessdataOut	-	Process data 1-16
NC records output -	boEnRestoreData	arreSpeedAxis	-	Velocity 1-16
Single block mode -	boMSingleBlock	uiCountDataSet	-	Output NC records
Single block output -	bostep	boError	-	Error
Stop function reset -	boResStop	uiError	-	Error ID
Ack synchr. M functions -	boConfMFunct	uiWarn	-	Warning ID
Movement immediate stop -	boStopMotion	parFollErr	-	Following distance 1-16
Influence of web speed -	reOverride	StValOutLineCircle	-	Output value 1-16
Configuration M function -	dwSyncMFunction			
Read actual position -	boReadinActPos			
Calculation max. velocity -	boEnCalcSpeedLimit			
Calculation tang. transf. -	boEnCalcTangCon			
Device structure -	stDevice			

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.
stInVal	STRUCT	_NCM_ST_GEO_IN Data structure with data of the NC record that is to be saved.
ardiConfigMaxRev	ARRAY	ARRAY{1..16]OF DINT Maximum speed of the motor of the axes 1-16 [rpm]
arreConfigResol	ARRAY	ARRAY{1..16]OF REAL Resolution of axes 1-16 [mm/rev.] / [°/rev.]
arreActPosAx	ARRAY	ARRAY[1..16]OF REAL Field of the actual positions
arboRoundAxis	ARRAY	ARRAY[1..16]OF BOOL Definition of the axis as round axis TRUE: Round axis For round axes, positioning is always performed along the shortest distance. **Example:** arreModuloposition[1]=360°; actual position =10°; target position =350° arboRoundAxis[1] = true --> axis [1] moves -20° to the target position (shortest distance). arboRoundAxis[1] = false --> axis [1] moves +340° to the target position.
arreModulopos	ARRAY	ARRAY[1..16]OF REALModoulo positions of the axes configured as round axes [°]
boStoreData	BOOL	With the edge change FALSE->TRUE, an NC record adjacent to 'stInVal' is stored in FiFo 1 The signal must be reset if 'boStoreDataDone' = TRUE (handshake).
boReset	BOOL	• Stops all movements of the axes. • Resets all error messages. • Resets all M functions. • Resets all internal states.
boEnRestoreData	BOOL	Enable signal for output of NC blocks. Only if 'boEnRestoreData' = TRUE can NC blocks be output.
boMSingleBlock	BOOL	The signal activates the single block mode, meaning one NC data block is processed at a time. The following data block is started with 'bostep'.
bostep	BOOL	Starts output of an NC block (only in conjunction with 'boMSingleBlock' = TRUE).
boResStop	BOOL	Acknowledgement signal for a programmed stop 'iCharElem'=9 (stop). For an edge change FALSE -> TRUE the next NC block is output.
boConfMFunct	BOOL	Acknowledgement signal for all synchronous M functions.
boStopMotion	BOOL	Signal stops all axis movements immediately.
reOverride	REAL	Factor for the programmed web speed (0-200%)

Name	Type	Description
dwSyncMFunction	DWORD	bit string for configuration of M functions 1..32 as synchronised or unsynchronised M function. • Unsynchronised means: M function becomes TRUE with start output of the corresponding NC block. With output of the next NC block, the M function becomes FALSE • Synchronised means: M function becomes TRUE with start output of the corresponding NC block. The output of the next NC block is blocked. When 'boConfMFunct' becomes TRUE, the M function becomes FALSE and the next NC block is output. Example for configuration of synchronous M functions: dwSyncMFunct:=16#FFFF0000 means: M function 1..16 unsynchronised M function 17..32 synchronised
boReadinActPos	BOOL	To make possible absolute positioning, it is necessary to read the actual position of the axes. Typically, 'boReadInActPos' is set to TRUE when the 1st NC block is saved. Thus the start point for the internal calculation is the actual position of the axes when saved. For all other NC blocks, 'boReadInActPos' = FALSE. Thus the start point for the internal calculation is always the end point of the previous data record.
boEnCalcSpeedLimit	BOOL	Enable signal for internal speed limit calculation due to the maximum speed ('ardiConfigMaxRev'). TRUE: Calculation is performed. FALSE: Calculation is not performed.
boEnCalcTangCon	BOOL	Enable signal for automatic calculation of a tangential transition between two NC motion blocks. TRUE: The calculation is performed automatically. If a tangential transition is recognised, both motion blocks are run with the programmed speed. No ramp down to V = 0 takes place after the 1st motion block. FALSE: No calculation is performed. A tangential transition can still be specified "manually" via the input structure '_NCM_ST_GEO_IN' boTangetial.
stDevice	STRUCT	ST_DEVICE The device description structure assigns the block a device. (See document Software description AmkBase Bibliothek, Part no.204986)

Output variables

Name	Type	Description
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled
boStoreDataDone	BOOL	FALSE -> TRUE: An NC block was successfully saved. The input signal 'boStoreData' becomes FALSE (handshake): TRUE -> FALSE
boLineDone	BOOL	FALSE -> TRUE: An NC block with line interpolation was successfully output. The next NC block is output: TRUE -> FALSE
bocircleDone	BOOL	FALSE -> TRUE: An NC block with circle interpolation was successfully output. The next NC block is output: TRUE -> FALSE
boStop	BOOL	FALSE -> TRUE: An NC block with programmed stop was successfully output. THe next NC block is output (boResStop = TRUE): TRUE -> FALSE.

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>boMessageFin</td><td>BOOL</td><td>FALSE -> TRUE: An NC block with programmed end was successfully output.
A new program is started or 'boReset' becomes TRUE: TRUE -> FALSE</td></tr>
<tr><td>dwMFunctOut</td><td>DWORD</td><td>Bit string with the set M functions.
Example:
NC block with the M functions
uiFirstMFunct = 5
uiSecondMFunct = 9
uiThirdMFunct=13
has dwMFunctOut the value 16#00001110.</td></tr>
<tr><td>boFifoOverflow</td><td>BOOL</td><td>FALSE -> TRUE: FiFo is full.
It is therefore not possible to save additional data blocks (boStoreDataDone = FALSE).</td></tr>
<tr><td>arreprocessdataOut</td><td>ARRAY</td><td>ARRAY[1..16]OF REAL
Output of the process data programmed in the data block (axis 1..16).</td></tr>
<tr><td>arreSpeedAxis</td><td>ARRAY</td><td>ARRAY[1..16]OF REAL
Velocity values of the axes</td></tr>
<tr><td>uiCountDataSet</td><td>UINT</td><td>Counter of the output NC blocks.</td></tr>
<tr><td>boError</td><td>BOOL</td><td>FALSE -> TRUE: The plausibility check detected an error.
'boReset' becomes TRUE: TRUE -> FALSE</td></tr>
<tr><td>uiError</td><td>UINT</td><td>Error identity number: Diagnostic number is output
<table>
<tr><td>0</td><td>No error</td></tr>
<tr><td>5</td><td>Circle radius is zero</td></tr>
<tr><td>10</td><td>Incorrectly programmed M function</td></tr>
<tr><td>11</td><td>Incorrectly programmed code (iCharElem)</td></tr>
<tr><td>12</td><td>Incorrectly programmed circle plane (iCharPlane)</td></tr>
<tr><td>13</td><td>Programmed web speed is zero</td></tr>
<tr><td>14</td><td>Programmed acceleration is zero</td></tr>
<tr><td>15</td><td>Programmed program is zero</td></tr>
</table></td></tr>
<tr><td>uiWarn</td><td>UINT</td><td>Warning identity number: Diagnostic number is output
<table>
<tr><td>0</td><td>No warning</td></tr>
<tr><td>1</td><td>Length of the line is zero</td></tr>
<tr><td>6</td><td>Radius at the start point of the circle is different to the radius at the end point of the circle.</td></tr>
</table>
'uiWarn' is reset when the next NC block is output.</td></tr>
<tr><td>parFollErr</td><td>POINTER</td><td>POINTER TO ARRAY[1..16]OF DINT
To enable an exact stop of the axes, it is necessary to compare the current following distance of each axis with an "in position" window 'ardiVerifyFollErr'.
Only once the following distance is within the "in position" window can the next NC block be output.
In the synchronous task (FPLC_PRG), the current following distances of axes 1-16 can be written to the array 'ardiFollErrAx [1..16]':
PLC_PRG.fbFifo1.parFollErr^[1] = following distance axis 1
:
:
PLC_PRG.fbFifo1.parFollErr^[16] = following distance axis 16
If the current following distances are not written to the array, internally the following distance=0
is used for calculation and thus the condition "in position" is fulfilled.</td></tr>
</table>

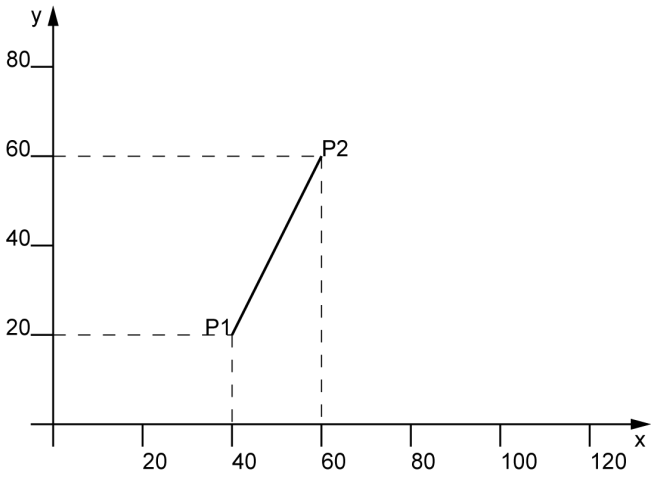
Name	Type	Description
StValOutLineCircle	STRUCT	_NCM_ST_VALOUT_LINE_CIRCLE Data structure that contains the output increments of axes 1..16. Example: PLC_PRG.fbFifo1.StValOutLineCircle.diOutVal[1] contains the output increments for axis 1

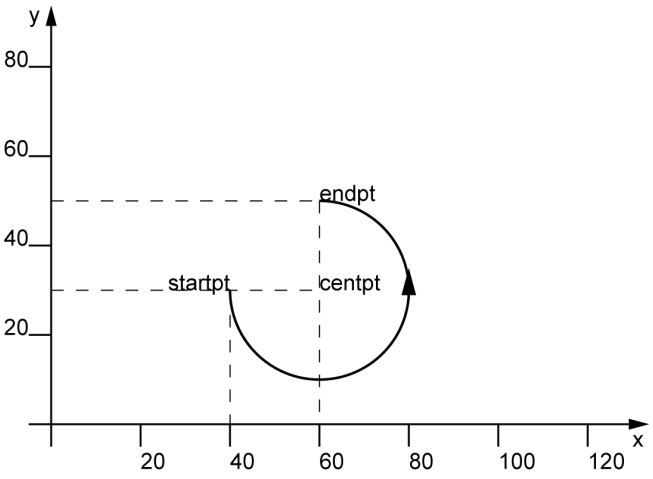
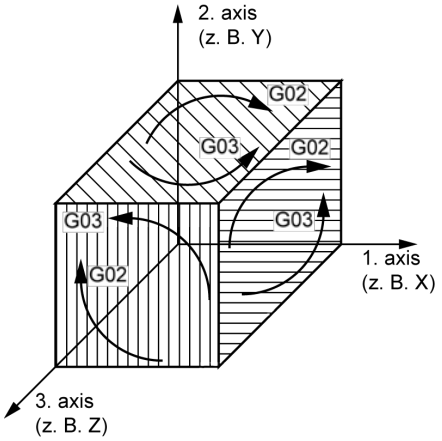
Usage note in the CoDeSys program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
_NCM_FIFO1	_NCM_FIFO1.FastFunctionFifo2

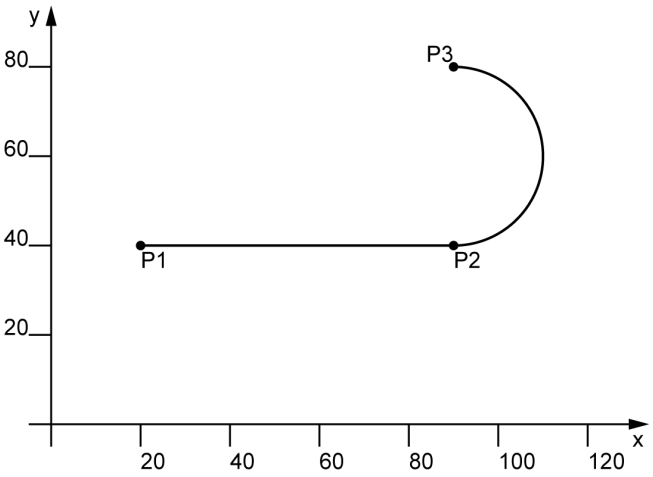
_NCM_ST_GEO_IN (ST)

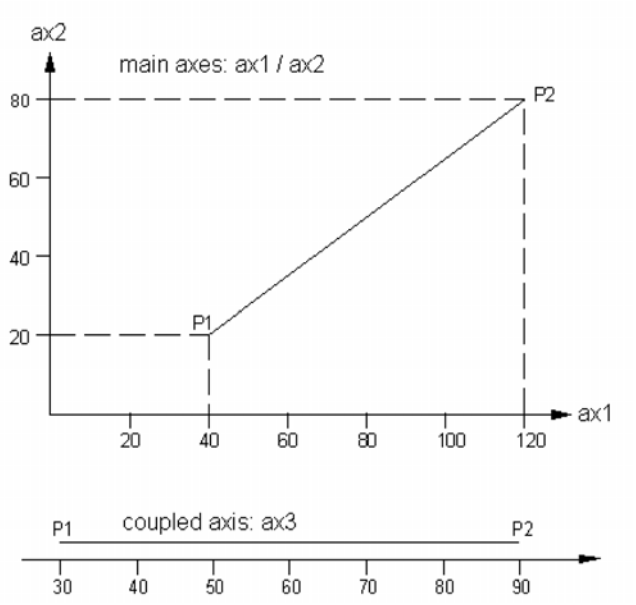
The structure '_NCM_ST_GEO_IN' contains all relevant data to configure an NC block.

Name	Type	Description
iCharElem	INT	Code for the NC record 1 – Line interpolation 2 – Circle interpolation clockwise (cw) 3 – Circle interpolation counter clockwise (ccw) 4 – Deceleration time 9 – Stop 30 – Program end
arreEndPosAx	ARRAY	ARRAY[1..16] OF REAL Specifies the end point (example: P2) of the corresponding axis in the NC block [mm]. Example: NC block for line interpolation iCharElem := 1 arreEndPosAx[1] := 60 mm arreEndPosAx[2] := 60 mm  P1: end point NC block (n) P2: end point NC block (n+1)

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>rePMAx1;
rePMAx2;
rePMAx3</td><td>REAL</td><td>Position circle centre [mm]
Axis 1 (X), axis 2 (Y), axis 3 (Z)
Example:
NC block for circle interpolation cw in plane X/Y
iCharElem := 3
iCharPlane := 17
rePMAx1 := 60 mm
rePMAx2 := 30 mm
arreEndPosAx[1] := 60 mm
arreEndPosAx[2] := 50 mm
 
</td></tr>
<tr><td>iCharPlane</td><td>INT</td><td>Code to define a circle plane
<table>
<tr><th>iCharPlane</th><th>Circle plane</th></tr>
<tr><td>17</td><td>X/Y (axis 1/axis 2)</td></tr>
<tr><td>18</td><td>X/Z (axis 1/axis 3)</td></tr>
<tr><td>19</td><td>Y/Z (axis 2/axis 3)</td></tr>
</table>

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

Name	Type	Description
uiFirstMFunct; uiSecondMFunct; uiThirdMFunct	UINT	**M functions** **In total, 32 M functions are available.** Per NC block, up to 3 M functions are programmable. Each M function can be configured as synchronised or unsynchronised M function. uiFirstMFunct (1st M function in the NC block) uiSecondMFunct (2nd M function in the NC block) uiThirdMFunct (3rd M function in the NC block) Example: An NC block with 3 M functions The 1st M function should be M function number 5. The 2nd M function should be M function number 9. The 3rd M function should be M number 13. uiFirstMFunct := 5 uiSecondMFunct := 9 uiThirdMFunct := 13
reDelaytime	REAL	Delay time up to output of the next NC block [s]. Example: Delay 3.5 s iCharElem := 4 reDelaytime := 3.5 s
ardiVerifyFollErr	ARRAY	ARRAY[1..16] OF DINT Comparison window for exact stop (see parFollError) Example: Comparison window for axis 1 should be 100 incr. ardiVerifyFollErr[1] := 100
reVelocity	REAL	Specification of web speed in NC block Linear: [mm/s]; circle: [°/s].
reAcceleration	REAL	Specification of the acceleration in the NC block Linear: [mm/s^2]; circle: [°/s^2].
reDecceleration	REAL	Specification of the deceleration in the NC block Linear: [mm/s^2]; circle: [°/s^2].
reACCJerk	REAL	Specification of jerk limitation during acceleration Linear: [mm/s^3]; circle: [°/s^3].  For reAccJerk=0, the jerk is infinite.
reDecJerk	REAL	Specification of jerk limitation during deceleration Linear: [mm/s^3]; circle: [°/s^3].  For reDecJerk=0, the jerk is infinite.

Name	Type	Description
boTangential	BOOL	In some applications, it makes sens when the web speed is braked to zero during the transition from NC motion block (n) to NC motion block (n+1). (Connection between line P1/P2 and circle P2/P3) boTangential := TRUE in the NC block of circle P2/P3 means: The movement of the axes accelerates at P1 until the programmed web speed is reached and decelerates to a standstill at P3. A tangential movement between two NC motion blocks is only possible if at least 2 NC blocks have been saved. 
boRelativ	BOOL	Relative positioning If an NC motion block should not be executed but rather relatively, 'boRelativ' must be set to true. Example: Relative line interpolation of axis 1 by 100 mm. iCharElem := 1 arreEndPosAx[1] := 100 mm boRelativ := TRUE
arreProzessdata	ARRAY	ARRAY[1..16] OF REAL There are a total of 16 process data per NC block. When outputting the NC block, the programmed process data are written to the output variable 'arreProcessdataOut'. Example: The cutting speed of the mill of an NC milling machine from NC block to NC block should be adjustable.

Name	Type	Description
wCoupledAxis	WORD	With the bit string 'wCoupledAxis', each axis can be configured as a coupled axis. The coordinate values of a coupled axis are not used in the calculation of the web speed of all other axes. Example: The web speed is given for the path from P1 to P2 for axis 1 and axis 2. The web speed for axis 3 is calculated as a function of the distance that it travels so that it arrives at the target point at the same time as axis 1 and axis 2. Axis 3 as coupled axis: wCoupledAxis := 16#0004 

4.2.1.2.8 08_Sequence

4.2.1.2.8.1 SEQUENCE_STATE_CHECK (FB)

The function block 'SEQUENCE_STATE_CHECK' allows the check of a step sequence.

The outputs 'uiStateTm..' show the states of the step sequence at the corresponding time. So you can see up to 10 state changes of the step sequence.

The call of this function block is permissible in both the asynchronous program section PLC_PRG as well as the synchronous program section FPLC_PRG.

User interface

	SEQUENCE_STATE_CHECK			
FB enable -	boEnable	boEnabAck	-	Ackn. "FB enable"
State of the step sequence -	uiState	uiStateTm1	-	uiState(t-1)
		uiStateTm2	-	uiState(t-2)
		uiStateTm3	-	uiState(t-3)
		uiStateTm4	-	uiState(t-4)
		uiStateTm5	-	uiState(t-5)
		uiStateTm6	-	uiState(t-6)
		uiStateTm7	-	uiState(t-7)
		uiStateTm8	-	uiState(t-8)
		uiStateTm9	-	uiState(t-9)
		uiStateTm10	-	uiState(t-10)

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.
uiState	UINT	State of the step sequence

Output variables

Name	Type	Description
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled
uiStateTm1	UINT	State of the step sequence at the time (t-1)
uiStateTm2	UINT	State of the step sequence at the time (t-2)
uiStateTm3	UINT	State of the step sequence at the time (t-3)
uiStateTm4	UINT	State of the step sequence at the time (t-4)
uiStateTm5	UINT	State of the step sequence at the time (t-5)
uiStateTm6	UINT	State of the step sequence at the time (t-6)
uiStateTm7	UINT	State of the step sequence at the time (t-7)
uiStateTm8	UINT	State of the step sequence at the time (t-8)
uiStateTm9	UINT	State of the step sequence at the time (t-9)
uiStateTm10	UINT	State of the step sequence at the time (t-10)

Usage note in the CoDeSys program

The call of this function block is permissible in both the asynchronous program section PLC_PRG as well as the synchronous program section FPLC_PRG.

Associated with this function block, a visualisation is prepared in CoDeSys.

Siehe 'Visualization of AFL blocks' auf Seite 806.

4.2.1.2.8.2 SEQUENCE_STATE_TIMEOUT (FB)

The function block 'SEQUENCE_STATE_TIMEOUT' is used for time monitoring. If the actual state / step will not be left within the specified time, the output 'boTimeOut' will be set.

The call of this function is permissible in both the asynchronous program section PLC_PRG as well as the synchronous program section FPLC_PRG.

User interface

	SEQUENCE_STATE_TIMEOUT			
FB enable -	boEnable	boEnabAck	-	Ackn. "FB enable"
Actual state -	uiActualState	boTimeOut	-	Time exceeded
Monitoring time -	tTimeOut			

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.
uiActualState	UINT	Actual state / step
tTimeOut	TIME	Timeout [ms] Default: tTimeOut = 10 s

Output variables

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

Usage note in the CoDeSys program

The call of this function block is permissible in both the asynchronous program section PLC_PRG as well as the synchronous program section FPLC_PRG.

Associated with this function block, a visualisation is prepared in CoDeSys.

Siehe 'Visualization of AFL blocks' auf Seite 806.

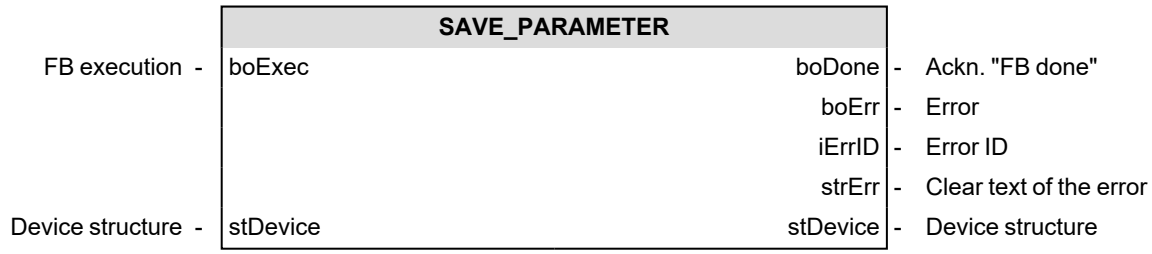
4.2.1.2.9 09_Parameter

4.2.1.2.9.1 SAVE_PARAMETER (FB)

The function block 'SAVE_PARAMETER' secures the parameter set of a drive. After you replace the hardware, the parameter set can be restored by the block 'RESTORE_PARAMETER'. The parameters are stored as a FILE on the controller.

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

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.

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
strErr	STRING	Clear text of the error
iErrID	INT	Error identity number: Diagnostic number is output iErrID = 0: No error iErrID ≠ 0, boErr = TRUE: Error iErrID ≠ 0, boErr = FALSE: Warning Value 1: Error reading the ID Value 2: ID read timeout Value 3: Write error file Value 4: Write file timeout Value 10: There was no file name 'strFileName' passed

Input and output variables

Name	Type	Description
stDevice	STRUCT	ST_DEVICE The device description structure assigns the block a device. (See document Software description AmkBase Bibliothek, Part no.204986)

Usage note in the CoDeSys program

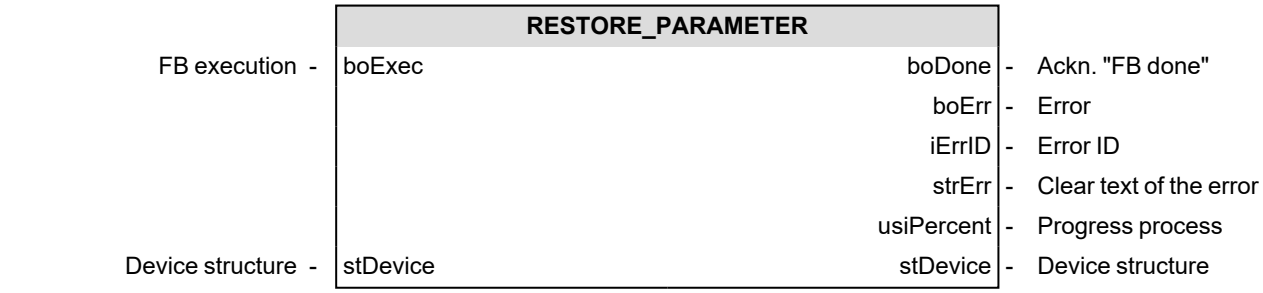
PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
SAVE_PARAMETER	

4.2.1.2.9.2 RESTORE_PARAMETER (FB)

With the function block 'RESTORE_PARAMETER', the stored parameters can be written to a drive. The parameters must have been previously secured by the block 'SAVE_PARAMETER'.

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

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.

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>Value</th><th>Meaning</th></tr>
<tr><td>1</td><td>Error reading file</td></tr>
<tr><td>2</td><td>Timeout reading file</td></tr>
<tr><td>3</td><td>Timeout ID write</td></tr>
<tr><td>4</td><td>List length ID greater than 4096 bytes</td></tr>
<tr><td>6</td><td>Error 'WRITE_ID_LIST'</td></tr>
<tr><td>7</td><td>Unknown file length of the ID</td></tr>
<tr><td>10</td><td>There was no file name 'strFileName' passed</td></tr>
</table></td></tr>
<tr><td>strErr</td><td>STRING</td><td>Clear text of the error</td></tr>
<tr><td>usiPercent</td><td>USINT</td><td>Progress process</td></tr>
</table>

Input and output variables

Name	Type	Description
stDevice	STRUCT	ST_DEVICE The device description structure assigns the block a device. (See document Software description AmkBase Bibliothek, Part no.204986)

Usage note in the CoDeSys program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
RESTORE_PARAMETER	

4.2.1.3 07_Visualization

4.2.1.3.1 NUMPAD (FB)

The function block 'NUMPAD' is the logic behind the visualisation ViNUMPAD. It realises the inputs and compares the current value with the specified limits.

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

User interface

	NUMPAD			
FB enable -	boEnable	boEnabAck	-	Ackn. "FB enable"
Integer -	boInteger	boDone	-	Ackn. "FB done"
Value before input -	reStartValue	reDisplay	-	Entered value as REAL
Lower limit -	reMinimalValue	diDisplay	-	Entered value as DINT
Upper limit -	reMaximalValue	strDisplay	-	Entered value as STRING
Button colour -	dwButtonColor			
Button alarm colour -	dwButtonAlarmColor			
Button text colour -	dwButtonTextColor			
Button text size -	wButtonTextHeight			
Display text size -	wDisplayTextHeight			
Display text colour -	dwDisplayTextColor			
Display colour -	dwDisplayColor			

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.
boInteger	BOOL	TRUE: Only integer entry possible
reStartValue	REAL	Value before input

Name	Type	Description
reMinimalValue	REAL	Lower limit of the input (reMinimalValue = reMaximalValue: no monitoring)
reMaximalValue	REAL	Upper limit of the input (reMinimalValue = reMaximalValue: no monitoring)
dwButtonColor	DWORD	Colour of the button (16#00BBGGRR) Default: Blue (16#00FF0000)
dwButtonAlarmColor	DWORD	Colour of the button for actuation (16#00BBGGRR) Default: Green (16#0000FF00)
dwButtonTextColor	DWORD	Text colour of the button (16#00BBGGRR) Default: White (16#00FFFFFF)
wButtonTextHeight	WORD	Text size of button
wDisplayTextHeight	WORD	Text size of display
dwDisplayTextColor	DWORD	Text colour of the display (16#00BBGGRR) Default: Red (16#000000FF)
dwDisplayColor	DWORD	Colour of the display (16#00BBGGRR) Default: Dark grey (16#00555555)

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.
reDisplay	REAL	Output of the entered value as floating point number
diDisplay	DINT	Output of the entered value as integer
strDisplay	STRING	Output of the entered value as string

Usage note in the CoDeSys program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
NUMPAD	

4.2.1.4 08_Devices

4.2.1.4.1 fstrGetControllerTyp (F)

The function 'fstrGetControllerTyp' returns the type of the control unit.
The function is called in the asynchronous program level PLC_PRG.

User interface

fstrGetControllerTyp		
fstrGetControllerTyp	-	Rückgabewert

Output variable

Name	Type	Description
fstrGetControllerTyp	STRING	Type of control unit

Usage note in the CoDeSys program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
fstrGetControllerTyp	

4.2.1.5 Appendix

EN_POS_J_ARITHMETIK

Additional information for input 'enArithmetik', used for example by AFL Applications block POSITION_JERK_1 (FB). Siehe 'POSITION_JERK_1 (FB)' auf Seite 362.

In the operation mode position control, 64 bit and 32 bit position function blocks are available.
64 bit position function blocks reach a higher accuracy because they have a higher resolution than 32 bits position function blocks.

With the input 'enArithmetik' the choice between 64 bit and 32 bit position function blocks is set.
In the default setting 'enAutomaticSelection' the selection is automatically adjusted, depending on the used PLC control and the used CODESYS version.
With 'en64BitArithmetik' or 'en32BitArithmetik', the selection can be done manually by the user.

The maximum size of data type, supported by the AMK controller depends on the used CODESYS version 2 or 3.
The connection between the controller and CODESYS version is shown in the table below.

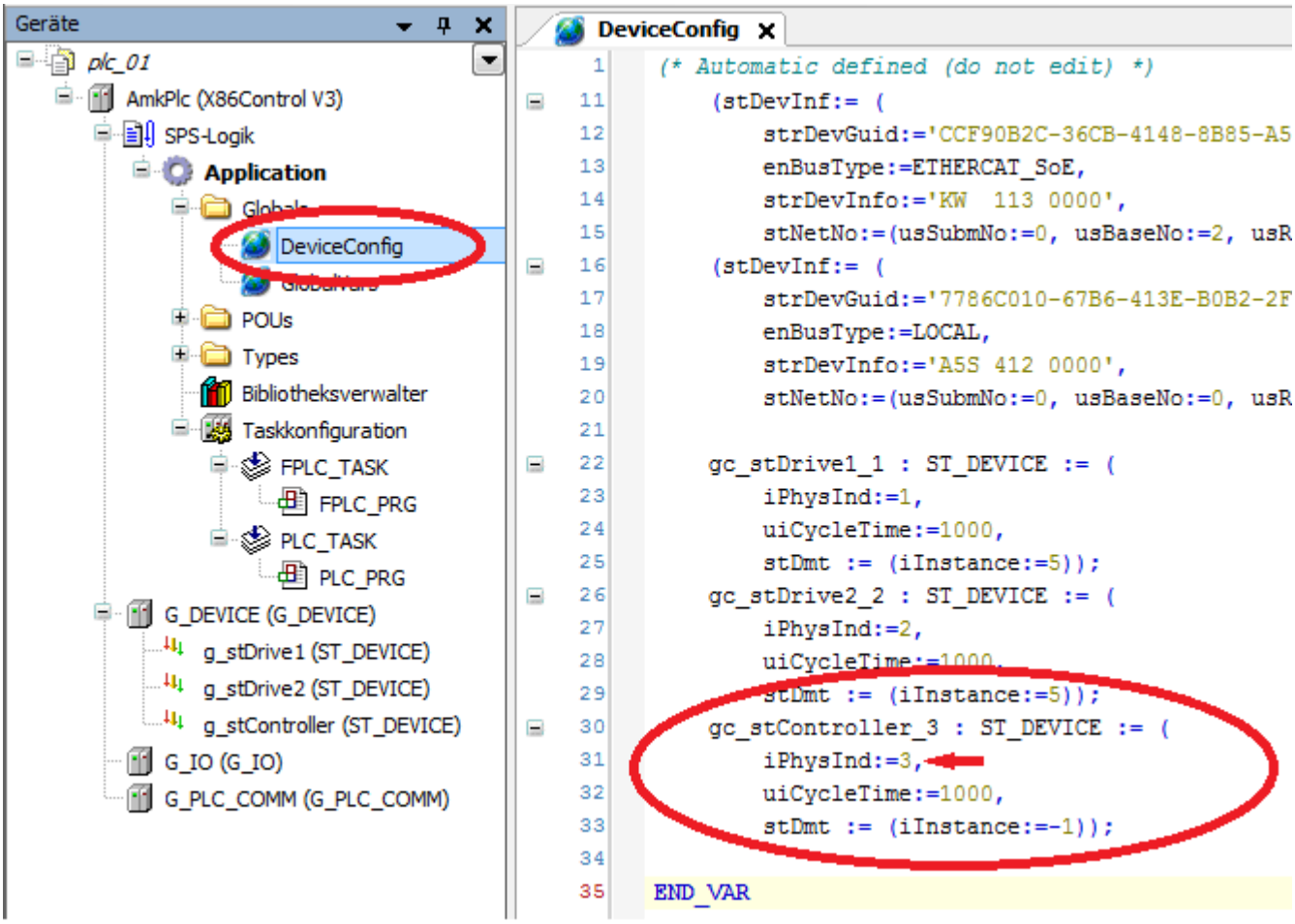
Controller	CODESYS V2	CODESYS V3
iSA	REAL (32 Bit)	LREAL (64 Bit)
A4	LREAL (32 Bit)	LREAL (64 Bit)
A5	LREAL (64 Bit)	LREAL (64 Bit)
A6	REAL (64 Bit)	LREAL (64 Bit)

64 bit position function blocks (en64BitArithmetik) may not be used with a 32 bit CODESYS version. A conversion of 64 bits into 32 bits is not possible without losses. Positioning is inaccurate and misses the target position. Consequence the state boDone/boInPos will not be TRUE.

32 bit position function blocks (en32BitArithmetik) can be used on a 64 bit CODESYS versions. The target position is reached, but by the non-used resolution accuracy is given away.

Example automatic determination

For example, the automatic determination of whether 32 bit or 64 bit arithmetic, is not working.
The symbolic device variables (type ST_DEVICE) were manually inserted in the order g_stDevice1, g_stDevice2, g_stController.
The controller (controller) is therefore listed at position 3 (iPhysInd := 3). The automatic determination only works if the controller is listed at position 1 (iPhysInd := 1).



4.2.2 Library description AmkAflConfig.lib

4.2.2.1 02_Parameter Access

4.2.2.1.1 ID_READ_10IDS_DINT (FB)

The function block 'ID_READ_10IDS_DINT' reads parameters, but not list parameters, from AMK devices. Up to 10 ID numbers in the format double integer (DINT) can be read. Every input for an ID number is assigned an output for the ID value.

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

User interface

		ID_READ_10IDS_DINT			
FB execution	-	boExec	boDone	-	Ackn. "FB done"
Read ID	-	uiIDNo1	diOutValNo1	-	ID number
Read ID	-	uiIDNo2	diOutValNo2	-	ID number
:	-	:	:	-	:
:	-	:	:	-	:
Read ID	-	uiIDNo10	diOutValNo10	-	ID number
Parameter instance	-	uiParInst	boErr	-	Error
			iErrID	-	Error ID
			enErrName	-	Name of faulty FB
Device structure	-	stDevice	stDevice	-	Device structure

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.
uiIDNo1 ... uiIDNo10	UINT	ID number to be read out
uiParInst	UINT	Parameter set number or instance number

Output variables

Name	Type	Description
boDone	BOOL	Response that the function block has been completely executed.
diOutValNo1 ... diOutValNo10	DINT	Read ID value for the ID number specified at input n
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
enErrName	ENUM	EN_FB_NAME Name of faulty block for which the diagnostic number is output. The diagnostic number is explained in the description of the corresponding library. Block: READ_ID_DINT — Library: AmkSystem.lib

Input and output variables

Name	Type	Description
stDevice	STRUCT	ST_DEVICE The device description structure assigns the block a device. (See document Software description AmkBase Bibliothek, Part no.204986)

Usage note in the CoDeSys program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
ID_READ_10IDS_DINT	

Associated with this function block, a visualisation is prepared in CoDeSys.
Siehe 'Visualization of AFL blocks' auf Seite 806.

4.2.2.1.2 ID_READ_5IDS_DINT (FB)

The function block 'ID_READ_5IDS_DINT' reads parameters, but not list parameters, from AMK devices. Up to 5 ID numbers in the format double integer (DINT) can be read. Every input for an ID number is assigned an output for the ID value.

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

User interface

		ID_READ_5IDS_DINT			
FB execution	-	boExec	boDone	-	Ackn. "FB done"
Read ID	-	uiIDNo1	diOutValNo1	-	ID number
Read ID	-	uiIDNo2	diOutValNo2	-	ID number
:	-	:	:	-	:
:	-	:	:	-	:
Read ID	-	uiIDNo5	diOutValNo5	-	ID number
Parameter instance	-	uiParInst	boErr	-	Error
			iErrID	-	Error ID
			enErrName	-	Name of faulty FB
Device structure	-	stDevice	stDevice	-	Device structure

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.
uiIDNo1 / 2 / 3 / 4 / 5	UINT	ID number to be read out
uiParInst	UINT	Parameter set number or instance number

Output variables

Name	Type	Description
boDone	BOOL	Response that the function block has been completely executed.
diOutValNo1 / 2 / 3 / 4 / 5	DINT	Read ID value for the ID number specified at input n
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
enErrName	ENUM	EN_FB_NAME Name of faulty block for which the diagnostic number is output. The diagnostic number is explained in the description of the corresponding library. Block: READ_ID_DINT — Library: AmkSystem.lib

Input and output variables

Name	Type	Description
stDevice	STRUCT	ST_DEVICE The device description structure assigns the block a device. (See document Software description AmkBase Bibliothek, Part no.204986)

Usage note in the CoDeSys program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
ID_READ_5IDS_DINT	

Associated with this function block, a visualisation is prepared in CoDeSys.
Siehe 'Visualization of AFL blocks' auf Seite 806.

4.2.2.1.3 ID_READ_NODE_LIST (FB)

The function block 'ID_READ_NODE_LIST' reads the node list (ID34211 'Node list 2') from the device declared with 'stDevice'. The nodes and the corresponding information are written to the structure 'stList'.

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

User interface

		ID_READ_NODE_LIST			
FB execution	-	boExec	boDone	-	Ackn. "FB done"
Instance	-	uiParInst	boErr	-	Error
			iErrID	-	Error ID
Node list	-	stList	stList	-	Node list
Device structure	-	stDevice	stDevice	-	Device structure

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.
uiParInst	UINT	Parameter set number or instance number, from which the node list shall be read (e.g. Instance 5 = EtherCAT at control unit A5)

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 Block: ID_READ_1 — Library: AmkBase.lib

Input and output variables

Name	Type	Description
stList	STRUCT	ST_NODE_LIST The structure contains the maximum and the actual size of the list, a header and the nodes with their contents
stDevice	STRUCT	ST_DEVICE The device description structure assigns the block a device. (See document Software description AmkBase Bibliothek, Part no.204986)

Usage note in the CoDeSys program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
ID_READ_NODE_LIST	

Associated with this function block, a visualisation is prepared in CoDeSys.
Siehe 'Visualization of AFL blocks' auf Seite 806.

4.2.2.1.4 ID_WRITE_10IDS_DINT (FB)

The function block' ID_WRITE_10IDS_DINT' writes parameters, but not list parameters, in AMK devices. Up to 10 ID numbers in the format double integer (DINT) can be written. Every input for an ID number is assigned an input for the ID value.
The function block is called in the asynchronous program level PLC_PRG.

User interface

		ID_WRITE_10IDS_DINT			
FB execution	-	boExec	boDone	-	Ackn. "FB done"
Write ID	-	uiIDNo1	boErr	-	Error
ID value	-	diValNo1	iErrID	-	Error ID
Write ID	-	uiIDNo2	enErrName	-	Name of faulty FB
ID value	-	diValNo2			
:	-	:			
:	-	:			
Write ID	-	uiIDNo10			
ID value	-	diValNo10			
Parameter instance	-	uiParInst			
Device structure	-	stDevice	stDevice	-	Device structure

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.
uiIDNo1 ... uiIDNo10	UINT	ID number to be read out
diValNo1 ... diValNo10	DINT	ID value to be written for the ID number specified at input n
uiParInst	UINT	Parameter set number or instance number

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
enErrName	ENUM	EN_FB_NAME Name of faulty block for which the diagnostic number is output. The diagnostic number is explained in the description of the corresponding library. Block: WRITE_ID_DINT; Library: AmkSystem.lib

Input and output variables

Name	Type	Description
stDevice	STRUCT	ST_DEVICE The device description structure assigns the block a device. (See document Software description AmkBase Bibliothek, Part no.204986)

Usage note in the CoDeSys program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
ID_WRITE_10IDS_DINT	

Associated with this function block, a visualisation is prepared in CoDeSys.
Siehe 'Visualization of AFL blocks' auf Seite 806.

4.2.2.1.5 ID_WRITE_5IDS_DINT (FB)

The function block 'ID_WRITE_5IDS_DINT' writes parameters, but not list parameters, in AMK devices. Up to 5 ID numbers in the format double integer (DINT) can be written. Every input for an ID number is assigned an input for the ID value.

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

User interface

	ID_WRITE_5IDS_DINT			
FB execution -	boExec	boDone	-	Ackn. "FB done"
Write ID -	uiIDNo1	boErr	-	Error
ID value -	diValNo1	iErrID	-	Error ID
Write ID -	uiIDNo2	enErrName	-	Name of faulty FB
ID value -	diValNo2			
: -	:			
: -	:			
Write ID -	uiIDNo5			
ID value -	diValNo5			
Parameter instance -	uiParInst			
Device structure -	stDevice	stDevice	-	Device structure

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.
uiIDNo1 ... uiIDNo5	UINT	ID number to be read out
diValNo1 ... diValNo5	DINT	ID value to be written for the ID number specified at input n
uiParInst	UINT	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
enErrName	ENUM	EN_FB_NAME Name of faulty block for which the diagnostic number is output. The diagnostic number is explained in the description of the corresponding library. Block: WRITE_ID_DINT — Library: AmkSystem.lib

Input and output variables

Name	Type	Description
stDevice	STRUCT	ST_DEVICE The device description structure assigns the block a device. (See document Software description AmkBase Bibliothek, Part no.204986)

Usage note in the CoDeSys program

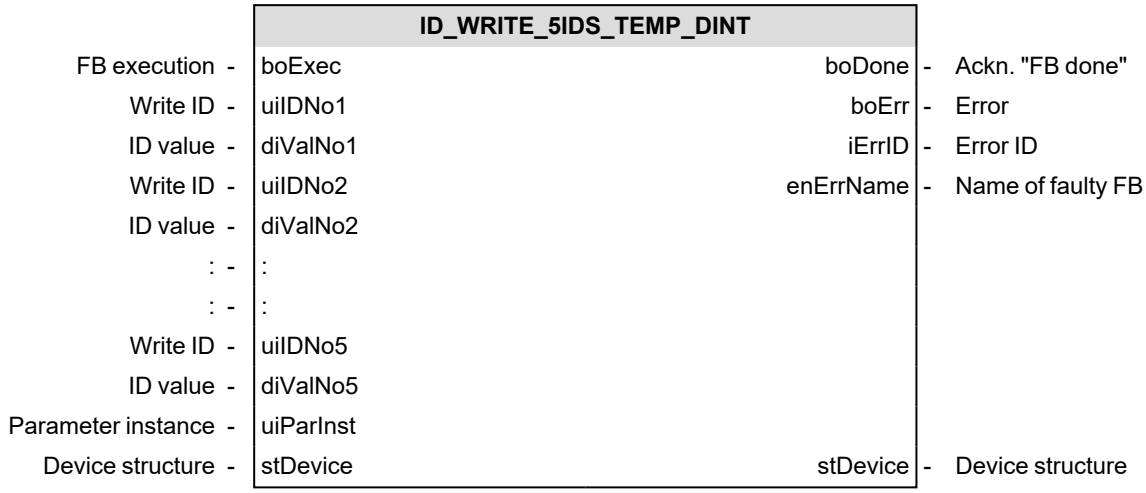
PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
ID_WRITE_5IDS_DINT	

Associated with this function block, a visualisation is prepared in CoDeSys.
Siehe 'Visualization of AFL blocks' auf Seite 806.

4.2.2.1.6 ID_WRITE_5IDS_TEMP_DINT (FB)

The function block 'ID_WRITE_5IDS_TEMP_DINT' writes temporary parameters, but not list parameters, in AMK devices. Up to 5 ID numbers in the format double integer (DINT) can be written. Every input for an ID number is assigned an input for the ID value.
The function block is called in the asynchronous program level PLC_PRG.

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.
uiIDNo1 ... uiIDNo5	UINT	ID number to be written
diValNo1 ... diValNo5	DINT	ID value to be written for the ID number specified at input n
uiParInst	UINT	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
enErrName	ENUM	EN_FB_NAME Name of faulty block for which the diagnostic number is output. The diagnostic number is explained in the description of the corresponding library. Block: WRITE_ID_DINT_TMP; Library: AmkSystem.lib

Input and output variables

Name	Type	Description
stDevice	STRUCT	ST_DEVICE The device description structure assigns the block a device. (See document Software description AmkBase Bibliothek, Part no.204986)

Usage note in the CoDeSys program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
ID_WRITE_5IDS_TEMP_DINT	

Associated with this function block, a visualisation is prepared in CoDeSys.
Siehe 'Visualization of AFL blocks' auf Seite 806.

4.2.2.1.7 ID_WRITE_ID26 (FB)

The function block 'ID_WRITE_ID26' writes the individual elements of ID26, 'Configuration status bits' to AMK devices.
The list elements 2 - 11 are pre-assigned as follows:

ID26 Element Nr.	Code	Meaning	
2	33036	Home position valid	
3	330	$n_{act} = n_{nom}$	within ID157 'Velocity window'
4	331	$n_{act} < n_{min}$	within ID124 'Zero velocity window'
5	332	$n_{act} < n_x$	according to ID125 'Velocity threshold'
6	333	$M_d \geq M_{dx}$	according to ID126 'Torque threshold'
7	334	$M_{nom} \geq M_{limit}$	ID83 'Negative torque limit' ≤ torque setpoint ≤ ID82 'Positive torque limit'
8	335	$n_{nom} \geq n_{limit}$	ID39 'Negative velocity limit' ≤ speed setpoint ≤ ID38 'Positive velocity limit'
9	33074	Warning active group warning	
10		Reserved	
11		Reserved	

For the list elements 12 - 17, additional codes can be selected freely. The codes are described in the parameter description, Binary Outputs chapter.
The function block is called in the asynchronous program level PLC_PRG.

User interface

		ID_WRITE_ID26			
FB execution	-	boExec	boDone	-	Ackn. "FB done"
Code ID26	-	uiID26_12	boErr	-	Error
Code ID26	-	uiID26_13	iErrID	-	Error ID
:	-	:	enErrName	-	Name of faulty FB
:	-	:			
Code ID26	-	uiID26_17			
Parameter instance	-	uiParInst			
Device structure	-	stDevice	stDevice	-	Device structure

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.
uiID26_12 ... uiID26_17	UINT	Transfer of a freely selectable status code for the element n of ID26
uiParInst	UINT	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
enErrName	ENUM	EN_FB_NAME Name of faulty block for which the diagnostic number is output. The diagnostic number is explained in the description of the corresponding library. Block: WRITE_ID_LIST — Library: AmkSystem.lib

Input and output variables

Name	Type	Description
stDevice	STRUCT	ST_DEVICE The device description structure assigns the block a device. (See document Software description AmkBase Bibliothek, Part no.204986)

Usage note in the CoDeSys program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
ID_WRITE_ID26	

Associated with this function block, a visualisation is prepared in CoDeSys.
Siehe 'Visualization of AFL blocks' auf Seite 806.

4.2.2.2 03_Actual Level

4.2.2.2.1 01_Status

4.2.2.2.1.1 GET_STATUSBITS (FB)

The function block 'GET_STATUSBITS' reads the device status QUE, QRF, SBM and additionally 16 freely configurable bit messages from the connected AMK devices. The read status information QUE, QRF, SBM assign the block its outputs, which can be evaluated in the user program. The 16 freely configurable status messages are stored in a structure. The structure is to be transferred as IN_OUT variable.

The freely configurable status bits are specified in parameter ID26. 'Configuration status bits' For configuration, the code of the desired status message is entered in an element of the parameter ID26. The status information is read out via ID144 'Status word'.

The selection list of the status messages is found in the parameter description, Binary Outputs chapter.

The call of this function block is permissible in both the asynchronous program section PLC_PRG as well as the synchronous program section FPLC_PRG.

From AFL version 3.10 onwards, this function block is replaced completely by 'GET_STATUSBITS_ACTUAL_VALUE_GEAR' and therefore should not be used for new applications.

User interface

		GET_STATUSBITS			
FB enable	-	boEnable	boEnabAck	-	Ackn. "FB enable"
			boErr	-	Error
			iErrID	-	Error ID
			enErrName	-	Name of faulty FB
			boQUE	-	Ackn. "DC bus ON"
			boSBM	-	System ready message
			boQRF	-	Ackn."Controller enable"
Configuration status	-	stConfigStatus	stConfigStatus	-	Configuration status
Device structure	-	stDevice	stDevice	-	Device structure

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
iErrID	INT	Error identity number: Diagnostic number is output iErrID = 0: No error iErrID ≠ 0, boErr = TRUE: Error iErrID ≠ 0, boErr = FALSE: Warning

Name	Type	Description
enErrName	ENUM	EN_FB_NAME Name of faulty block for which the diagnostic number is output. The diagnostic number is explained in the description of the corresponding library. **Block** / **Library** GET_STAT_DC_BUSENABLE_ACK_x_QUE / AmkDevAccess.lib GET_STAT_INVERTER_ON_ACK_x_QRF / AmkDevAccess.lib GET_STAT_SYSTEM_READY_X_SBM / AmkDevAccess.lib GET_STATUS_ID144 / AmkDevAccess.lib
boQUE	BOOL	Acknowledgement DC bus ON
boSBM	BOOL	System ready message
boQRF	BOOL	Acknowledgement controller enable

Input and output variables

Name	Type	Description
stConfigStatus	STRUCT	ST_CONFIG_STATUS_ID26 Status information structure boStatusID26_n Status after ID26 element n (n = 2 ... 17)
stDevice	STRUCT	ST_DEVICE The device description structure assigns the block a device. (See document Software description AmkBase Bibliothek, Part no.204986)

Usage note in the CoDeSys program

The call of this function block is permissible in both the asynchronous program section PLC_PRG as well as the synchronous program section FPLC_PRG.

Associated with this function block, a visualisation is prepared in CoDeSys.

Siehe 'Visualization of AFL blocks' auf Seite 806.

4.2.2.2.2 02_Actual_Value

4.2.2.2.2.1 GET_RECTANGULAR_PULSE_INPUT (FB)

The function block 'GET_RECTANGULAR_PULSE_INPUT' reads the rectangular pulse input from connected AMK devices. ID32948, 'Message 4x32' must be configured accordingly.

The function block is called in the synchronous program level FPLC_PRG.

User interface

	GET_RECTANGULAR_PULSE_INPUT			
FB enable -	boEnable	boEnabAck	-	Ackn. "FB enable"
		boErr	-	Error
		iErrID	-	Error ID
		enErrName	-	Name of faulty FB
		boRefPulse	-	Homing pulse
		diOutVal	-	Pulse encoder input
		diOffset	-	Shift Homing.Pulse
Device structure -	stDevice	stDevice	-	Device structure

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
iErrID	INT	Error identity number: Diagnostic number is output iErrID = 0: No error iErrID ≠ 0, boErr = TRUE: Error iErrID ≠ 0, boErr = FALSE: Warning
enErrName	ENUM	EN_FB_NAME Name of faulty block for which the diagnostic number is output. The diagnostic number is explained in the description of the corresponding library. Block: GET_SETPOINT_SCR1 — Library: AmkDevAccess.lib Block: DI_TO_COUNT — Library: AmkBase.lib
boRefPulse	BOOL	Homing pulse: Pulse at which the current counter value is entered in the reference counter
diOutVal	DINT	Current counter value of the pulse encoder [increments]
diOffset	DINT	Offset value [increments] of the counter value to the homing pulse

Input and output variables

Name	Type	Description
stDevice	STRUCT	ST_DEVICE The device description structure assigns the block a device. (See document Software description AmkBase Bibliothek, Part no.204986)

Usage note in the CoDeSys program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
	GET_RECTANGULAR_PULSE_INPUT

Associated with this function block, a visualisation is prepared in CoDeSys.

Siehe 'Visualization of AFL blocks' auf Seite 806.

4.2.2.2.2.2 GET_STATUSBITS_ACTUAL_VALUE_GEAR (FB)

The function block 'GET_STATUSBITS_ACTUAL_VALUE_GEAR' reads the status and actual value information from the connected AMK devices. Here, the set gear behaviour is taken into account.

The status information and the current values are output from the block directly as an output. In addition, the 16 configurable status values are read out of the status list and are available according to their configuration order (order of the codes in ID26) as output. The status codes are described in the parameter description, Binary Outputs chapter.

The function block is called in the synchronous program level FPLC_PRG.

It replaces the application blocks:

- GET_STATUSBITS
- GET_ACTUAL_VALUE

User interface

		GET_STATUSBITS_ACTUAL_VALUE_GEAR			
FB enable	-	boEnable	boEnabAck	-	Ackn. "FB enable"
Gear multiplier	-	diMultiplier	boErr	-	Error
Gear divider	-	udDivider	iErrID	-	Error ID
			enErrName	-	Name of faulty FB
			boQUE	-	Ackn. "DC bus ON"
			boSBM	-	System ready message
			boQRF	-	Ackn."Controller enable"
			boStatusID26_2	-	State ID26, element 2
			boStatusID26_3	-	State ID26, element 3
			:	-	:
			:	-	:
			boStatusID26_17	-	State ID26, element 17
			diActualPosition	-	Actual position value
			diActualVelocity	-	Actual velocity value
			diActualTorque	-	Actual torque value
			diRectangularPulsInput	-	Pulse square-wave encoder
Device structure	-	stDevice	stDevice	-	Device structure

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.
diMultiplier	DINT	Gear multiplier/factor [1]
udDivider	UDINT	Gear divider [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></td></tr>
<tr><td>enErrName</td><td>ENUM</td><td>EN_FB_NAME
Name of faulty block for which the diagnostic number is output.
The diagnostic number is explained in the description of the corresponding library.
<table>
<tr><th>Block</th><th>Library</th></tr>
<tr><td>GET_STAT_DC_BUSENABLE_ACK_x_QUE</td><td>AmkDevAccess.lib</td></tr>
<tr><td>GET_STAT_INVERTER_ON_ACK_x_QRF</td><td>AmkDevAccess.lib</td></tr>
<tr><td>GET_STAT_SYSTEM_READY_x_SBM</td><td>AmkDevAccess.lib</td></tr>
<tr><td>GET_STATUS_ID144</td><td>AmkDevAccess.lib</td></tr>
<tr><td>GET_ACTUAL_POSITION</td><td>AmkDevAccess.lib</td></tr>
<tr><td>GET_ACTUAL_SPEED</td><td>AmkDevAccess.lib</td></tr>
<tr><td>GET_ACTUAL_TORQUE</td><td>AmkDevAccess.lib</td></tr>
</table></td></tr>
<tr><td>boQUE</td><td>BOOL</td><td>Acknowledgement DC bus ON</td></tr>
<tr><td>boSBM</td><td>BOOL</td><td>System ready message</td></tr>
<tr><td>boQRF</td><td>BOOL</td><td>Acknowledgement controller enable</td></tr>
<tr><td>boStatusID26_n</td><td>BOOL</td><td>n = 2 ... 17;</td></tr>
<tr><td>diActualPosition</td><td>DINT</td><td>Actual position value [increments]</td></tr>
<tr><td>diActualVelocity</td><td>DINT</td><td>Actual velocity value [0.0001 rpm]</td></tr>
<tr><td>diActualTorque</td><td>DINT</td><td>Actual torque value [0.1% Mn]</td></tr>
<tr><td>diOutRectangularPulsInput_X132</td><td>DINT</td><td>Actual value of the pulse encoder input X132 [increments]</td></tr>
</table>

Input and output variables

Name	Type	Description
stDevice	STRUCT	ST_DEVICE The device description structure assigns the block a device. (See document Software description AmkBase Bibliothek, Part no.204986)

Usage note in the CoDeSys program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
	GET_STATUSBITS_ACTUAL_VALUE_GEAR

Associated with this function block, a visualisation is prepared in CoDeSys.
Siehe 'Visualization of AFL blocks' auf Seite 806.

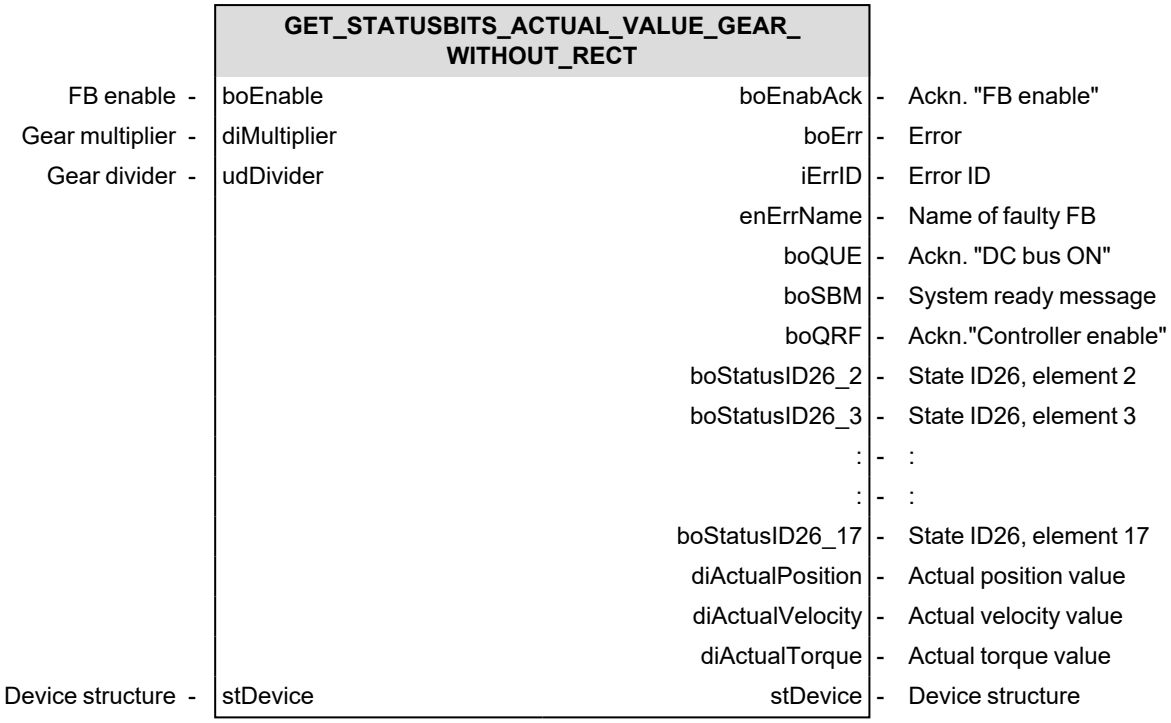
4.2.2.2.2.3 GET_STATUSBITS_ACTUAL_VALUE_GEAR_WITHOUT_RECT (FB)

The function block 'GET_STATUSBITS_ACTUAL_VALUE_GEAR_WITHOUT_RECT' reads the status and actual value information from the connected AMK devices. Here, the set gear behaviour is taken into account.

The status information and the current values are output from the block directly as an output. In addition, the 16 configurable status values are read out of the status list and are available according to their configuration order (order of the codes in ID26) as output. The status codes are described in the parameter description, Binary Outputs chapter.

The function block is called in the synchronous program level FPLC_PRG.

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.
diMultiplier	DINT	Gear multiplier/factor [1]
udDivider	UDINT	Gear divider [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
iErrID	INT	Error identity number: Diagnostic number is output iErrID = 0: No error iErrID ≠ 0, boErr = TRUE: Error iErrID ≠ 0, boErr = FALSE: Warning

Name	Type	Description
enErrName	ENUM	EN_FB_NAME Name of faulty block for which the diagnostic number is output. The diagnostic number is explained in the description of the corresponding library. **Block** / **Library** GET_STAT_DC_BUSENABLE_ACK_x_QUE / AmkDevAccess.lib GET_STAT_INVERTER_ON_ACK_x_QRF / AmkDevAccess.lib GET_STAT_SYSTEM_READY_x_SBM / AmkDevAccess.lib GET_STATUS_ID144 / AmkDevAccess.lib GET_ACTUAL_POSITION / AmkDevAccess.lib GET_ACTUAL_SPEED / AmkDevAccess.lib GET_ACTUAL_TORQUE / AmkDevAccess.lib
boQUE	BOOL	Acknowledgement DC bus ON
boSBM	BOOL	System ready message
boQRF	BOOL	Acknowledgement controller enable
boStatusID26_n	BOOL	n = 2 ... 17;
diActualPosition	DINT	Actual position value [increments]
diActualVelocity	DINT	Actual velocity value [0.0001 rpm]
diActualTorque	DINT	Actual torque value [0.1% Mn]

Input and output variables

Name	Type	Description
stDevice	STRUCT	ST_DEVICE The device description structure assigns the block a device. (See document Software description AmkBase Bibliothek, Part no.204986)

Usage note in the CoDeSys program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
	GET_STATUSBITS_ACTUAL_VALUE_GEAR_WITHOUT_RECT

Associated with this function block, a visualisation is prepared in CoDeSys.
Siehe 'Visualization of AFL blocks' auf Seite 806.

4.2.2.3 04_Setpoint_Level

4.2.2.3.1 01_Setpoint

4.2.2.3.1.1 SET_SET_POINTS_AND_FEED_FORWARD_GEAR (FB)

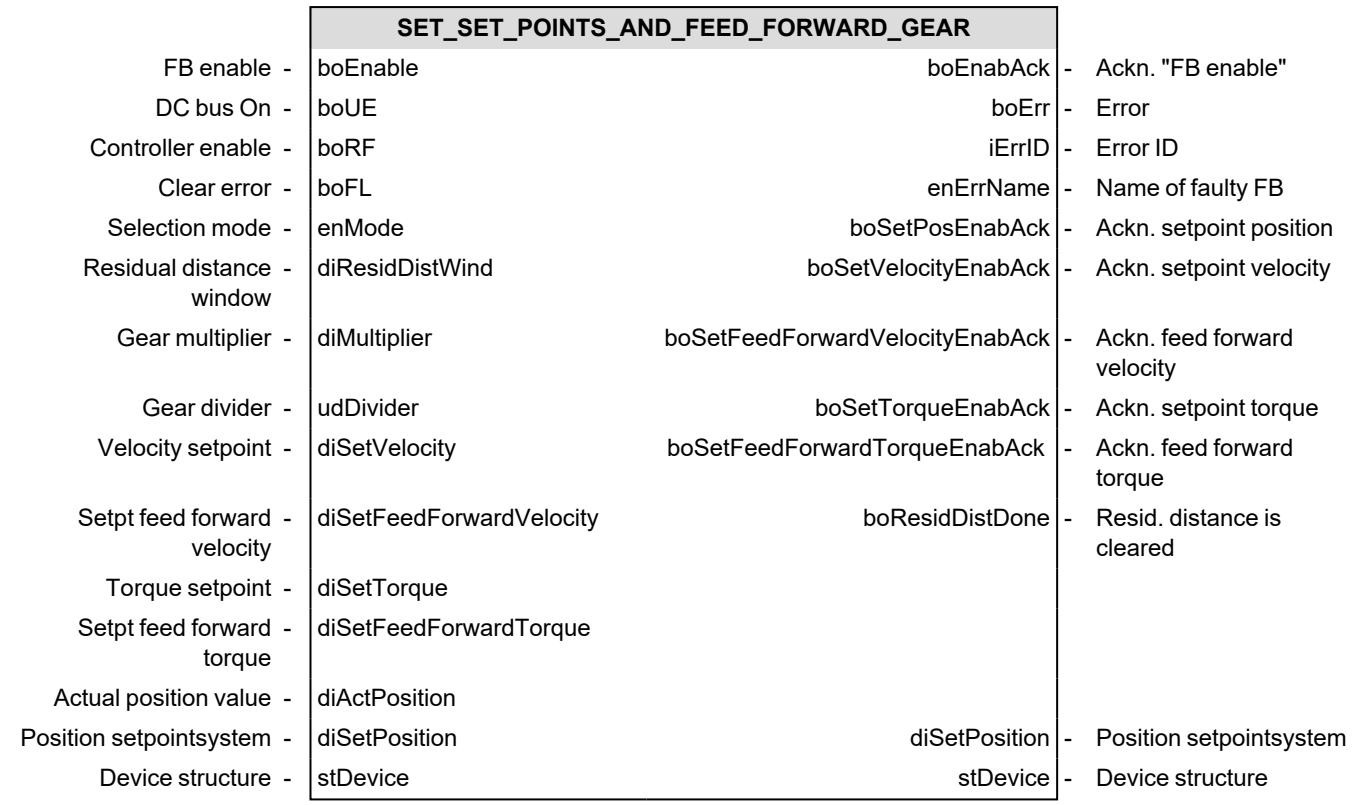
The function block 'SET_SET_POINTS_AND_FEED_FORWARD_GEAR' writes control bits, setpoints and feedforward values in AMK devices.

In addition, the position setpoint system is adjusted by a gear stage.

The function block is called in the synchronous program level FPLC_PRG.

It replaces the block:
SET_SET_POINTS_AND_FEED_FORWARD

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.
boUE	BOOL	Bit for switching the converter (DC bus voltage on). The corresponding return message QUE is a precondition for RF.
boRF	BOOL	Bit for enabling the controller
boFL	BOOL	Bit for error deletion
enMode	ENUM	EN_SETPOINT Selection mode AMK_SETPOINT_NONE: Setpoint sinks deactivated AMK_SETPOINT_POSITION: Position setpoint sink active AMK_SETPOINT_VELOCITY: Speed setpoint sink active AMK_SETPOINT_TORQUE: Torque setpoint sink active
diResidDistWind	DINT	The residual distance window specifies the area within which the axis returns to the initial position for RF return. This settings is only relevant for position setpoint specification.
diMultiplier	DINT	Gear multiplier/factor [1]
udDivider	UDINT	Gear divider [1]
diSetVelocity	DINT	Velocity setpoint [0.0001 rpm]
diSetFeedForwardVelocity	DINT	Velocity forward-control value [0.0001 rpm]
diSetTorque	DINT	Specification of the torque setpoint [0.1% Mn]

Name	Type	Description
diSetFeedForwardTorque	DINT	Torque feed-forward control [0.1% Mn]
diActPosition	DINT	Actual position value [increments]
diSetPosition	DINT	Specification of the position setpoint (position setpoint system) [increments]

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
enErrName	ENUM	EN_FB_NAME Name of faulty block for which the diagnostic number is output. The diagnostic number is explained in the description of the corresponding library.
boSetPosEnabAck	BOOL	Acknowledgement of enable for position setpoint.
boSetVelocityEnabAck	BOOL	Acknowledgement of enable for velocity setpoint.
boSetFeedForwardVelocity EnabAck	BOOL	Acknowledgement of enable for velocity control with feedforward
boSetTorqueEnabAck	BOOL	Acknowledgement of enable for torque setpoint.
boSetFeedForwardTorque EnabAck	BOOL	Acknowledgement of enable for torque control with feedforward
boResidDistDone	BOOL	Residual distance cleared.

Block	Library
SET_CTRL_DC_BUSENABLE_x_UE	AmkDevAccess.lib
SET_CTRL_ERR_RESET_x_FL	AmkDevAccess.lib
GET_STAT_ERR_RESET_ACK_x_QFL	AmkDevAccess.lib
SET_CTRL_INVERTER_ON_x_RF	AmkDevAccess.lib
SET_SETPOINT_POSITION	AmkDevAccess.lib
SET_SETPOINT_SPEED	AmkDevAccess.lib
SET_PRE_SETPOINT_SPEED	AmkDevAccess.lib
SET_SETPOINT_TORQUE	AmkDevAccess.lib
SET_PRE_SETPOINT_TORQUE	AmkDevAccess.lib
RATIO_INC_1	AmkBase.lib

Input and output variables

Name	Type	Description
stDevice	STRUCT	ST_DEVICE The device description structure assigns the block a device. (See document Software description AmkBase Bibliothek, Part no.204986)
diSetPosition	DINT	Specification of the position setpoint (position setpoint system) [increments]

Usage note in the CoDeSys program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
	SET_SET_POINTS_AND_FEED_FORWARD_GEAR

Associated with this function block, a visualisation is prepared in CoDeSys.
Siehe 'Visualization of AFL blocks' auf Seite 806.

4.2.2.3.1.2 SET_SET_POINTS_GEAR (FB)

The function block 'SET_SET_POINTS_GEAR' writes control bits and setpoints in AMK devices. The position setpoint system is adjusted by a gear stage.

The specifications of the setpoints must be performed synchronously, therefore the block must be called in the real-time task.

User interface

		SET_SET_POINTS_GEAR			
FB enable	-	boEnable	boEnabAck	-	Ackn. "FB enable"
DC bus On	-	boUE	boErr	-	Error
Controller enable	-	boRF	iErrID	-	Error ID
Clear error	-	boFL	enErrName	-	Name of faulty FB
Selection mode	-	enMode	boSetPosEnabAck	-	Ackn. setpoint position
Residual distance window	-	diResidDistWind	boSetVelocityEnabAck	-	Ackn. setpoint velocity
Gear multiplier	-	diMultiplier	boSetTorqueEnabAck	-	Ackn. setpoint torque
Gear divider	-	udDivider	boResidDistDone	-	Resid. distance is cleared
Velocity setpoint	-	diSetVelocity			
Torque setpoint	-	diSetTorque			
Actual position value	-	diActPosition			
Position setpoint	-	diSetPosition	diSetPosition	-	Position setpoint
Device structure	-	stDevice	stDevice	-	Device structure

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.
boUE	BOOL	Bit for switching the converter (DC bus voltage on). The corresponding return message QUE is a precondition for RF.
boRF	BOOL	Bit for enabling the controller
boFL	BOOL	Bit for error deletion
enMode	ENUM	EN_SETPOINT Selection mode AMK_SETPOINT_NONE: Setpoint sinks deactivated AMK_SETPOINT_VELOCITY: Speed setpoint sink active AMK_SETPOINT_POSITION: Position setpoint sink active AMK_SETPOINT_TORQUE: Torque setpoint sink active
diResidDistWind	DINT	The residual distance window specifies the area within which the axis returns to the initial position for RF return. This settings is only relevant for position setpoint specification.

Name	Type	Description
diMultiplier	DINT	Gear multiplier/factor [1]
udDivider	UDINT	Gear divider [1]
diSetVelocity	DINT	Velocity setpoint [0.0001 rpm]
diSetTorque	DINT	Specification of the torque setpoint [0.1% Mn]
diActPosition	DINT	Actual position feedback value [increments]

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
enErrName	ENUM	EN_FB_NAME Name of faulty block for which the diagnostic number is output. The diagnostic number is explained in the description of the corresponding library.
boSetPosEnabAck	BOOL	Acknowledgement of enable for position setpoint.
boSetVelocityEnabAck	BOOL	Acknowledgement of enable for velocity setpoint.
boSetTorqueEnabAck	BOOL	Acknowledgement of enable for torque setpoint.
boResidDistDone	BOOL	Residual distance cleared.

Block	Library
SET_CTRL_DC_BUSENABLE_x_UE	AmkDevAccess.lib
SET_CTRL_ERR_RESET_x_FL	AmkDevAccess.lib
GET_STAT_ERR_RESET_ACK_x_QFL	AmkDevAccess.lib
SET_CTRL_INVERTER_ON_x_RF	AmkDevAccess.lib
SET_SETPOINT_POSITION	AmkDevAccess.lib
SET_SETPOINT_SPEED	AmkDevAccess.lib
SET_SETPOINT_TORQUE	AmkDevAccess.lib
RATIO_INC_1	AmkBase.lib

Input and output variables

Name	Type	Description
diSetPosition	DINT	Specification of the position setpoint (position setpoint system) [increments]
stDevice	STRUCT	ST_DEVICE The device description structure assigns the block a device. (See document Software description AmkBase Bibliothek, Part no.204986)

Usage note in the CoDeSys program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
	SET_SET_POINTS_GEAR

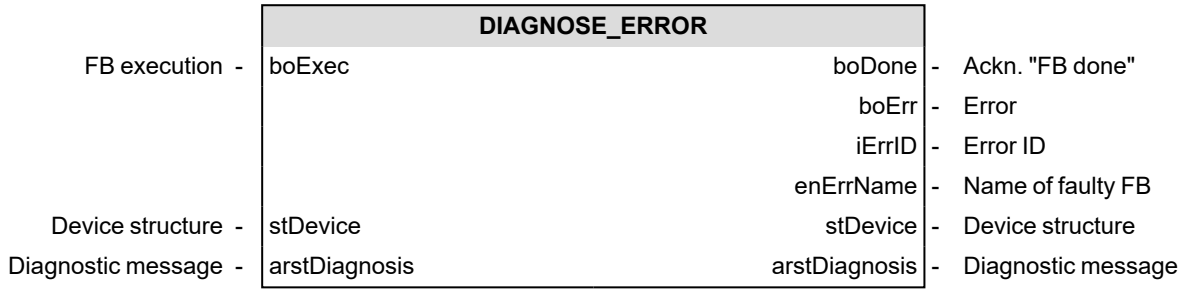
Associated with this function block, a visualisation is prepared in CoDeSys.
Siehe 'Visualization of AFL blocks' auf Seite 806.

4.2.2.4 06_Diagnose

4.2.2.4.1 DIAGNOSE_ERROR (FB)

The function block 'DIAGNOSE_ERROR' reads the error information from AMK devices (ID32840 'Diagnostic list').
The function block is called in the asynchronous program level PLC_PRG.

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.

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
enErrName	ENUM	EN_FB_NAME Name of faulty block for which the diagnostic number is output. The diagnostic number is explained in the description of the corresponding library. Block: READ_ID_LIST; Library: AmkSystem.lib

Input and output variables

Name	Type	Description
stDevice	STRUCT	ST_DEVICE The device description structure assigns the block a device. (See document Software description AmkBase Bibliothek, Part no.204986)

Name	Type	Description
arstDiagnosis	ARRAY	ARRAY [1..10] OF ST_AXIS_ERROR Diagnostic messages

Seq. no.	Diagnostic message	Info 1
1	Diagnosis no. n	
:	:	
10	Diagnosis no. m	

Usage note in the CoDeSys program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
DIAGNOSE_ERROR	

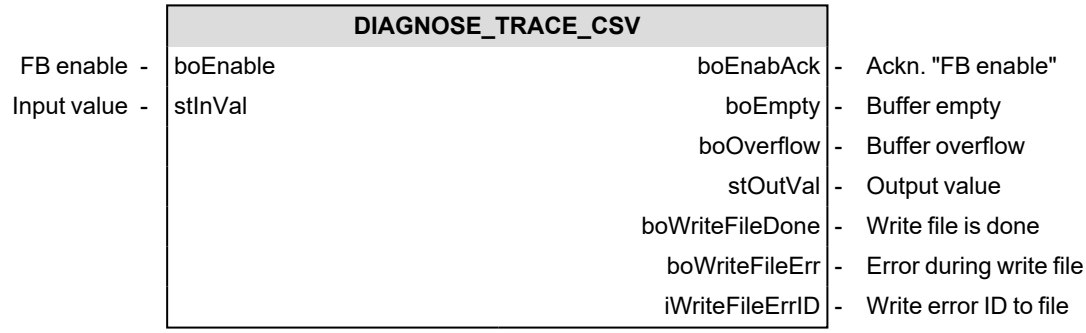
Associated with this function block, a visualisation is prepared in CoDeSys.
Siehe 'Visualization of AFL blocks' auf Seite 806.

4.2.2.4.2 DIAGNOSE_TRACE_CSV (FB)

The function block 'DIAGNOSE_TRACE_CSV' writes a value with 40 entries to a FIFO and saves the data as CSV file in AMK controllers.

The call of this function block is permissible in both the asynchronous program section PLC_PRG as well as the synchronous program section FPLC_PRG.

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.
stInVal	STRUCT	ST_TRACE_CSV Input value

Output variables

Name	Type	Description
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled
boEmpty	BOOL	The FiFo buffer is empty
boOverflow	BOOL	Buffer overflow: A new value is to be saved to the buffer, but no free position is available.

Name	Type	Description
stOutVal	STRUCT	ST_TRACE_CSV Output value
boWriteFileDone	BOOL	Acknowledgement "File was written to the controller"
boWriteFileErr	BOOL	Message "Error while writing file"
iWriteFileErrID	INT	Error identity number: Diagnostic number is output

Usage note in the CoDeSys program

The call of this function block is permissible in both the asynchronous program section PLC_PRG as well as the synchronous program section FPLC_PRG.

Associated with this function block, a visualisation is prepared in CoDeSys.

Siehe 'Visualization of AFL blocks' auf Seite 806.

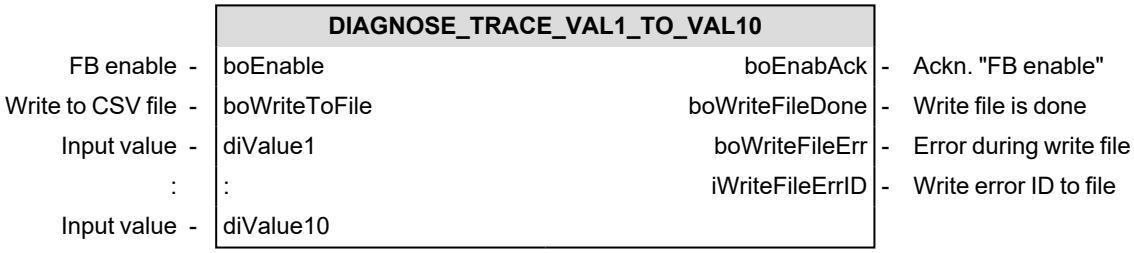
4.2.2.4.3 DIAGNOSE_TRACE_VAL1_TO_VAL10 (FB)

The function block 'DIAGNOSE_TRACE_VAL1_TO_VAL10' writes up to 10 different values with the length of 40 entries to a FIFO and saves the data as CSV file in AMK controllers.

It is based on the block 'DIAGNOSE_TRACE_CSV'.

The call of this function block is permissible in both the asynchronous program section PLC_PRG as well as the synchronous program section FPLC_PRG.

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.
boWriteToFile	BOOL	With a positive edge, the recorded FIFO contents are written to a CSV file on the controller.
diValue1 ... diValue10	DINT	Values that are recorded in a FIFO. With the block 'boEnabAck' = TRUE activated, all inputs values 'diValue1' ... 'diValue10' are written to the FIFO when one or more input values changes.

Output variables

Name	Type	Description
boEnabAck	BOOL	Acknowledgement: Function block is initialised and enabled
boWriteFileDone	BOOL	Acknowledgement "File was written to the controller"
boWriteFileErr	BOOL	Message "Error while writing file"
iWriteFileErrID	INT	Error identity number: Diagnostic number is output

Usage note in the CoDeSys program

The call of this function block is permissible in both the asynchronous program section PLC_PRG as well as the synchronous program section FPLC_PRG.

Associated with this function block, a visualisation is prepared in CoDeSys.
Siehe 'Visualization of AFL blocks' auf Seite 806.

4.2.2.5 old version

4.2.2.5.1 GET_ACTUAL_VALUE (FB)

The function block 'GET_ACTUAL_VALUE' reads the current actual value information

- Position
- Velocity
- Torque

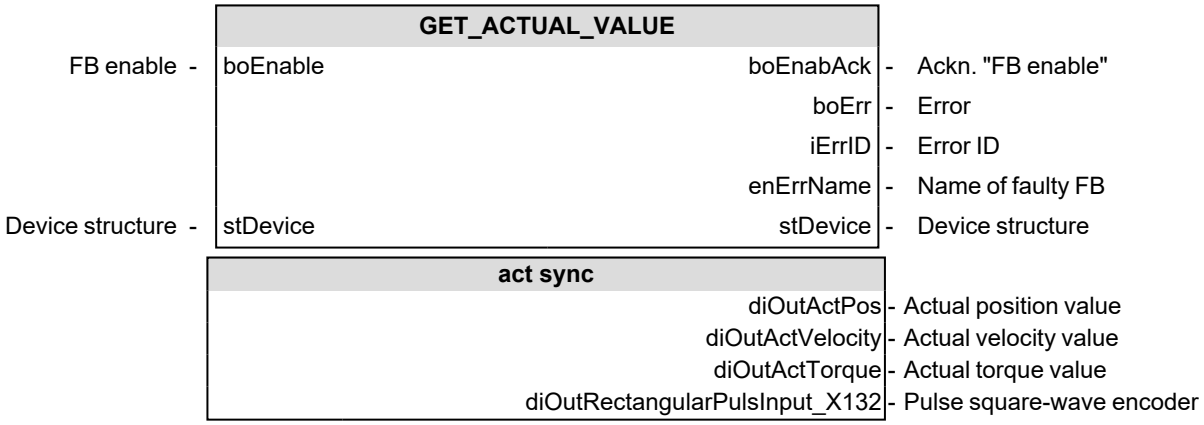
from the connected AMK devices. These values are assigned to the output variables.

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

Setpoints and current values are transferred in a synchronous action. The synchronous action must be called in the synchronous program level FPLC_PRG.

From AFL version 3.10 onwards, this function block is replaced completely by
'GET_STATUSBITS_ACTUAL_VALUE_GEAR'
and therefore should not be used for new applications.

User interface



Input variables of the asynchronous program levels (PLC_PRG)

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 of the asynchronous program levels (PLC_PRG)

<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>enErrName</td><td>ENUM</td><td>EN_FB_NAME
Name of faulty block for which the diagnostic number is output.
The diagnostic number is explained in the description of the corresponding library.
<table><tr><th>Block</th><th>Library</th></tr><tr><td>GET_ACTUAL_POSITION</td><td>AmkDevAccess.lib</td></tr><tr><td>GET_ACTUAL_SPEED</td><td>AmkDevAccess.lib</td></tr><tr><td>GET_ACTUAL_TORQUE</td><td>AmkDevAccess.lib</td></tr></table></td></tr>
</table>

Output variables of the synchronous program levels (FPLC_PRG)

Name	Type	Description
diOutActPos	DINT	Actual position value [increments]
diOutActVelocity	DINT	Actual velocity value [0.0001 rpm]
diOutActTorque	DINT	Actual torque value [0.1% Mn]
diOutRectangularPulsInput_X132	DINT	Actual value of the pulse encoder input X132 [increments]

Input and output variables

Name	Type	Description
stDevice	STRUCT	ST_DEVICE The device description structure assigns the block a device. (See document Software description AmkBase Bibliothek, Part no.204986)

Usage note in the CoDeSys program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
GET_ACTUAL_VALUE	GET_ACTUAL_VALUE.actSync

Associated with this function block, a visualisation is prepared in CoDeSys.
Siehe 'Visualization of AFL blocks' auf Seite 806.

4.2.2.5.2 GET_STATUSBITS_ACTUAL_VALUE (FB)

The function block 'GET_STATUSBITS_ACTUAL_VALUE' reads the status and actual value information from the connected AMK devices.

It combines the application blocks 'GET_STATUSBITS' and 'GET_ACTUAL_VALUE'.

The status information QUE, QRF and SBM are output from the block directly as an output. In addition, the 16 configurable status values from ID144 'Status word' are read out and are available according to their configuration order in a structure. The status codes are described in the parameter description, Binary Outputs chapter.

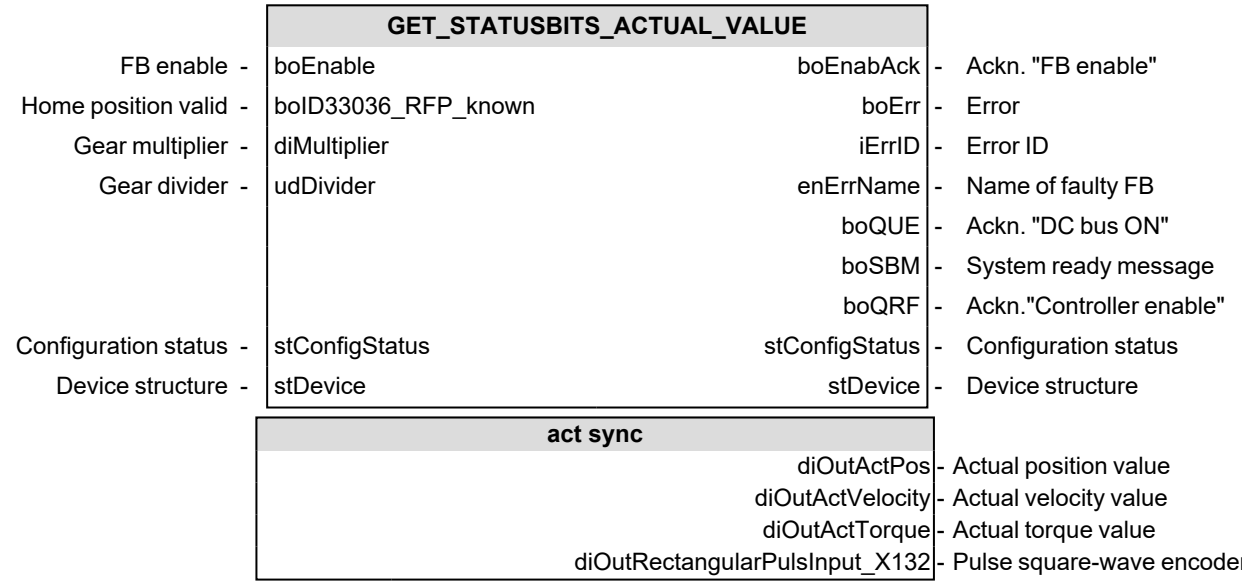
The actual value information current position, current velocity and current torque is also available as output.

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

Setpoints and current values are transferred in a synchronous action. The synchronous action must be called in the synchronous program level FPLC_PRG.

From AFL version 3.10 onwards, this function block is replaced completely by 'GET_STATUSBITS_ACTUAL_VALUE_GEAR' and therefore should not be used for new applications.

User interface



Input variables of the asynchronous program levels (PLC_PRG)

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.
boID33036	BOOL	Home position valid
diMultiplier	DINT	Gear multiplier/factor [1]
udDivider	UDINT	Gear divider [1]

Output variables of the asynchronous program levels (PLC_PRG)

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
enErrName	ENUM	EN_FB_NAME Name of faulty block for which the diagnostic number is output. The diagnostic number is explained in the description of the corresponding library.

Block	Library
GET_STAT_DC_BUSENABLE_ACK_x_QUE	AmkDevAccess.lib
GET_STAT_INVERTER_ON_ACK_x_QRF	AmkDevAccess.lib
GET_STAT_SYSTEM_READY_x_SBM	AmkDevAccess.lib
GET_STATUS_ID144	AmkDevAccess.lib
GET_ACTUAL_POSITION	AmkDevAccess.lib
GET_ACTUAL_SPEED	AmkDevAccess.lib
GET_ACTUAL_TORQUE	AmkDevAccess.lib

Output variables of the synchronous program levels (FPLC_PRG)

Name	Type	Description
diOutActPos	DINT	Actual position value [increments]
diOutActVelocity	DINT	Actual velocity value [0.0001 rpm]
diOutActTorque	DINT	Actual torque value [0.1% Mn]
diOutRectangularPulsInput_X132	DINT	Actual value of the pulse encoder input X132 [increments]

Input and output variables

Name	Type	Description
stConfigStatus	STRUCT	ST_CONFIG_STATUS_ID26 Status information structure boStatusID26_n: Status after ID26 element n (n = 2 ... 17)
stDevice	STRUCT	ST_DEVICE The device description structure assigns the block a device. (See document Software description AmkBase Bibliothek, Part no.204986)

Usage note in the CoDeSys program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
GET_STATUSBITS_ACTUAL_VALUE	GET_STATUSBITS_ACTUAL_VALUE.actSync

Associated with this function block, a visualisation is prepared in CoDeSys.
Siehe 'Visualization of AFL blocks' auf Seite 806.

4.2.2.5.3 SET_CONTROLBITS_SETPOINT_SETVALUES (FB)

The function block 'SET_CONTROLBITS_SETPOINT_SETVALUES' writes control bits and setpoints in AMK devices. It constitutes the position setpoint system in the controller.

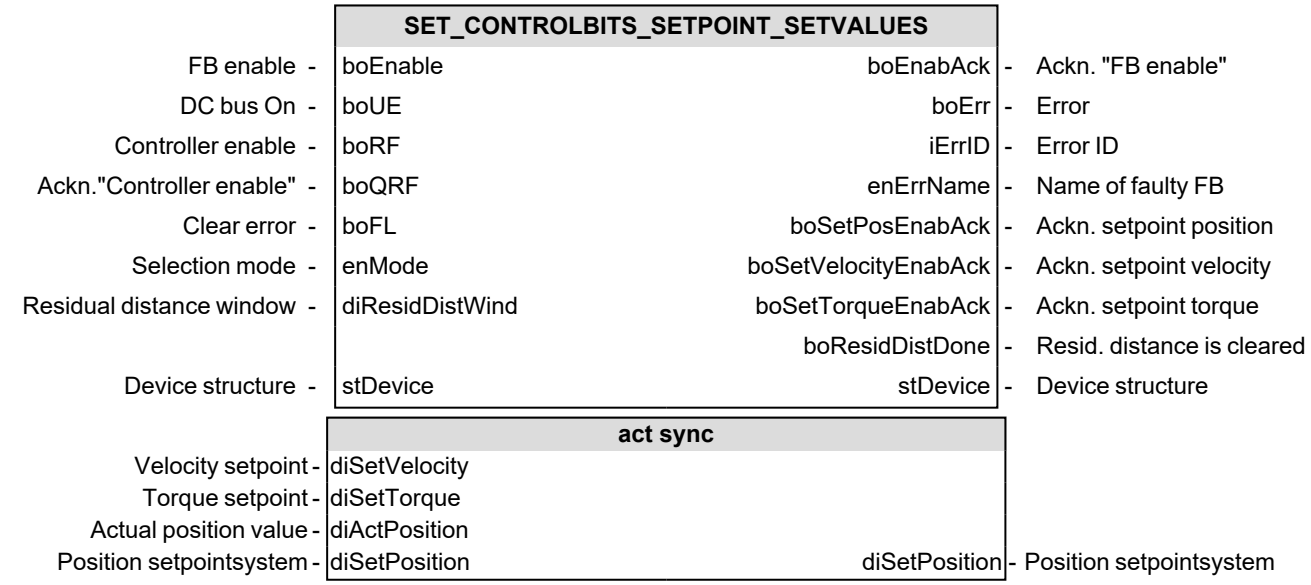
- Position, speed and torque setpoints can be written to the drive with this block.
- The selection of which setpoint is written in the drive is made at the mode input of the block.
  The individual setpoints are block-internally interlocked with respect to one another. This means: Position, speed and torque setpoints can only be written separately and independently from one another to the drive.
- If not setpoint mode is transferred, the setpoint reductions of the PLC are deactivated. The active module switches the drive to secondary operating mode 2 (speed control), ensuring that when the controller enable is switched, no adjustment movement occurs.
- For switching on the controller enable RF, it is necessary to first set the control bit converter On UE. If UE is set, then through the control bit RF the controller enable is switched on.
- Before the block writes the RF control bit to the drive, the position is put on. This prevents an abrupt movement of the axis.
- If the axis should retract to its initial position when RF is switched on, this can be defined by specifying a window. This functionality is only possible in position control.
- If the switch-on position is below the defined window, the residual distance will be cleared and the axis will be energised without movement. Deleting the residual distance is displayed with an output message.
- By setting the error delete bit FL, the drive error is deleted.
- The output bit supplies the block the return messages on the active setpoint sink.

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

Setpoints and current values are transferred in a synchronous action. The synchronous action must be called in the synchronous program level FPLC_PRG.

From AFL version 3.10 onwards, this function block is replaced completely by 'SET_SET_POINTS_GEAR' and therefore should not be used for new applications.

User interface



Input variables of the asynchronous program levels (PLC_PRG)

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
boUE	BOOL	Bit for switching the converter (DC bus voltage on). The corresponding return message QUE is a precondition for RF.
boRF	BOOL	Bit for enabling the controller
boQRF	BOOL	Acknowledgement controller enable
boFL	BOOL	Bit for error deletion
enMode	ENUM	EN_SETPOINT Selection mode AMK_SETPOINT_NONE: Setpoint sinks deactivated AMK_SETPOINT_POSITION: Position setpoint sink active AMK_SETPOINT_VELOCITY: Speed setpoint sink active AMK_SETPOINT_TORQUE: Torque setpoint sink active
diResidDistWind	DINT	The residual distance window specifies the area within which the axis returns to the initial position for RF return. This settings is only relevant for position setpoint specification.

Input variables of the synchronous program levels (FPLC_PRG)

Name	Type	Description
diSetVelocity	DINT	Velocity setpoint [0.0001 rpm]
diSetTorque	DINT	Specification of the torque setpoint [0.1% Mn]
diActPosition	DINT	Actual position value [increments]

Output variables of the asynchronous program levels (PLC_PRG)

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
enErrName	ENUM	EN_FB_NAME Name of faulty block for which the diagnostic number is output. The diagnostic number is explained in the description of the corresponding library.
boSetPosEnabAck	BOOL	Acknowledgement of enable for position setpoint.
boSetVelocityEnabAck	BOOL	Acknowledgement of enable for velocity setpoint.

Block	Library
SET_CTRL_DC_BUSENABLE_x_UE	AmkDevAccess.lib
SET_CTRL_ERR_RESET_x_FL	AmkDevAccess.lib
GET_STAT_ERR_RESET_ACK_x_QFL	AmkDevAccess.lib
SET_CTRL_INVERTER_ON_x_RF	AmkDevAccess.lib
SET_SETPOINT_POSITION	AmkDevAccess.lib
SET_SETPOINT_SPEED	AmkDevAccess.lib
SET_SETPOINT_TORQUE	AmkDevAccess.lib

Name	Type	Description
boSetTorqueEnabAck	BOOL	Acknowledgement of enable for torque setpoint.
boResidDistDone	BOOL	Residual distance cleared.

Input and output variables

Name	Type	Description
diSetPosition	DINT	Specification of the position setpoint (position setpoint system) [increments]
stDevice	STRUCT	ST_DEVICE The device description structure assigns the block a device. (See document Software description AmkBase Bibliothek, Part no.204986)

Usage note in the CoDeSys program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
SET_CONTROLBITS_SETPOINT_SETVALUES	SET_CONTROLBITS_SETPOINT_SETVALUES.actSync

Associated with this function block, a visualisation is prepared in CoDeSys.
Siehe 'Visualization of AFL blocks' auf Seite 806.

4.2.2.5.4 SET_SET_POINTS_AND_FEED_FORWARD (FB)

The function block 'SET_SET_POINTS_AND_FEED_FORWARD' writes control bits, setpoints and pilot control values in AMK devices. It constitutes the functionality of the FB 'SET_CONTROLBITS_SETPOINT_SETVALUES' and expands it with the specification of speed and torque pilot control values.

The torque pilot control values can be specified in all operating modes (position control, speed control, torque control).

Speed pilot control values can be specified in the operating modes position and speed control.

Here, the pilot control channels are always active, meaning that if a value > 0 is written to the corresponding pilot control input, it is passed on directly to the drive. Separate activation of the pilot control specifications is no longer necessary.

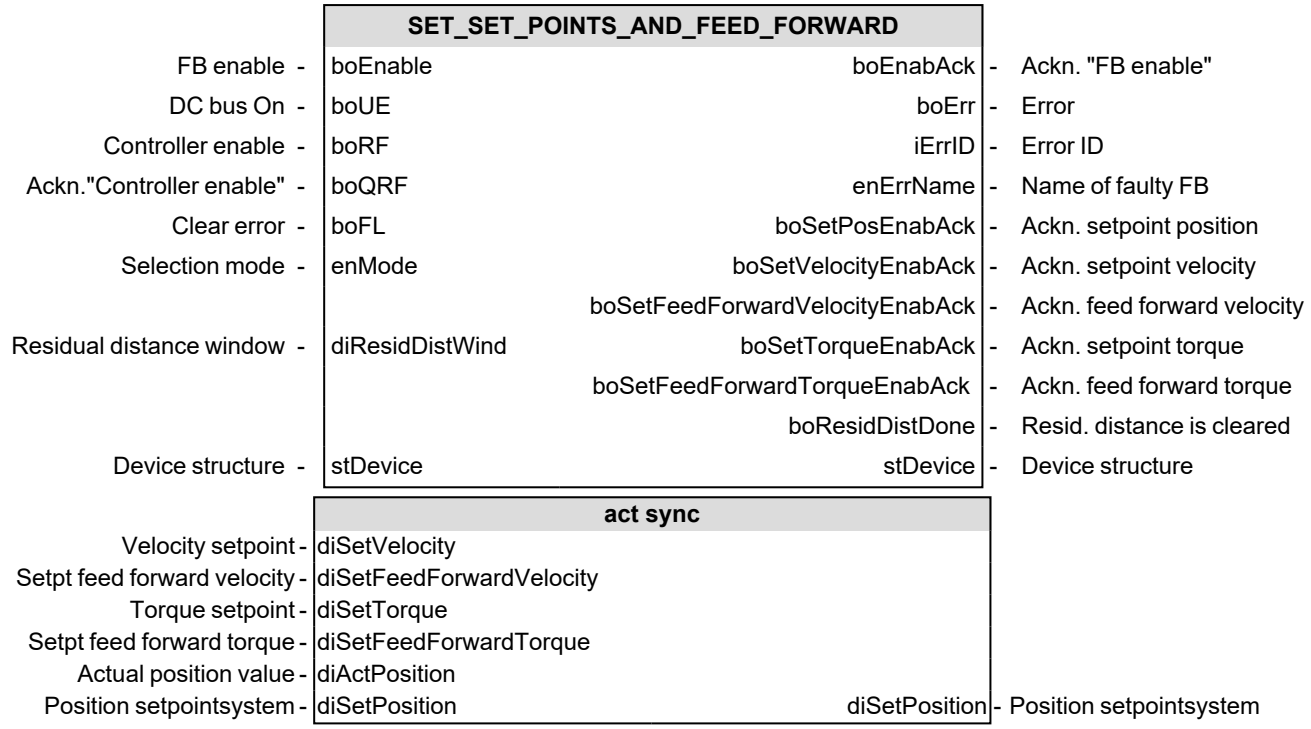
Through the output variables, activation of the control with feedforward is displayed.

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

Setpoints and current values are transferred in a synchronous action. The synchronous action must be called in the synchronous program level FPLC_PRG.

From AFL version 3.10 onwards, this function block is replaced completely by
'SET_SET_POINTS_AND_FEED_FORWARD_GEAR'
and therefore should not be used for new applications.

User interface



Input variables of the asynchronous program levels (PLC_PRG)

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.
boUE	BOOL	Bit for switching the converter (DC bus voltage on). The corresponding return message QUE is a precondition for RF.
boRF	BOOL	Bit for enabling the controller
boQRF	BOOL	Acknowledgement controller enable
boFL	BOOL	Bit for error deletion
enMode	ENUM	EN_SETPOINT Selection mode AMK_SETPOINT_NONE: Setpoint sinks deactivated AMK_SETPOINT_POSITION: Position setpoint sink active AMK_SETPOINT_VELOCITY: Speed setpoint sink active AMK_SETPOINT_TORQUE: Torque setpoint sink active
diResidDistWind	DINT	The residual distance window specifies the area within which the axis returns to the initial position for RF return. This settings is only relevant for position setpoint specification.

Input variables of the synchronous program levels (FPLC_PRG)

Name	Type	Description
diSetVelocity	DINT	Velocity setpoint [0.0001 rpm]
diSetFeedForwardVelocity	DINT	Velocity forward-control value [0.0001 rpm]
diSetTorque	DINT	Specification of the torque setpoint [0.1% Mn]
diSetFeedForwardTorque	DINT	Torque feed-forward control [0.1% Mn]
diActPosition	DINT	Actual position value [increments]

Output variables of the asynchronous program levels (PLC_PRG)

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
enErrName	ENUM	EN_FB_NAME Name of faulty block for which the diagnostic number is output. The diagnostic number is explained in the description of the corresponding library.
boSetPosEnabAck	BOOL	Acknowledgement of enable for position setpoint.
boSetVelocityEnabAck	BOOL	Acknowledgement of enable for velocity setpoint.
boSetFeedForwardVelocity EnabAck	BOOL	Acknowledgement of enable for velocity control with feedforward
boSetTorqueEnabAck	BOOL	Acknowledgement of enable for torque setpoint.
boSetFeedForwardTorque EnabAck	BOOL	Acknowledgement of enable for torque control with feedforward
boResidDistDone	BOOL	Residual distance cleared.

Block	Library
SET_CTRL_DC_BUSENABLE_x_UE	AmkDevAccess.lib
SET_CTRL_ERR_RESET_x_FL	AmkDevAccess.lib
GET_STAT_ERR_RESET_ACK_x_QFL	AmkDevAccess.lib
SET_CTRL_INVERTER_ON_x_RF	AmkDevAccess.lib
SET_SETPOINT_POSITION	AmkDevAccess.lib
SET_SETPOINT_SPEED	AmkDevAccess.lib
SET_PRE_SETPOINT_SPEED	AmkDevAccess.lib
SET_SETPOINT_TORQUE	AmkDevAccess.lib
SET_PRE_SETPOINT_TORQUE	AmkDevAccess.lib

Input and output variables

Name	Type	Description
diSetPosition	DINT	Specification of the position setpoint (position setpoint system) [increments]
stDevice	STRUCT	ST_DEVICE The device description structure assigns the block a device. (See document Software description AmkBase Bibliothek, Part no.204986)

Usage note in the CoDeSys program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
SET_SET_POINTS_AND_FEED_FORWARD	SET_SET_POINTS_AND_FEED_FORWARD.actSync

Associated with this function block, a visualisation is prepared in CoDeSys.
Siehe 'Visualization of AFL blocks' auf Seite 806.

4.2.2.5.5 SET_SETPOINT_POSITION_ABSOLUTE (FB)

The function block 'SET_SETPOINT_POSITION_ABSOLUTE' writes an absolute position setpoint in ID47, 'Position command value' of the drive.

The function block is called in the synchronous program level FPLC_PRG.

From AFL version 3.10 onwards, this function block is replaced completely by 'SET_SETPOINT_POSITION'
and therefore should not be used for new applications.

User interface

	SET_SETPOINT_POSITION_ABSOLUTE			
FB enable -	boEnable	boEnabAck	-	Ackn. "FB enable"
Position setpoint -	diSetPosition	boErr	-	Error
		iErrID	-	Error ID
Device structure -	stDevice	stDevice	-	Device structure

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

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

Input and output variables

Name	Type	Description
stDevice	STRUCT	ST_DEVICE The device description structure assigns the block a device. (See document Software description AmkBase Bibliothek, Part no.204986)

Usage note in the CoDeSys program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
	SET_SETPOINT_POSITION_ABSOLUTE

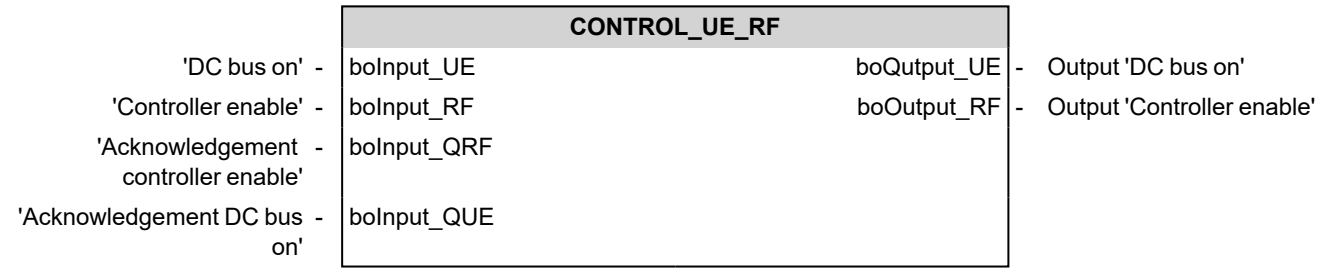
4.2.2.6 Support

4.2.2.6.1 CONTROL_UE_RF (FB)

The function block 'CONTROL_UE_RF' sets the outputs UE 'DC bus on' and RF 'Controller enable' according to the four inputs. The function block is called in the asynchronous program level PLC_PRG.

'Support' functions and function blocks will not be used directly. They are called as subroutines by other functions or function blocks.

User interface



Input variables

Name	Type	Description
boInput_UE	BOOL	Input command 'DC bus on'
boInput_RF	BOOL	Input command 'Controller enable'
boInput_QRF	BOOL	Input 'Acknowledgement controller enable'
boInput_QUE	BOOL	Input 'Acknowledgement DC bus on'

Output variables

Name	Type	Description
boOutput_UE	BOOL	Output 'DC bus on' (boInput_UE = TRUE + delay time 150 ms => boOutput_UE = TRUE)
boOutput_RF	BOOL	Output 'Controller enable' (boInput_UE = TRUE AND boInput_QUE = TRUE AND boInput_RF = TRUE => boOutput_RF = TRUE)

Verwendungshinweis im CoDeSys-Programm

PLC_PRG (asynchroner Programmteil)	FPLC_PRG (synchroner Programmteil)
CONTROL_UE_RF	

4.2.2.6.2 RATIO_INC_ABS (FB)

The function block 'RATIO_INC_ABS' is an absolute counting gear, for which the ratio between input and output increments can be set to any value.

The function block is called in the synchronous program level FPLC_PRG.

'Support' functions and function blocks will not be used directly. They are called as subroutines by other functions or function blocks.

User interface

		RATIO_INC_ABS			
FB enable	-	boEnable	boEnabAck	-	Ackn. "FB enable"
Input value	-	diInVal	boErr	-	Error
Gear multiplier	-	diMultiplier	iErrID	-	Error ID
Gear divider	-	udDivider	diOutVal	-	Output value

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
diMultiplier	DINT	Gear multiplier/factor [1]
udDivider	UDINT	Gear divider [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
iErrID	INT	Error identity number: Diagnostic number is output iErrID = 0: No error iErrID ≠ 0, boErr = TRUE: Error iErrID ≠ 0, boErr = FALSE: Warning
diOutVal	DINT	Output value

Usage note in the CoDeSys program

PLC_PRG (asynchronous part of the program)	FPLC_PRG (synchronous part of the program)
	RATIO_INC_ABS

4.3 FIRST STEPS with AFL Standard blocks

4.3.1 FIRST STEPS with CODESYS V3 and AFL Standard blocks

'FIRST STEPS' describes how:

- to do the AIPEX PRO 'Basic adjustment'
- to generate a new PLC project with AIPEX PRO
- to import AFL Standard Function Blocks and instance for a controller, input and axis
- to access the axis structures (actual and setpoint values)
- to transfer the PLC program

The chapter 'Functions' contains further examples with the following topics:

- IO access
- Additional variable access
- Communication between PLC and PLC
- Cam
- Importing a PLC project
- Control Variants
- Visualization

The chapter 'General' contains basic information such as:

- Automatic bus configuration
- Library Administrator
- Default task configuration
- Diagnostic information
- Device descriptions
- Control Configuration

4.3.1.1 Prerequisites

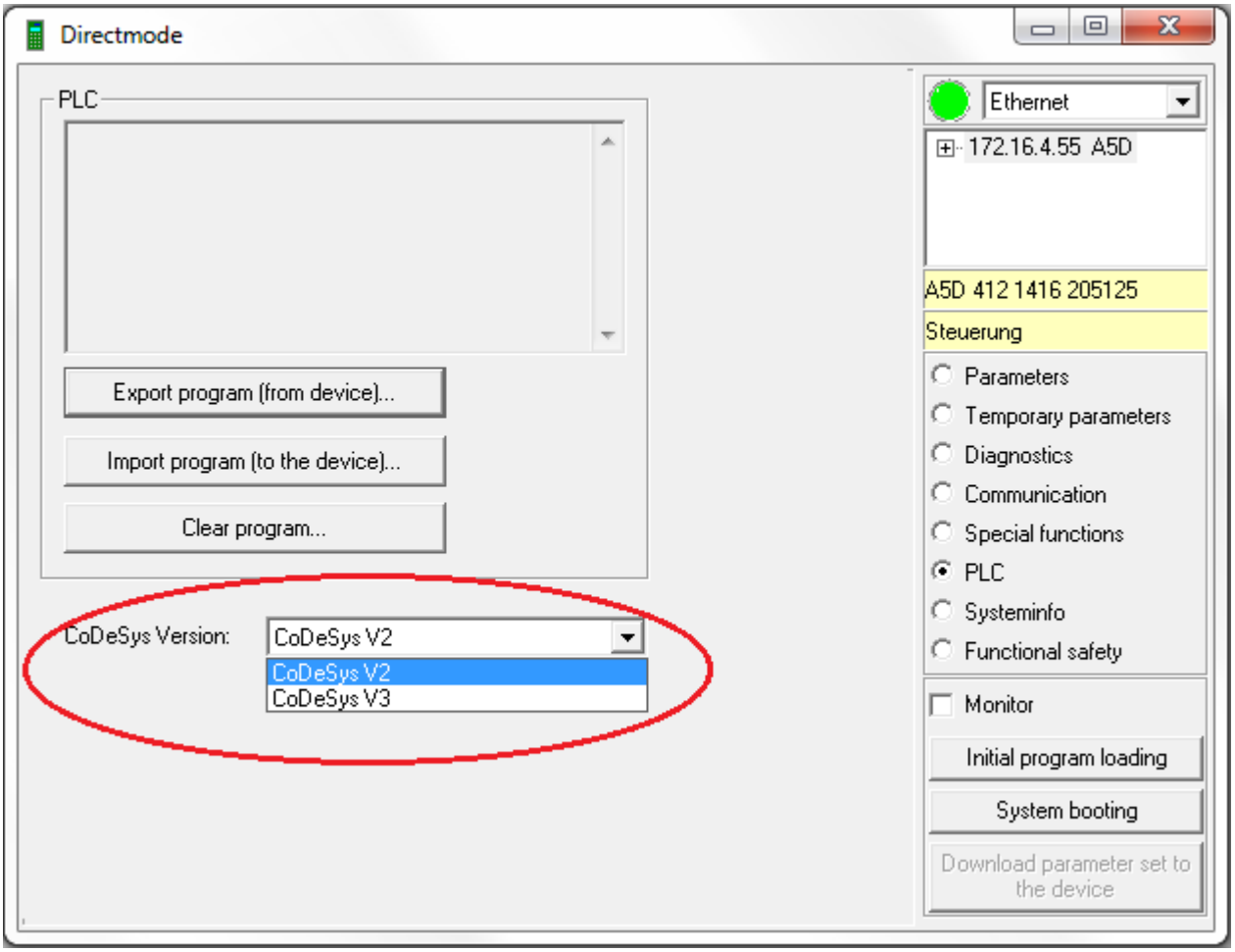
Hardware / Firmware	Software / AFL 4 Version
Hardware: AMKAMAC A4 controller Firmware: ≥ V4.21 2017/08 part-no. 206755	Software: ≥ AIPEX PRO Version 3.03 2015/41 part-no. 205890 AFL Library: ≥ AFL V4 Version 3.5.5.0 2015/41 part-no. 206004
Hardware: AMKAMAC A5 controller Firmware: ≥ A5 V4.11 2013/50 part-no. 204756	Software: ≥ AIPEX PRO Version 3.00 2013/50 part-no. 204905 AFL Library: ≥ AFL V4 Version 3.5.3.0 2014/06 part-no. 204786
Hardware: AMKAMAC A6 controller Firmware: ≥ A6 V4.11 2014/24 part-no. 204760	
Hardware: AMKASMART iSA controller Firmware: ≥ iSA V4.20 2015/26 part-no. 205729	Software: ≥ AIPEX PRO Version 3.03 2015/41 part-no. 205890 AFL Library: ≥ AFL V4 Version 3.5.5.0 2015/41 part-no. 206004

AMK A5 controls with firmware ≥ A5 V4.11 2013/50 AMK part-no. 204756 can be programmed with CODESYS V3 or CODESYS V2. The default setting is CODESYS V2. Change to V3: Siehe 'Version change' auf Seite 638.

4.3.1.2 Version change

AMK A5 controls with firmware A5_411_1350 AMK parts no. 204756 can be programmed with CODESYS V3 or CODESYS V2. The default setting is CODESYS V2. The version change to CODESYS V3 takes place in 'Direct mode'.

Open the Direct mode. Siehe 'Direct mode' auf Seite 122.



Restart the computer to activate the change.

Direct access via parameter

ID34175 'Controller settings' Bit 4 = 1 CODESYS V3

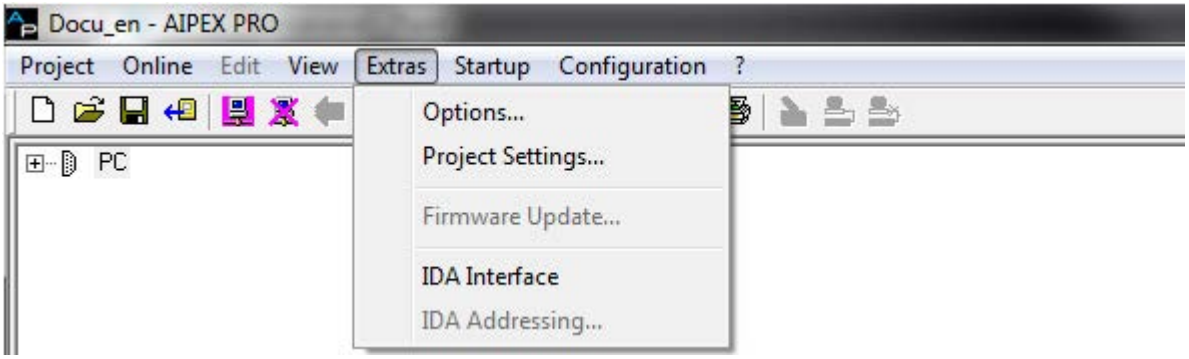
ID34175 'Controller settings' Bit 4 = 0 CODESYS V2

Status bar of display control

The active CODESYS version for display controls is displayed in the status bar during the system reboot.

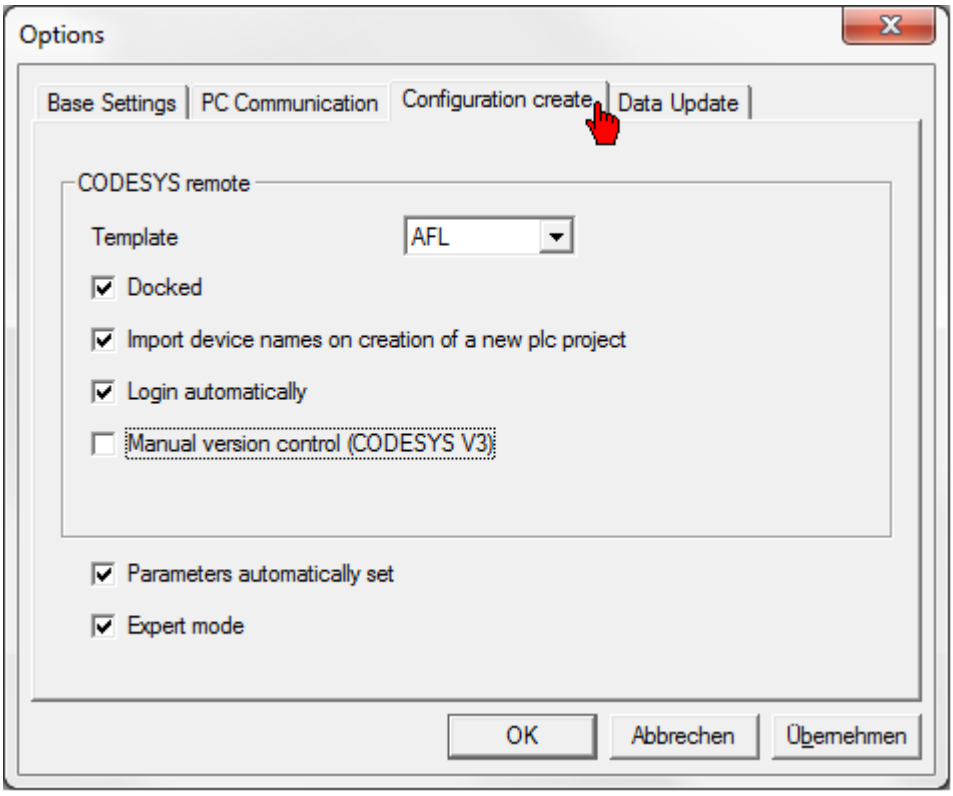
4.3.1.3 Basic AIPEX PRO Settings

Open the menu **'Extras'** → **'Options'**



Implement the following basic settings

The template must be on 'AFL' stand before the CODESYS project is generated.

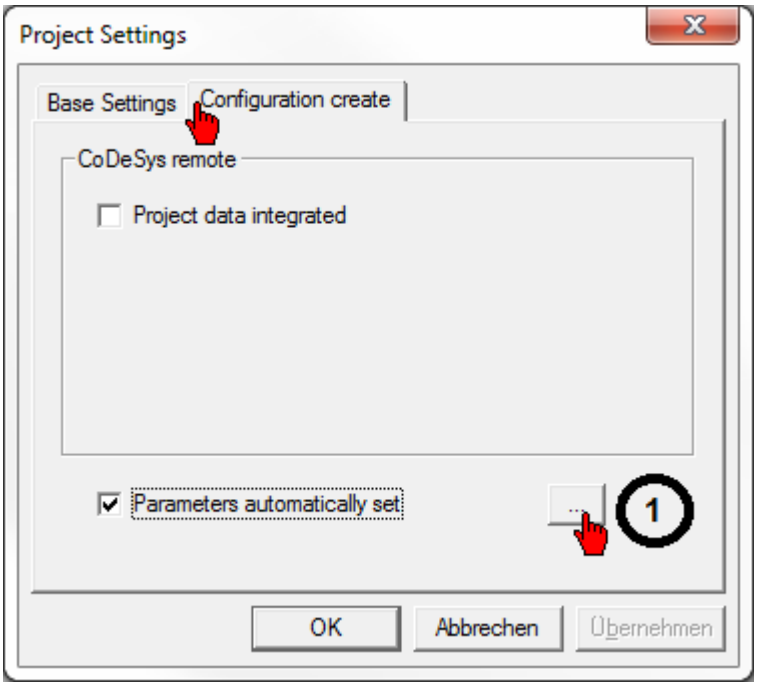


Description of the menu items

Siehe 'Configuration create' auf Seite 103.

Open the menu **'Extras'** → **'Project settings'** → **'Configuration create'**

Implement the following project settings. The project settings can be changed each time before generating.



Description of the menu items:

Siehe 'Project Settings - Configuration create (project specific)' auf Seite 108.

4.3.1.4 Creating PLC project (online project)

This section describes how to create a new PLC project. In the example, AIPEX PRO is connected to a demo model. The device properties are transferred directly from the demo model.

Start AIPEX PRO.

Prerequisite:

'Communication/Icon': Status 'Green' (Interface wide)

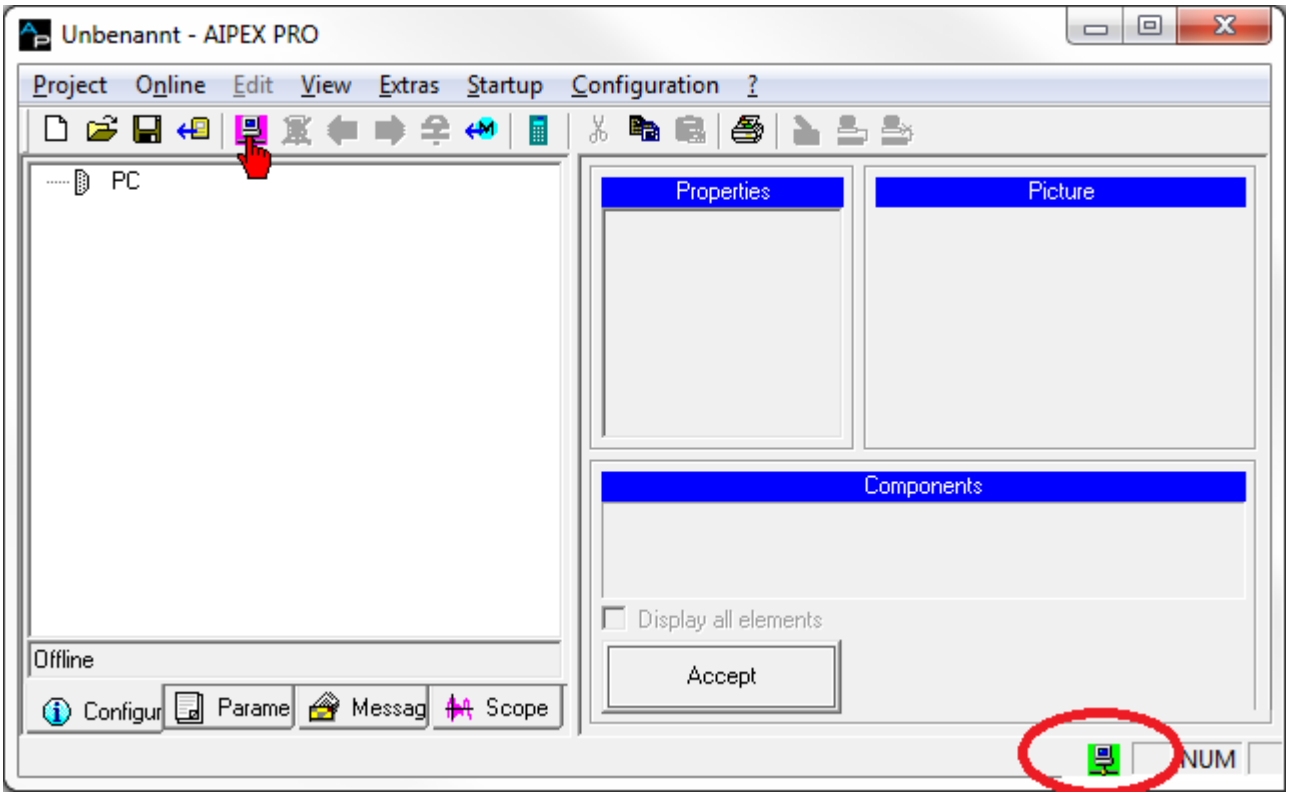
Icon: Status 'Red' or no icon present, check the following points:

Initial start-up of field bus completed:
See document Initial start-up KE/KW (AMK parts no. 204539) document Initial start-up of field bus or document Initial start-up of iC/iX/iDT5 (AMK parts no. 204737) Initial start-up of field bus

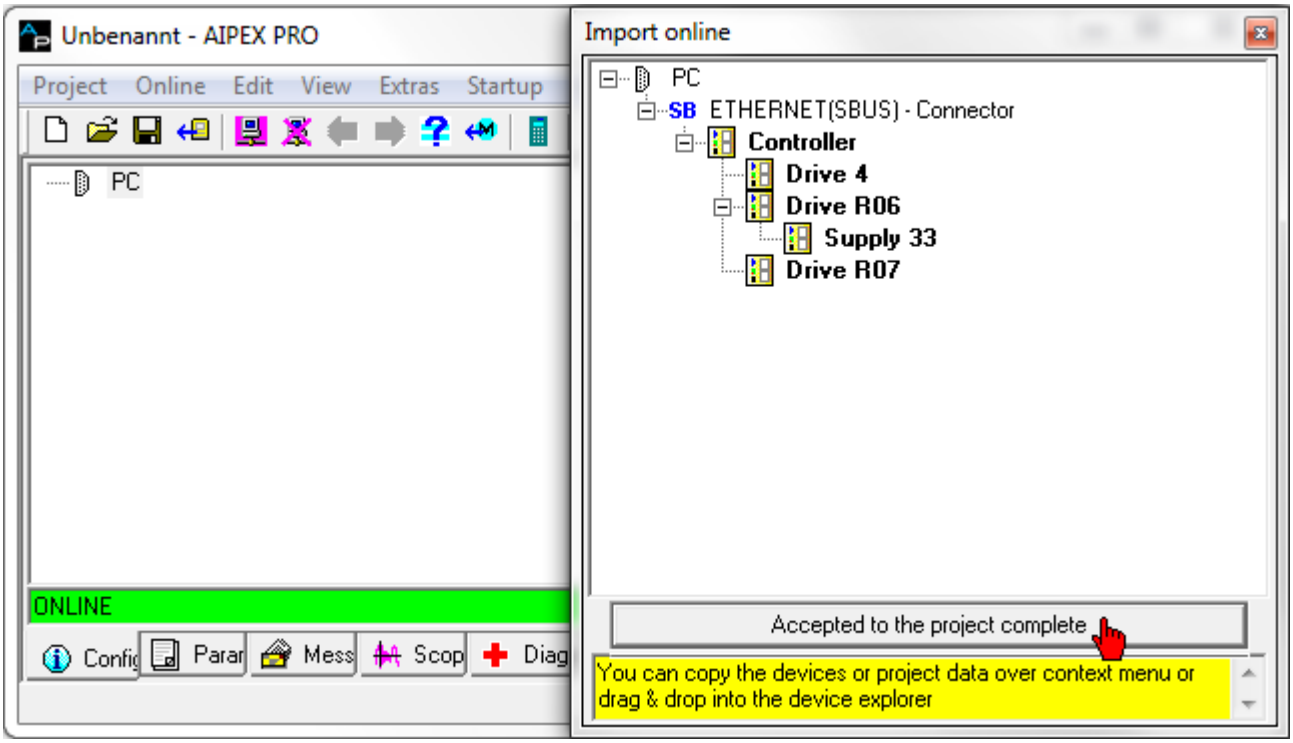
Is direct connection via Ethernet active?:
See document Device description of Controllers A series (AMK parts no. 202975) section Direct connection via Ethernet.

Before creating the AIPEX PRO project, the CODESYS version must be set.
CODESYS Version change: Siehe 'Version change' auf Seite 638.

Press **'Logon'** to create a connection with the connected devices.

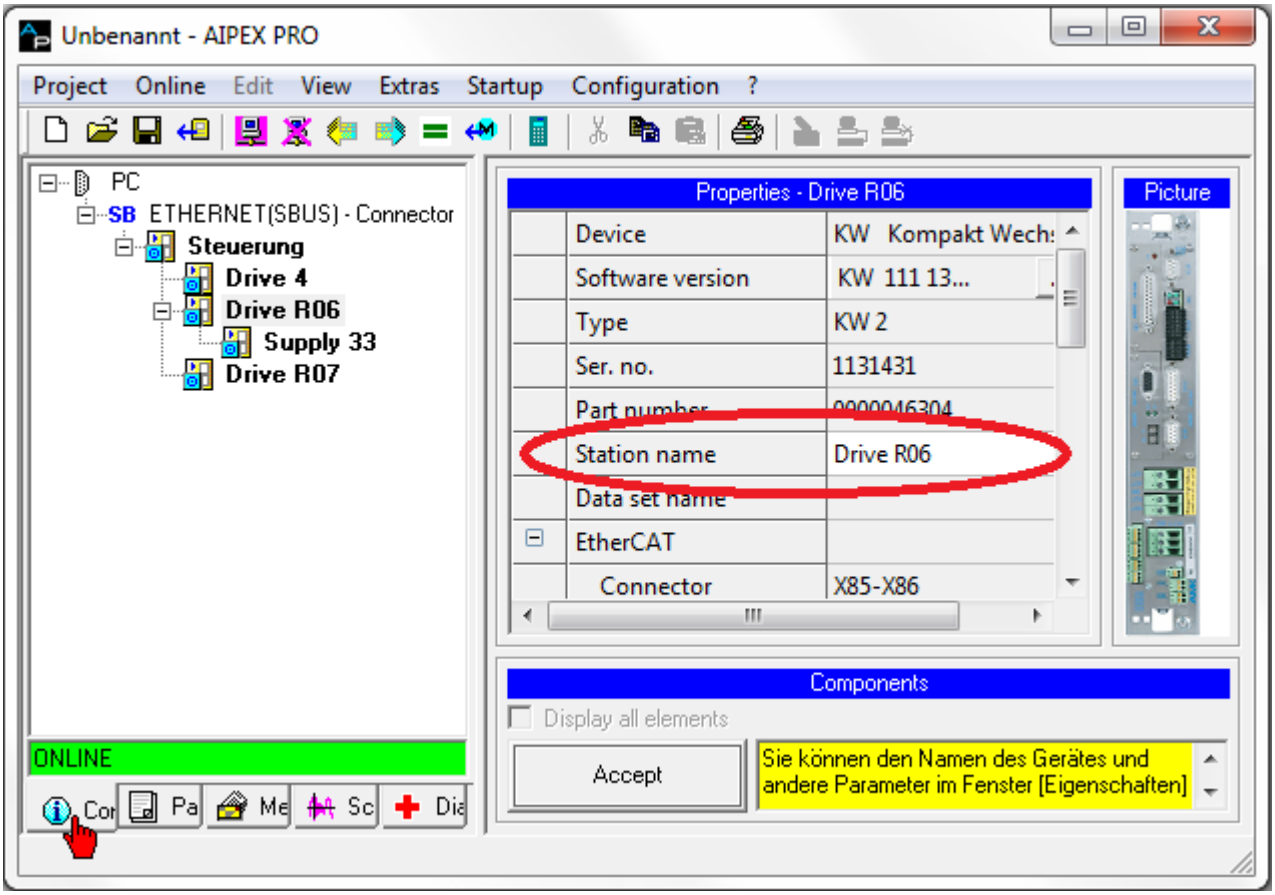


In the 'Import online' window, check whether all physical devices have been detected.
Press **'Accepted to the project complete'** to import the device data.

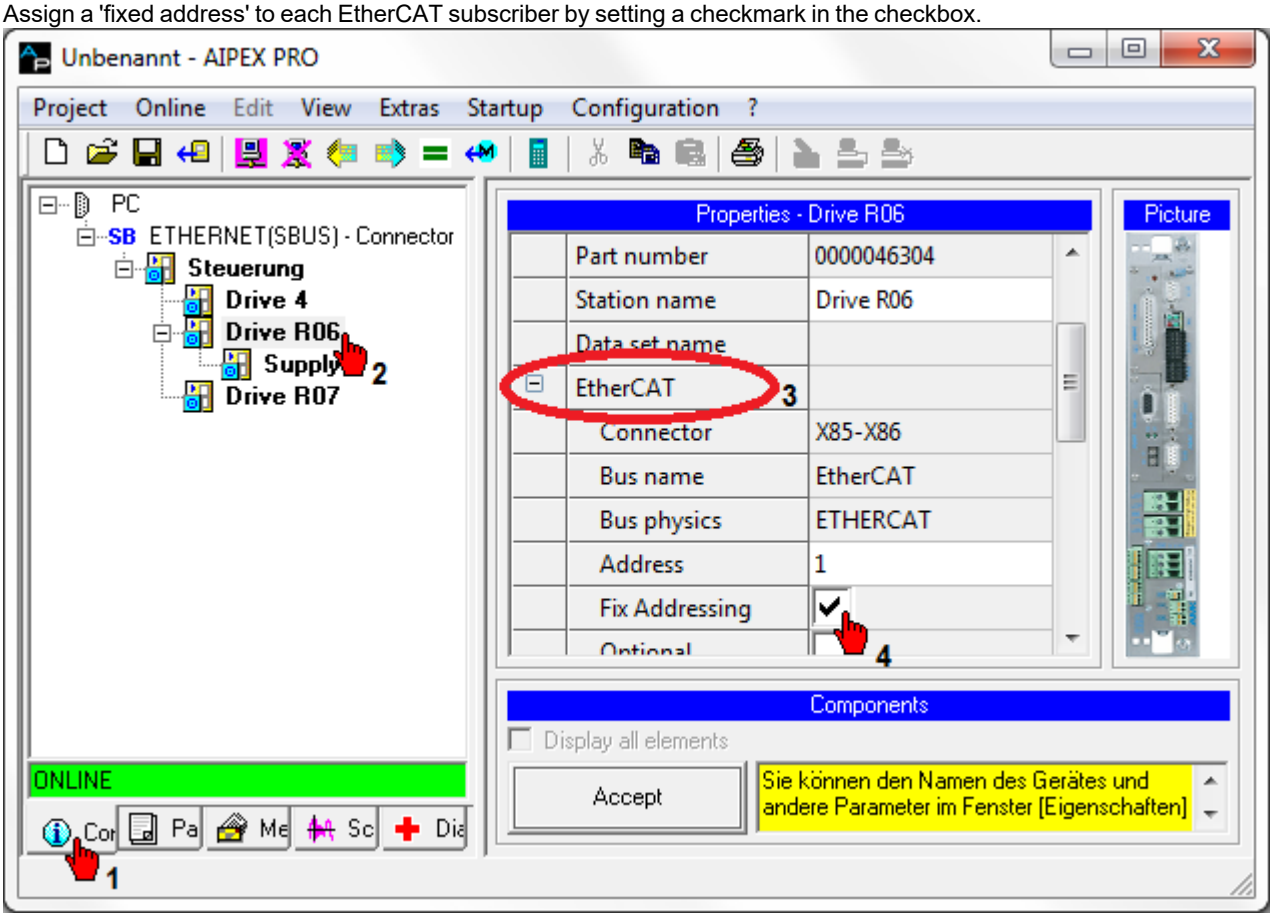


Check the wiring and the system statuses if devices are not shown.

You can customize the device names ('Station names') to your application. The device names are added to the PLC project automatically as symbolic device names (PLC variables).

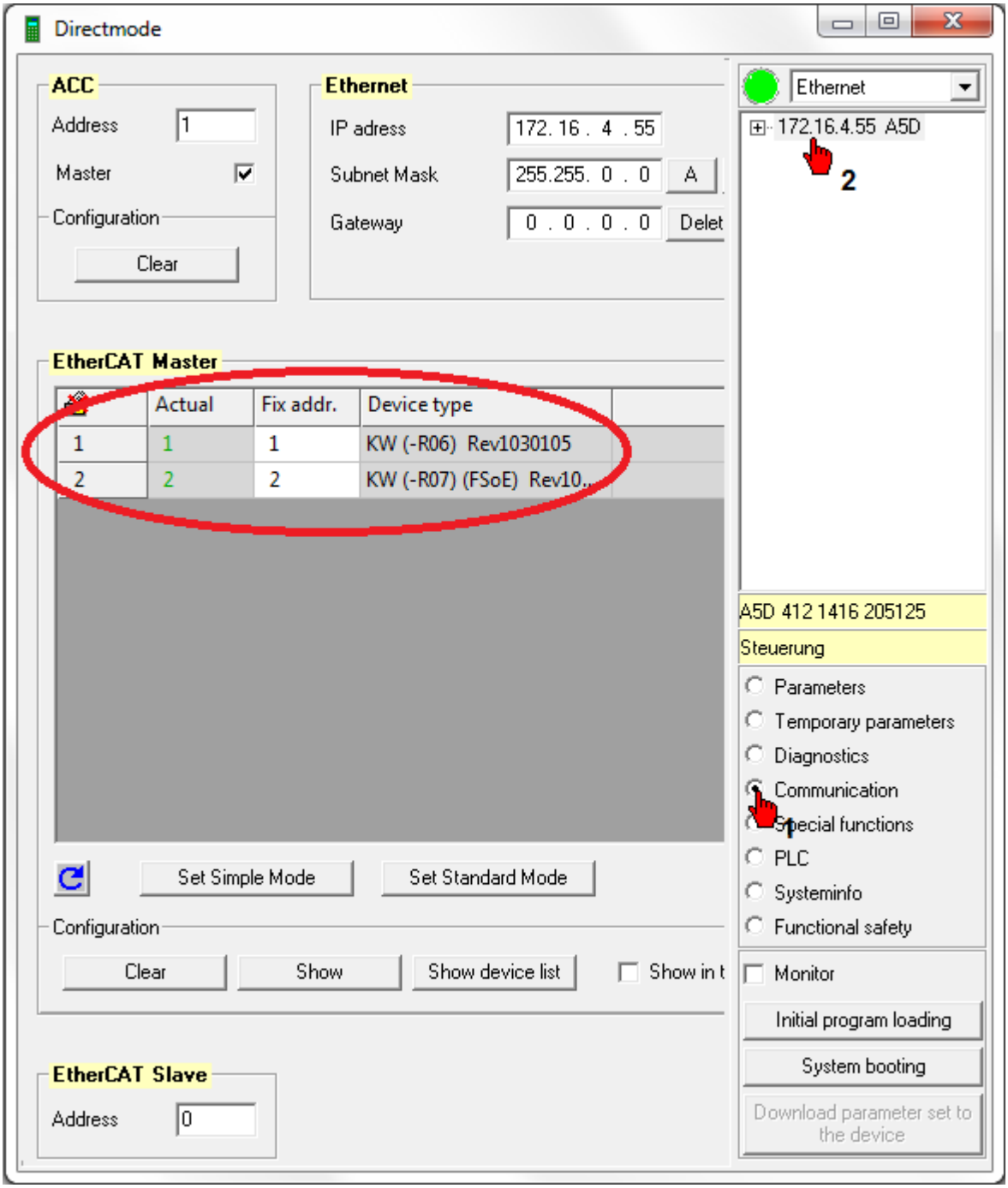


Assign a 'fixed address' to each EtherCAT subscriber by setting a checkmark in the checkbox.

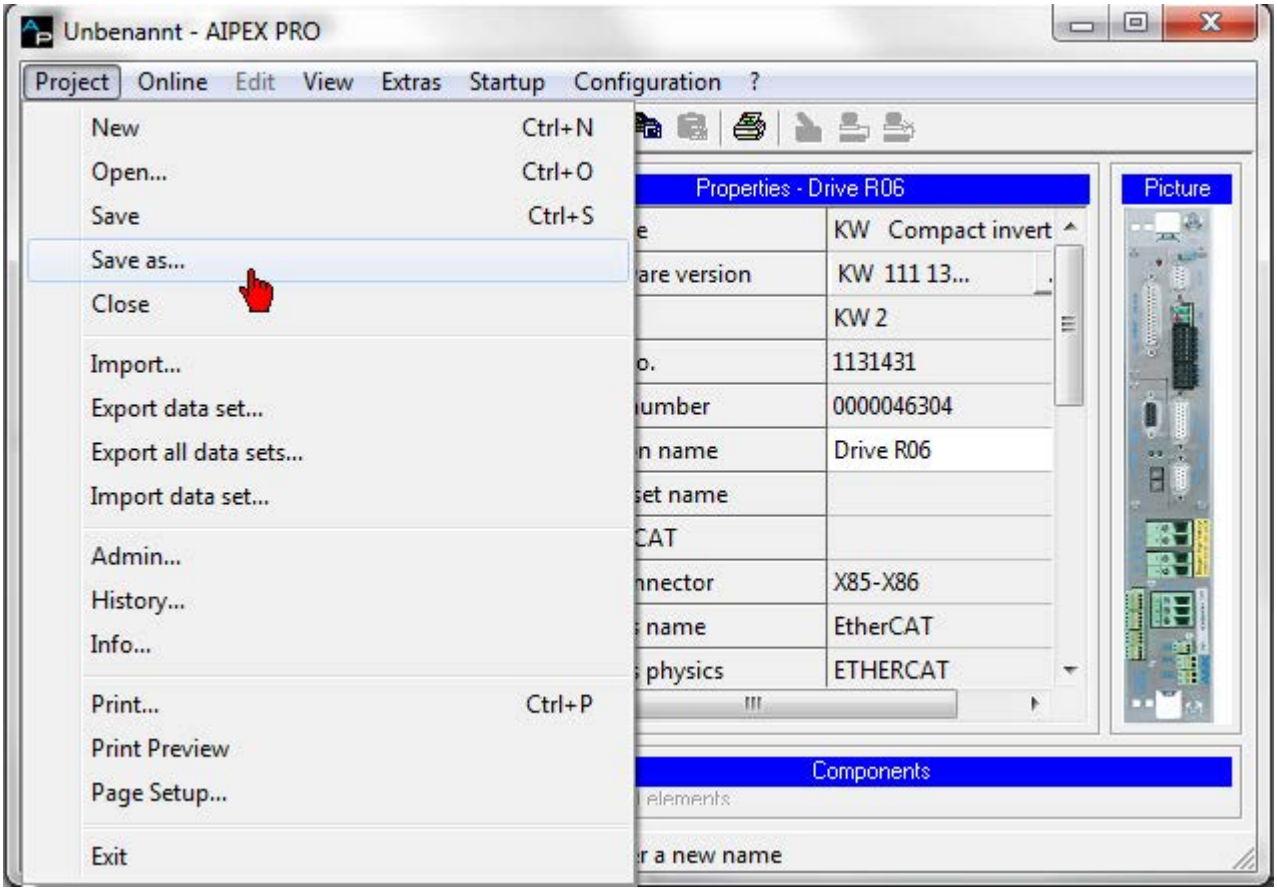


Restart the computer to activate the change.

Once the system reboot is fully completed, you can check the addressing in the **'Direct mode'** → **'Communication'** menu.



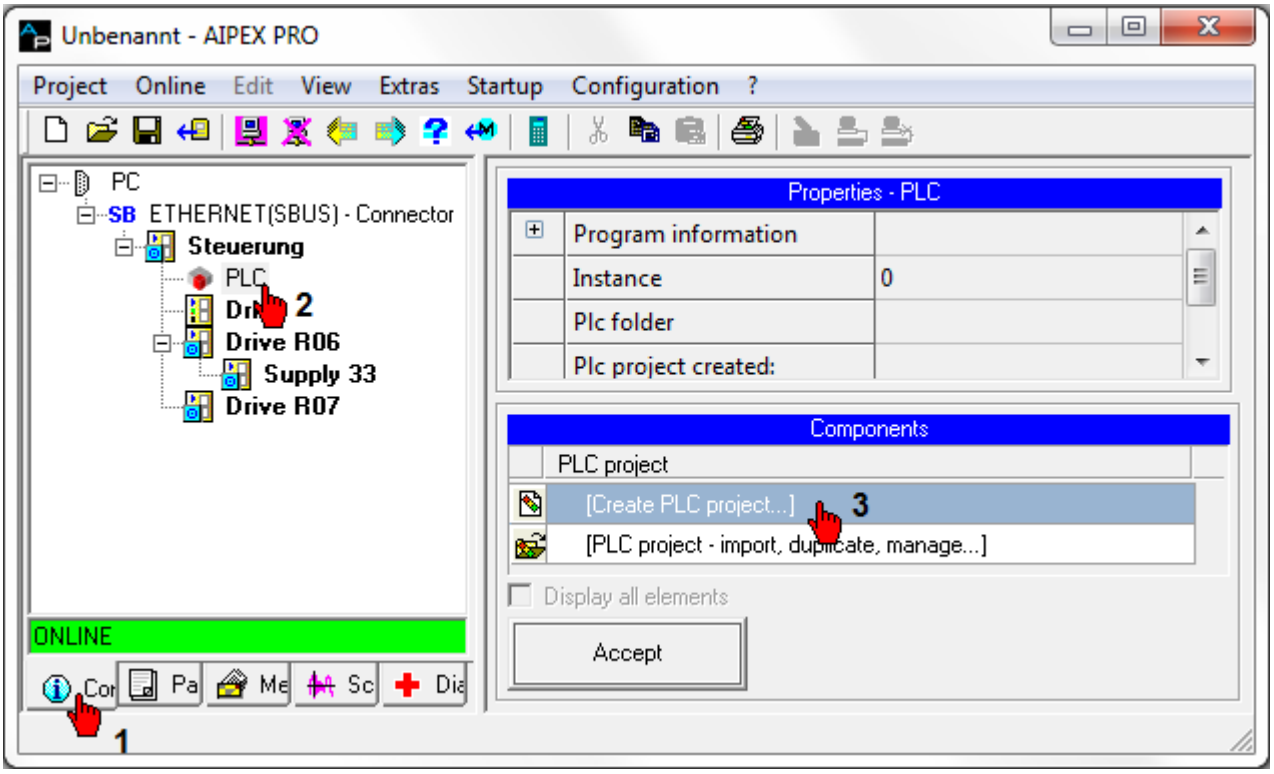
Save the imported project under **'Save as ...'**. You can assign a different name to the PLC project later.



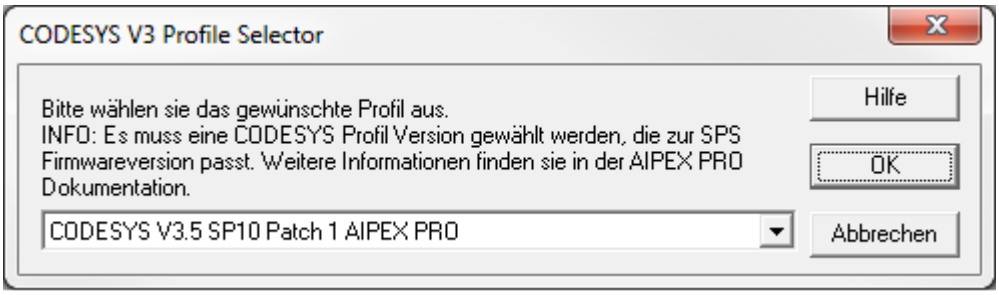
The CODESYS version set on the controller determines which CODESYS version is opened. CODESYS Version change: Siehe 'Version change' auf Seite 638.

If you only change the CODESYS version now, you will have to create the AIPEX PRO project anew. Procedure Menu 'Project'→ 'New'. See beginning of section.

Select the 'PLC' icon in the device tree.
You can then open the PLC CODESYS programming system by selecting **'Create new PLC project...'**.



Icon '**PLC**' missing:
Right-click in the device tree. You can customize the view under 'Select view'.
Siehe 'Program overview - Display filter' auf Seite 39.

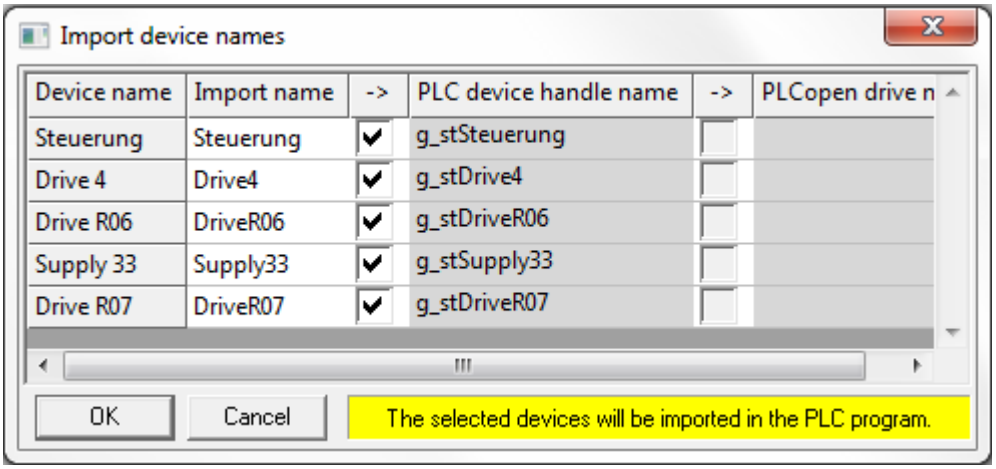


Siehe 'Version control CODESYS V3' auf Seite 709.

Assign a different name to the PLC project

CODESYS V3 starts

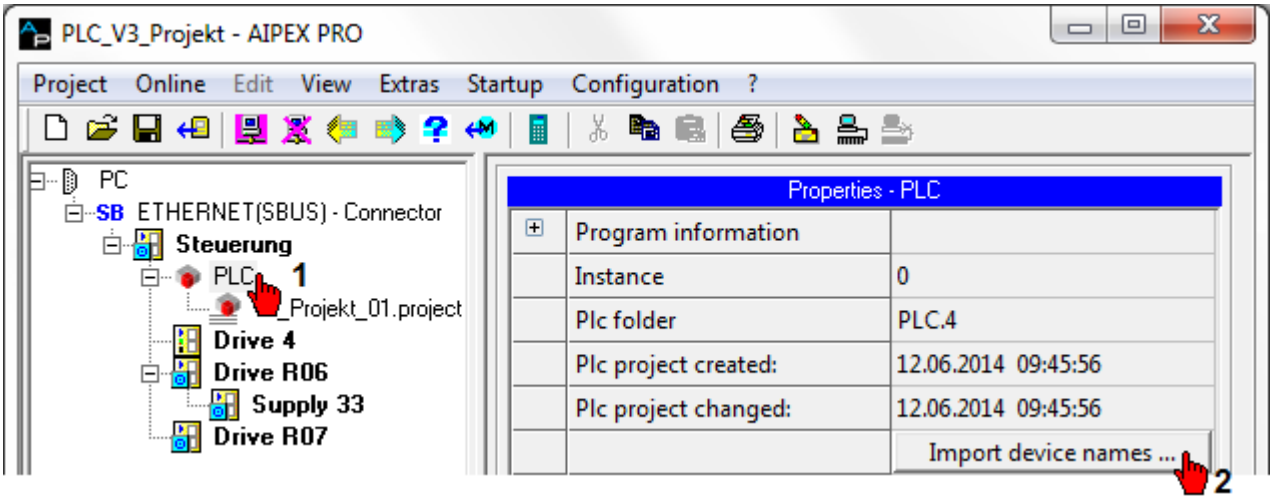
The physically existing device names (from the AIPEX PRO device tree) are imported into the PLC project automatically as 'PLC device handle name' (described in the documentation as symbolic device names). The extension 'g_st' stands for a global structure variable.



This function is activated and deactivated in the menu **'Extras'** → **'Options'** → **'Configuration create'** → **'Import device names when creating a PLC project'**.

Additional information:

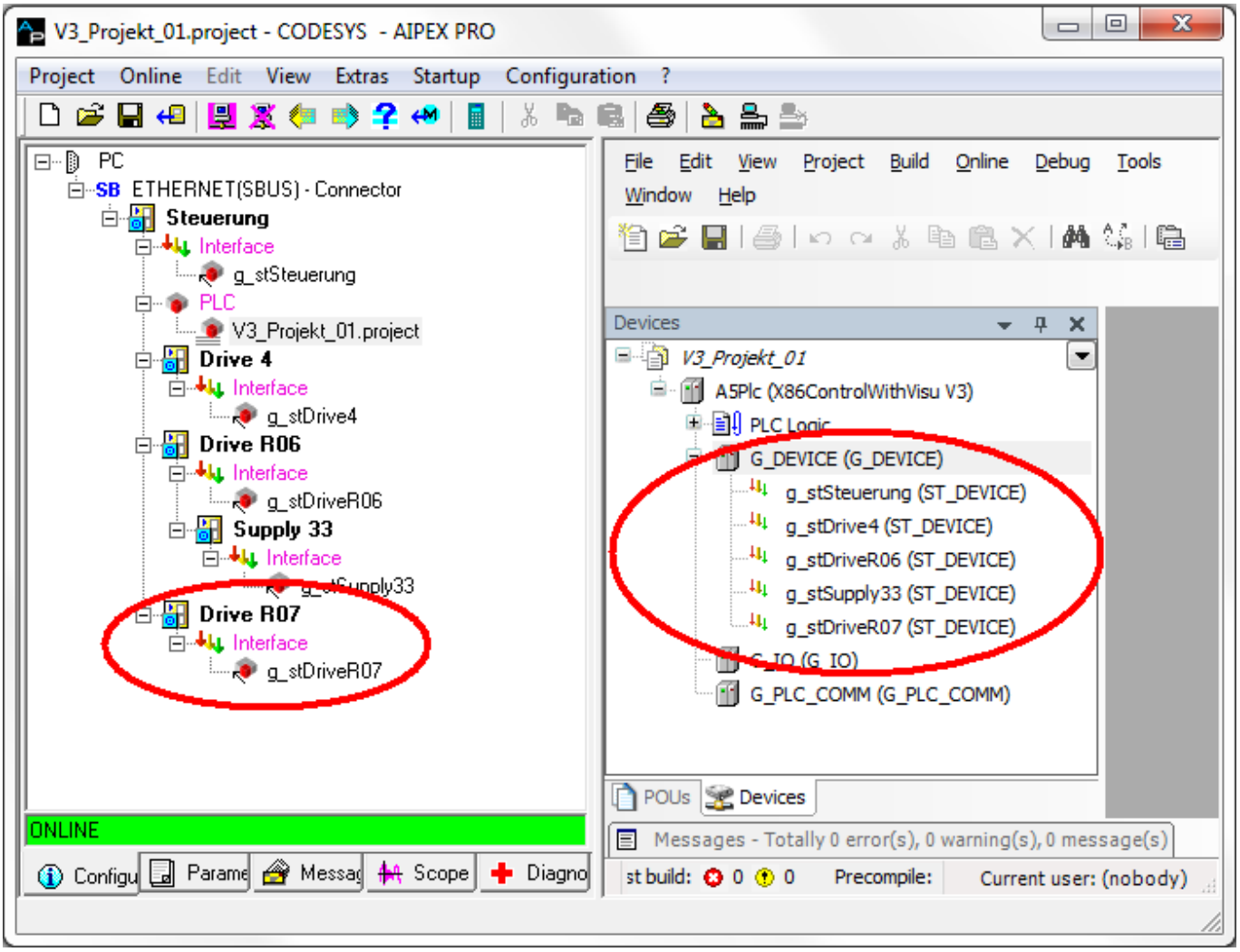
The device names can be imported manually by pressing the **'Import device names'** button.



Switching between the AIPEX and CODESYS interface

Click on any AIPEX icon to switch to the AIPEX interface.
Click on the CODESYS PLC project name to switch to the CODESYS interface.

You will find the symbolic device names at **'Devices'** -> **'G_DEVICE (G_DEVICE)'** .
In the device tree from AIPEX PRO is the PLC device handle name linked with the attendant device icon 'Interface' automatically.
The automatic message configuration use this information to create a message configuration file.

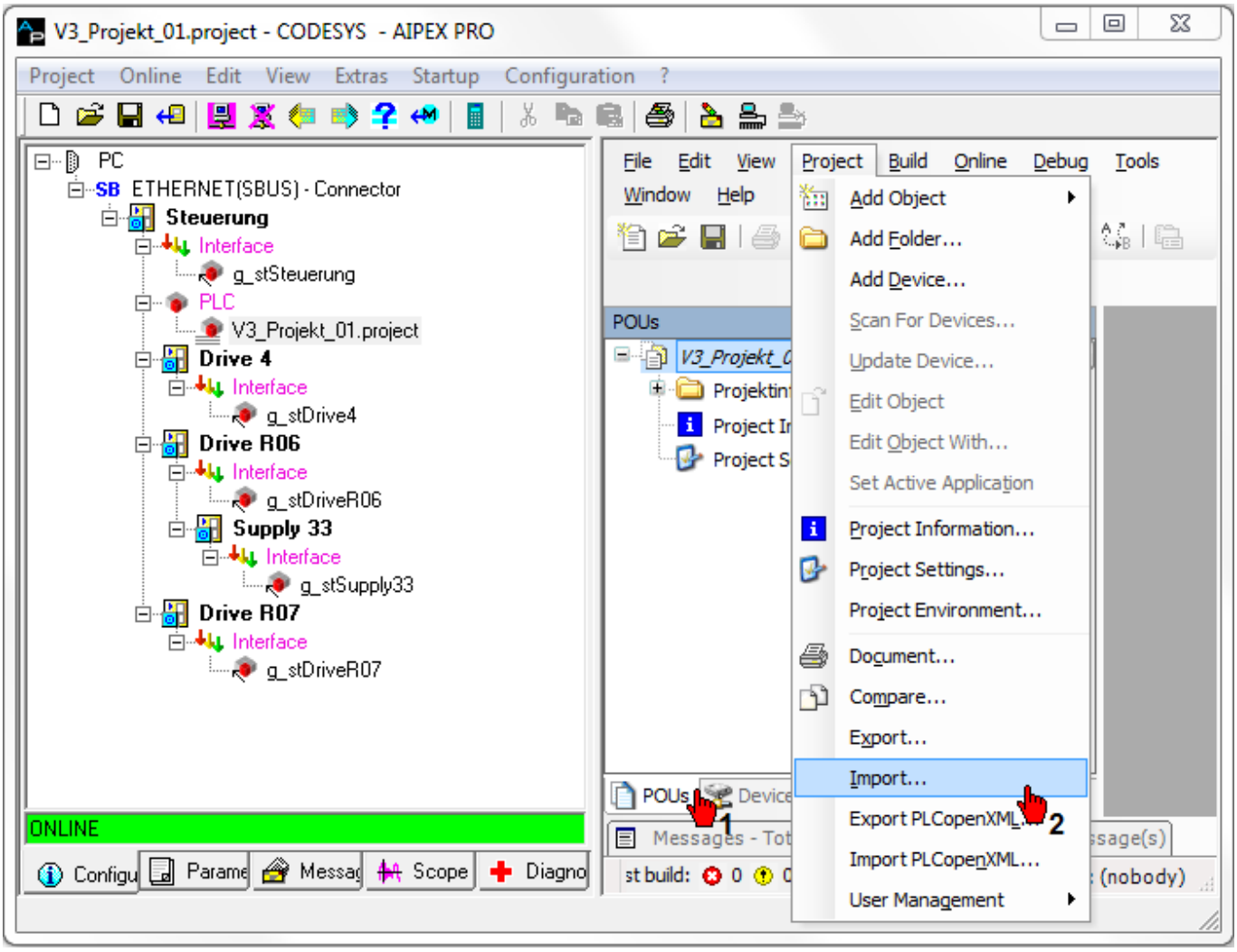


Additional information:

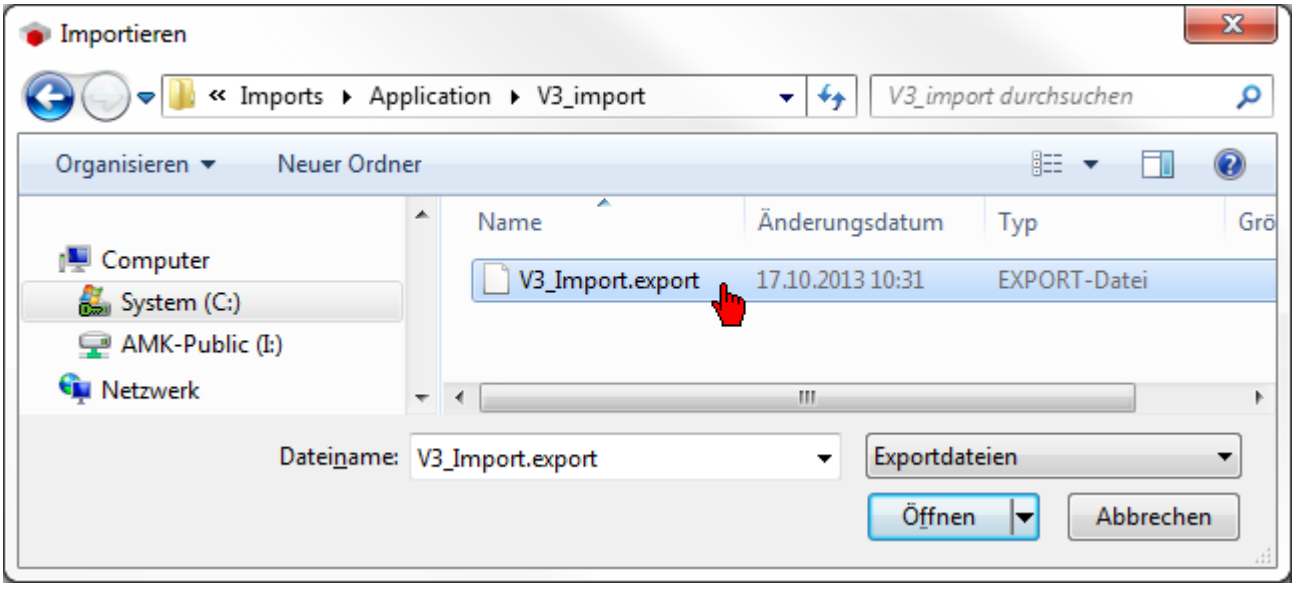
The symbolic device names can also be created manually. In this case, the link between devices interface and symbolic device names must be assigned manually. This takes place after you have called the AIPEX menu '**Configuration**' → '**Configuration create**'. Siehe 'Automatic bus configuration' auf Seite 711.

4.3.1.5 Importing standard function blocks

Change to the 'POUs' tab before calling the import function.

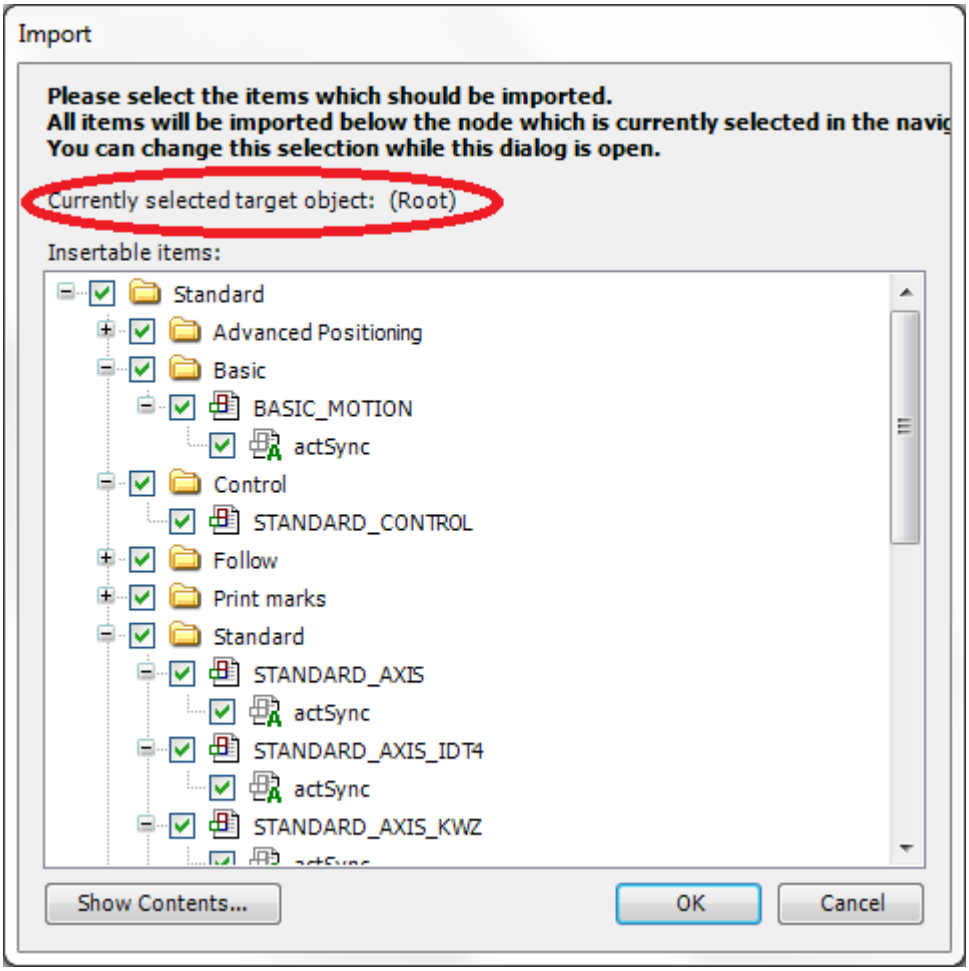


Open the 'V3_Import.export' file. You can find the file under the Windows 7 path:
C:\Programme (x86)\3S CODESYS\AmkAddition\Imports\Application\V3_import



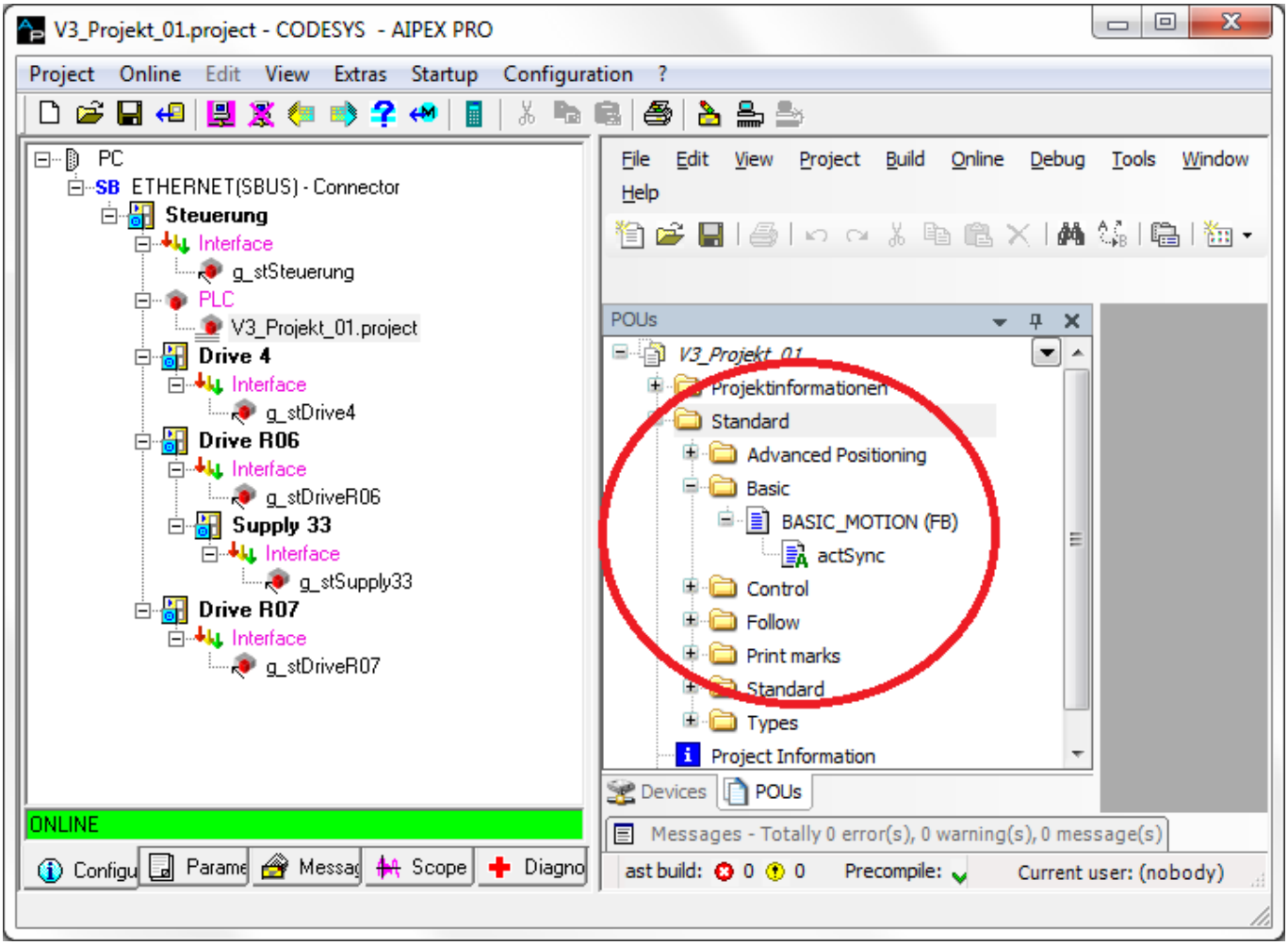
Import all objects.

Experienced users can select the objects according to specific projects.



Is nothing displayed in the 'Import' window? Change to the 'POUs' CODESYS tab.

A folder with the name 'Standard' is created under 'POUs' for the imported objects.



4.3.1.6 Standard Control

The following basic functionalities for the controller are available:

- Status bits (SBM System ready message, QFL Acknowl. error cleared, network and bus information)
- The controlling of FL
- The reading out of diagnostic messages

Additional information:

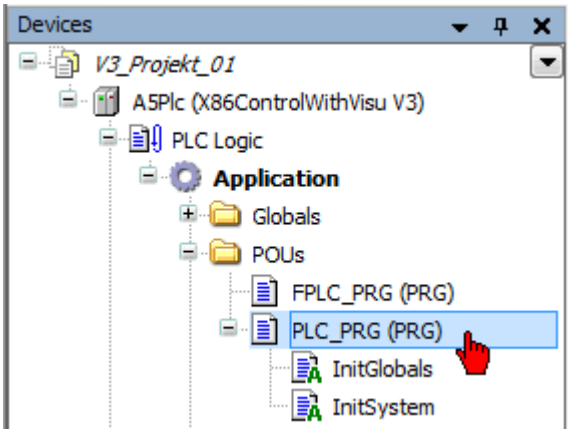
Description of function block: Siehe 'STANDARD_CONTROL (FB)' auf Seite 195.

The 'STANDARD_CONTROL' function block needs the 'ST_CONTROL' structure.

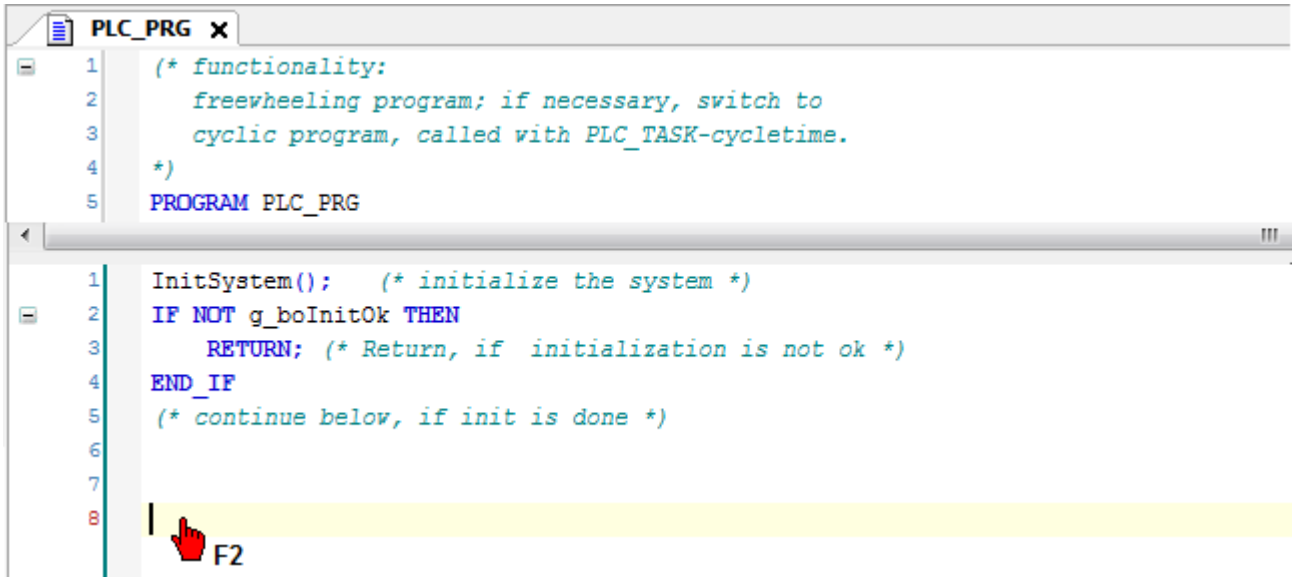
Description of structure: Siehe 'ST_CONTROL (ST)' auf Seite 189.

Switch to the 'Devices' tab

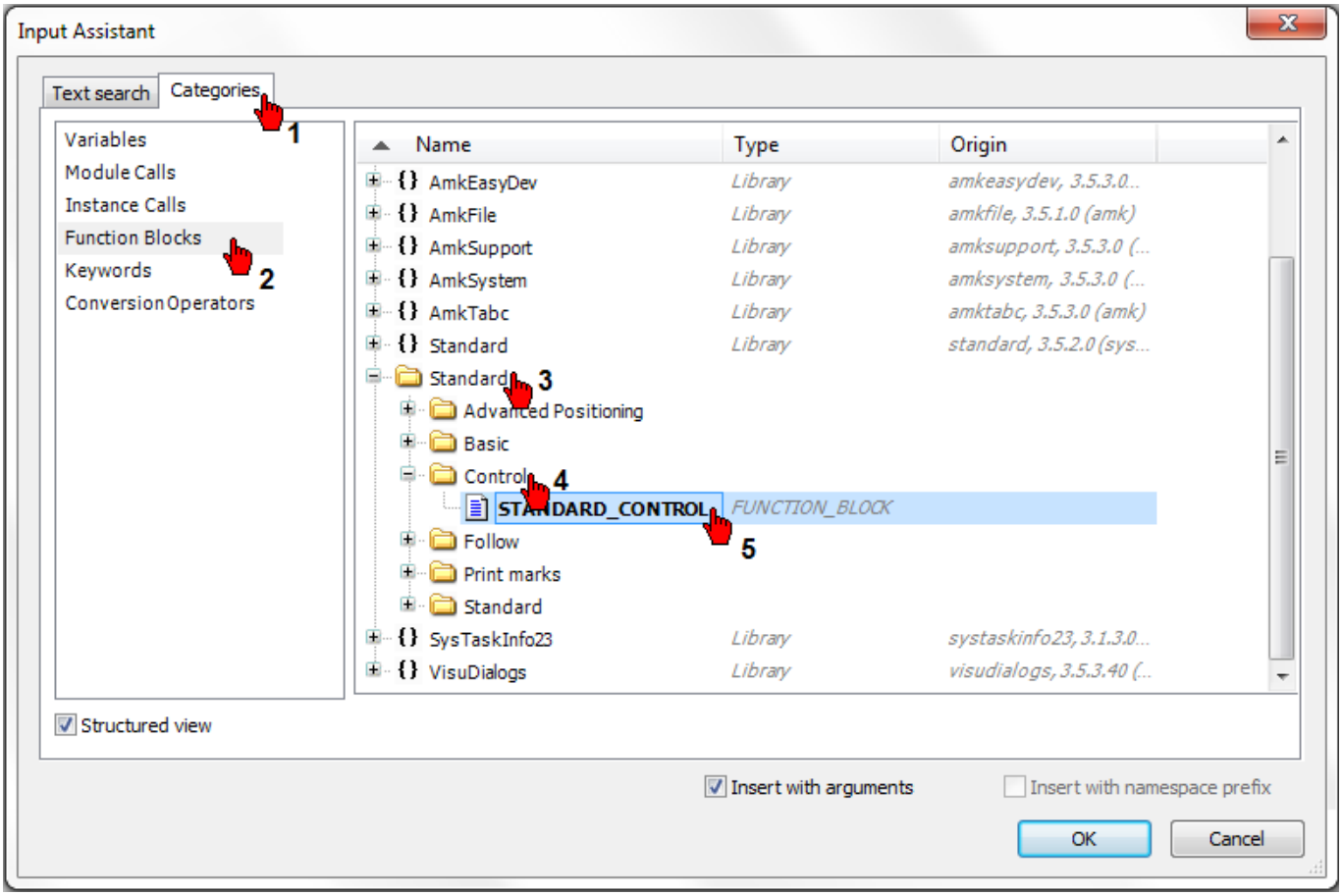
Open the program object 'PLC_PRG (PRG)'



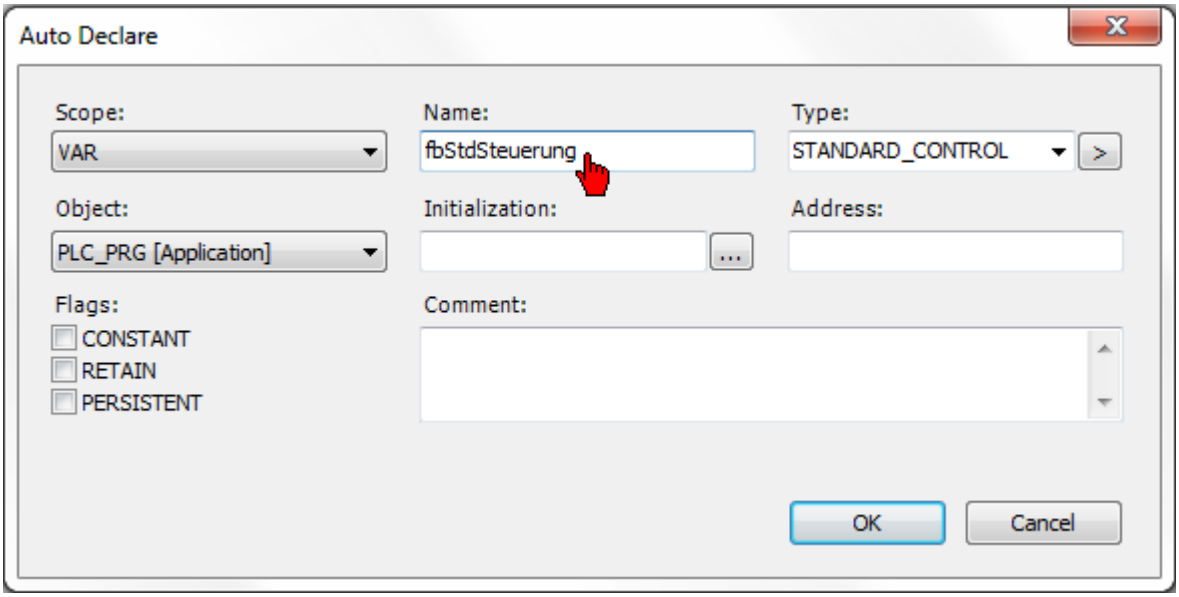
Click in an empty line in the program editor. Press the **'F2'** button to open the 'Input assistant'.



Insert the 'STANDARD_CONTROL' function block.



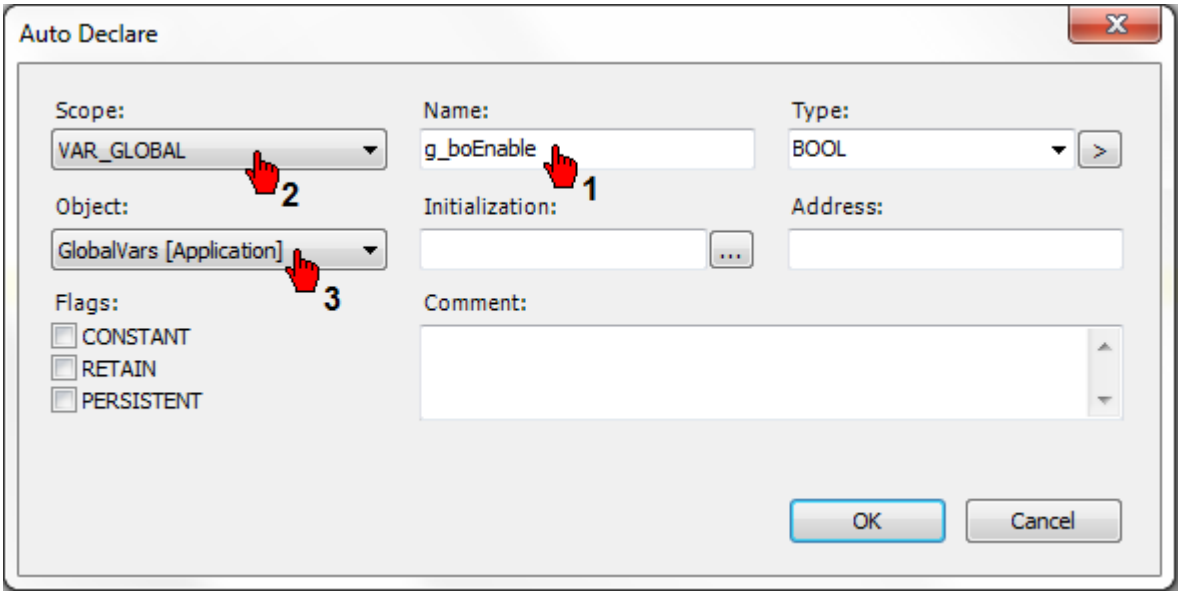
You can instantiate the function block by assigning its own name to it.



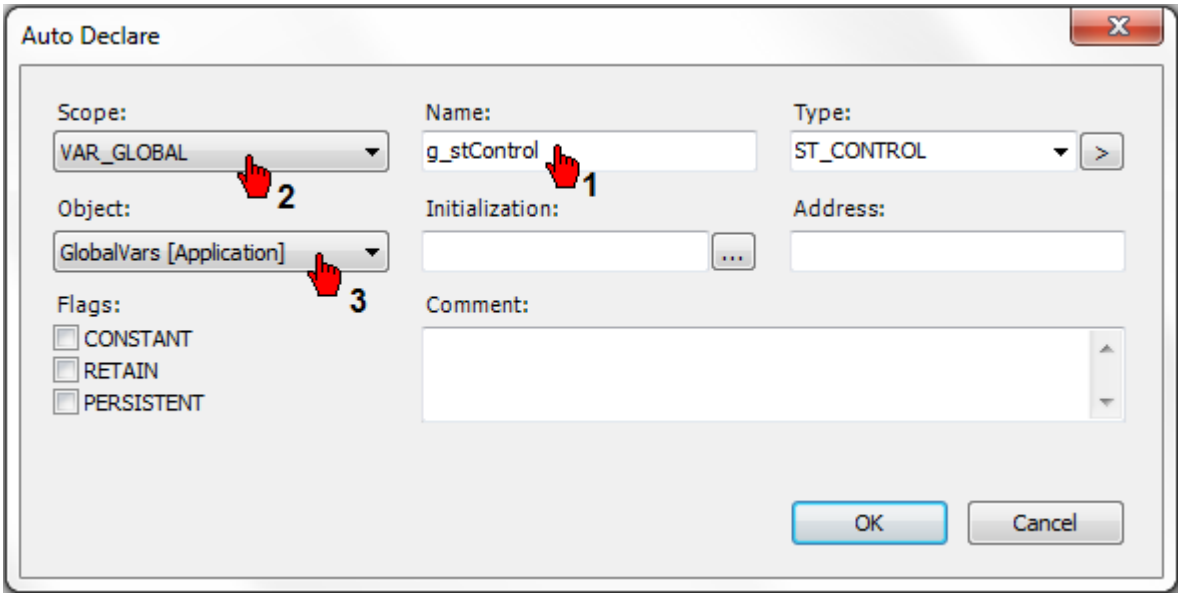
Create the required variables,
enter the symbolic device names from the 'controller configuration' in the 'stDevice' variable.

```
fbStdSteuerung(
    boEnable:= TRUE,     (* Start PLC, block enable set on TRUE *)
    boErrorReset:= ,
    uiBusInstance:= 5 ,      (* Instance Ethercat *)
    boEnabAck=> ,
    boSBM=> ,
    boNetworkReady=> ,
    boBusReady=> g_boEnable,     (* Global enable signal for more function blocks *)
    boBusWarning=> ,
    boBusError=> ,
    boErr=> ,
    iErrID=> ,
    enErrName=> ,
    stControl:= g_stControl ,    (* AMK structure for controller data *)
    stDevice:= g_stSteuerung);   (* Symbolic device name *)
```

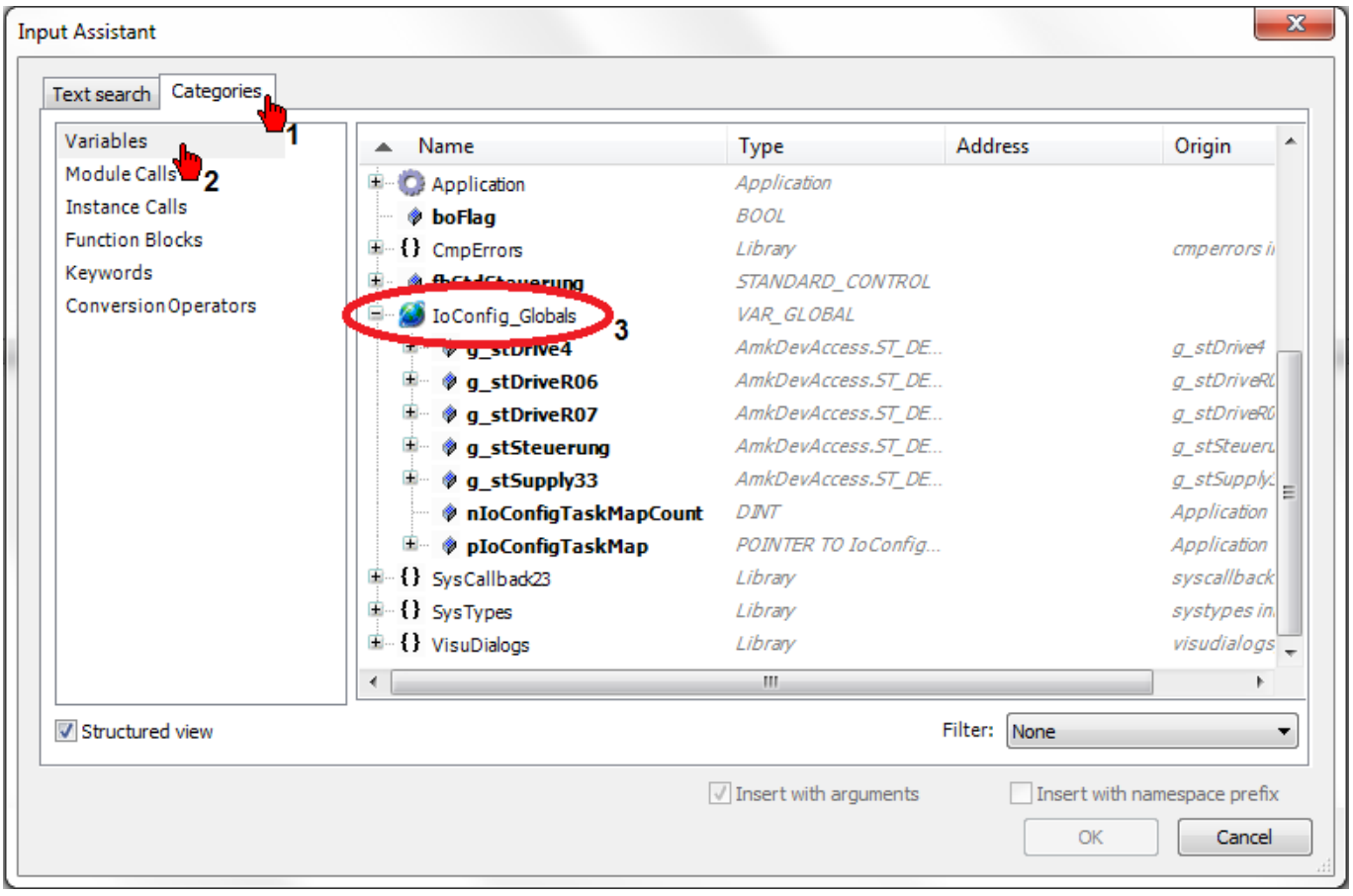
Declare 'g_boEnable' global variable.



Declare 'g_stControl' global structure variable.



The symbolic device names can be found under 'IoConfig_Globals'.



4.3.1.7 Standard Power Supply

The following basic functionalities for the input are available:

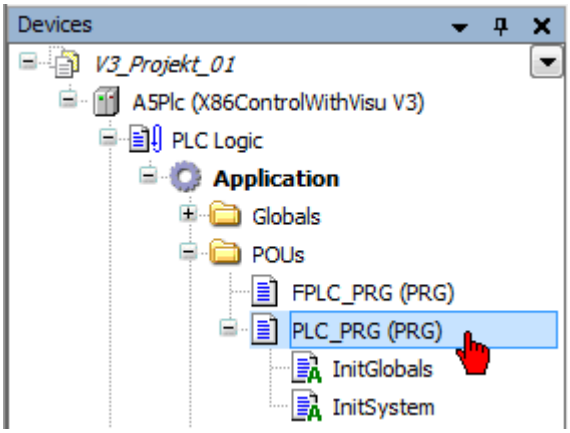
- Status bits (QUE Acknowl. DC bus ON, SBM System ready message)
- Control bits (UE DC bus ON, FL Clear error)
- Output of diagnostics number

Additional information:

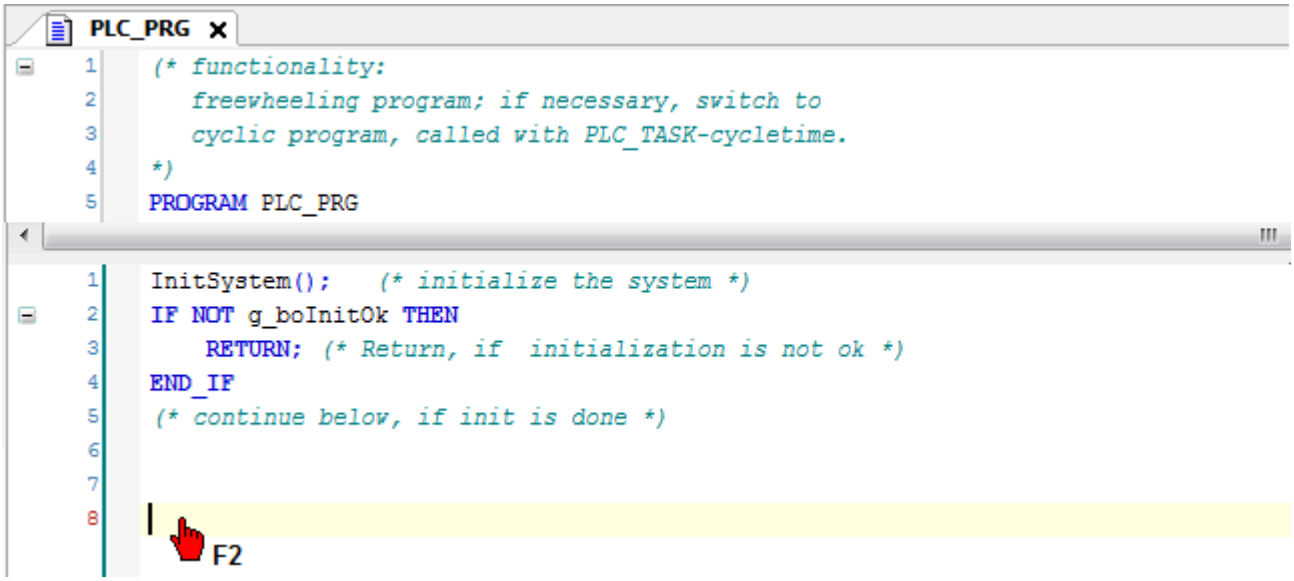
Description of function block: Siehe 'STANDARD_CONTROL (FB)' auf Seite 195.

Switch to the 'Devices' tab

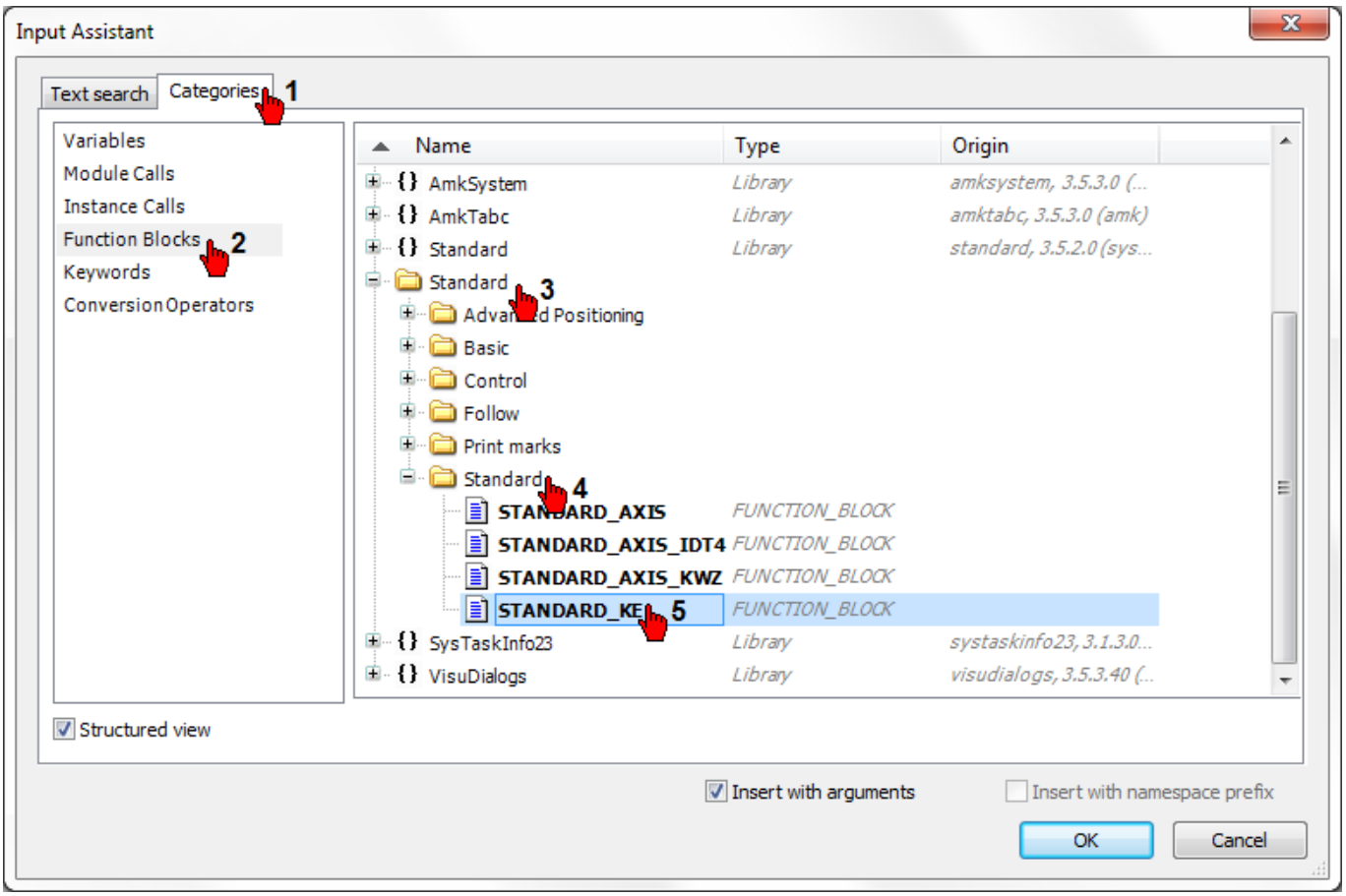
Open the program object 'PLC_PRG (PRG)'



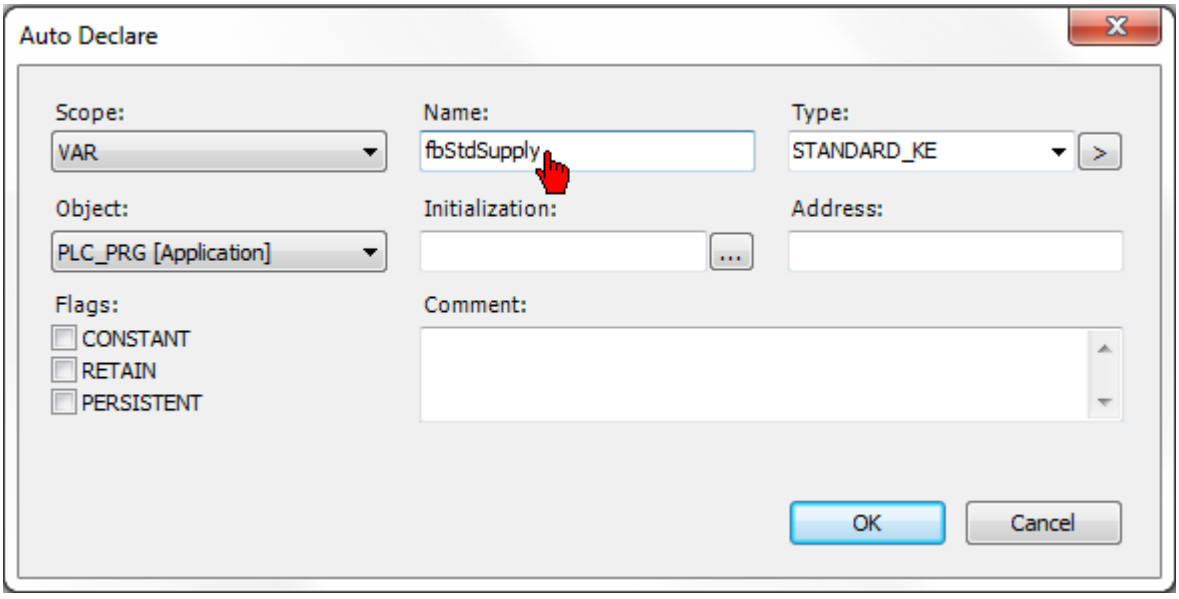
Click in an empty line in the program editor. Press the **'F2'** button to open the 'Input assistant'.



Insert the 'STANDARD_KE' function block.



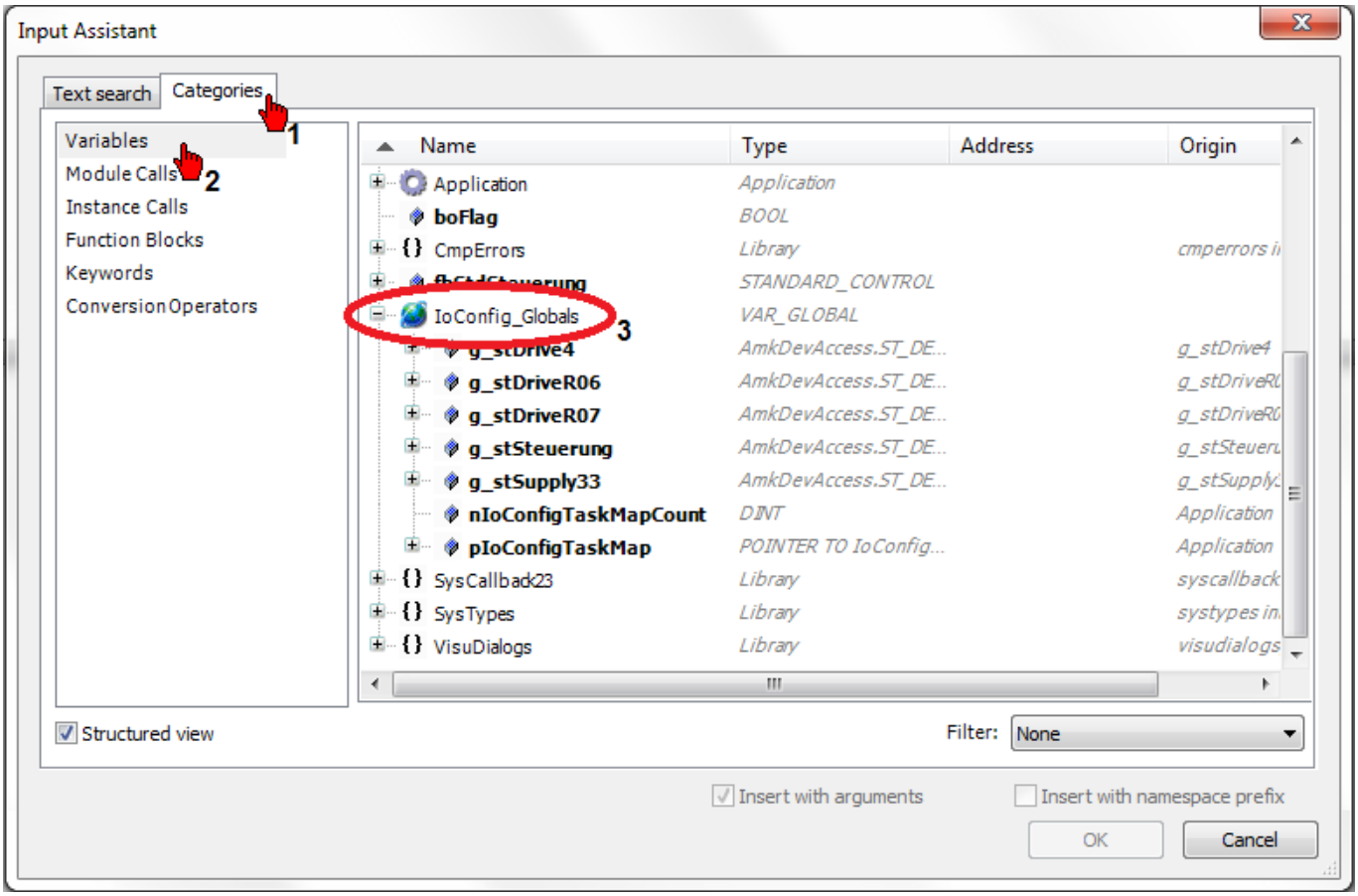
You can instantiate the function block by assigning its own name to it.



Create the required variables,
enter the symbolic device names from the 'controller configuration' in the 'stDevice' variable.

```
fbStdSupply(
    boEnable:= g_boEnable,   (* Global enable signal *)
    boUE:= ,
    boFL:= ,
    stDevice:= g_stSupply33,     (* Symbolic device name *)
    boEnabAck=> ,
    uiDiagnosticNr=> ,
    boQUE=> ,
    boSBM=> ,
    boErr=> ,
    iErrID=> ,
    enErrName=> );
```

The symbolic device names can be found under 'IoConfig_Globals'.



4.3.1.8 Standard AXIS

The following basic functionalities for the axis are available:

Inputs/outputs on the function block

- Control bits (RF Controller enable, FL Clear error)
- Emergency stop
- Speed control
- Homing cycle
- Positioning (absolute/relative)
- Jog mode (minus/plus)
- Status signals of axis functions

Access via program code

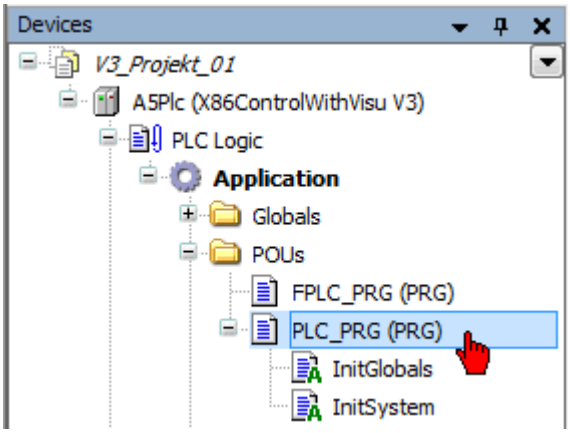
- Configuration of status messages (ID26 'Configuration status bits')
- Reading of parameters according to a specifications list
- Actual position cached
- Control with feedforward (torque and speed)

The 'STANDARD_AXIS' function block needs the 'BASIC_MOTION' function block and 'ST_AXIS_DRIVE' structure.
'STANDARD_AXIS' consists of an asynchronous functional part that is called in the PLC_PRG and a synchronous functional part that is called in the FPLC_PRG.

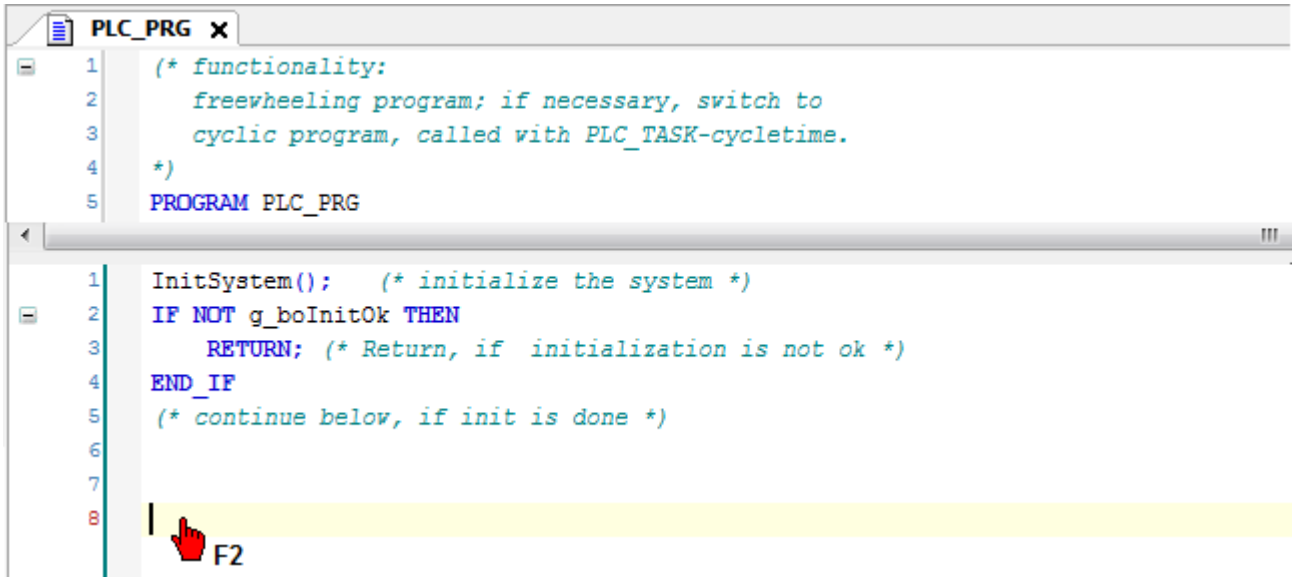
Additional information:

Description of function block: Siehe 'STANDARD_AXIS (FB) (STANDARD_AXIS_KWZ, STANDARD_AXIS_IDT4, STANDARD_AXIS_ihX)' auf Seite 201.
Description of structure: Siehe 'ST_AXIS_DRIVE (ST)' auf Seite 191.

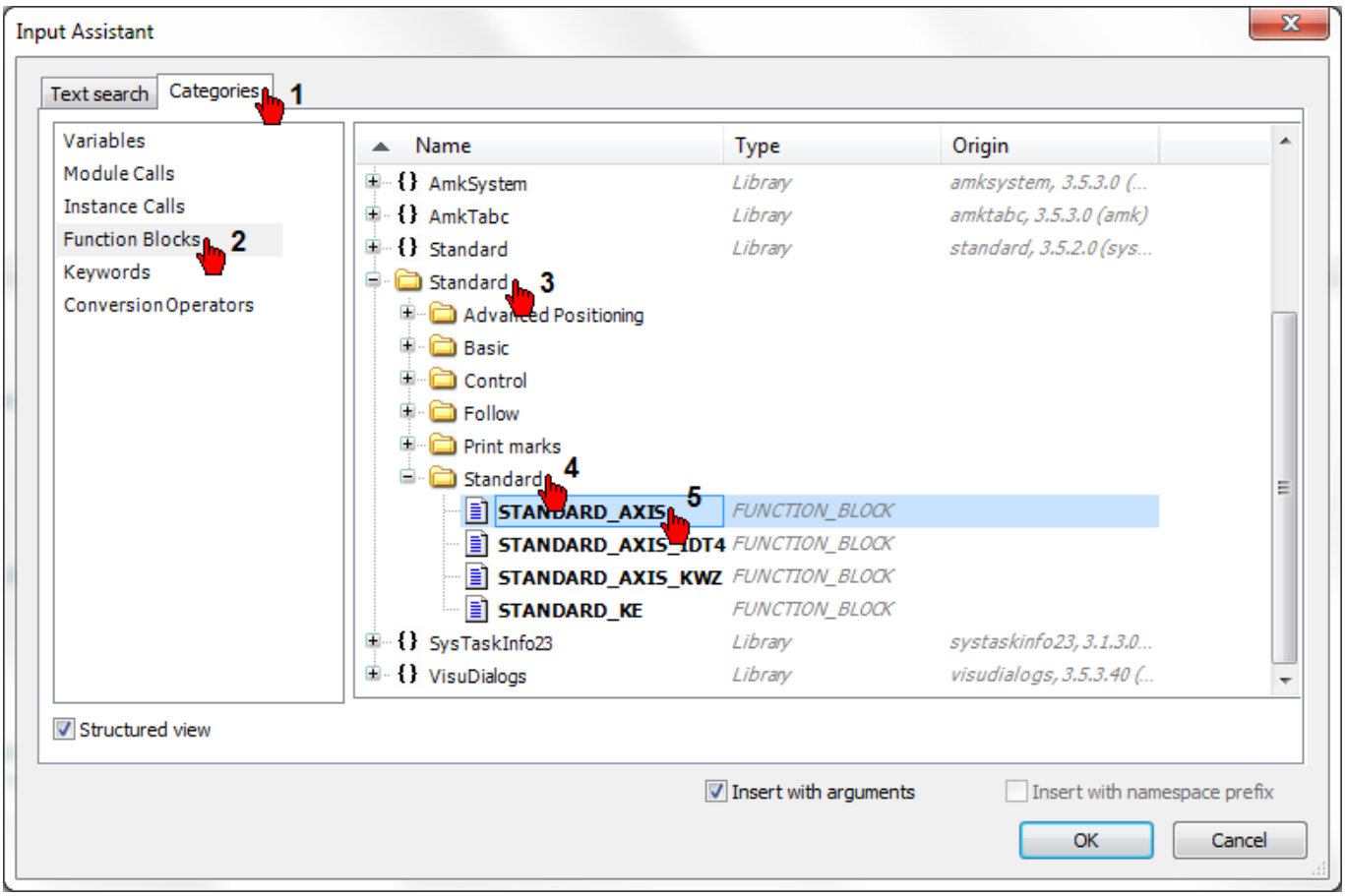
Switch to the 'Devices' tab
Open the program object 'PLC_PRG (PRG)'



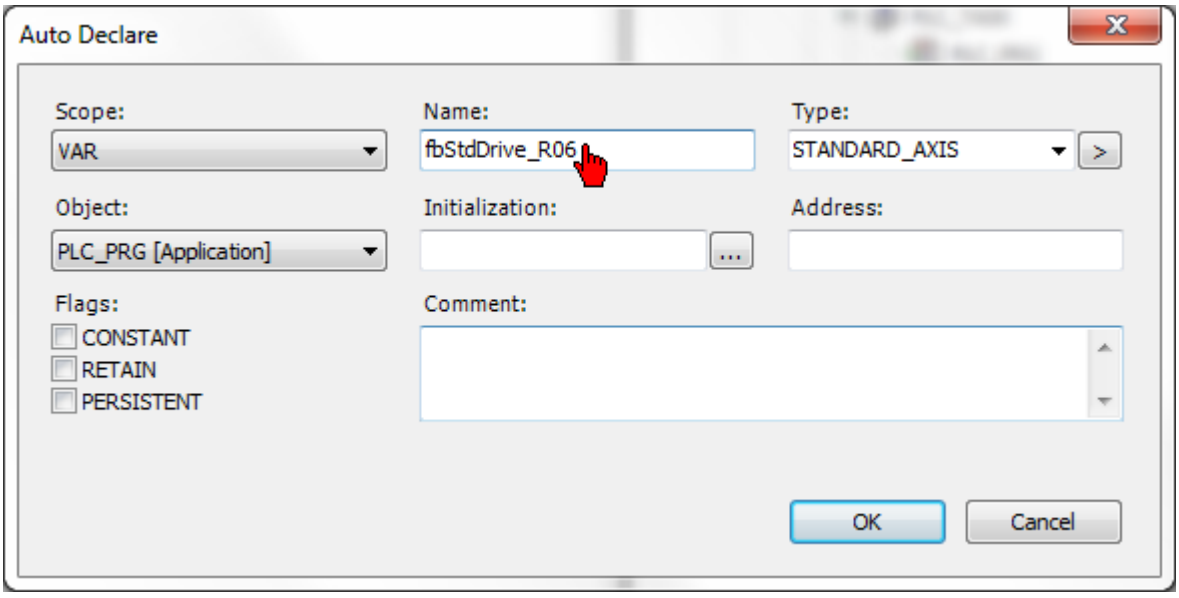
Click in an empty line in the program editor. Press the **'F2'** button to open the 'Input assistant'.



Add the function block 'STANDARD_AXIS'.



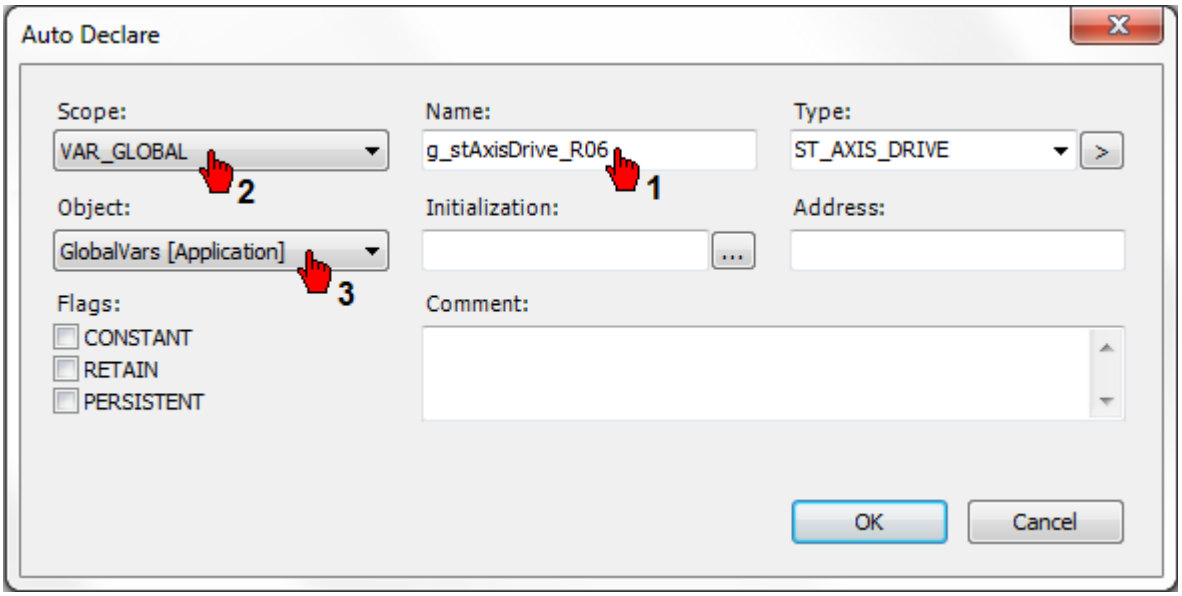
You can instantiate the function block by assigning its own name to it.



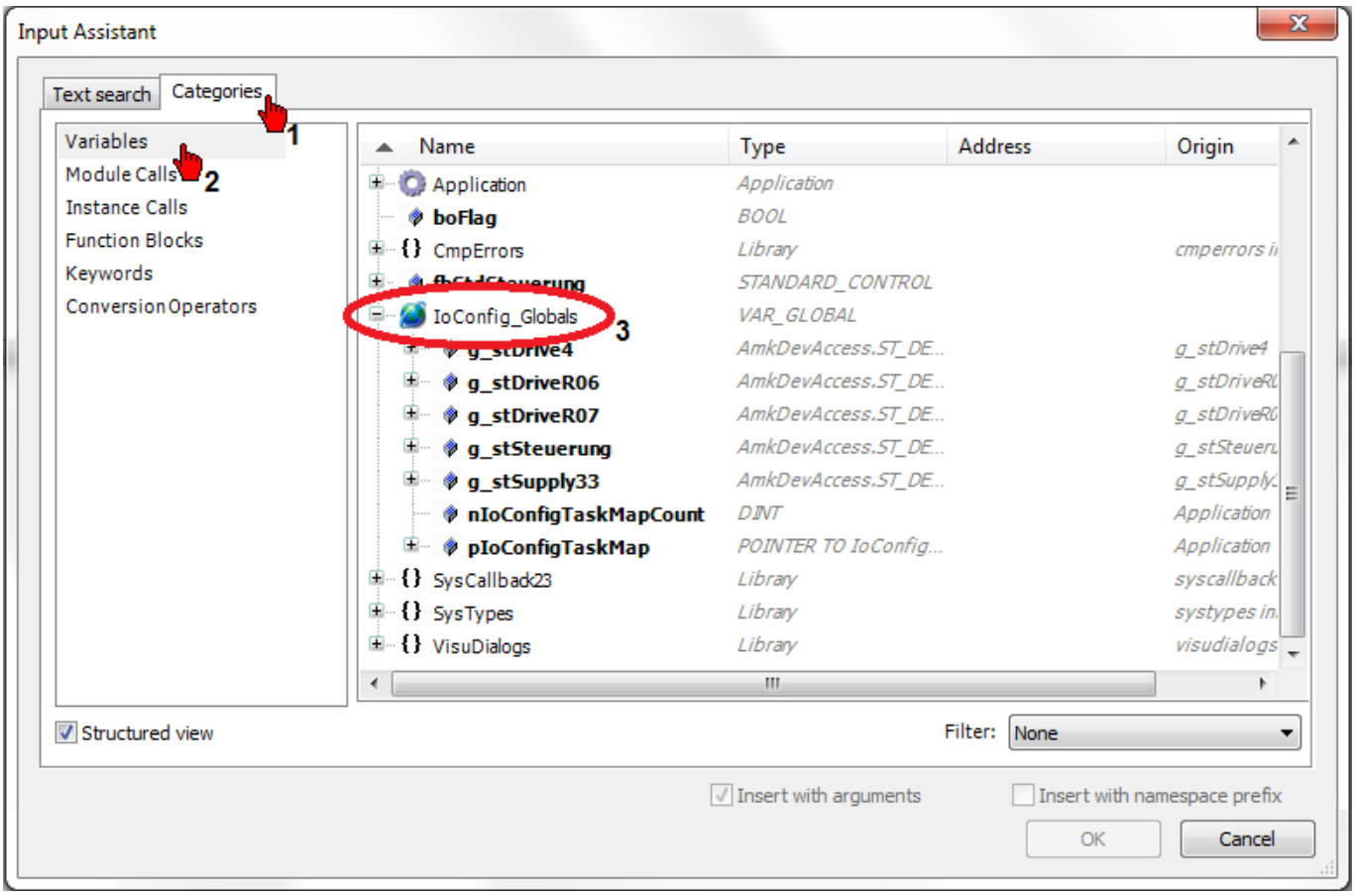
Create the required variables,
enter the symbolic device names from the 'controller configuration' in the 'stDevice' variable.

```
fbStdDrive_R06(
    boEnable:= g_boEnable,   (*Global enable signal *)
    boRF:= ,
    boFL:= ,
    boEmergency_Stop:= ,
    boSpeed:= ,
    boHome:= ,
    boPosAbs:= ,
    boPosRel:= ,
    boJogPlus:= ,
    boJogMinus:= ,
    udPosVelocity:= ,
    diPosition:= ,
    lreAccel:= ,
    lreDecel:= ,
    lreJerkStart:= ,
    lreJerkEnd:= ,
    udVelocityJog:= ,
    diSpeedVelocity:= ,
    boEnabAck=> ,
    boSBM=> ,
    boQRF=> ,
    boDone=> ,
    boErr=> ,
    iErrID=> ,
    enErrName=> ,
    boPosBusy=> ,
    boJogBusy=> ,
    boHomeBusy=> ,
    boSpeedBusy=> ,
    stAxisDrive:= g_stAxisDrive_R06 ,    (* AMK structure for unique drive data *)
    stDevice:= g_stDriveR06);    (* Symbolic device name *)
```

Declare 'g_stAxisDrive_R06' global structure variable

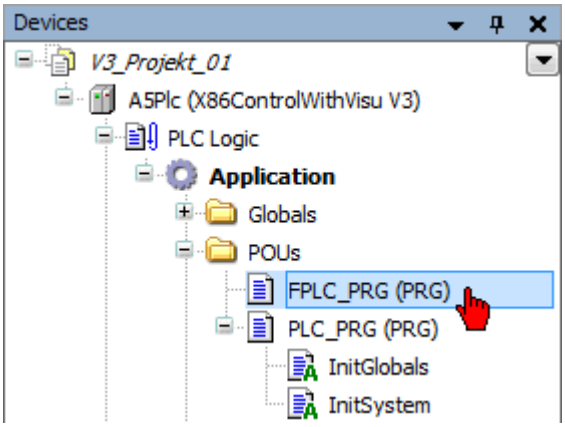


The symbolic device names can be found under 'IoConfig_Globals'.



Switch to the 'Devices' tab

Open the program object 'FPLC_PRG (PRG)'



Click in an empty line in the program editor.

```
FPLC_PRG
1  (* functionality:
2     external event-program FPLC_PRG,
3     called by FPLC_TASK in PGT-cycletime (ID2).
4   *)
5  PROGRAM FPLC_PRG
```

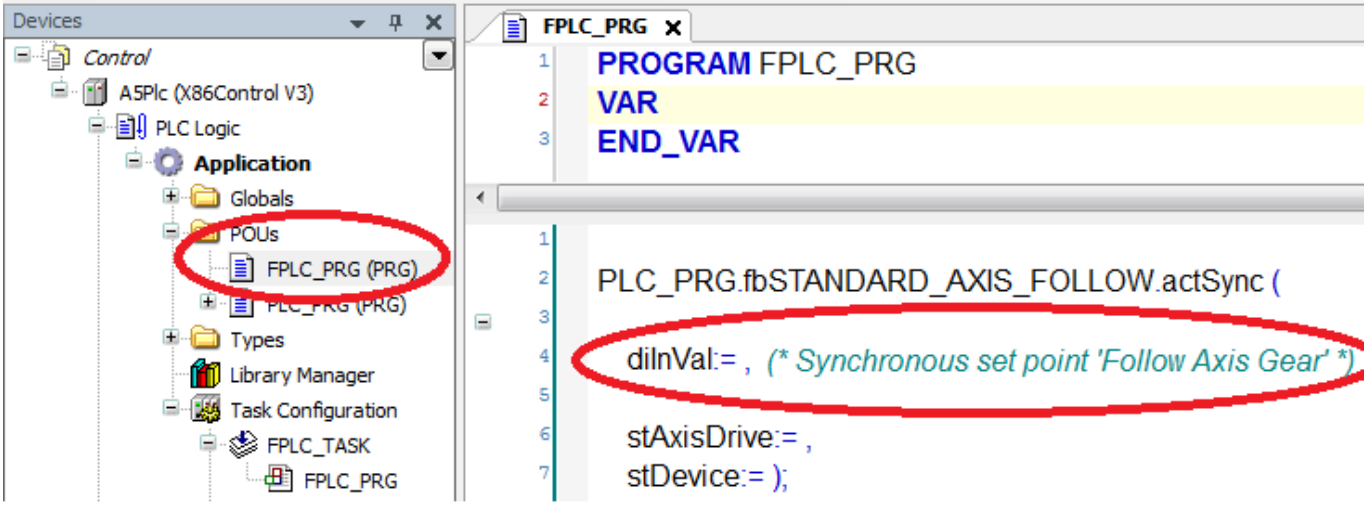
```
1  IF NOT g_boInitOk THEN
2      RETURN; (* Return, if  initialization is not ok *)
3  END_IF
4  (* continue below, if init is done *)
5
6
```

The 'STANDARD_AXIS' function block contains a synchronous action that is synchronized with the drive.
The synchronous action must be called in the 'FPLC_PRG'. The function block cannot be switched on and the data will not be updated without this call.

To do this, generate the following program code in the 'FPLC_PRG'.
PLC_PRG.STANDARD_AXIS .actSync (* in the example PLC_PRG.fbStdAntrieb_R06.actSync *)
(
stAxisDrive := axes structure (* in the example g_stAxisDrive_R06 *),
stDevice := symbolic device names (* in the example g_stAntrieb_R06*)
);

```
PLC_PRG.fbStdDrive_R06.actSync
(
stAxisDrive:= g_stAxisDrive_R06 ,    (* AMK structure for unique drive data *)
    stDevice:= g_stDriveR06 );   (* Symbolic device name *)
```

Synchronous set point using as example STANDARD_FOLLOW_AXIS_GEAR (FB)
The synchronous setpoint is specified in FPLC_PRG / *.actSync call.
Add the Input Variable 'diInVal' (Setpoint specification for follow operation) as transfer variable.

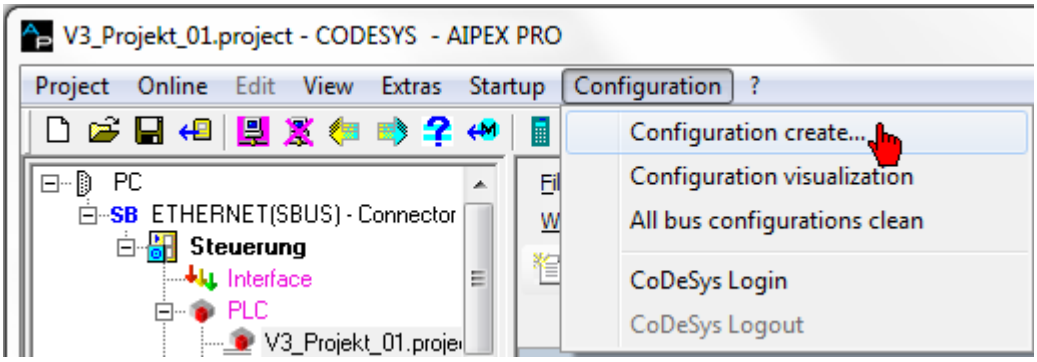


4.3.1.9 Creating PLC program and message configuration

AIPEX PRO must be 'Online' for the following step.

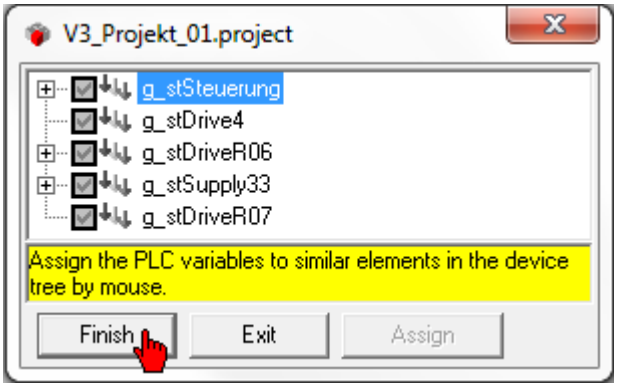
Before the PLC program is transferred to the controller, it must first be translated without error and the message configuration must be generated.

Start this process under '**Configuration**' → '**Configuration create**'.



g_stSteuerung [Arrow grey] The symbolic device name is assigned to a device (interface) in the AIPEX PRO device tree

g_stSteuerung [Arrow colored] The symbolic device name is not assigned. Siehe 'Automatic bus configuration' auf Seite 711.

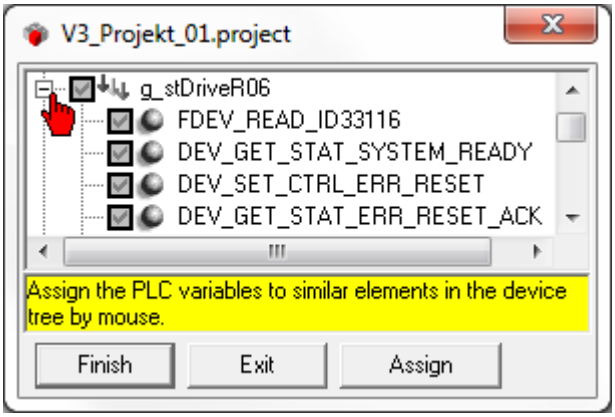


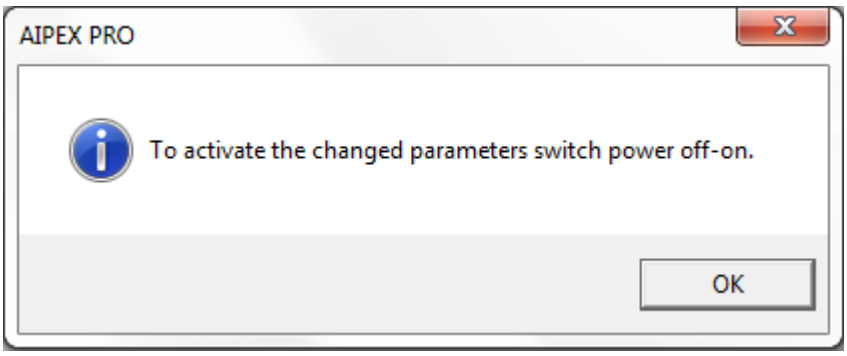
Additional information:

If activated, the configurator changes the relevant parameters automatically. Click on the + icon. You will see which function blocks are assigned to a symbolic device name.

Activating/deactivating or function block specific changes:

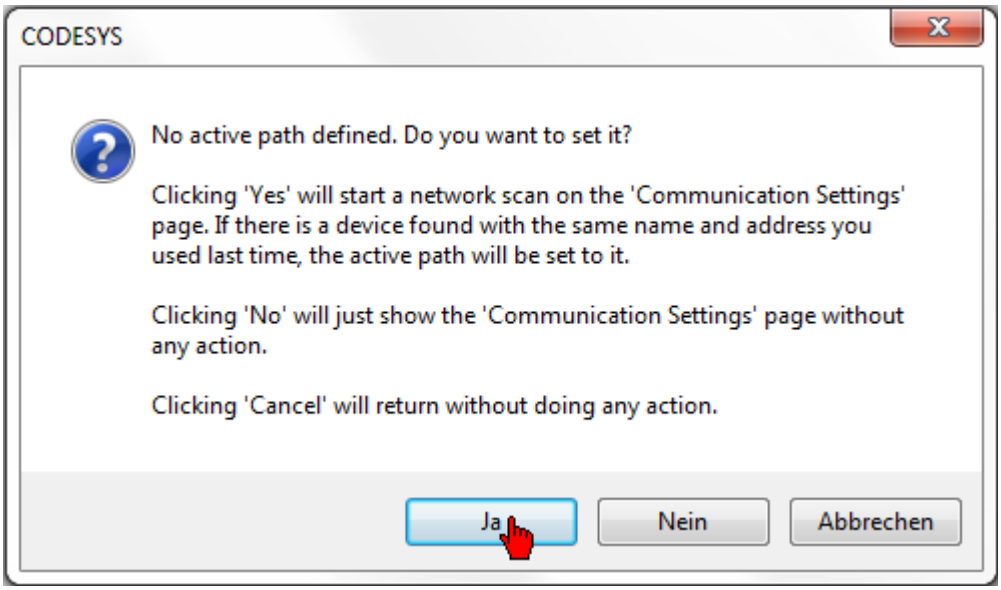
Siehe 'Project Settings - Configuration create (project specific)' auf Seite 108.





Communication settings

The communication settings between the PC and controller must be set separately in CODESYS V3. They are not transferred from AIPEX PRO as in CODESYS V2.

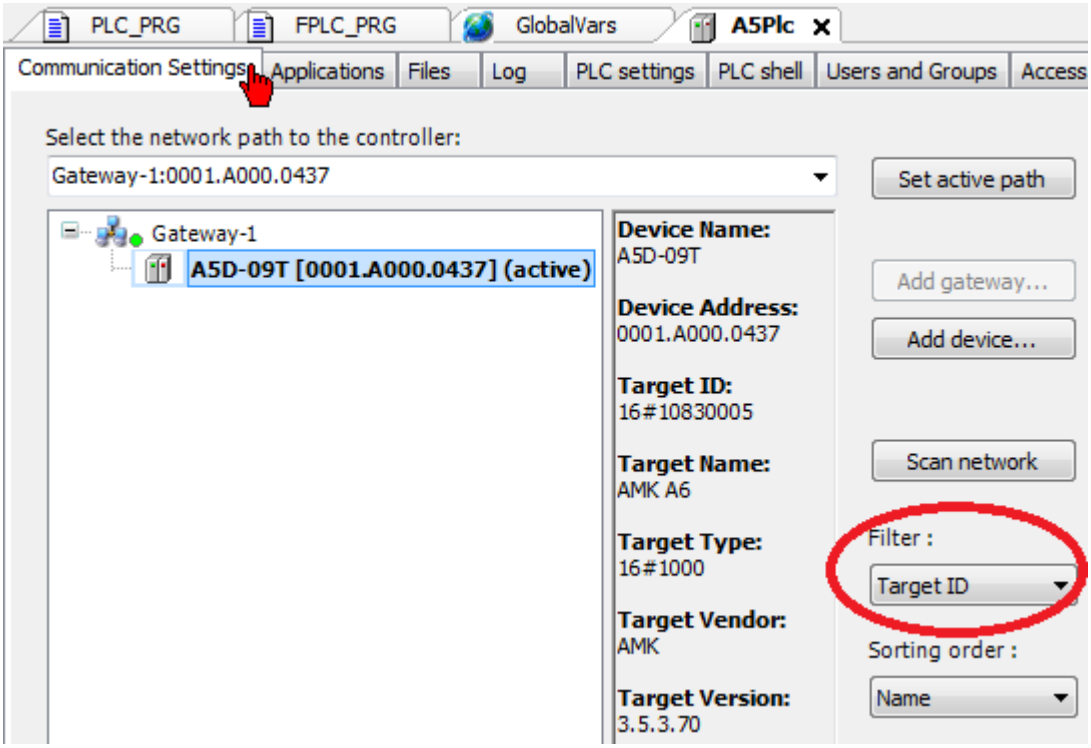


With the 'Target ID' active, those controllers suited to your project are displayed only automatically. Make your choice based on the 'device address'.

Representation of the AMK controller's TCP/IP address under CODESYS V3. Only the last 4 digits are decisive.

Example: The controller is in a network with the TCP/IP address 172.16.xxx.xxx.

CODESYS V3 **Device address**	**AMK controller** **TCP/IP address**
0000.047C (hex)	172.16.4.124 (dec)
0001.A000.0462 (hex)	172.16.4.098 (dec)



Declaration of device properties: Siehe 'Device identification' auf Seite 720.

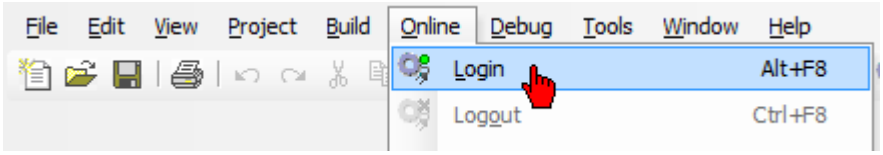
4.3.1.10 Testing PLC project

Objective: Activation of the motor control with the help of the 'declaration editor in online mode'.

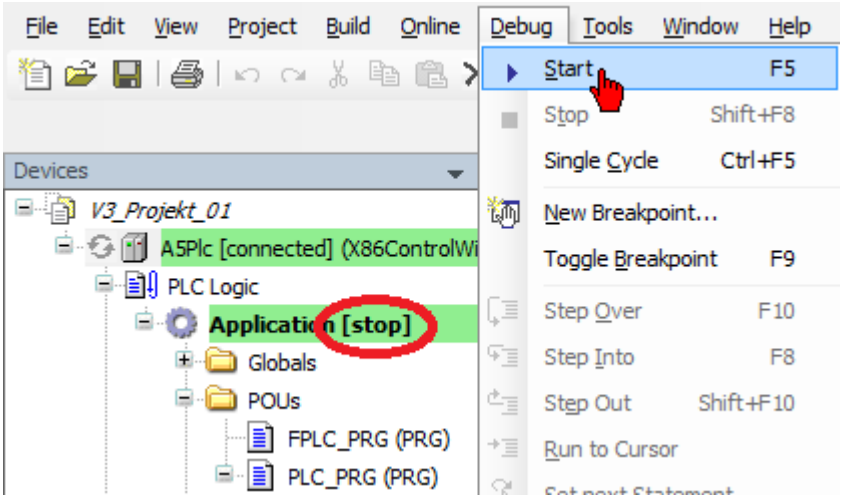
Prerequisite:

- Initial start-up of motor completed
- EF/EF2 (STO) hardware signals present
- Any emergency-stop circuits are closed

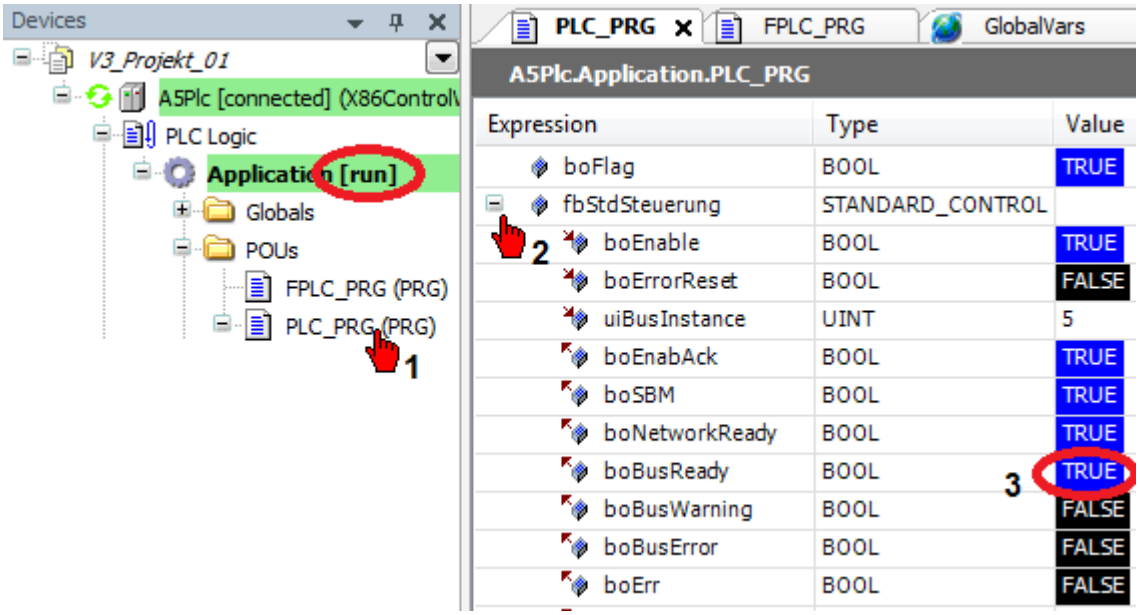
Log on



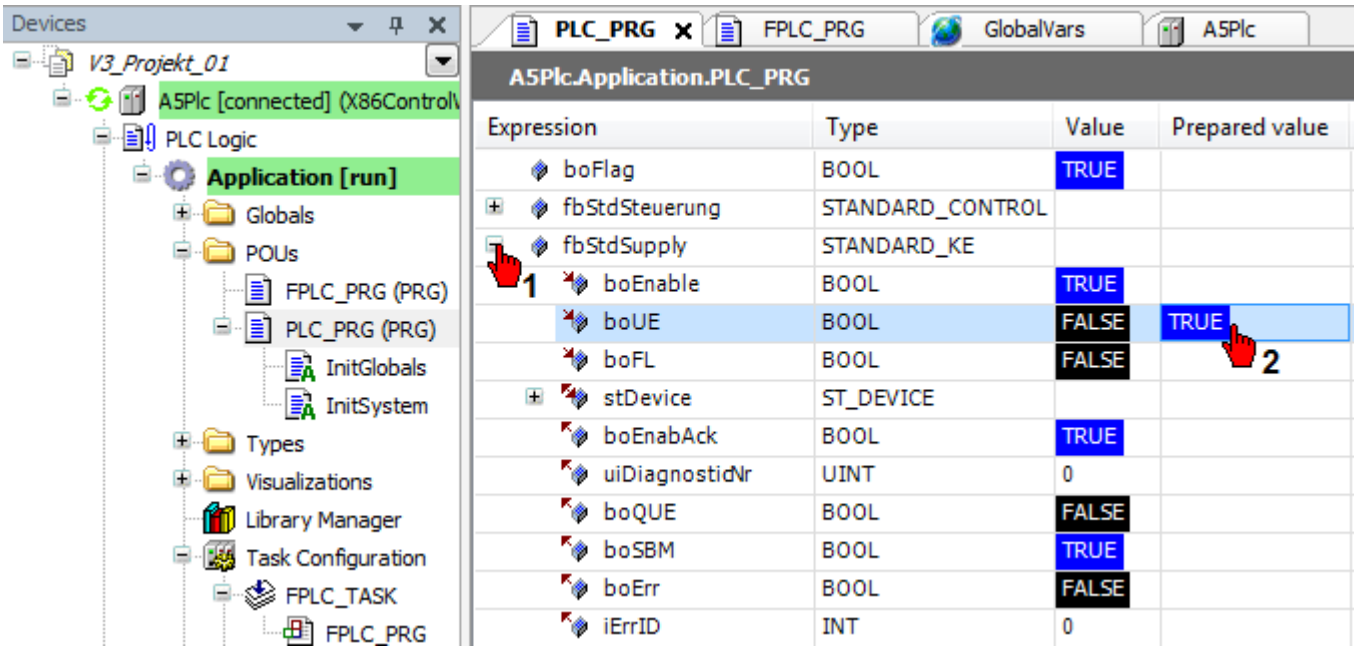
Start the PLC



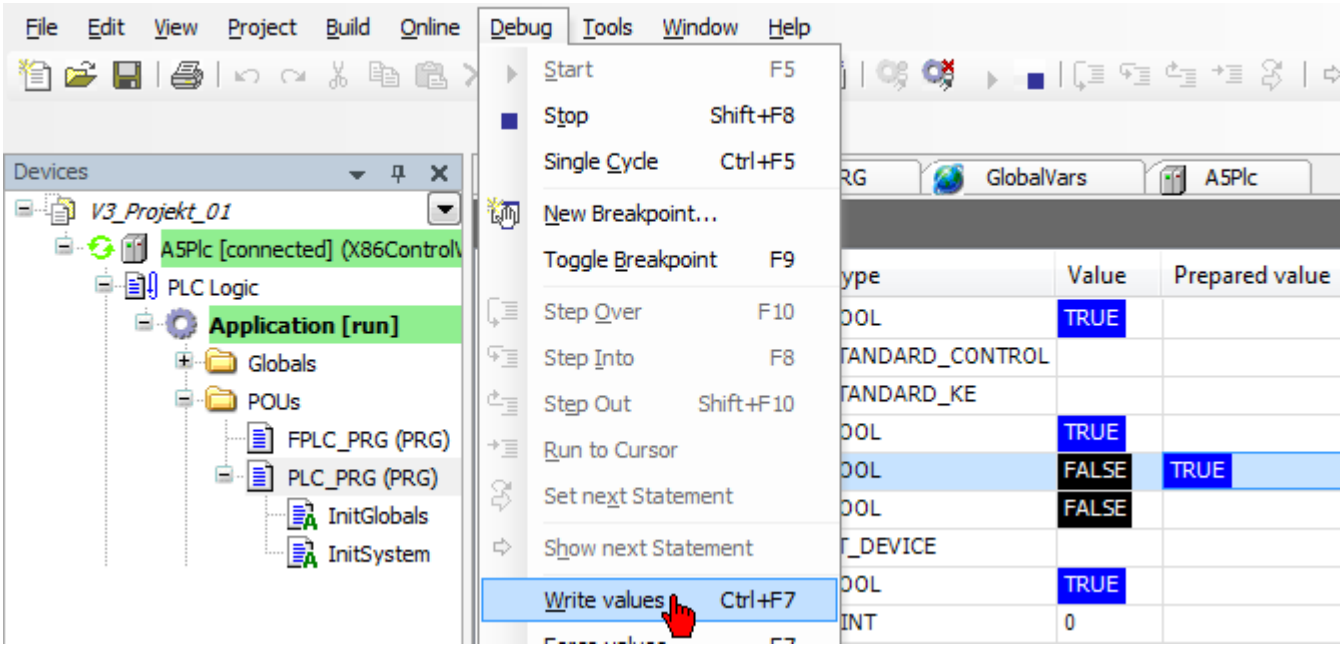
The 'Application' status changes to [run]. Open the program object 'PLC_PRG'. In the properties of the 'fbStdSteuerung' function block you can now see (boBusReady = TRUE) whether the AMK devices have been initialized without error and whether the EtherCAT Bus has reached the 'Operational mode'.



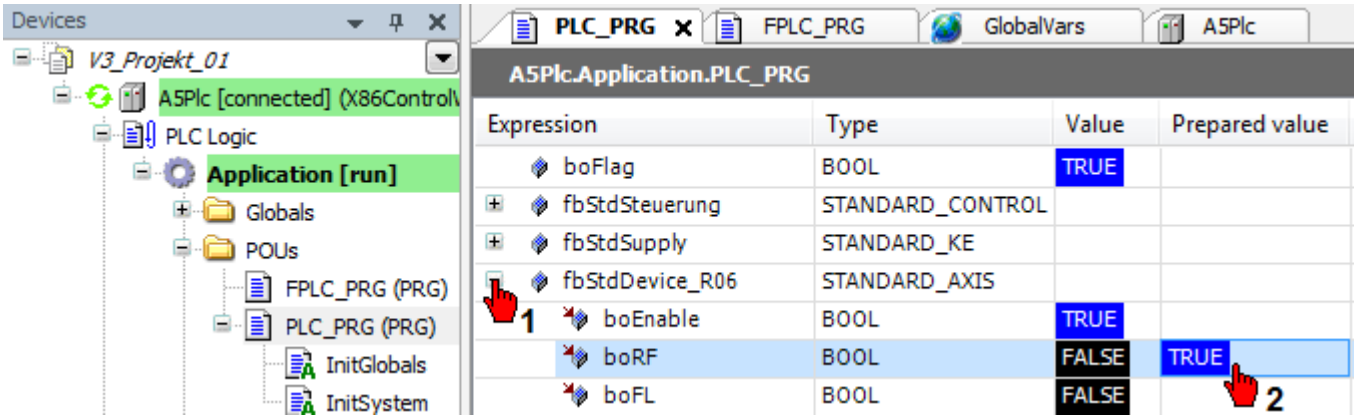
Load the intermediate DC link by setting the 'boUE' input variable of the 'fbStdEinspeisung' function block to 'TRUE'. (acknowledgement signal for error-free loading 'boQUE = TRUE')



'boUE = TRUE' is applied with 'Write values [Ctrl + F7]'.



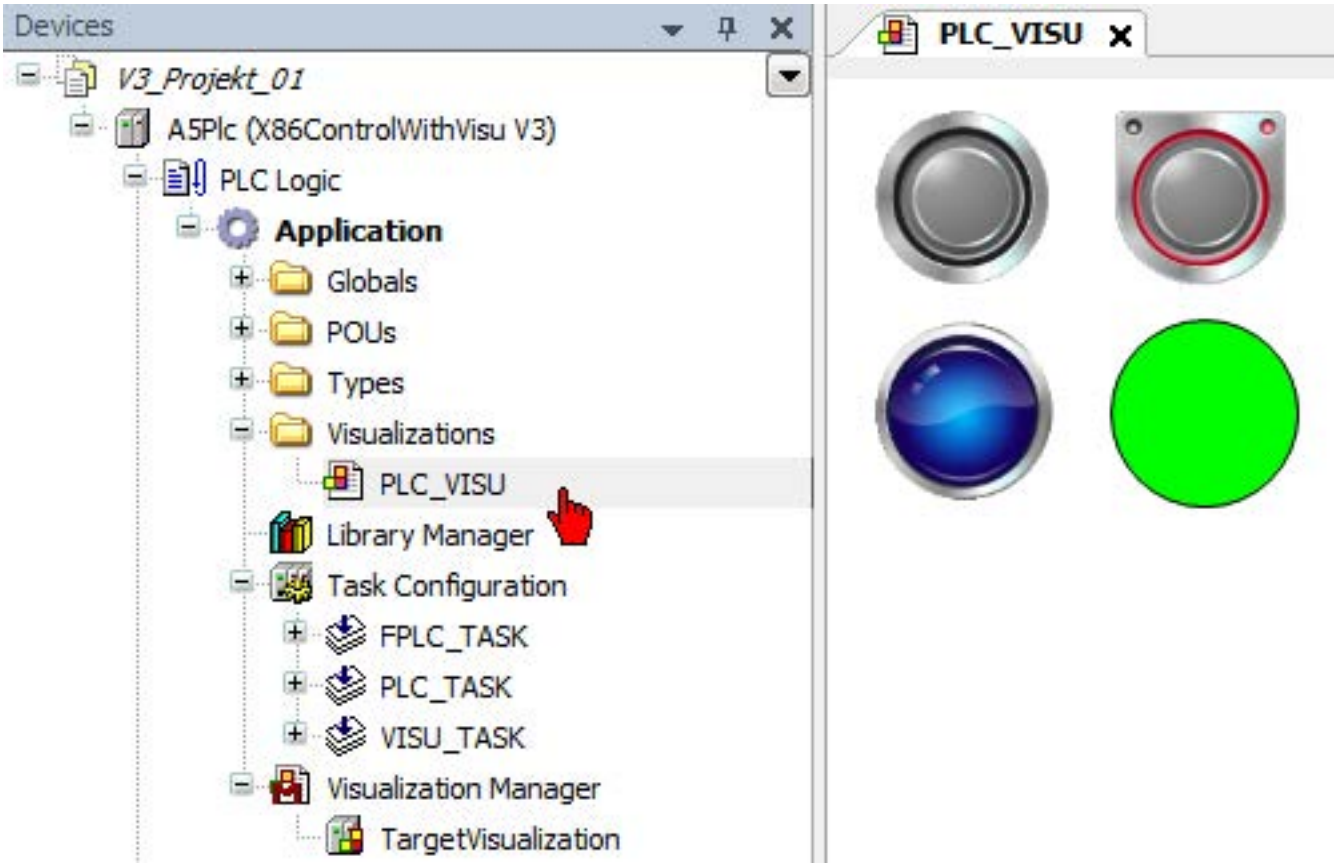
Activate the motor control by setting the 'boRF' input variable of the 'fbStdAntrieb_R06' function block to 'TRUE'.
'boRF = TRUE' is applied with 'Write values [Ctrl + F7]'.



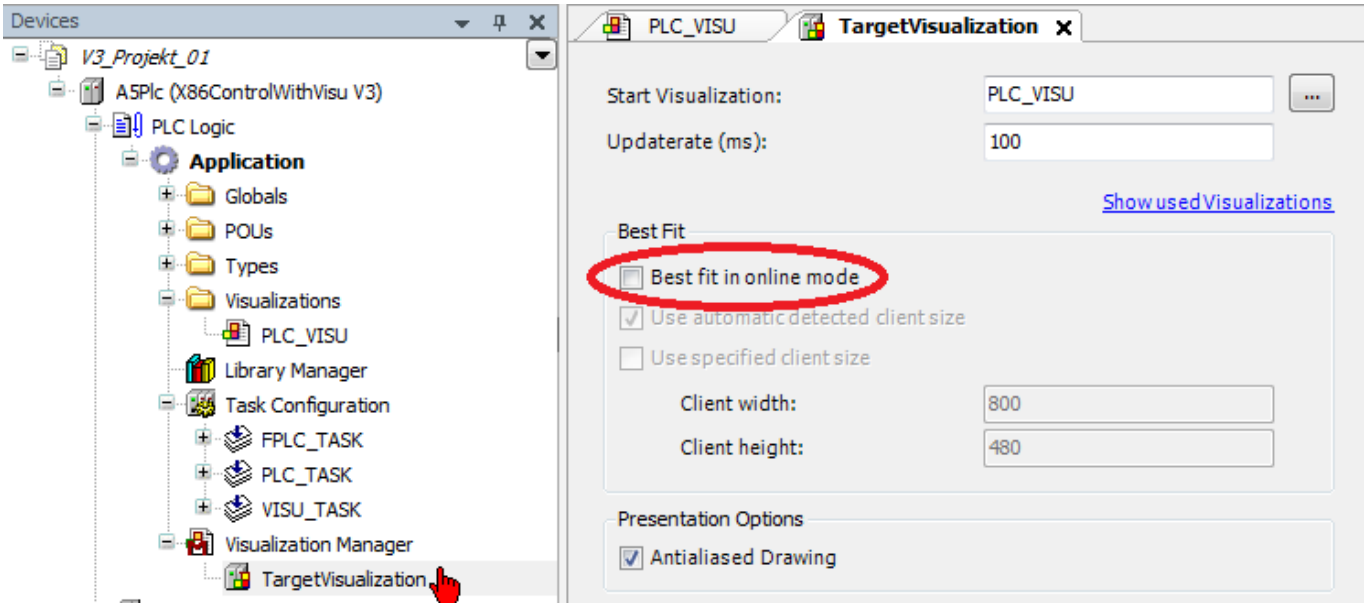
Acknowledgement signal for error-free activation of the motor control 'boQRF = TRUE'
The drive system is ready to process setpoints.

4.3.1.11 Visualization

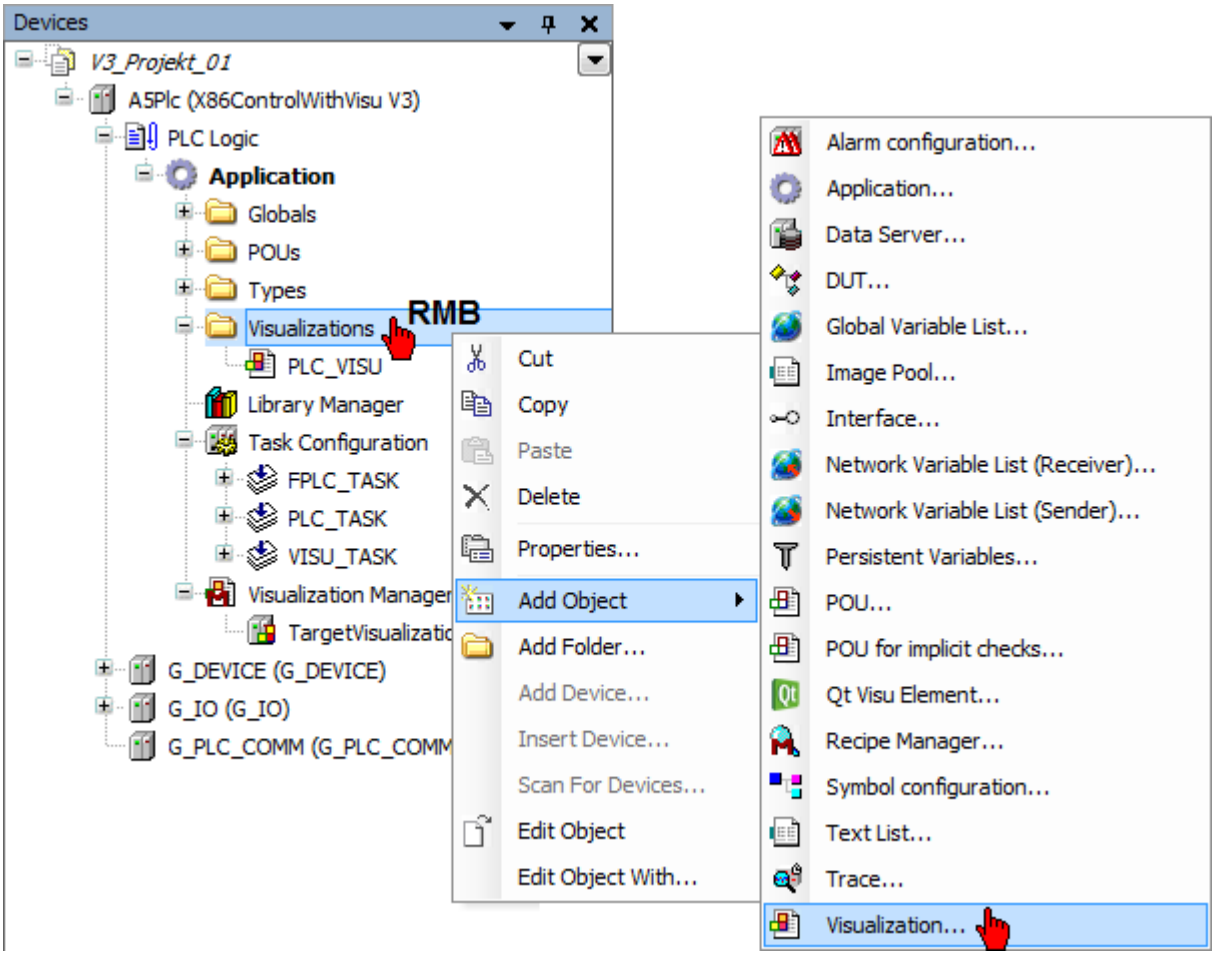
PLC_VISU is always the first visualization page called.



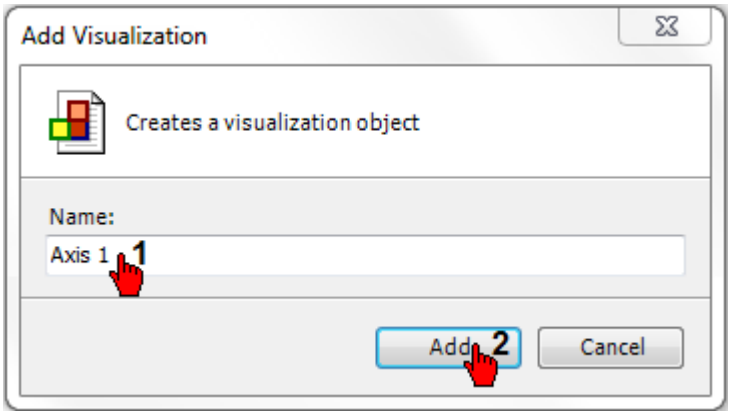
The basic settings are set on the 'TargetVisualization' page.



Create an empty page

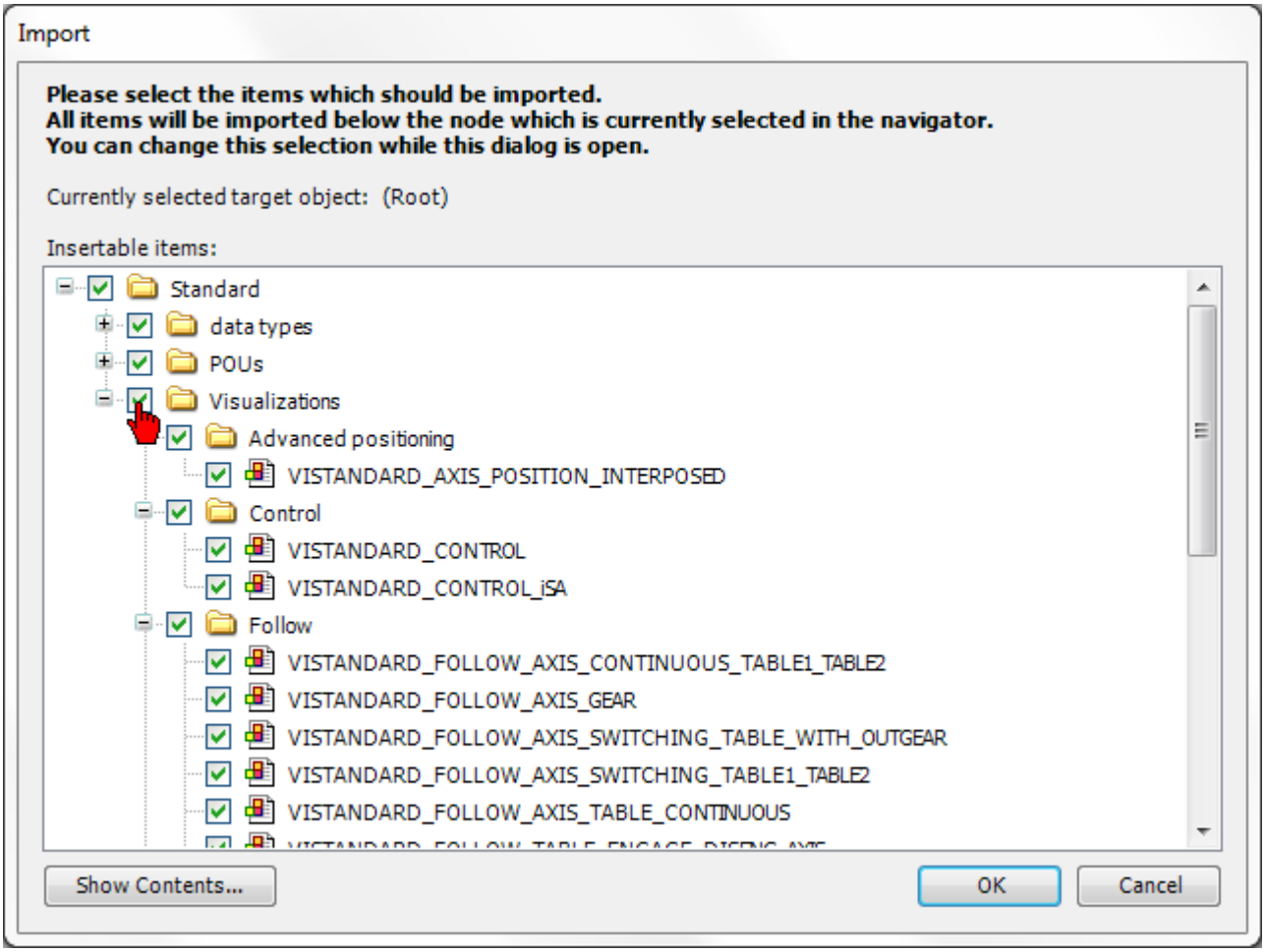


Assign a name



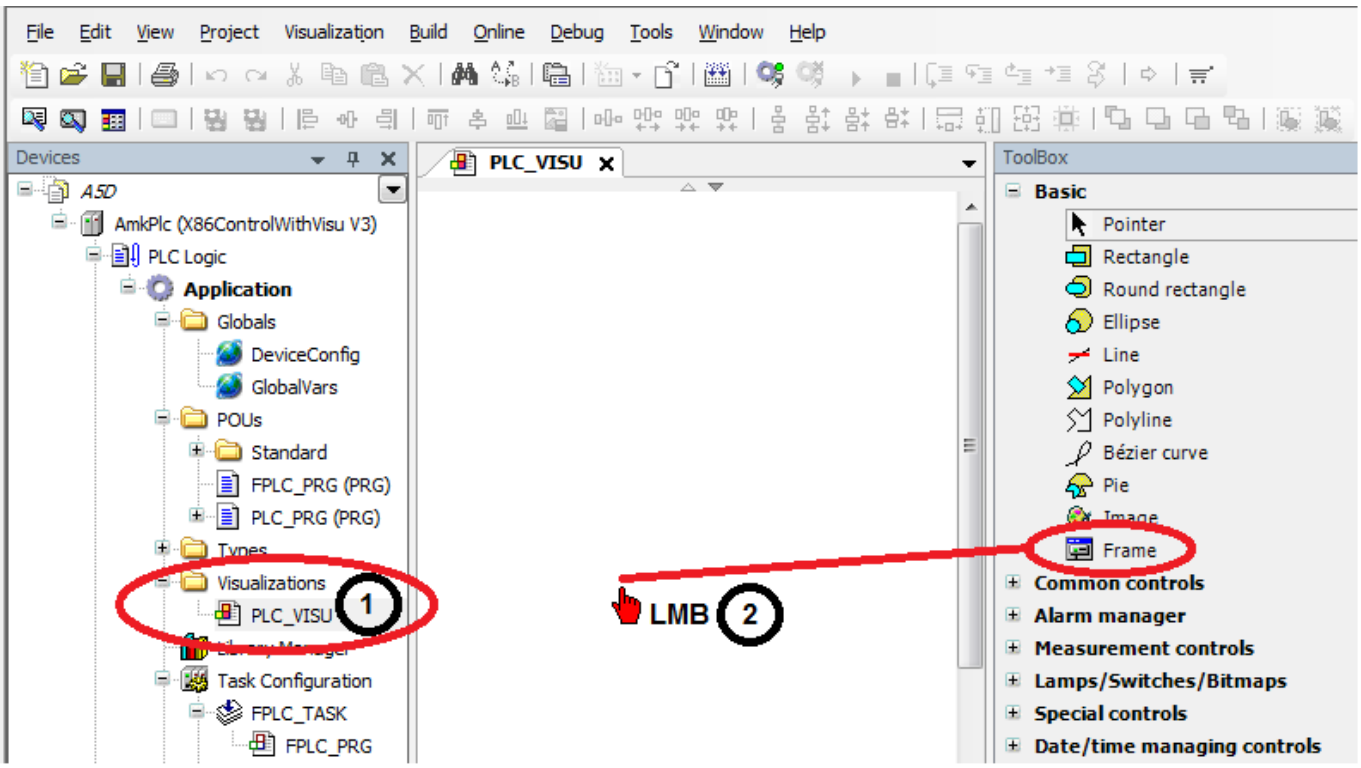
4.3.1.11.1 Visualization Standard blocks

The Standard block visualizations will be integrated (with CODESYS V3) inside the actual project by using the 'Import' function.
CODESYS menu 'Project' → 'Import'.
C:\Programs (x86)\3S CODESYS\AmkAddition\Imports\Application\V3_import.

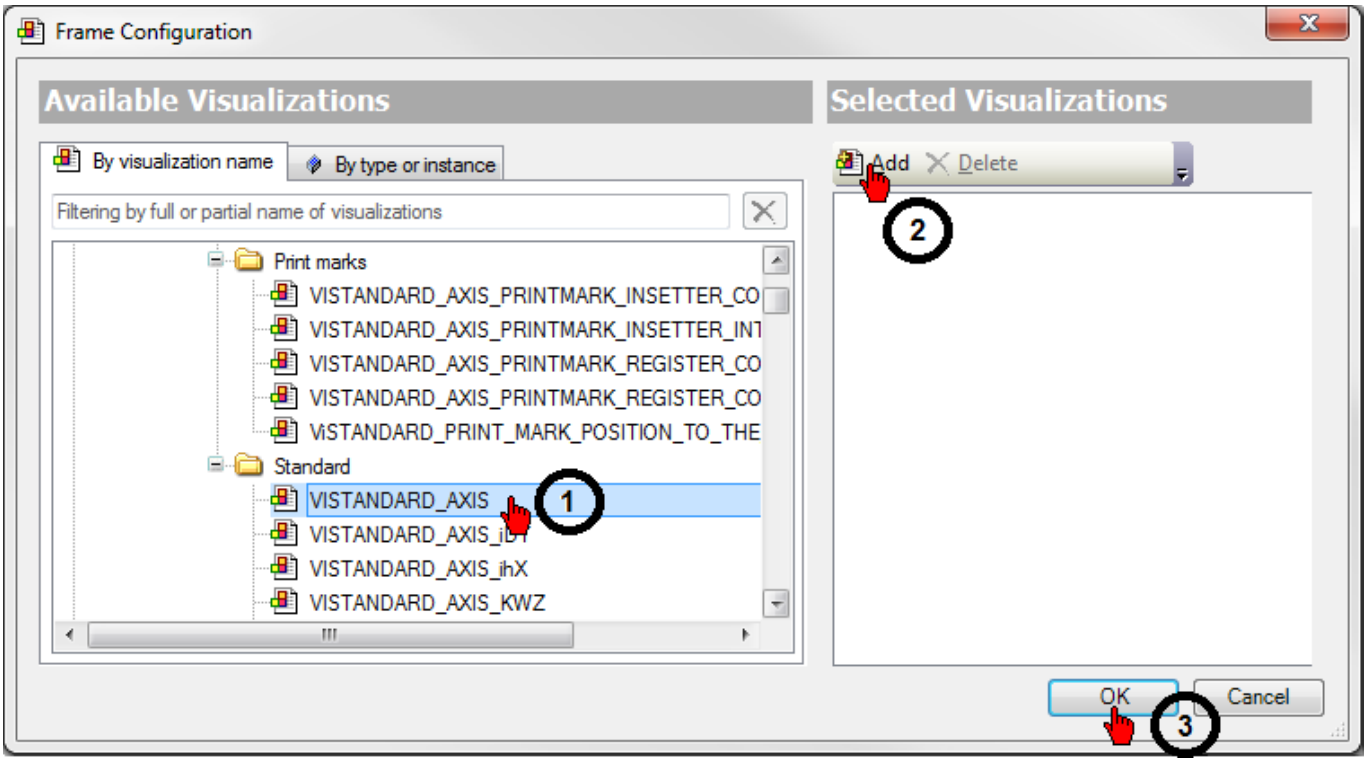


(The AFL Application block visualizations contained in an additional library. Siehe 'Visualization of AFL blocks' auf Seite 806.)

The visualizations are integrated with Vis-tool 'Frame'.

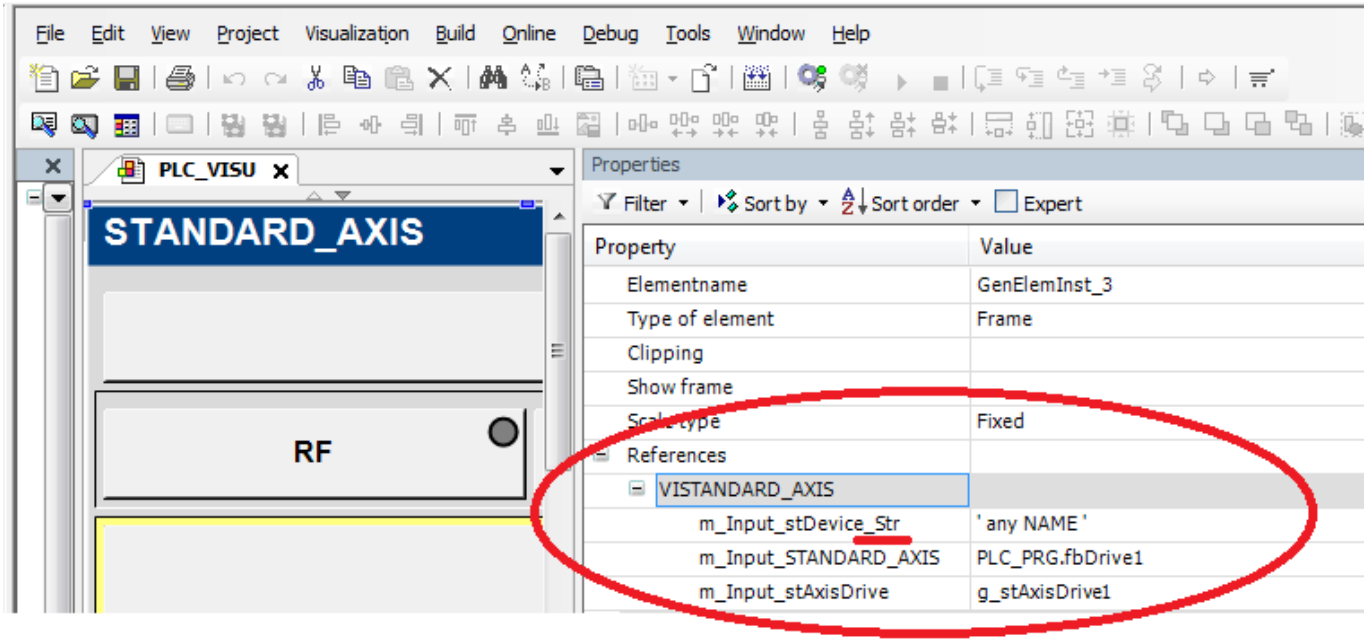


Chose the desired visualization with the button 'add'.

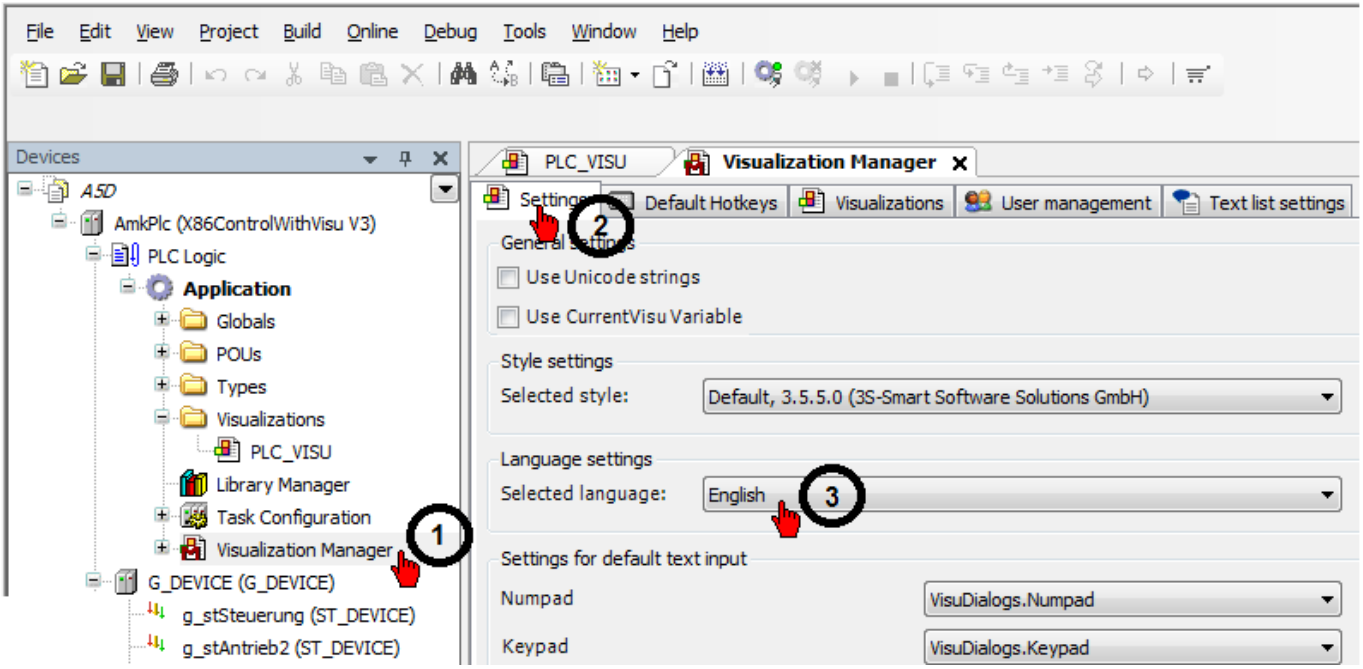


The inserted frame in the visualization must be linked to the instanced AFL Standard block and the associated structure ST_AXIS_DRIVE.

With m_Input_stDevice_Str specified a name (STRING) which will be displayed in the visualization.



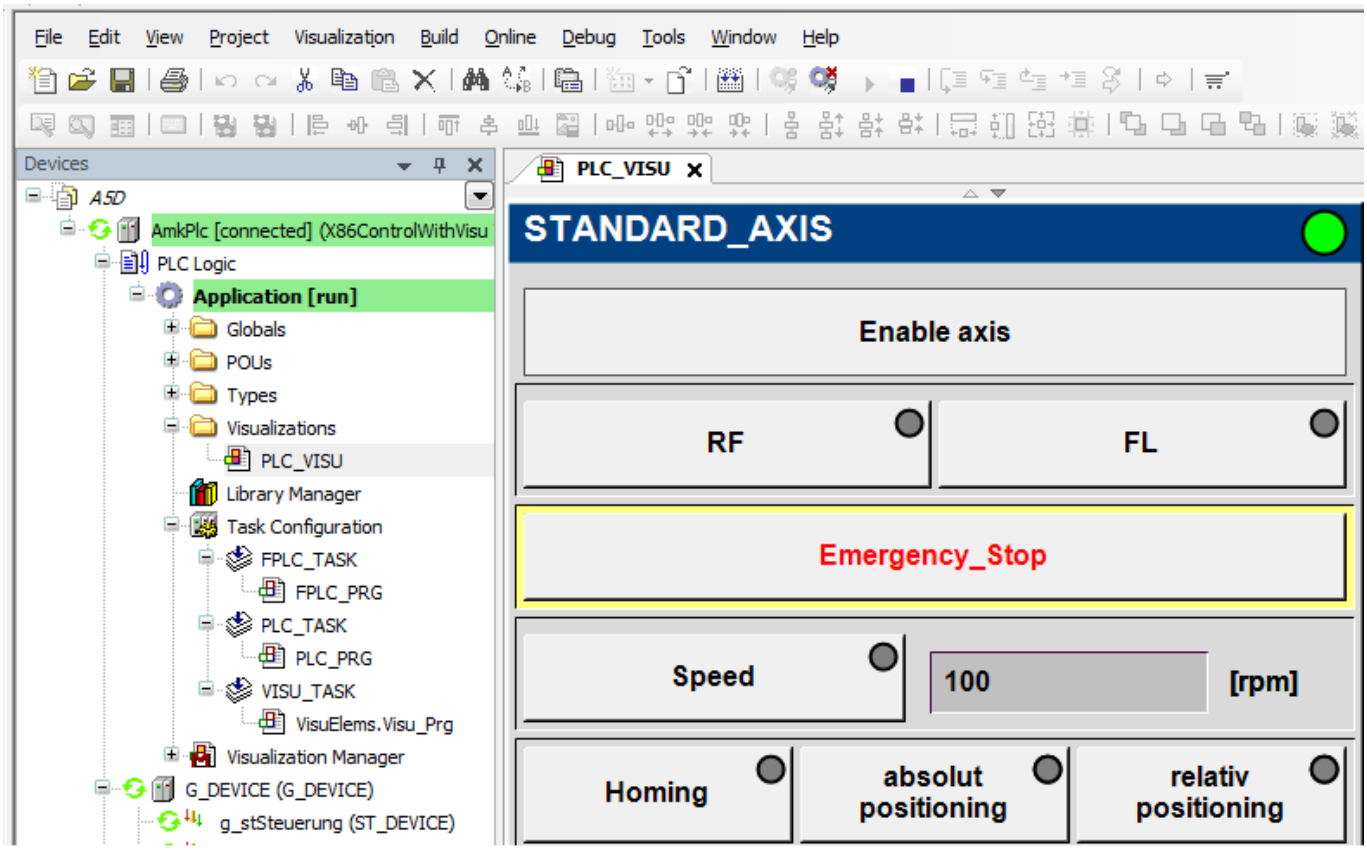
Selection visualization language.



Language-dependent texts are displayed only in the online mode.

The displayed texts (German and English) are stored in a text list: folder 'POUs' → 'Standard' → 'Visualizations' → 'Standard' → 'TextListStandard'

Display in 'RUN mode'



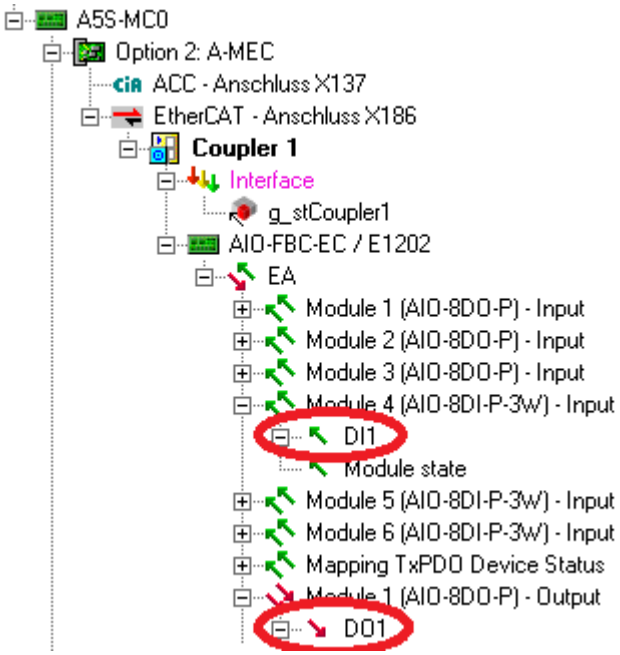
4.3.1.12 Functions

4.3.1.12.1 I/O access

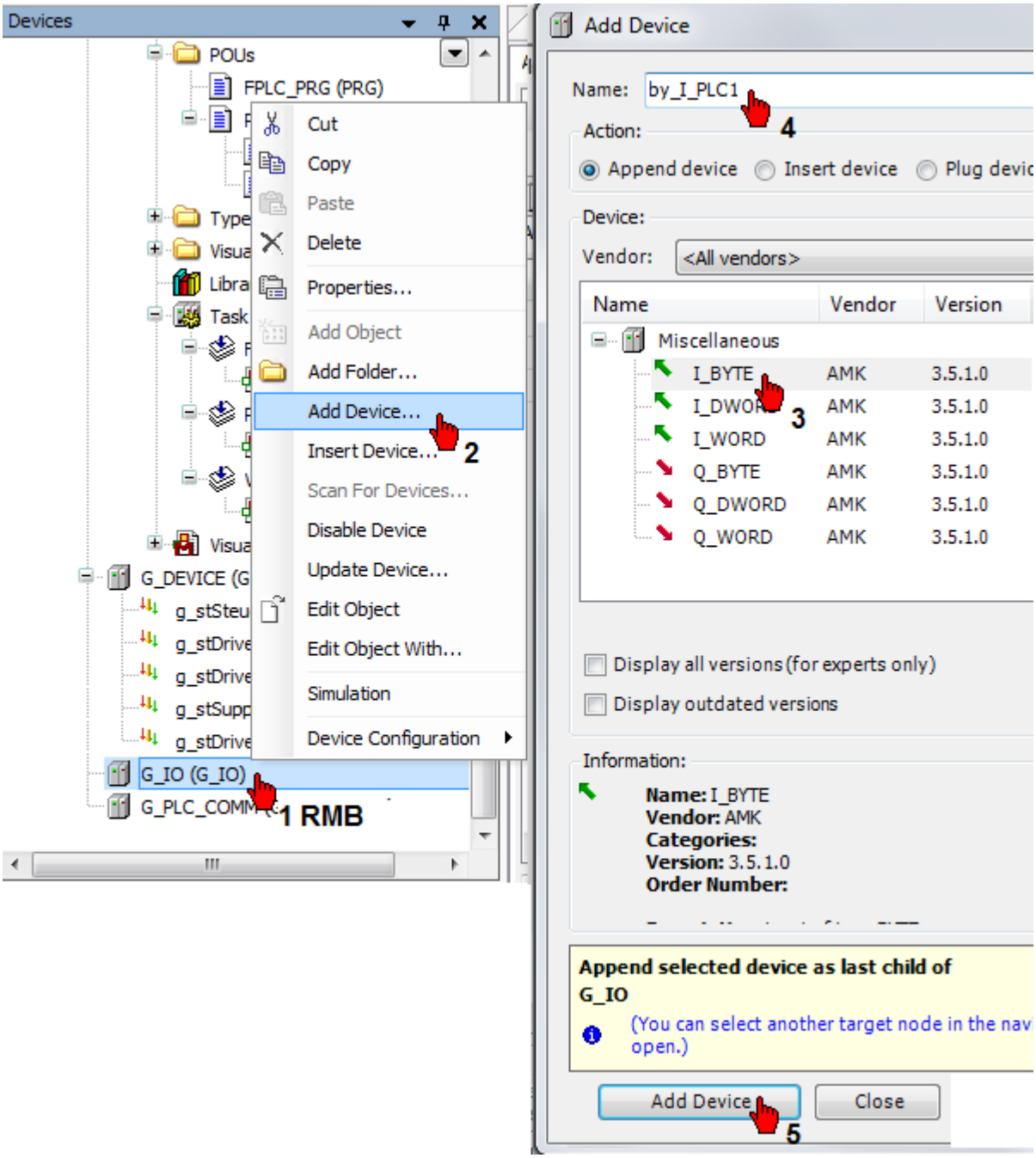
4.3.1.12.1.1 External I/O terminal (asynchronous)

Integration of an external I/O terminal. The example shows an AIO-FBC-EC fieldbus coupler with three AIO-8DO-P and three AIO-8DI-P-3W modules. The fieldbus coupler is detected automatically by AIPEX PRO.

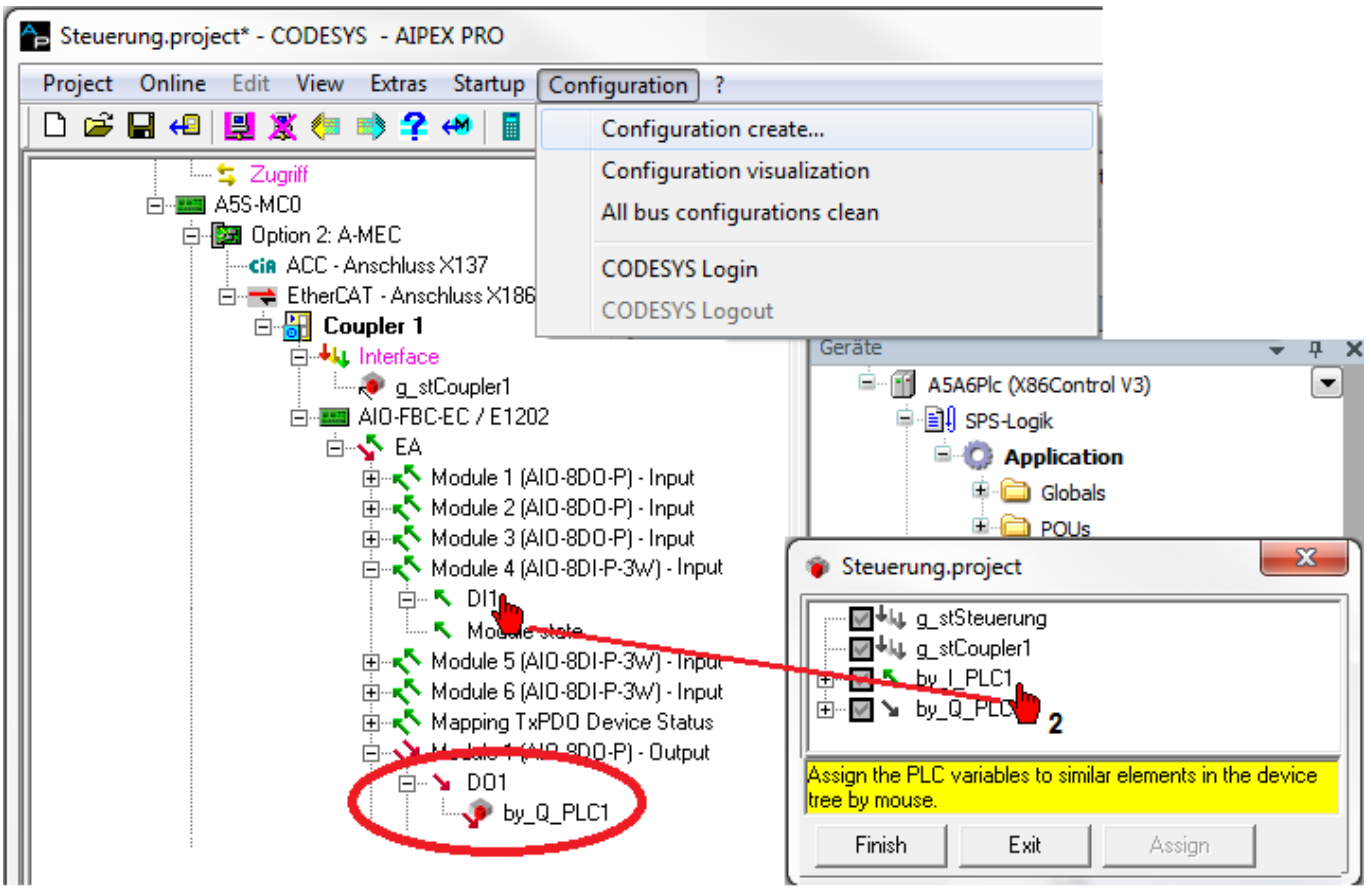
The I/Os are read (at program start) and written (at program end) within one program cycle.



A global I/O variable must be set in CODESYS for each input and output block.
In the example, an input byte by_I_PLC1 and an output byte by_Q_PLC1 is created.



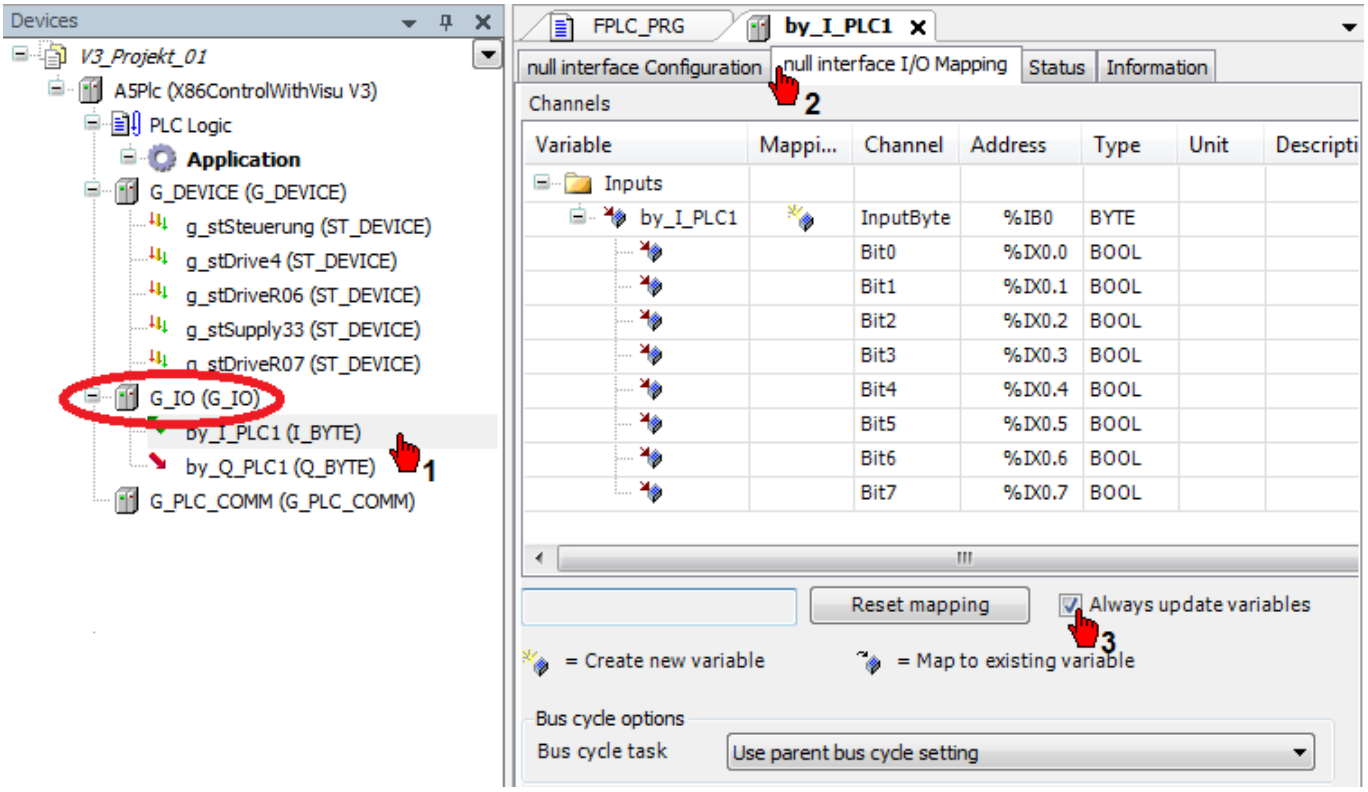
Call **'Configuration create'**. Assign the symbolic device names to the I/O interfaces in the AIPEX PRO device tree.



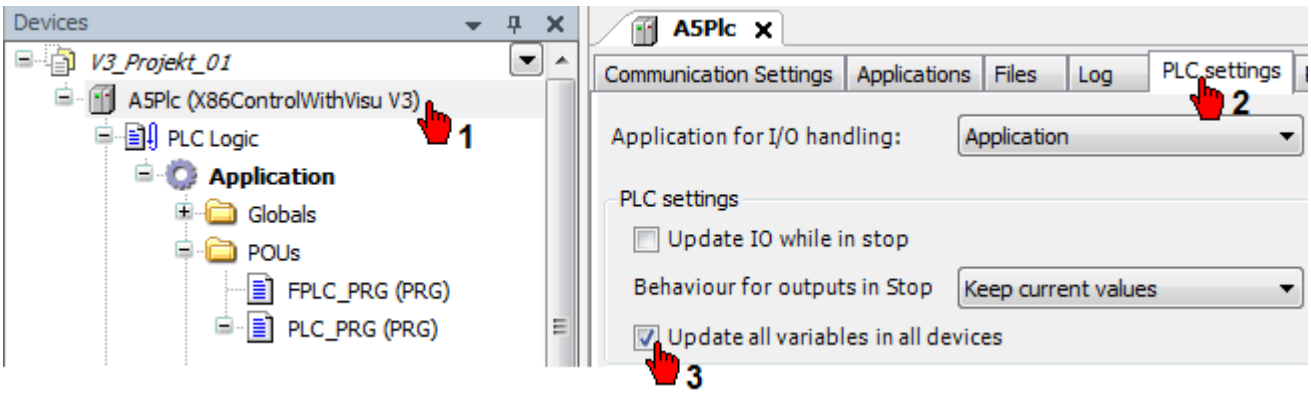
Testing

The following options are available for updating the I/O variables in the PLC project:

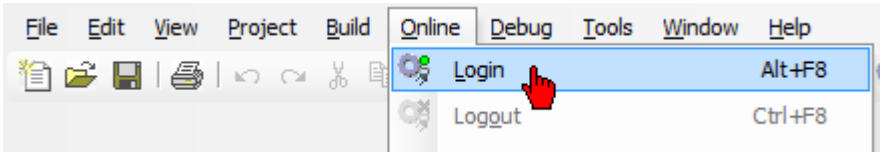
- Link the symbolic device names with PLC variables
- Set the option 'Always update variable' (see illustration)



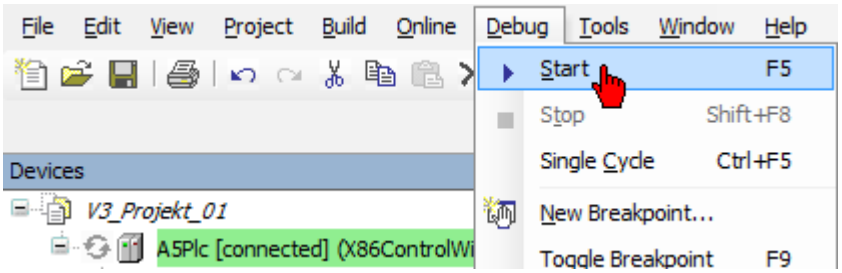
Alternatively, you can activate the 'Update all variables in all devices' option.



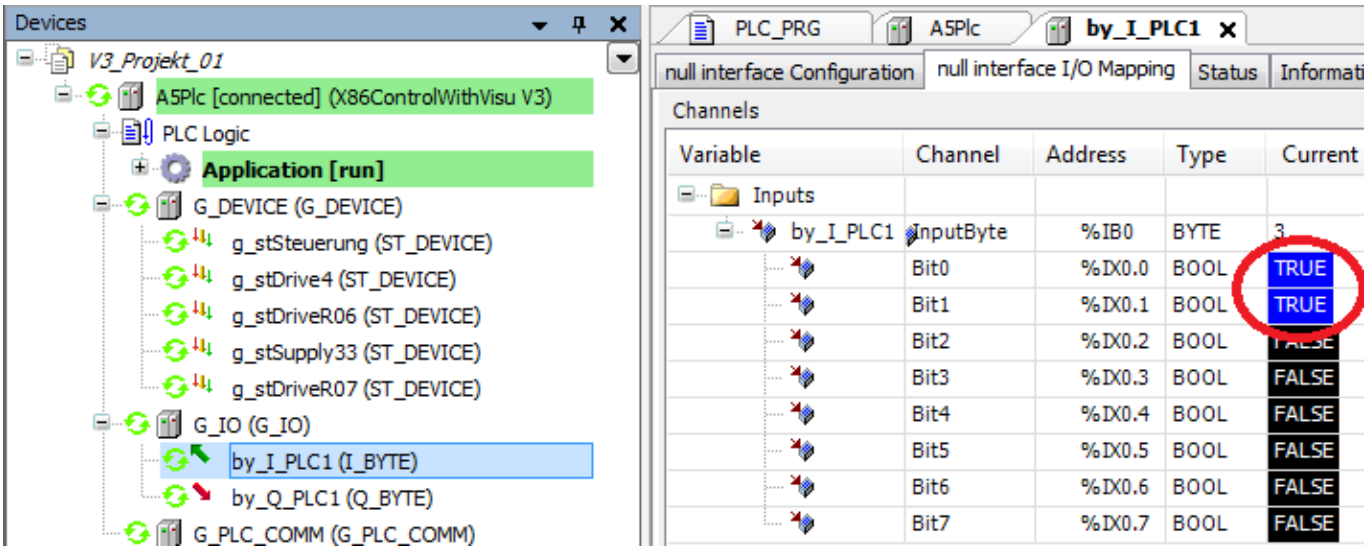
Call the **'Login'** function (code is generated automatically).



Start the application on the controller.



Set binary inputs on the controller for testing.

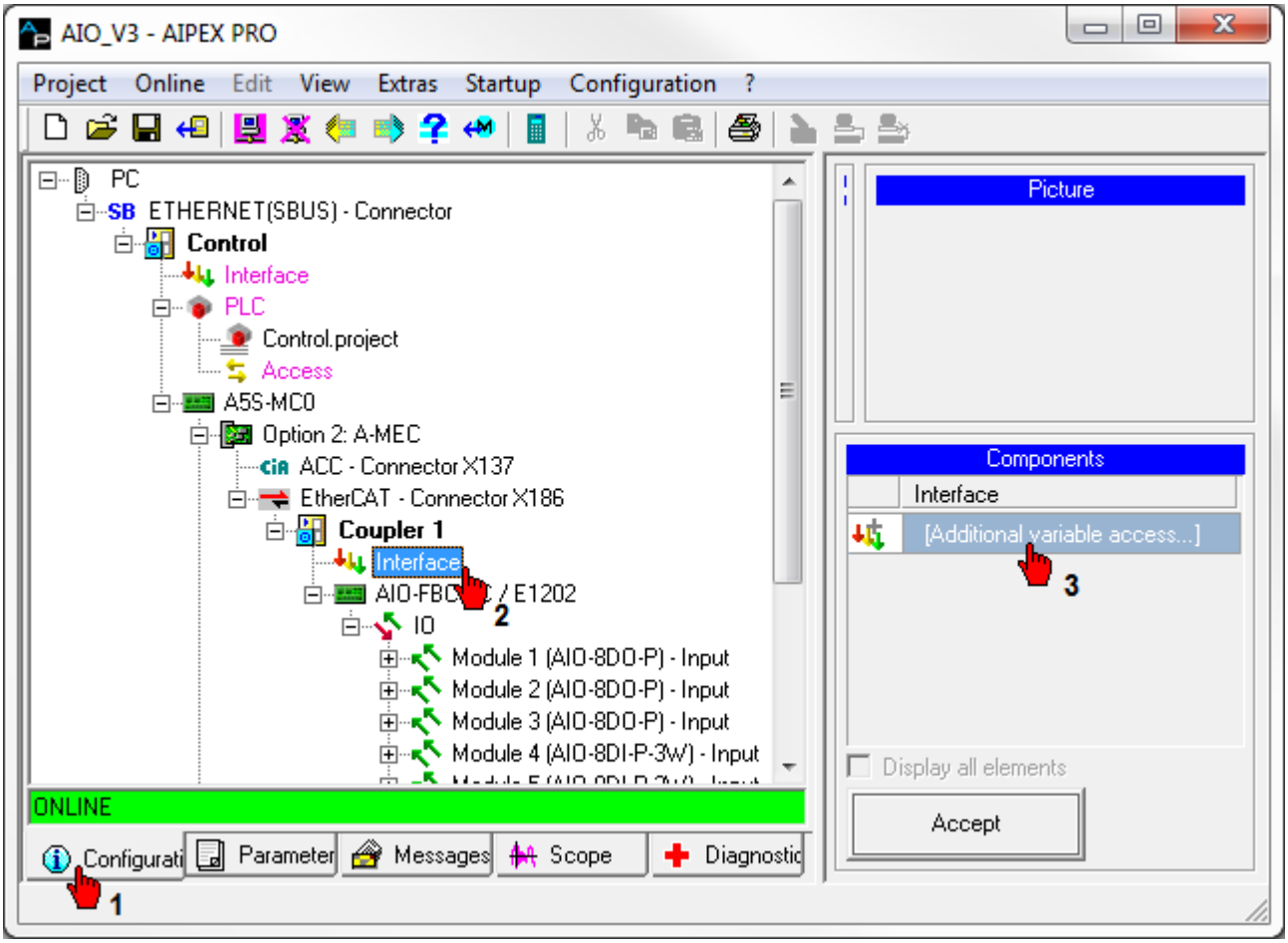


4.3.1.12.1.2 External I/O terminal (cyclical)

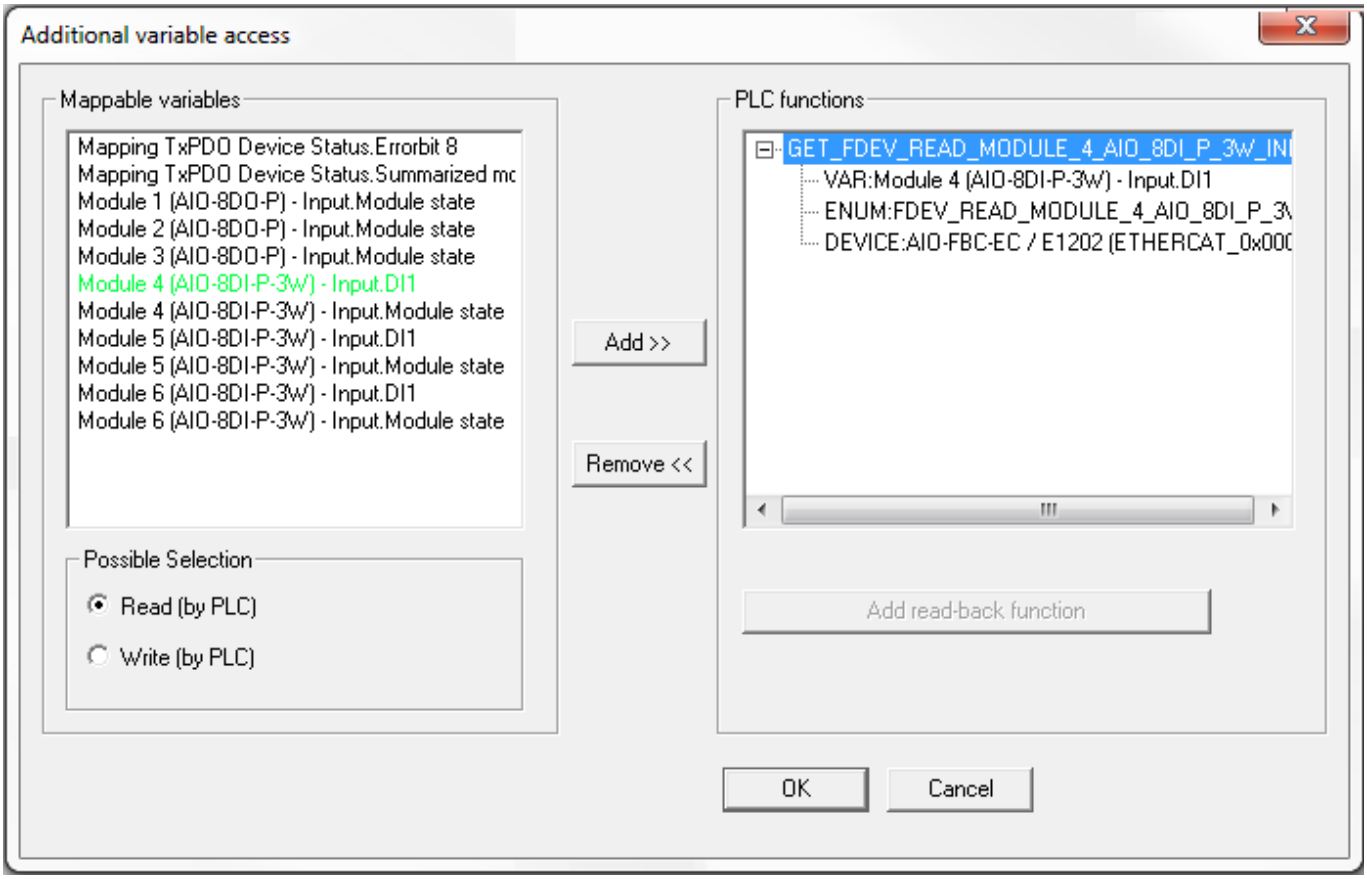
Integration of an external I/O terminal. The example shows an AIO-FBC-EC fieldbus coupler with three AIO-8DO-P and three AIO-8DI-P-3W modules. The fieldbus coupler is detected automatically by AIPEX PRO.

The procedure is based on the 'Additional variable access' principle. Siehe 'Additional variable access' auf Seite 687.

The cyclical reading and writing of an I/O map is performed via the device interface.
This is done by connecting an **'Interface'** to your terminal and open the **'Additional variable access'** window.

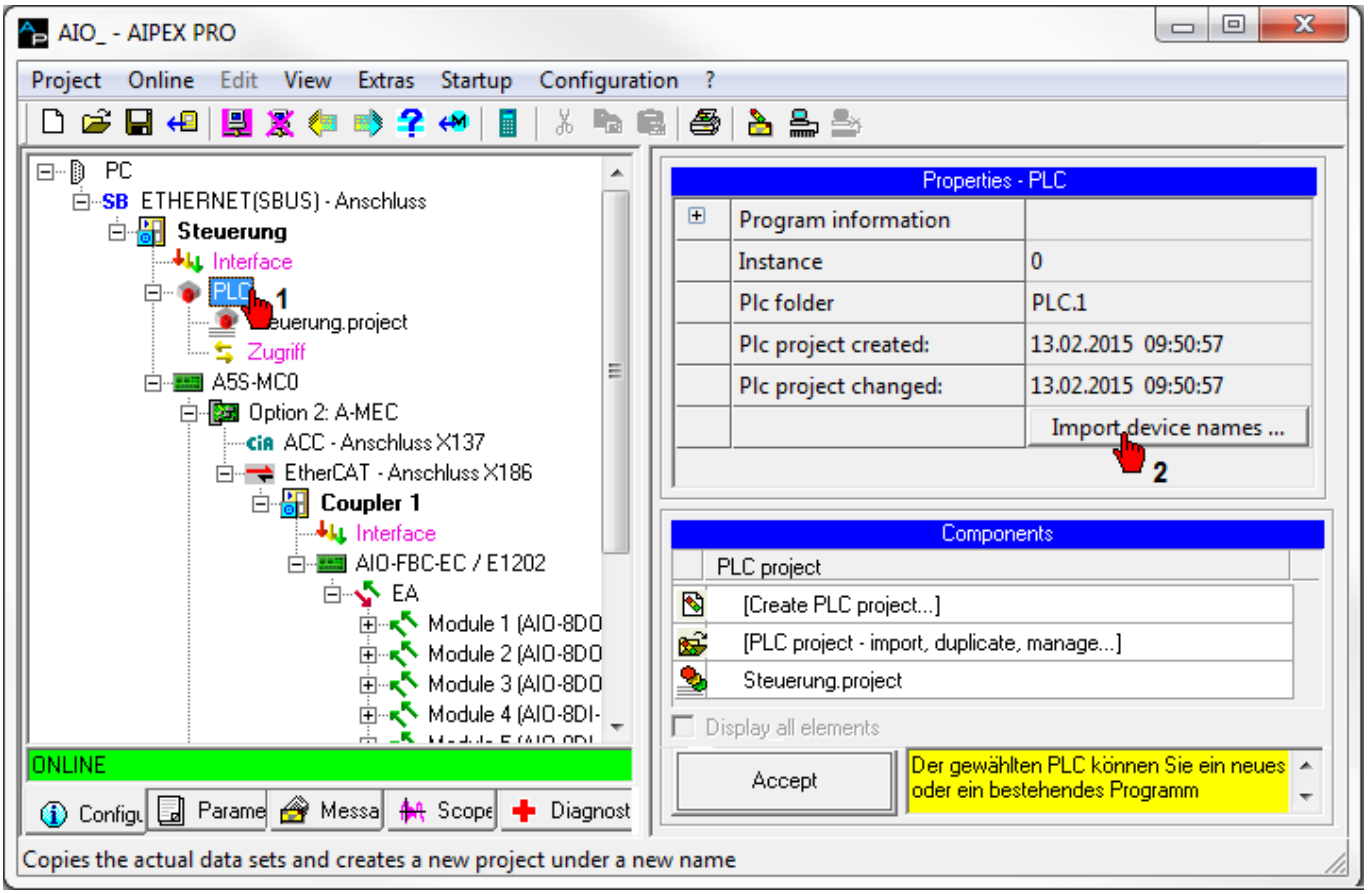


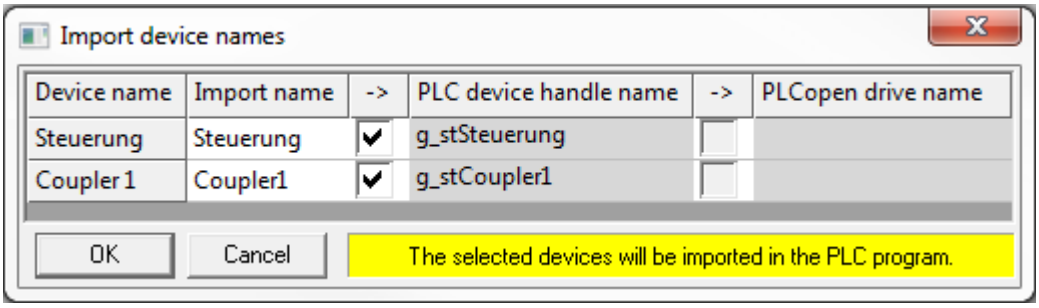
The possible selection of connectable variables is stored in the corresponding devices XML file.



Generate a 'Write' function block for the output module (AIO-8DO-P) in the same way

Add the device names to the PLC project. To do this, you can use the automatic **'Import device names'** function. Alternatively, you can create the terminals manually in the PLC controller configuration (G_DEVICE (G_DEVICE) and then assign them to the 'Interface' icon.

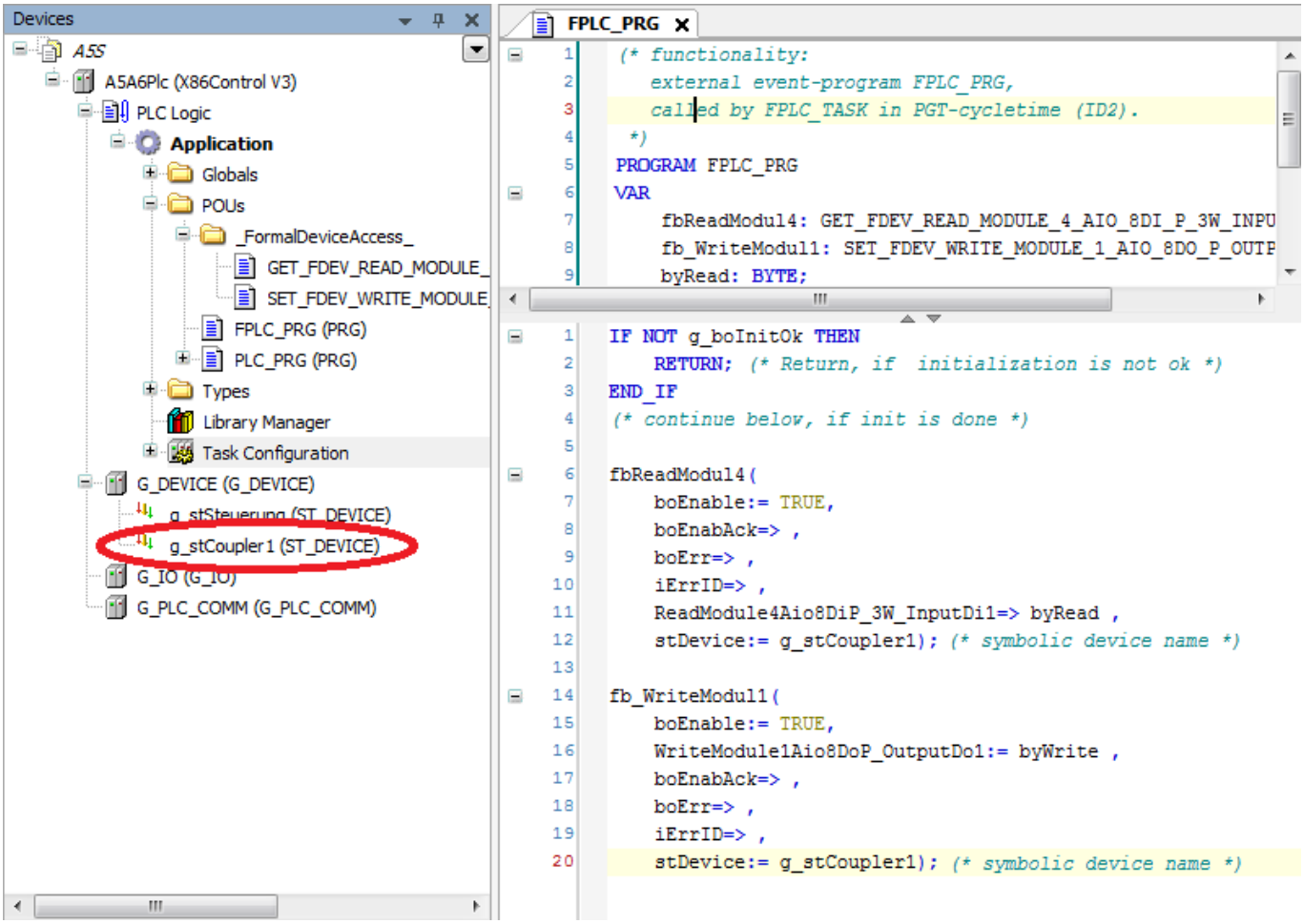




Open the CODESYS project. (Steuerung.project)

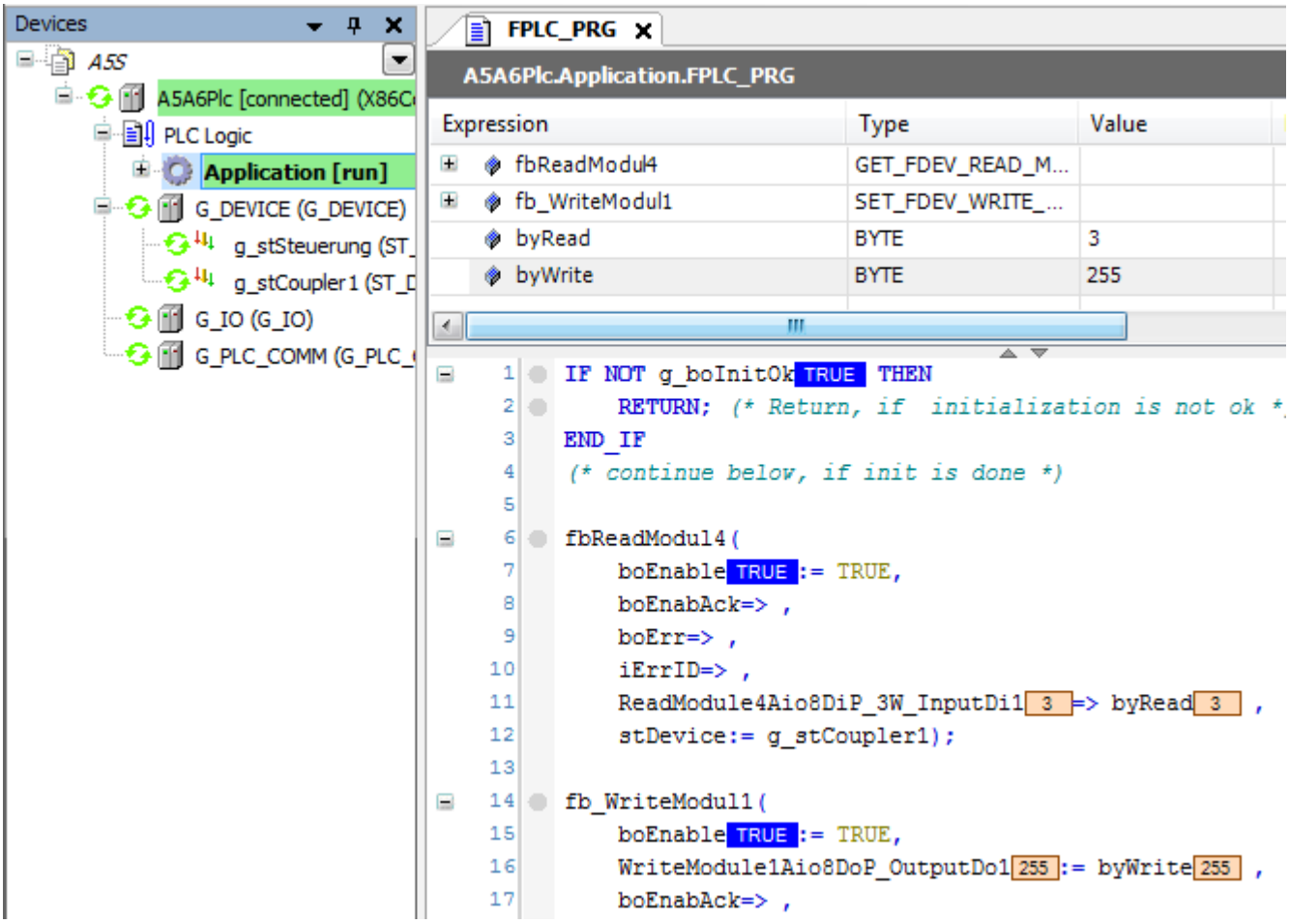
In the real time level (FPLC_PRG), add one instance each of the generated modules 'GET_FDEV_READ_MODULE_4_AIO_8DI_P_3W_INPUT_DI1' and 'SET_FDEV_WRITE_MODULE_1_AIO_8DO_P_OUTPUT_DO1'.

Connect each 'stDevice' variable with the corresponding symbolic device name.



Testing

Create the project: Siehe 'Creating PLC program and message configuration' auf Seite 663.



4.3.1.12.1.3 Time stamp IOs and XFC terminals

The section describes the access to the time stamp IOs of the A5 control with the 'IO extension'. The PLC function blocks specified are used for the time stamp IOs of the A5 control and for the XFC terminals EL 1252 und EL 2252. The functionality of the XFC terminals is compatible with the time stamp IOs of the A5 control.

PLC function blocks for the time stamp IO access are provided with the 'AmkDevAccess' AMK library.

See: Folder: 'POUs' → 'DeviceAccessSync' → 'TimeStamp'

Function block GET_TS_INPUTS (time stamp inputs)

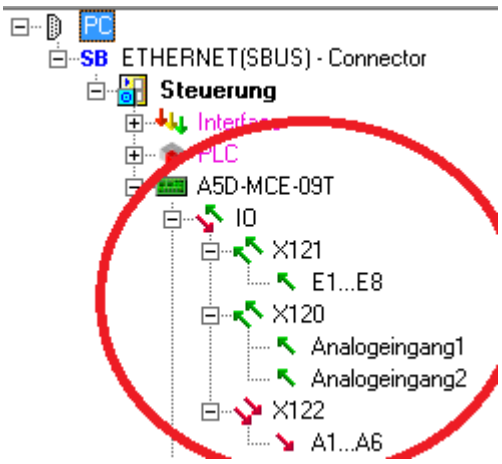
Function block SET_TS_OUTPUTS (time stamp outputs)

Description of the measurement inputs and time stamp outputs of the A5 control:

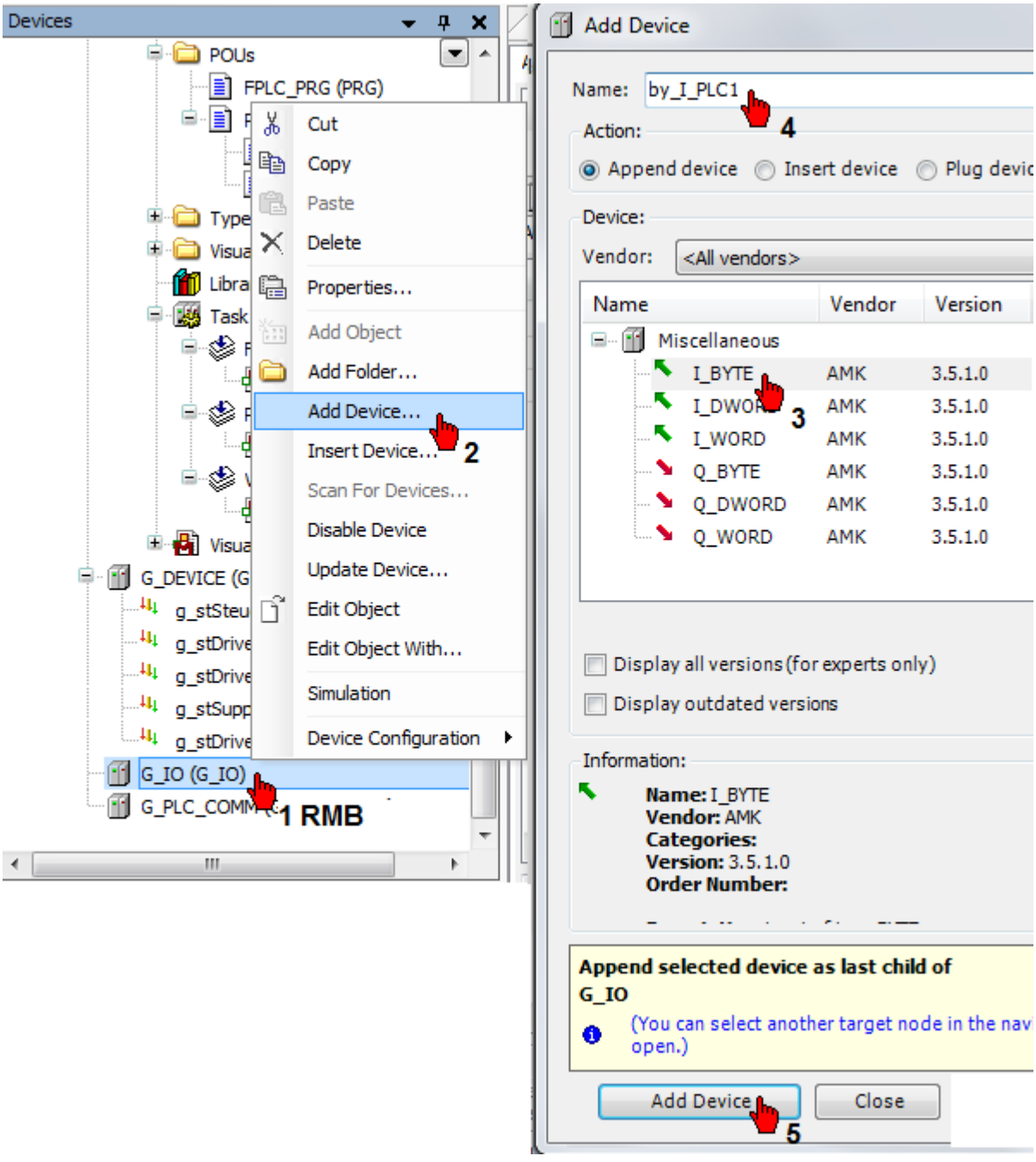
See product description Controllers A4 / A5 / A6 Section: 'Measurement Inputs' and Section 'Time Stamp Outputs'

4.3.1.12.1.4 IO Terminal control (asynchronous)

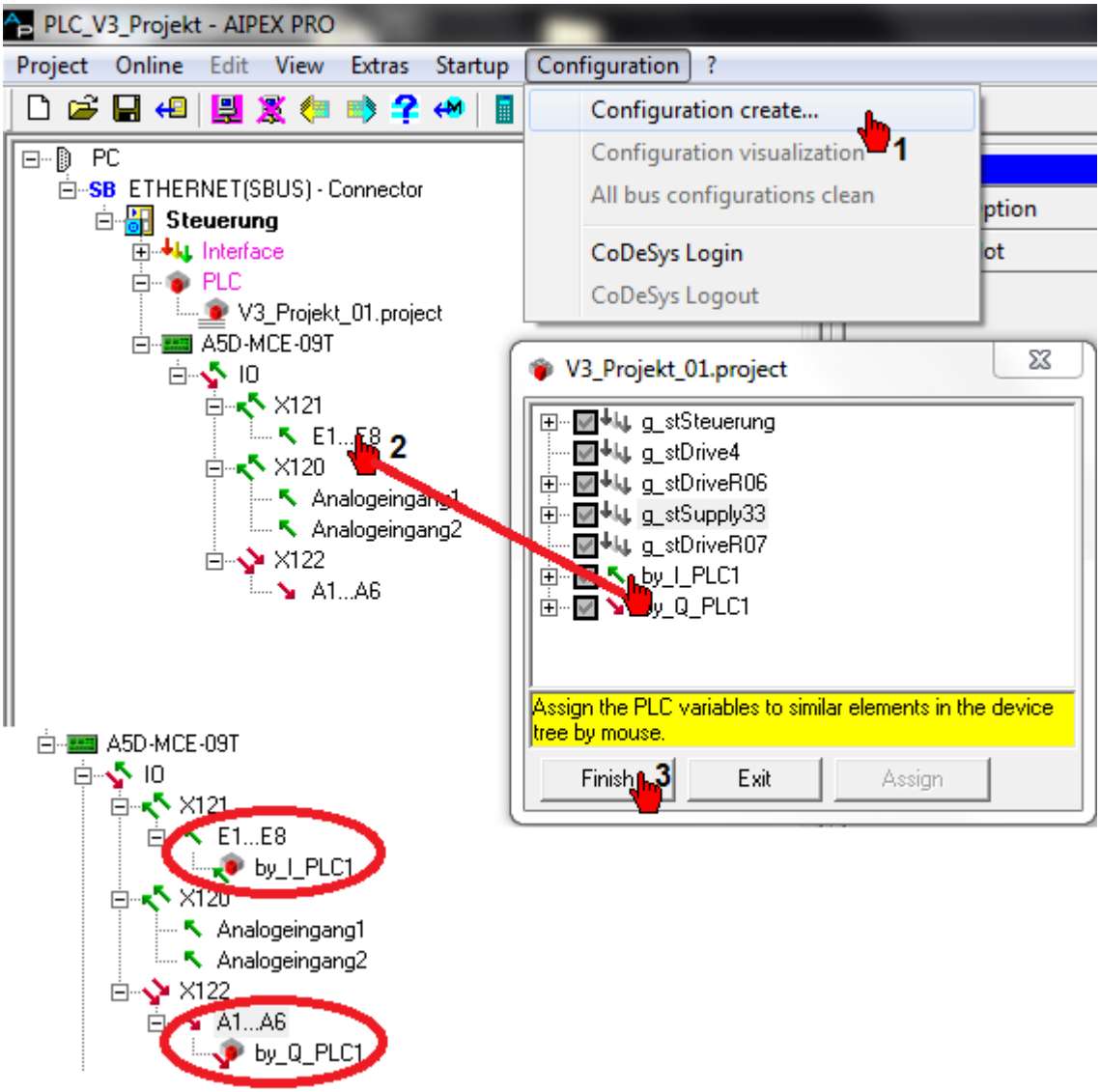
The I/Os are read (at program start) and written (at program end) within one program cycle.



A global I/O variable must be set in CODESYS for each input and output block.
In the example, an input byte by_I_PLC1 and an output byte by_Q_PLC1 is created.



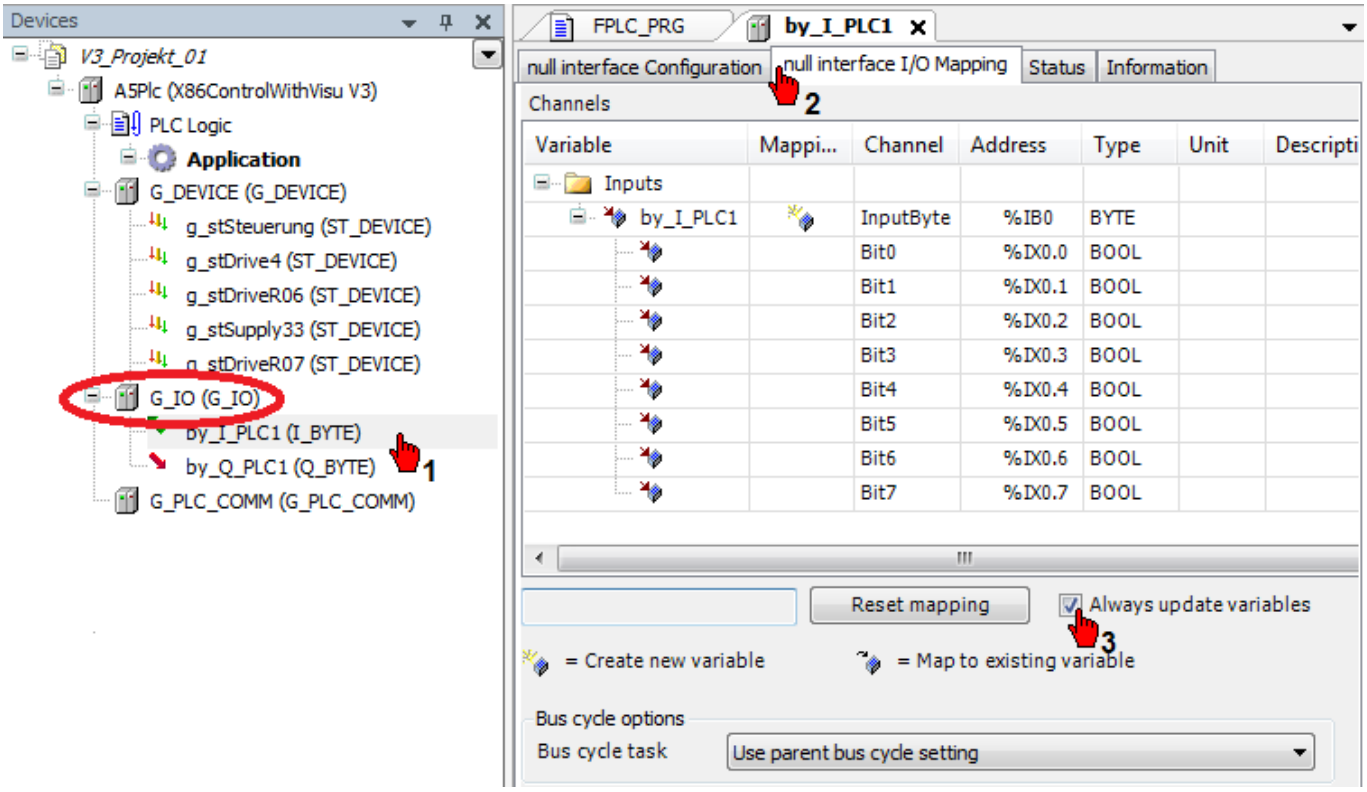
Call **'Configuration create'**. Assign the symbolic device names to the I/O interfaces in the AIPEX PRO device tree.



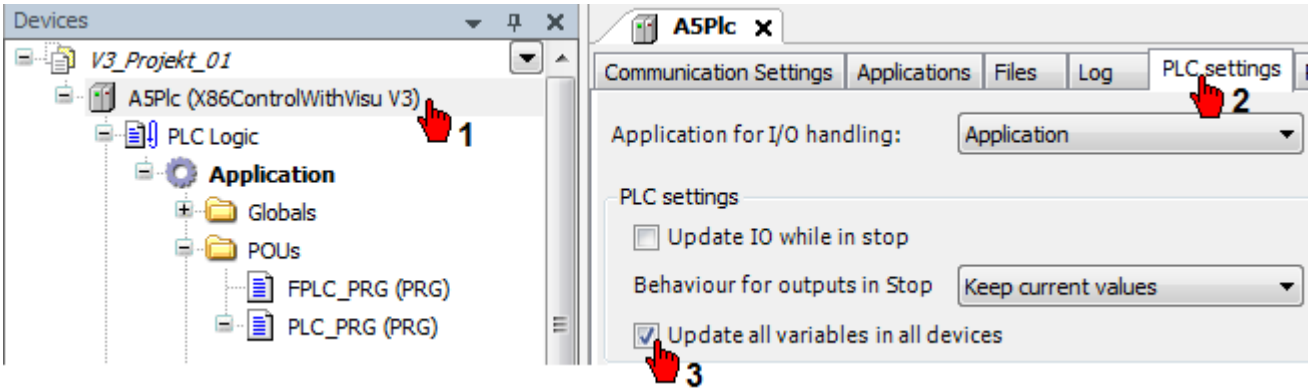
Testing

The following options are available for updating the I/O variables in the PLC project:

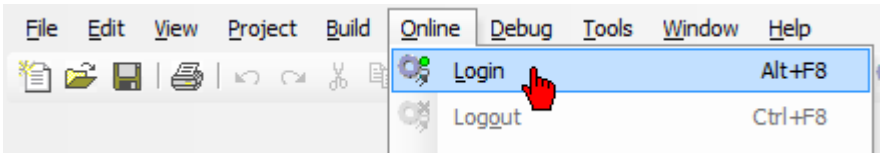
- Link the symbolic device names with PLC variables
- Set the option 'Always update variable' (see illustration)



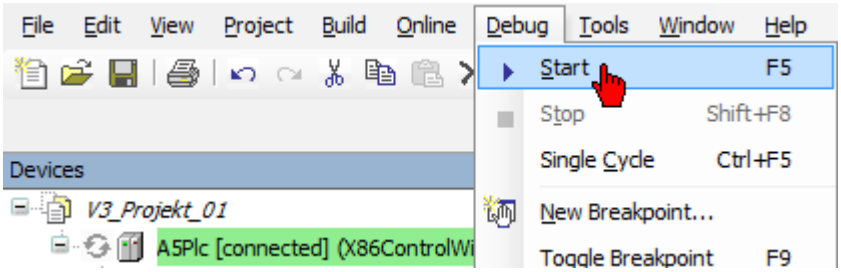
Alternatively, you can activate the 'Update all variables in all devices' option.



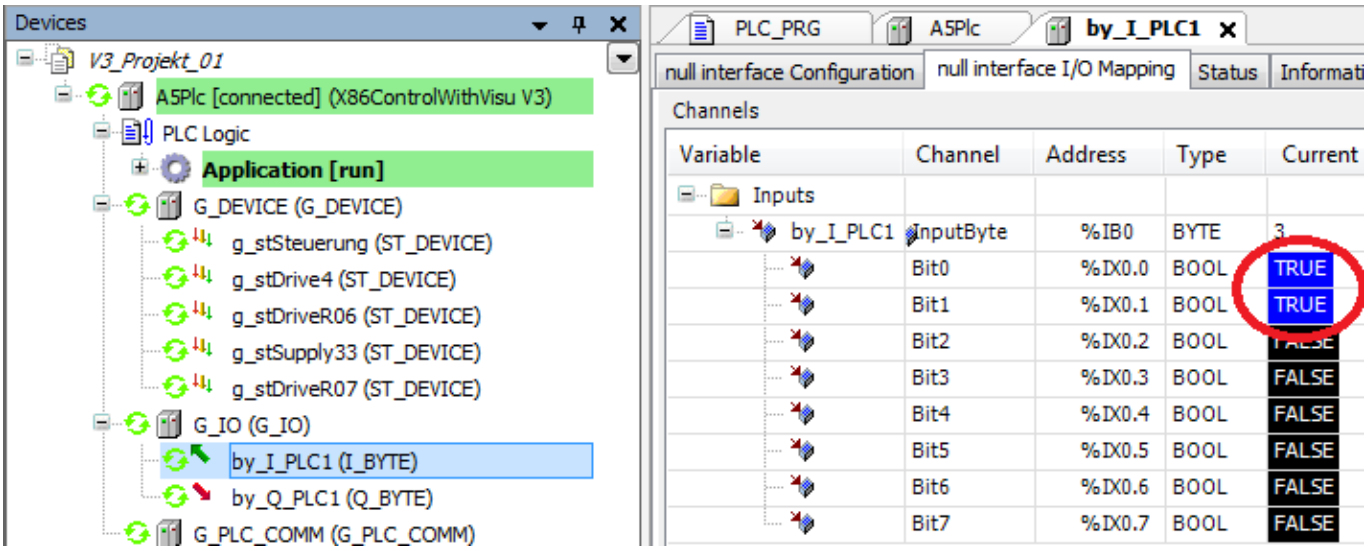
Call the **'Login'** function (code is generated automatically).



Start the application on the controller.



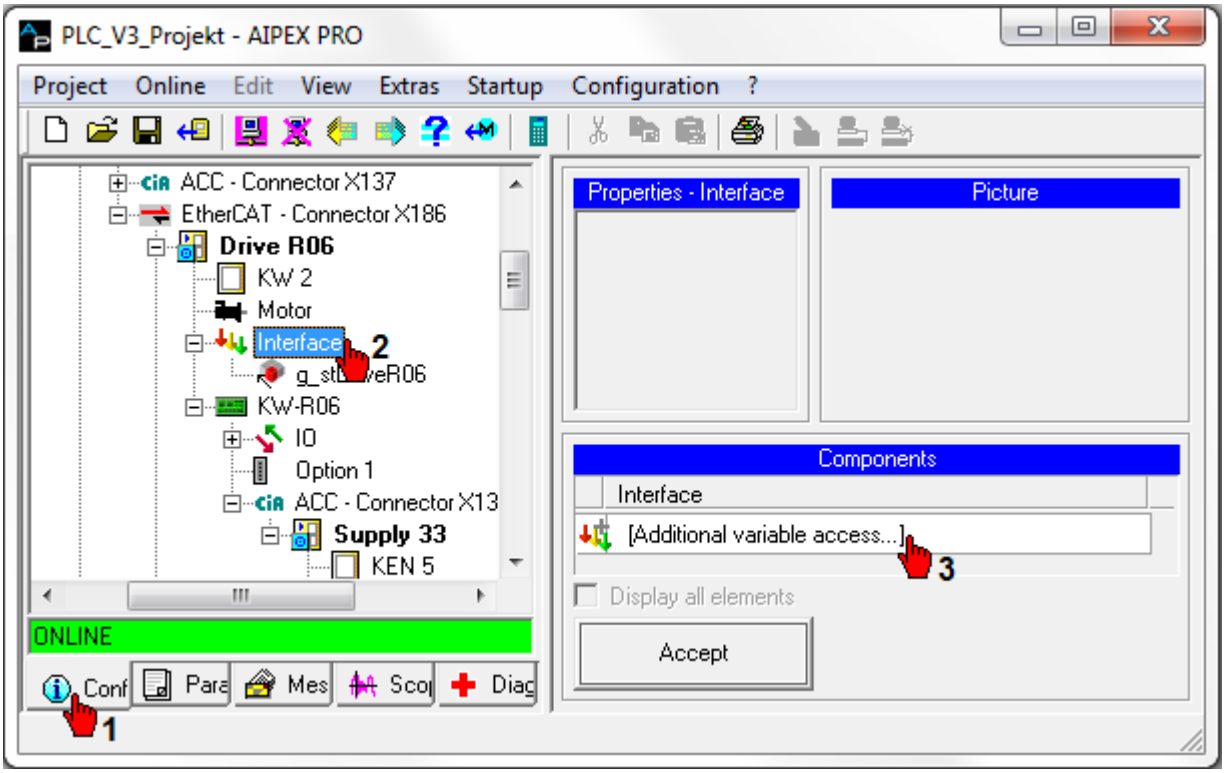
Set binary inputs on the controller for testing.



4.3.1.12.2 Additional variable access

The AMK libraries provide a selection of function blocks to exchange real-time bit signals, actual values, setpoints, etc., between drive and PLC. Nonexistent (formal) modules can be created using the 'additional variable access' option.

To do this, select 'Interface' for which a new function block is to be created.

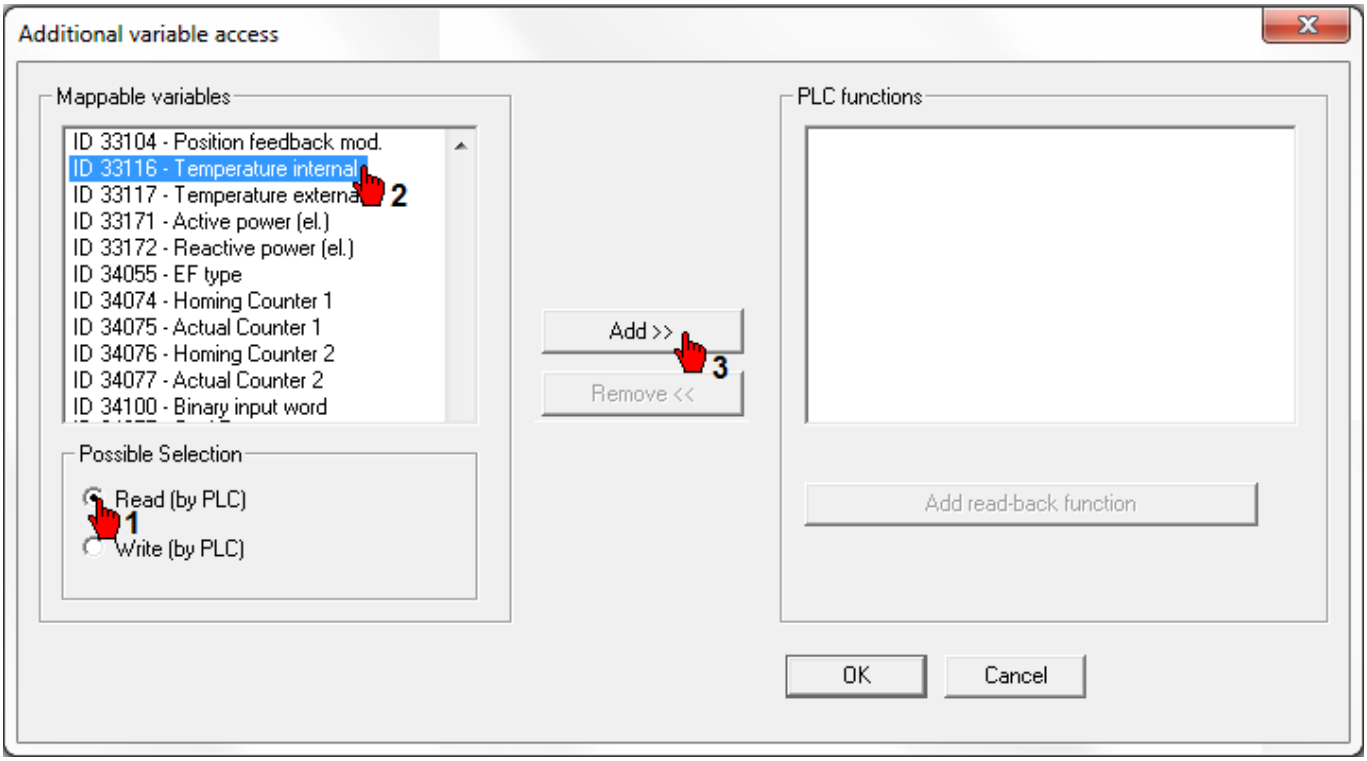


In the first step, choose whether the PLC should have reading or writing access to the drive.

All values available in the drive can be found in the variable list.

The variable selection is added by pressing the **'Add'** button.

The 'Readback function allows 'GET_FDEV_...' modules to be generated with which you can read the values written with 'SET_FDEV_...' modules.

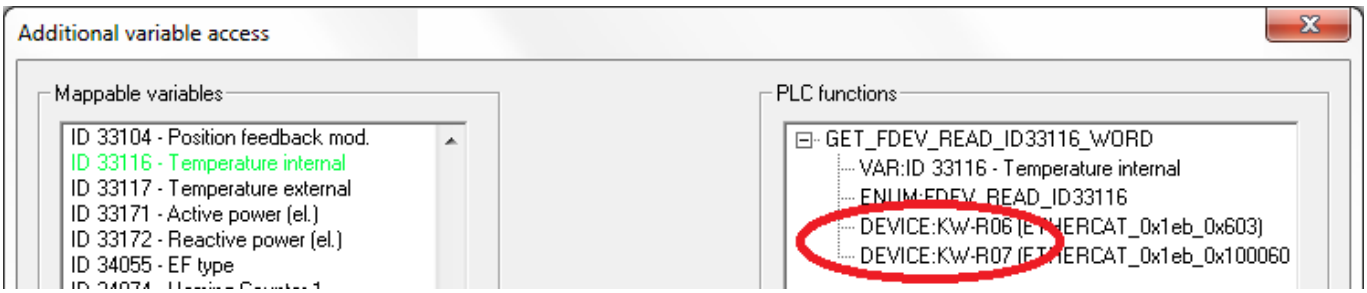


Additional information:

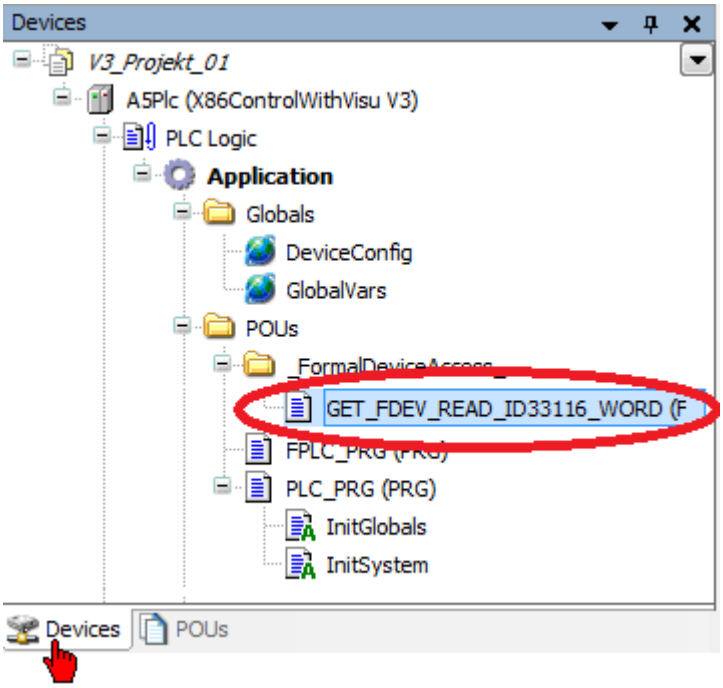
You can instantiate the created FDEV module for all devices in the PLC project that have the same device type. E.G. DEVICE: KW-R06.

If your PLC project contains different device types (e.g. an additional DEVICE: KW-R07), you have to select the 'interface' of this device and execute the 'Additional variable access' function in exactly the same way.

FDEV formal module, linked to the devices KW-R06 and KW-R07

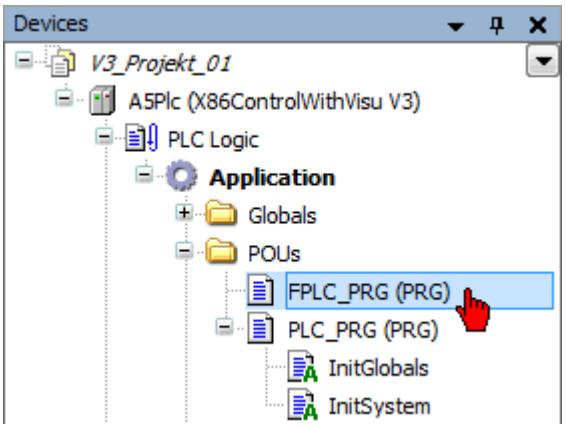


Open the programming editor.
You can find the generated function blocks under 'Devices' → 'Application' → 'POUs' → _FormalDeviceAccesses_.

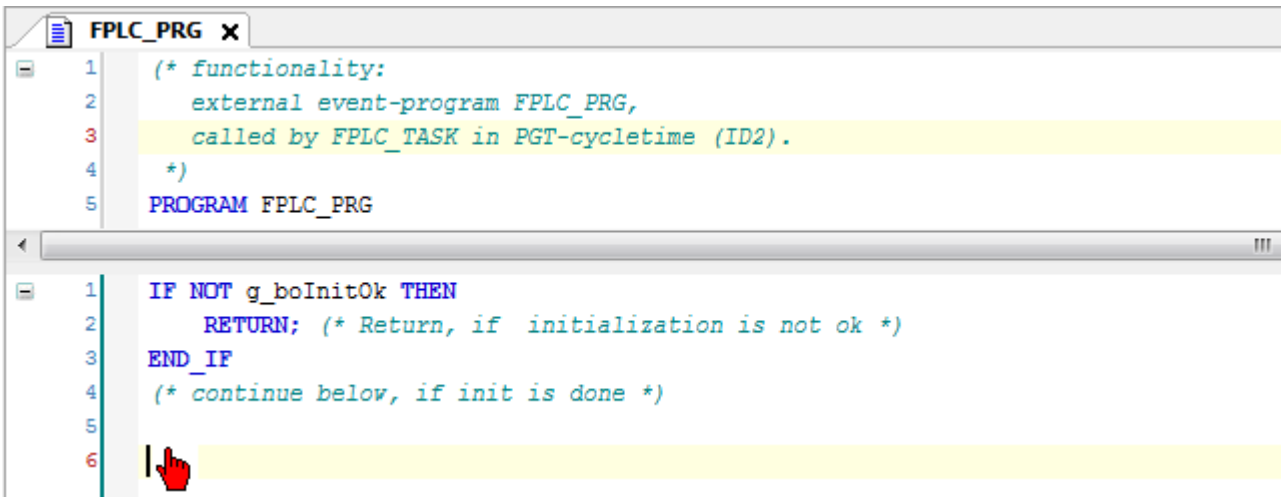


Consistent data transmission is only guaranteed with an instance call in the synchronous FPLC_PRG program module.

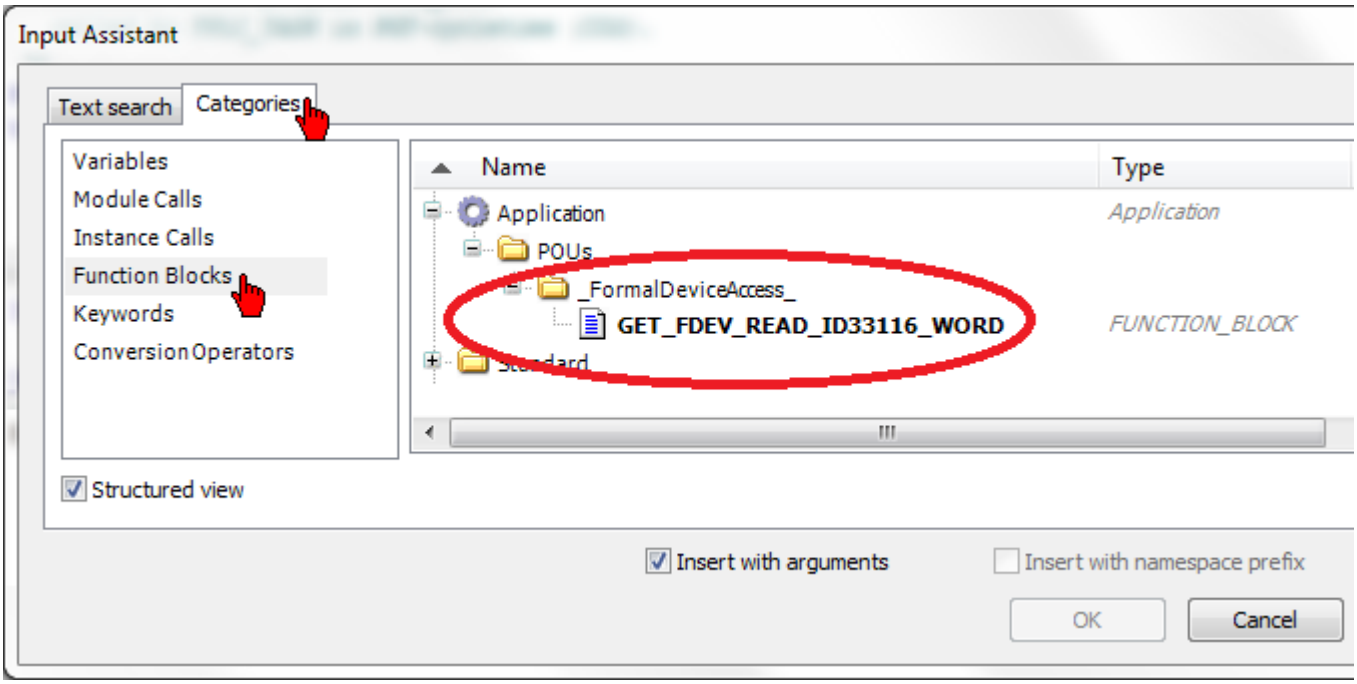
Switch to the 'Devices' tab
Open the program object 'FPLC_PRG (PRG)'



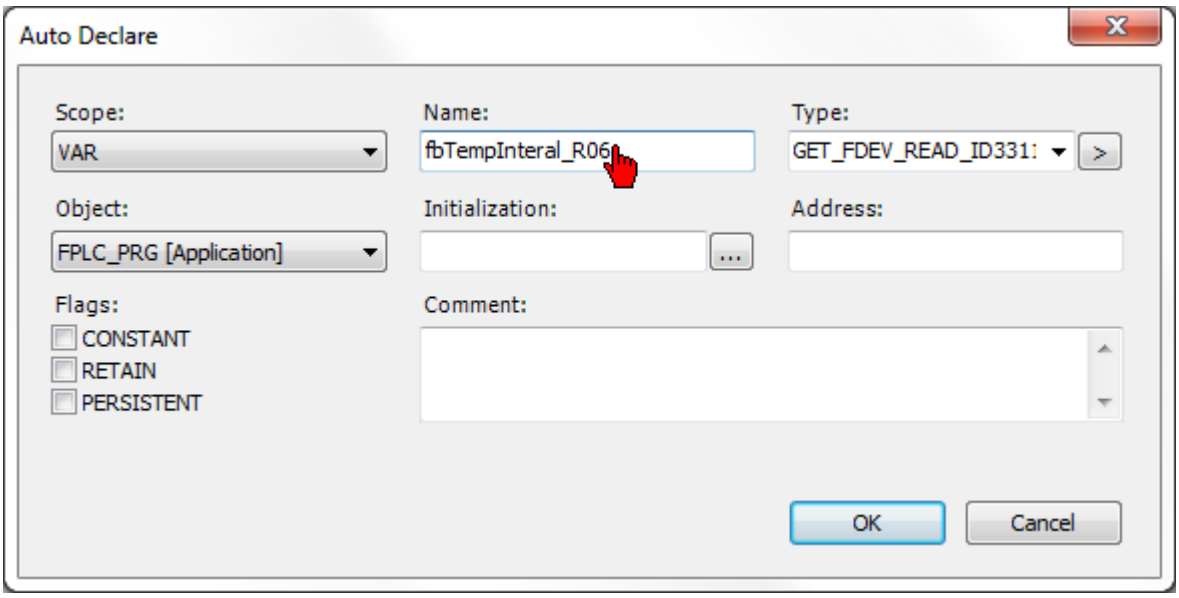
Click in an empty line in the program editor.



Press the **'F2'** button to open the Input assistant.
You can find the generated function blocks under **'Categories'** → **'Function blocks'** → **'Application' -> 'POUs' -> '_FormalDeviceAccesses_'**.
Click on the function block to add it to the project.



You can instantiate the function block by assigning its own name to it.



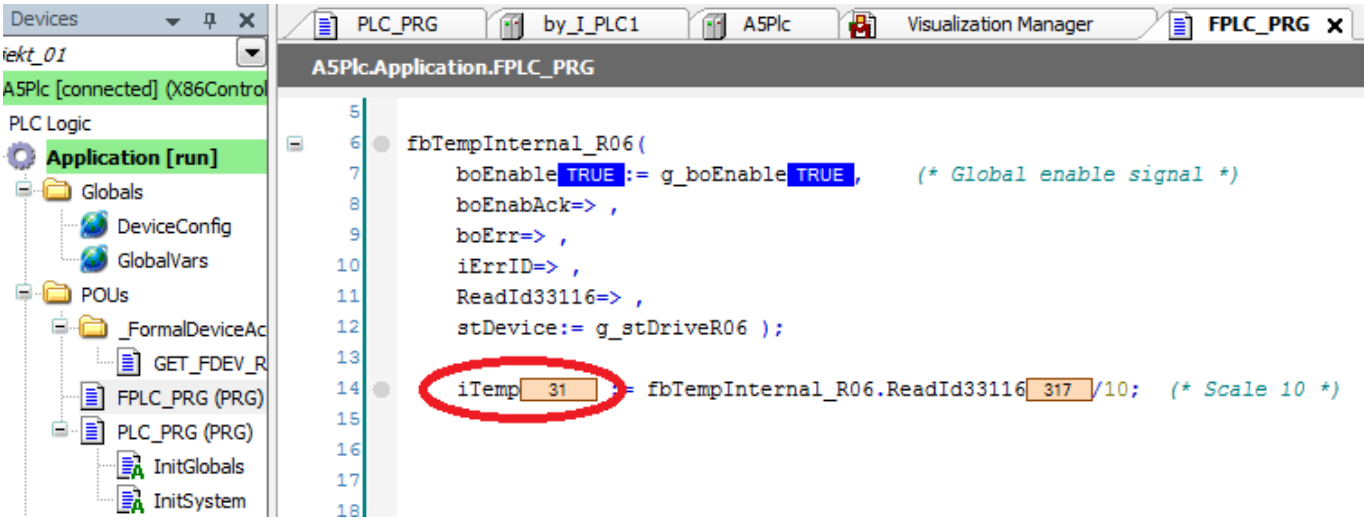
Example:

```
fbTempInteral_R06(
    boEnable:= g_boEnable,   (* Global enable signal *)
    boEnabAck=> ,
    boErr=> ,
    iErrID=> ,
    ReadId33116=> ,
    stDevice:= g_stDriveR06 );

    iTemp := fbTempInternal_R06.ReadId33116/10; (* Scale 10 *)
```

Testing

Create the project: Siehe 'Creating PLC program and message configuration' auf Seite 663.

Start the PLC: Siehe 'Testing PLC project' auf Seite 666.



4.3.1.12.3 EtherCAT Cross Communication between two AMK controllers

The example shows how to exchange data between AMK controllers (writing and reading).

The PLC program of the PLC_Master controller sends an integer variable which is received from the PLC program of the PLC_Slave_02 controller.

4.3.1.12.3.1 Preparation

Required wiring and configuration of the controllers:
See Product Description Controllers A4 / A5 / A6 section: 'Cross communication'

Creating an online project
See Product Description Controllers A4 / A5 / A6 section: 'Cross communication' - Importing and configuring online project with AIPEX PRO'.

Parameterization of controllers

In the cross communication between AMK controls the following parameters must be configured:
Use the function AIPEX PRO 'Direct Mode'

EtherCAT Master

Instance	Parameter	Value	Meaning
5	ID1204 'XML file'		Bus configuration file
	ID1205 'XML file'		Bus configuration file
	ID1206 'XML file'		Bus configuration file
	ID1207 'XML file'		Bus configuration file
	ID34023 'BUS address participant'	0xFF	Default address, can be assigned in the bus system only once
	ID34024 'BUS transmit rate'	100.000	By default, equivalent to 100 Mbps,
	ID34025 'BUS mode'	0x2	Master
	ID34026 'BUS mode attribute'	0x0	-
	ID34140 'AS BUS protocol'	0x41	EtherCAT Master / Slave
	ID34143 'Usage port'	0x2	CC bus (cross communication between controllers)

EtherCAT Slave

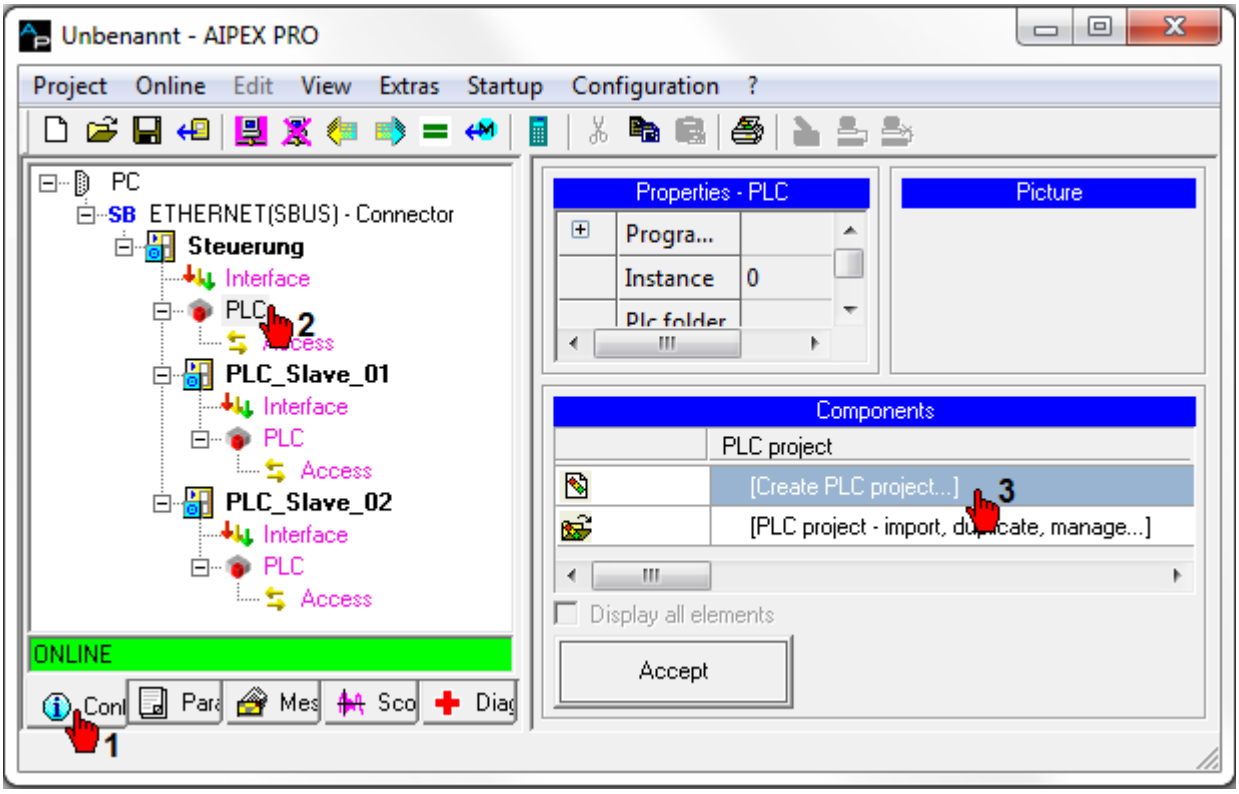
Instance	Parameter	Value	Meaning
2	ID34023 'BUS address participant'	0	Default address, automatic addressing
	ID34024 'BUS transmit rate'	100.000	By default, equivalent to 100 Mbps,
	ID34025 'BUS mode'	0	Slave
	ID34026 'BUS mode attribute'	0	-
	ID34140 'AS BUS protocol'	0x41	EtherCAT slave option A-SEC
	ID34143 'Usage port'	2	CC bus (cross communication between controllers)

The run up of the cross communication must be completed faster than the drive field bus. If needed you can use at the drive field bus master a run up delay time via ID34026 'BUS mode attribute'.

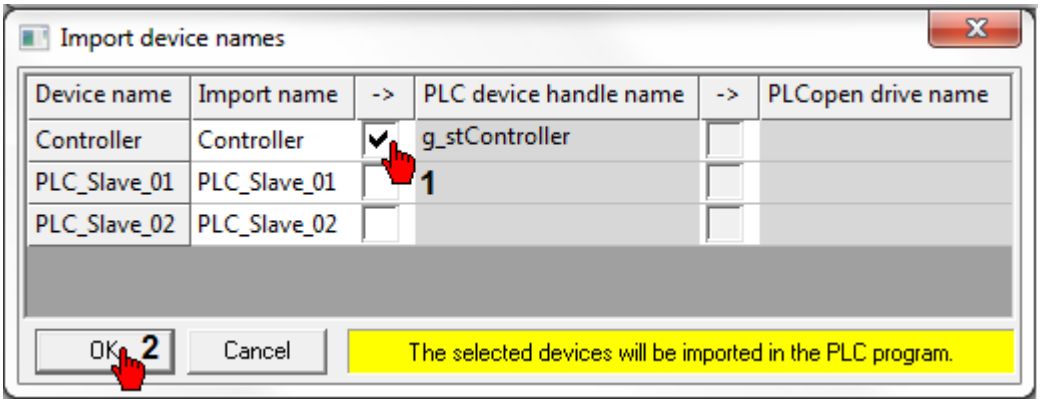
After parameterization, both (all) controllers must be restarted (24 VDC OFF / ON)
After restarting the cross communication is active. The ACC master is connected via the ACC (CC) connection with the ACC slave.
Read a configuration and save the project: AIPEX PRO 'Online' menu → 'Login'
Save the project: AIPEX PRO menu 'Project' → 'Save as...'

4.3.1.12.3.2 PLC Program Master Controller

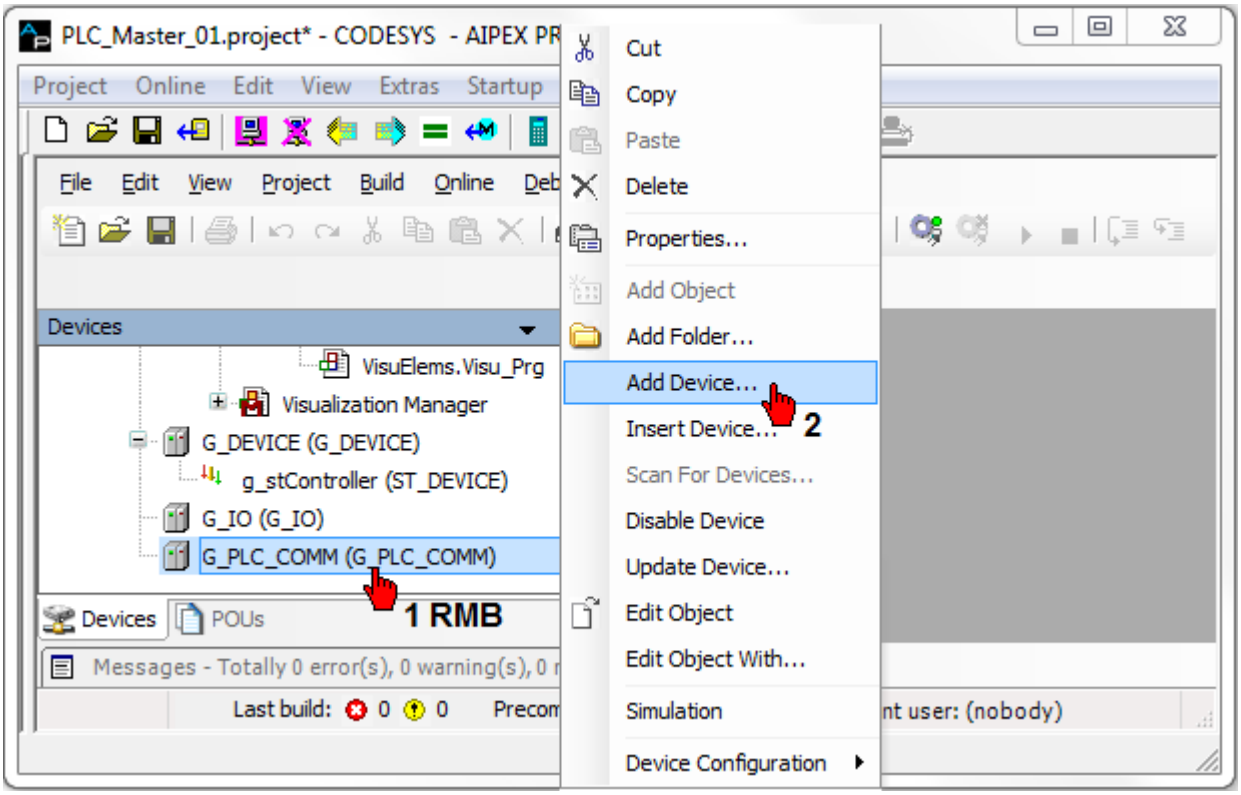
Create and save the PLC program for the PLC_Master controller.



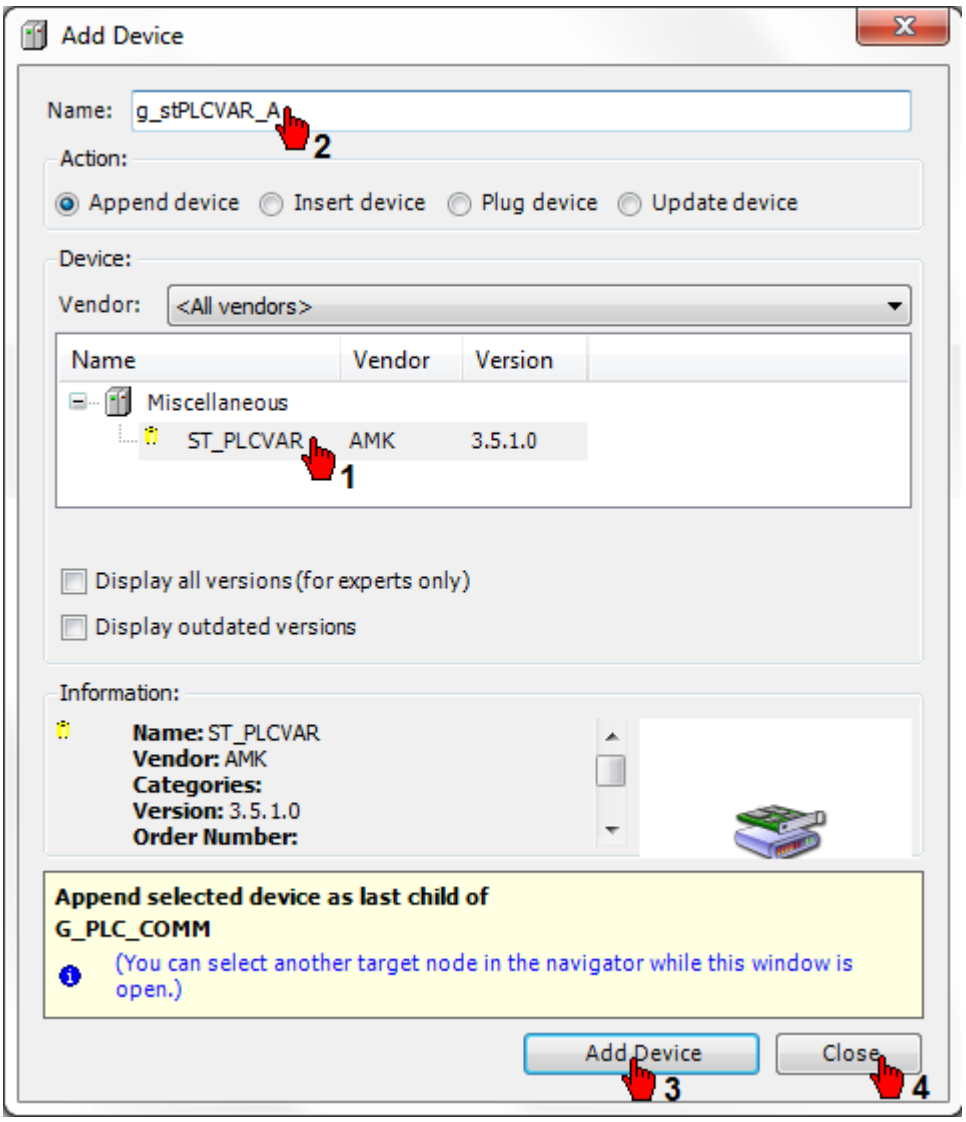
Transfer the 'PLC device handle name' (symbolic device name) of the PLC_Master controller to the PLC program.



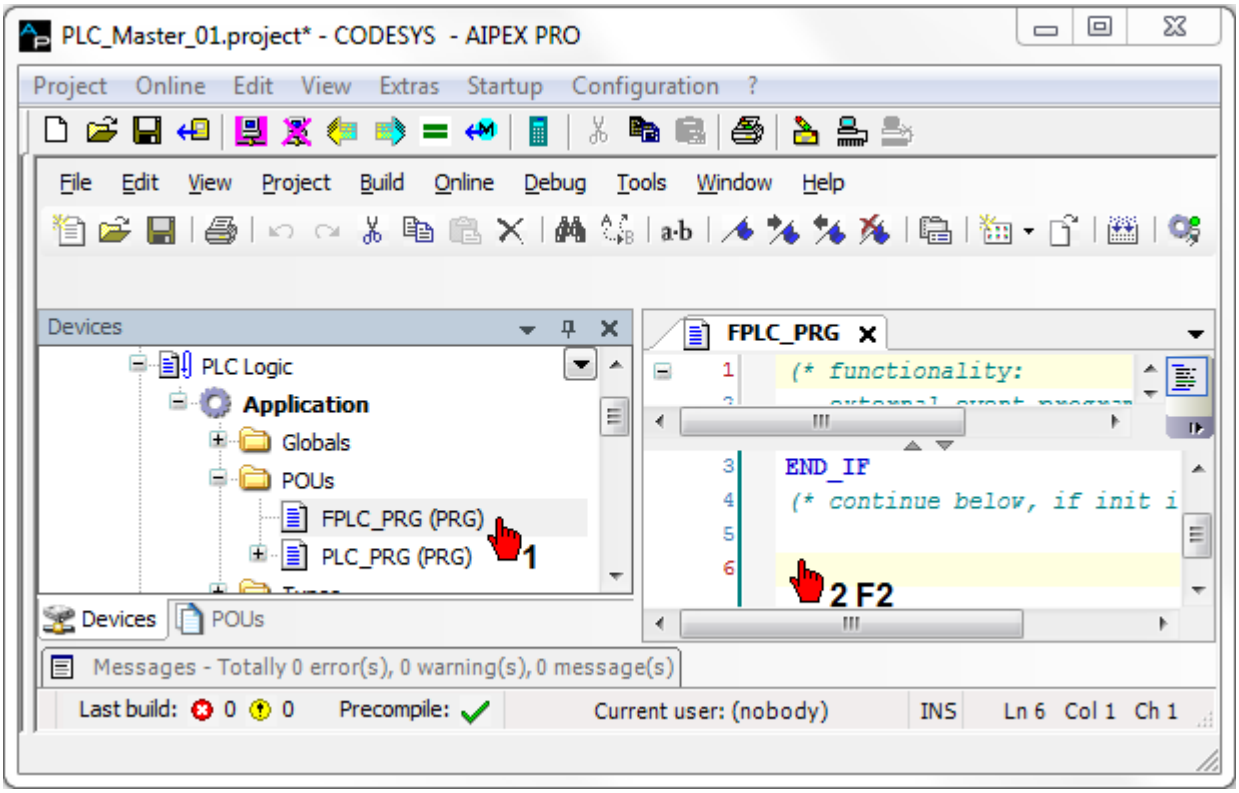
Change to the 'Controller configuration'.
Add a communication variable of type ST_PLCVAR to the controller configuration G_PLC_COMM (g_stPLCVAR_A).



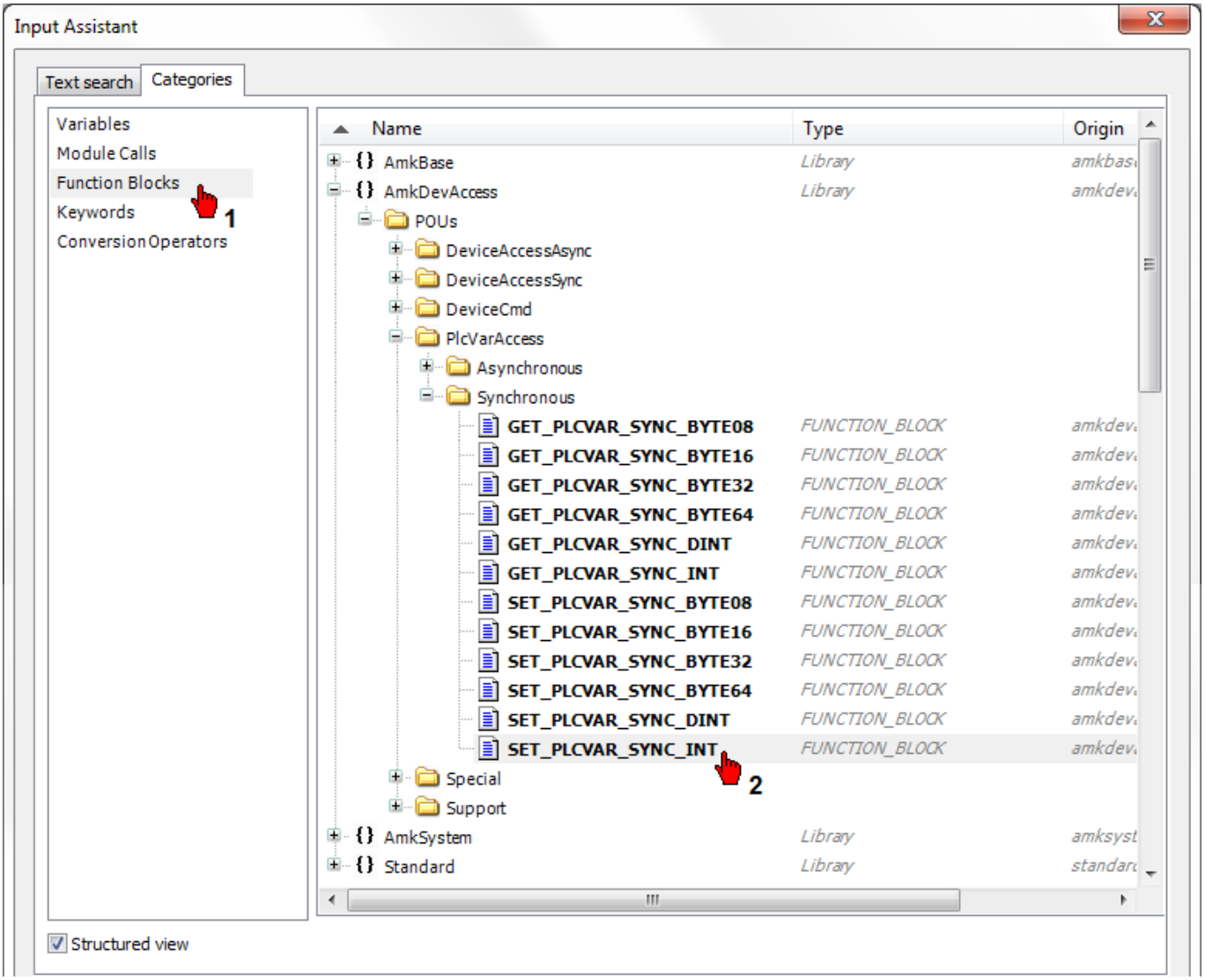
Create a communication variable (in the example g_stPLCVAR_A).



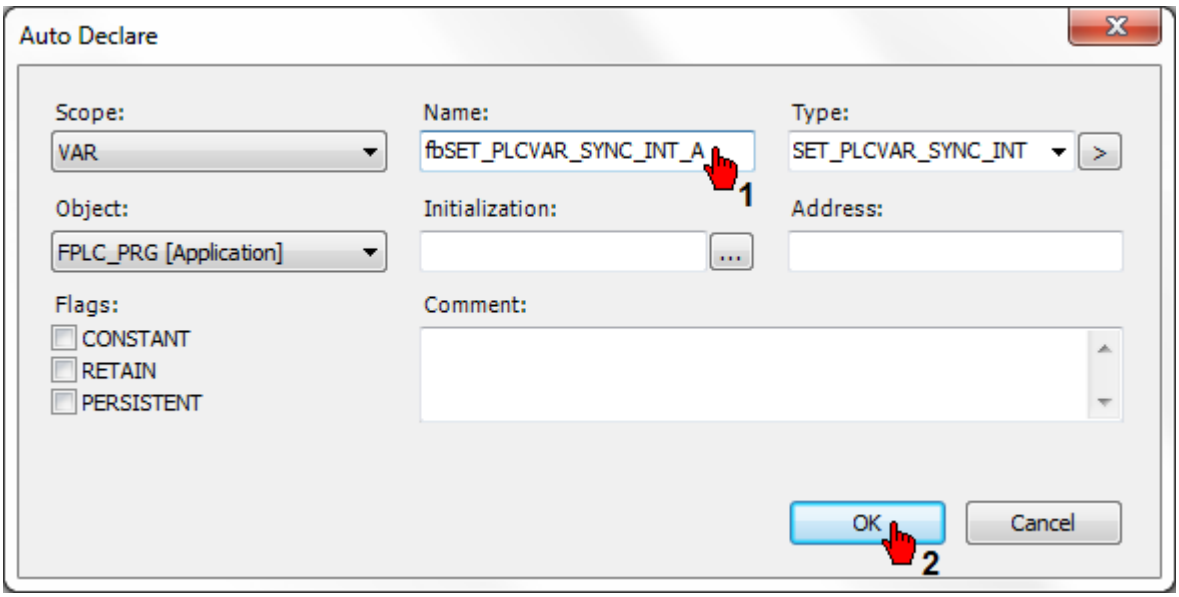
Add the PLC writing module (in the example, an integer variable in the synchronous level is transferred).
Instantiate in the real-time task FPLC_PRG a function block type SET_PLCVAR_SYNC_INT.



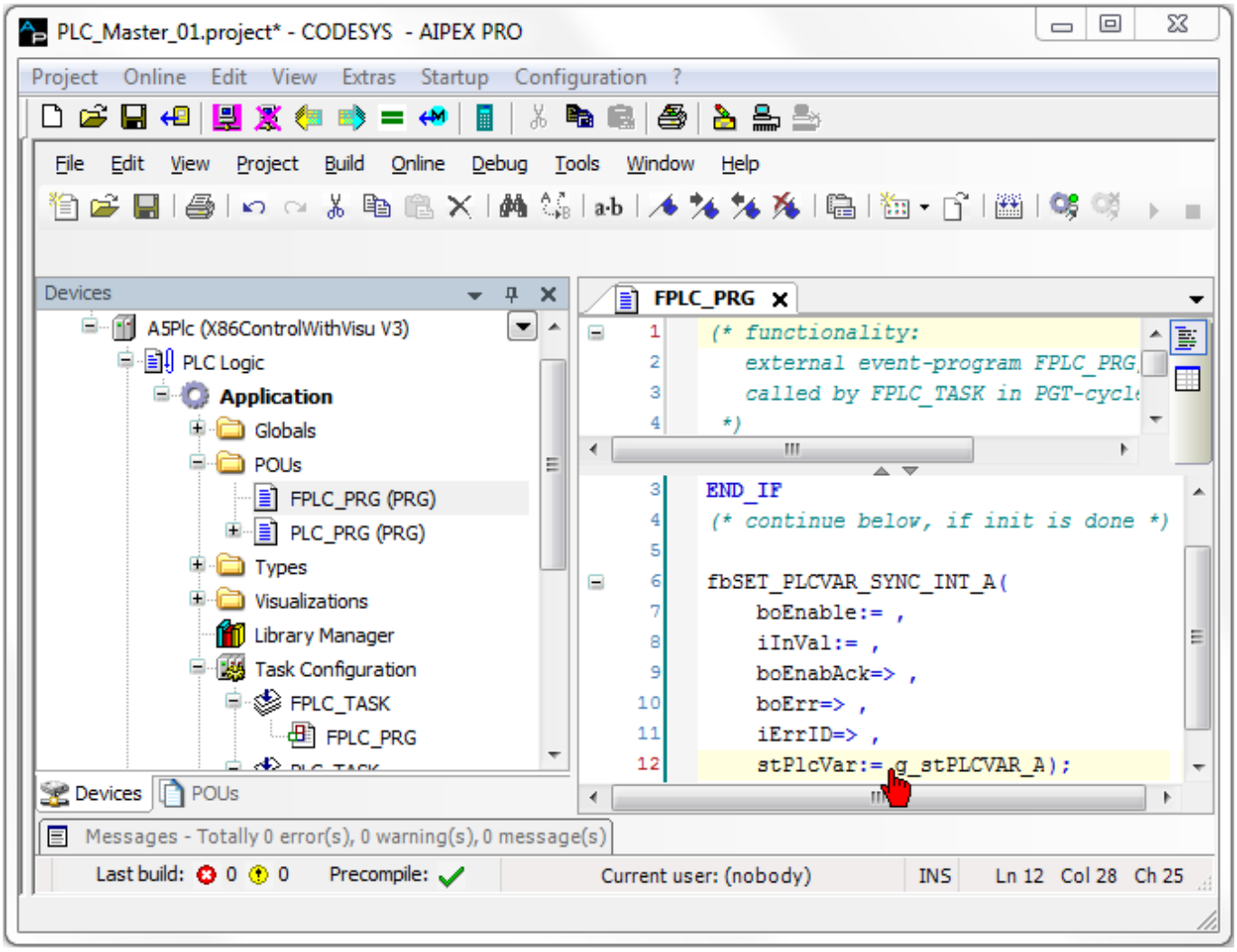
PLC function blocks for communication exchange between the controllers are provided with the 'AmkDevAccess' AMK library.
See asynchronous data exchange: Folder: 'POUs' → 'PlcVarAccess' → 'Asynchronous'
See synchronous data exchange: Folder: 'POUs' → 'PlcVarAccess' → 'Synchronous'
Function block SET_: write
Function block GET_: read



You can instantiate the function block by assigning its own name to it.



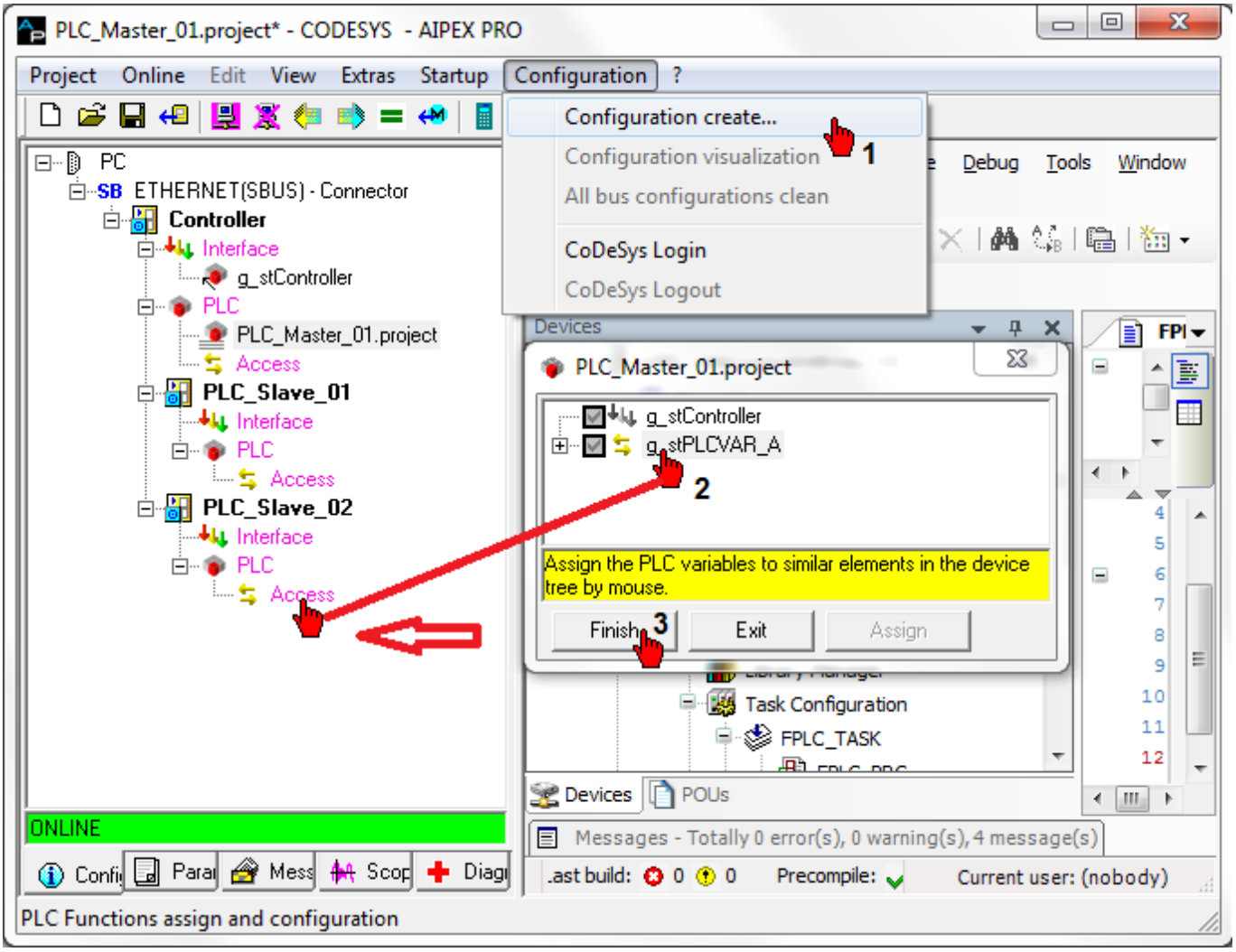
Link the function block variable 'stPlcVar' with the communication variable (g_stPLCVAR_A).



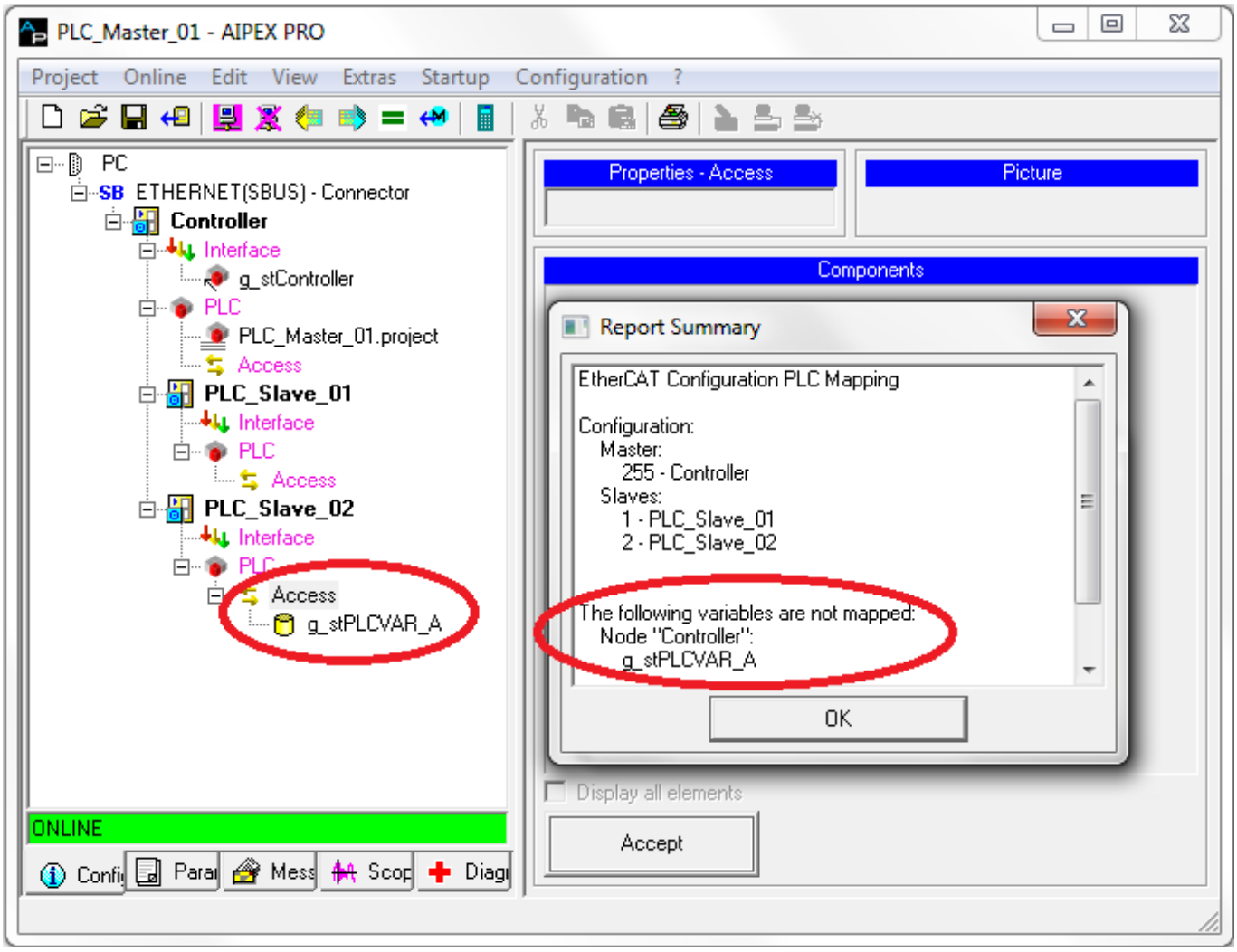
Create message configuration for the EtherCAT master

Click the AIPEX PRO menu 'Configuration ...' → 'Configuration create'

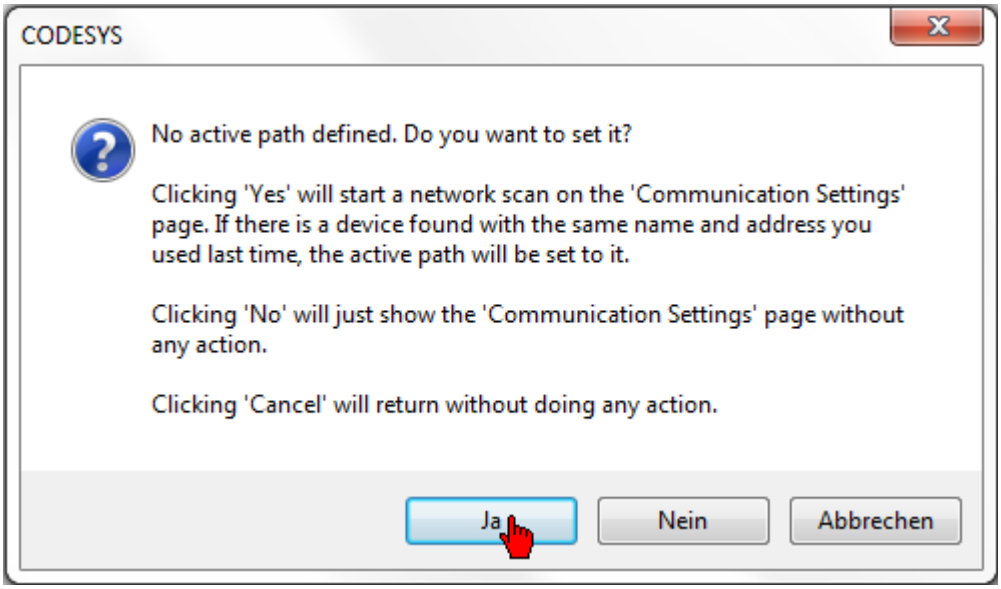
Assign the communication variable via 'drag and drop' to the EtherCAT slave (interface 'access')



The final variable mapping takes place after the slave PLC is programmed and the 'Configuration create' function is called.



Communication settings

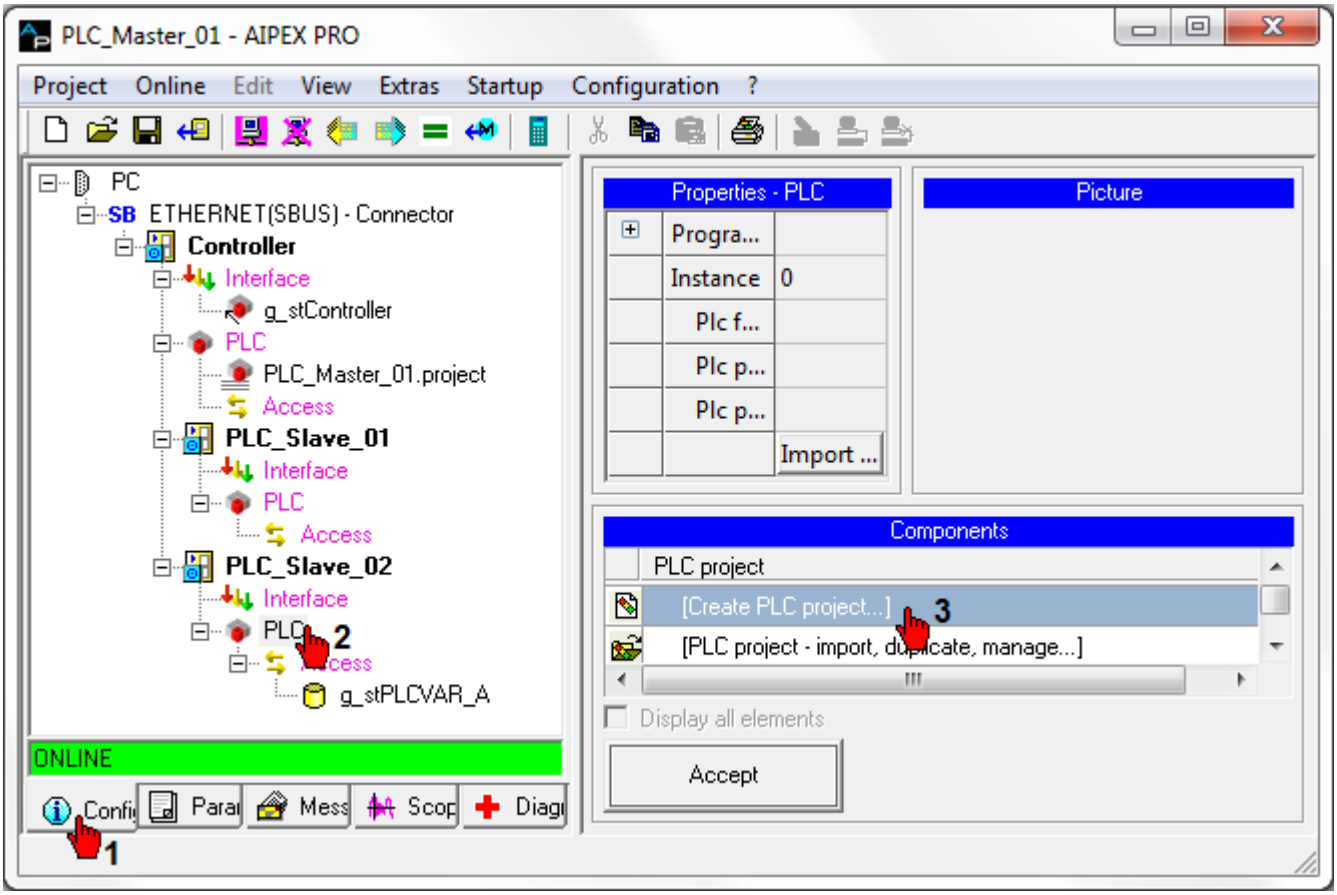


Siehe 'Creating PLC program and message configuration' auf Seite 663.

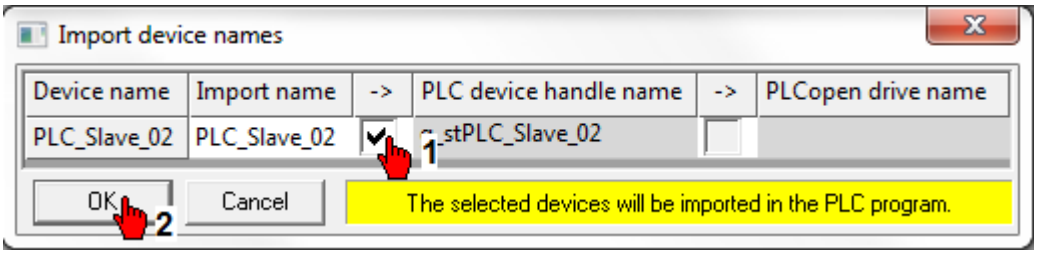
Communication settings section

4.3.1.12.3.3 PLC Program Slave Controller

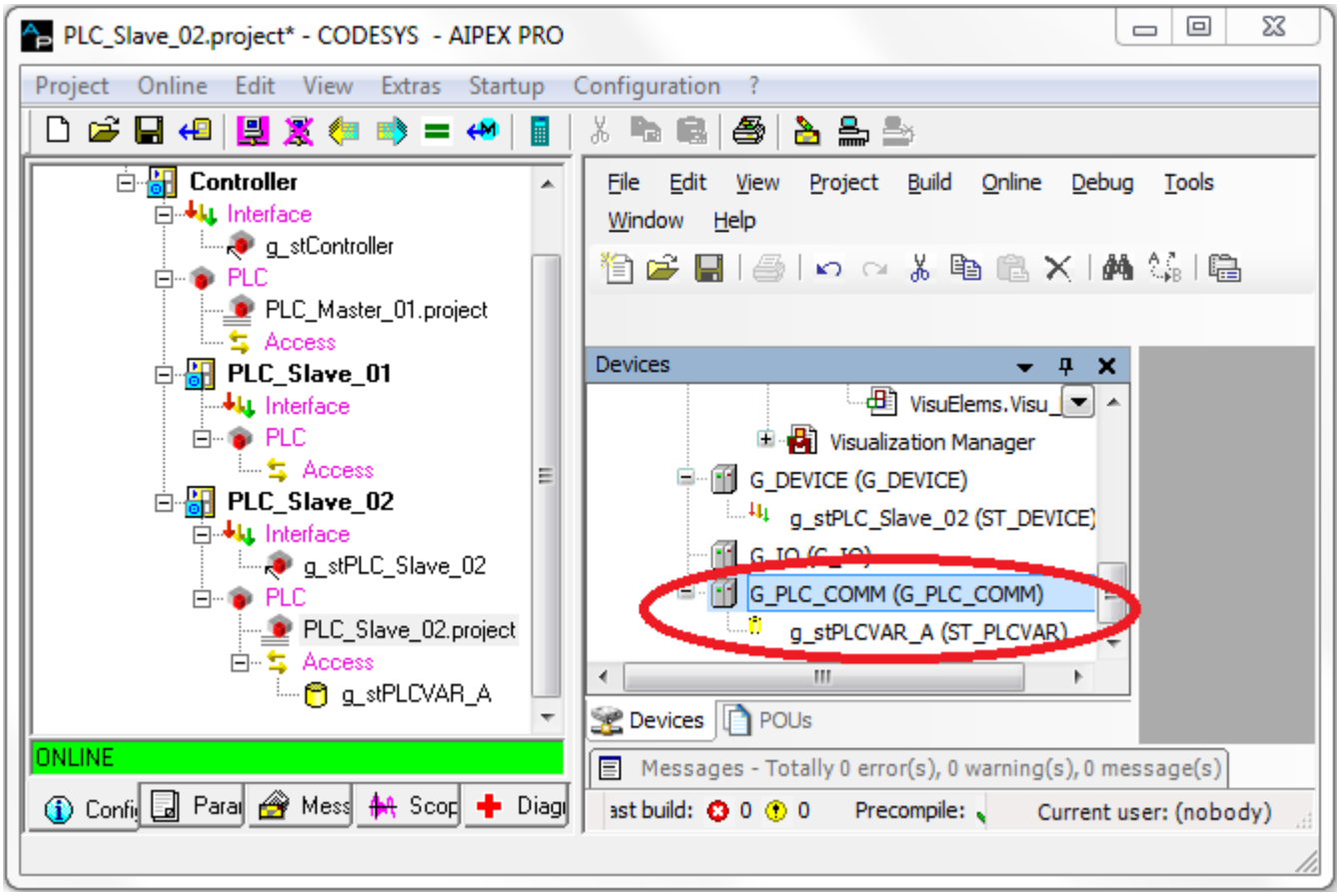
Create and save the PLC program for the PLC_Slave_02 controller.



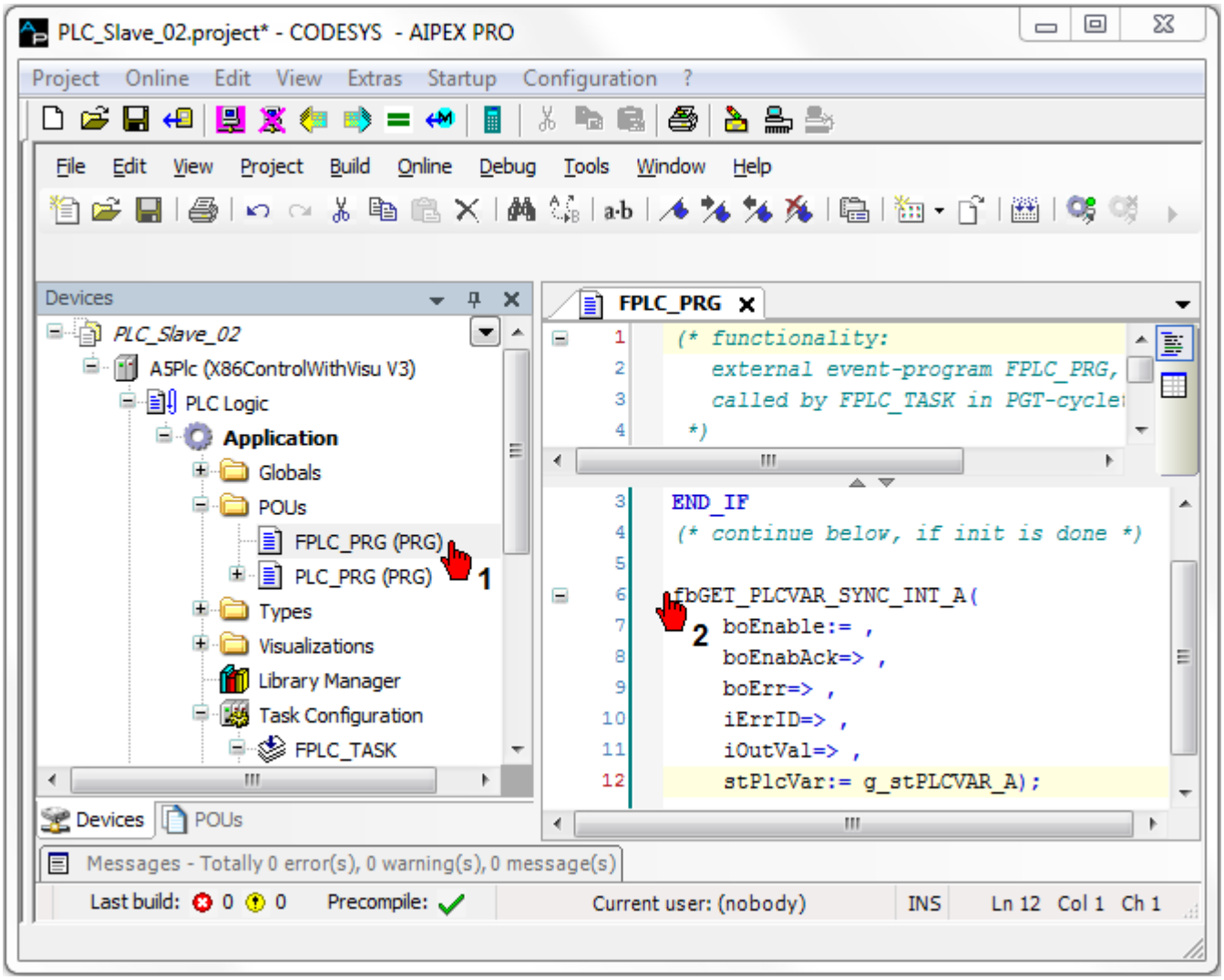
Transfer the 'PLC device handle name' (symbolic device name) of the PLC_Slave_02 controller to the PLC program.



The assigned PLC communication variable from the PLC_Master controller (in the example g_stPLCVAR_A) is imported automatically to the 'G_PLC_COMM' controller configuration.

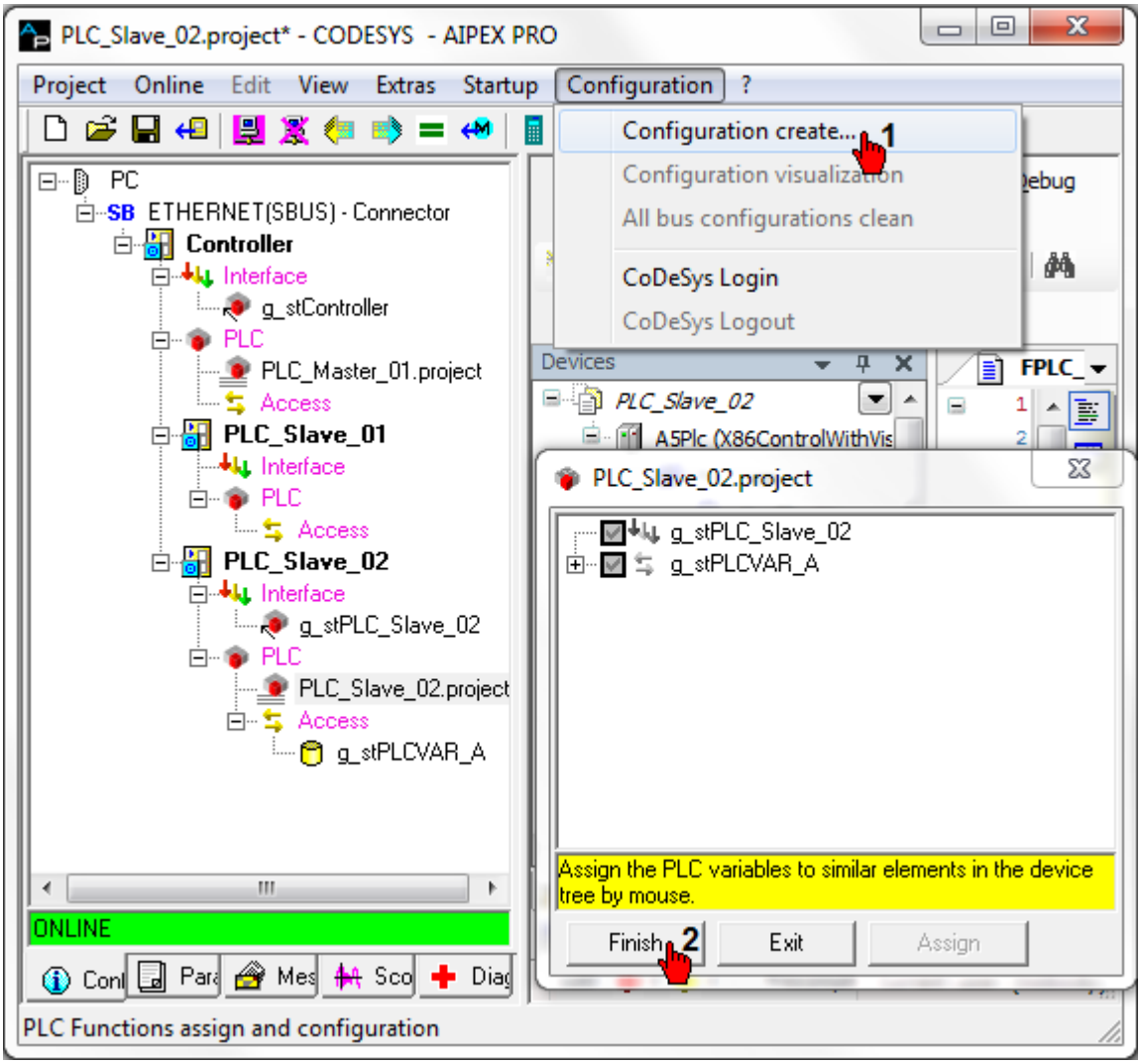


Instantiate in the real-time task FPLC_PRG a function block type GET_PLCVAR_SYNC_INT.
Link the function block variable 'stPlcVar' with the communication variable (g_stPLCVAR) .

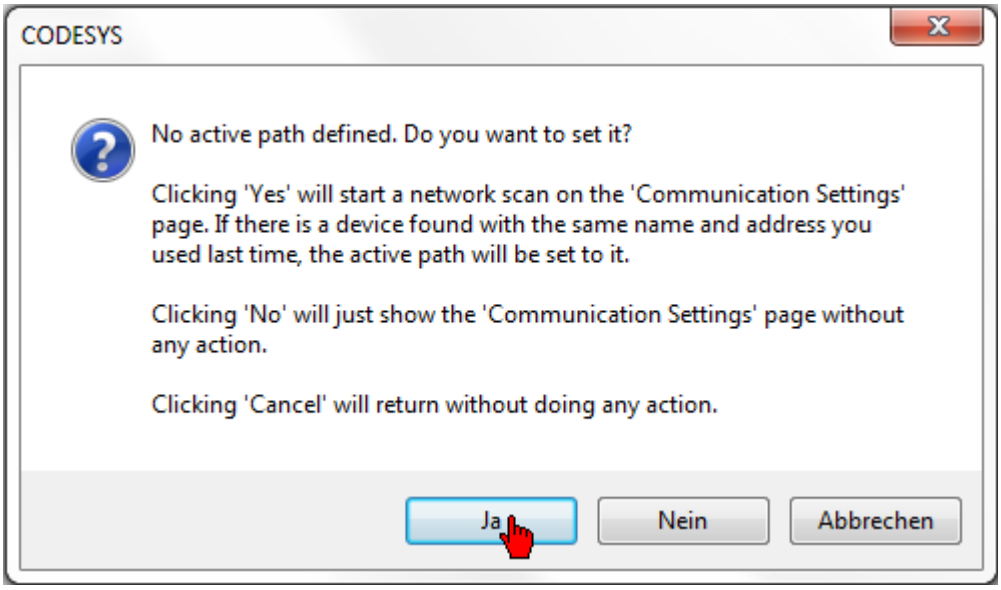


Create message configuration for the EtherCAT slave

Click the AIPEX PRO menu 'Configuration ...' → 'Configuration create'



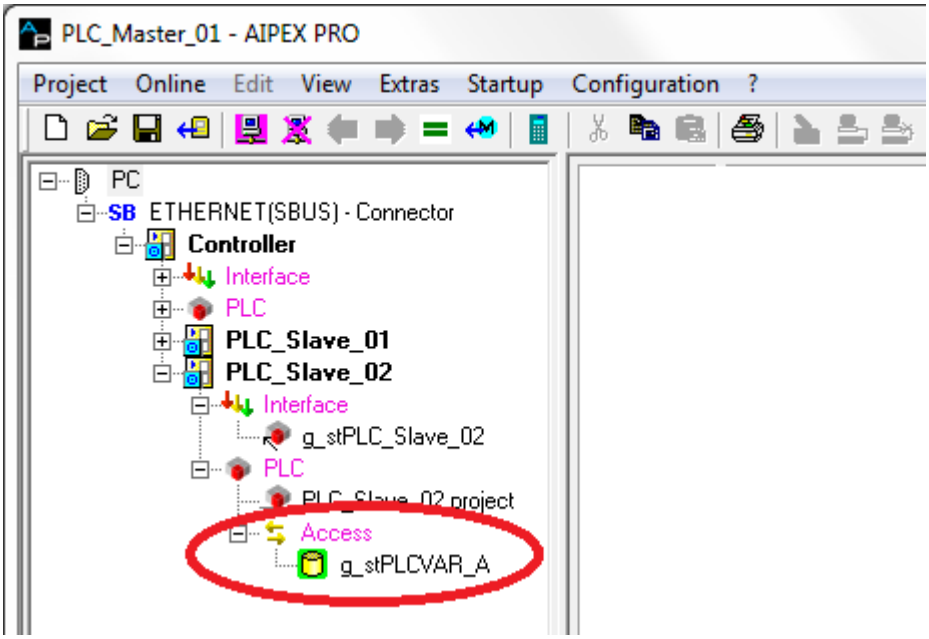
Communication settings



Siehe 'Creating PLC program and message configuration' auf Seite 663.

Communication settings section

The icon of the mapped PLC communication variable is highlighted in 'green'.



The configuration is finished! The program can be tested.

- A new PLC communication variable must be created for each SET_ transmission block.
- One PLC communication variable can be linked to any number of PLC slave controllers.
- Follow the same procedure if you want to exchange data between two slave PLCs.

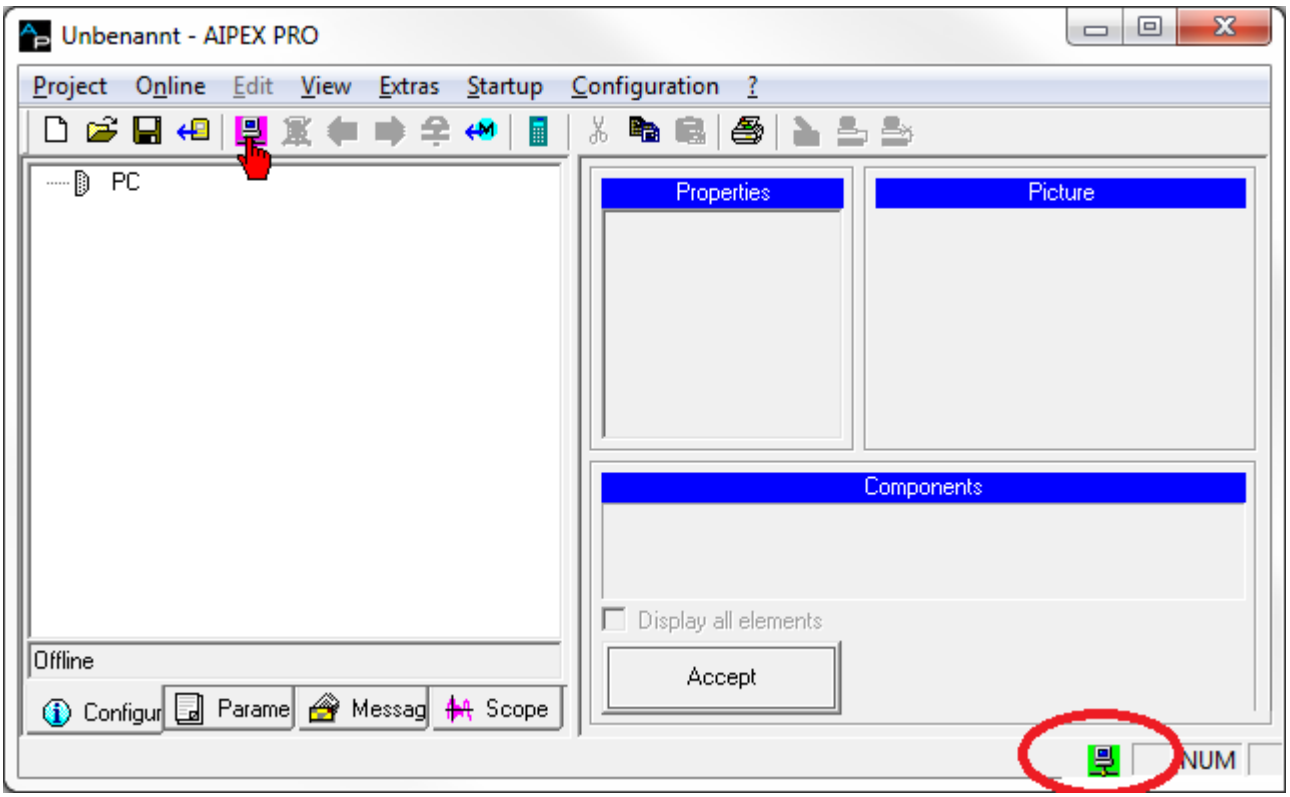
4.3.1.12.4 Cam

Prerequisite of Cam:
A5 devices with additional option PCO (PLCopen) or A5D device with additional option PNC (PLCopen CNC).

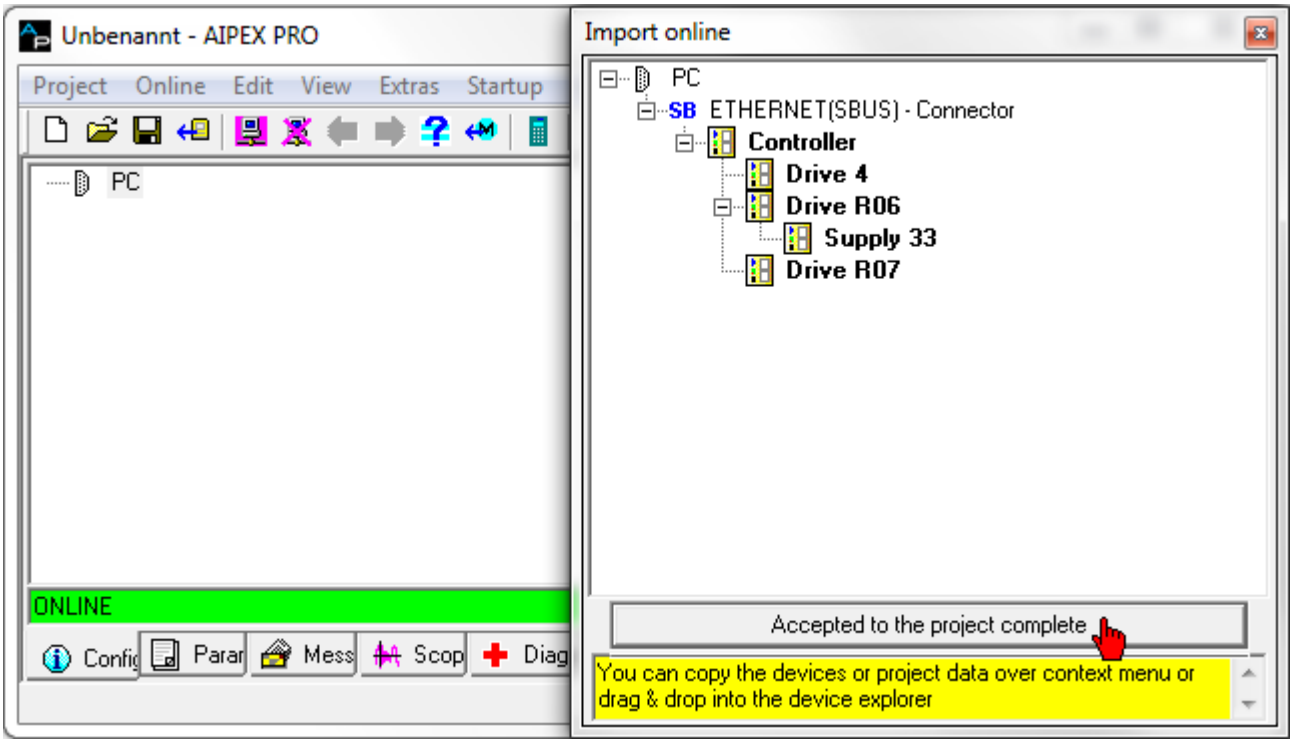
4.3.1.12.5 Importing a PLC project

This section describes how to import a PLC project from another AIPEX PRO project and to transfer it to an existing application.

Press **'Logon'** to create a connection with the connected devices.

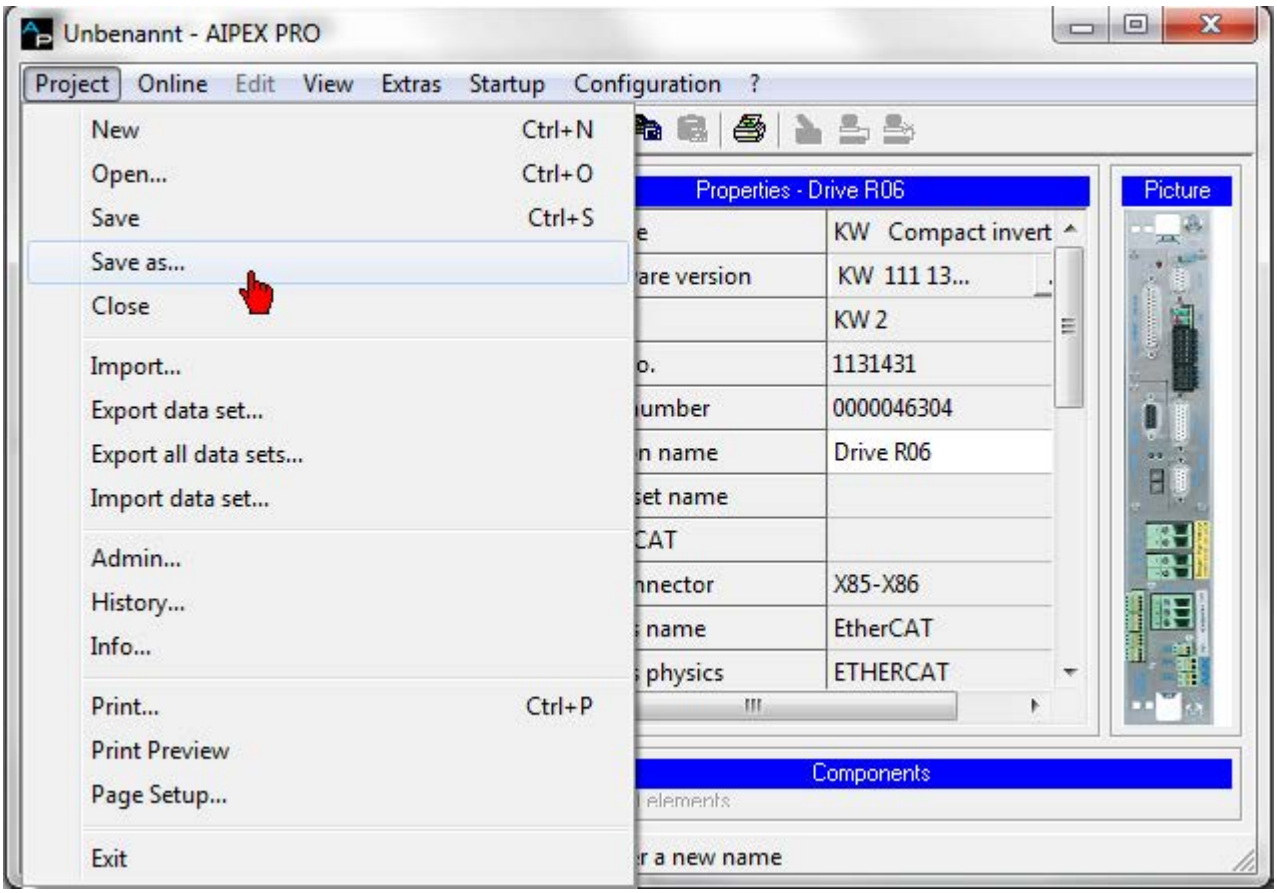


In the 'Import online' window, check whether all physical devices have been detected.
Press **'Accepted to the project complete'** to import the device data.

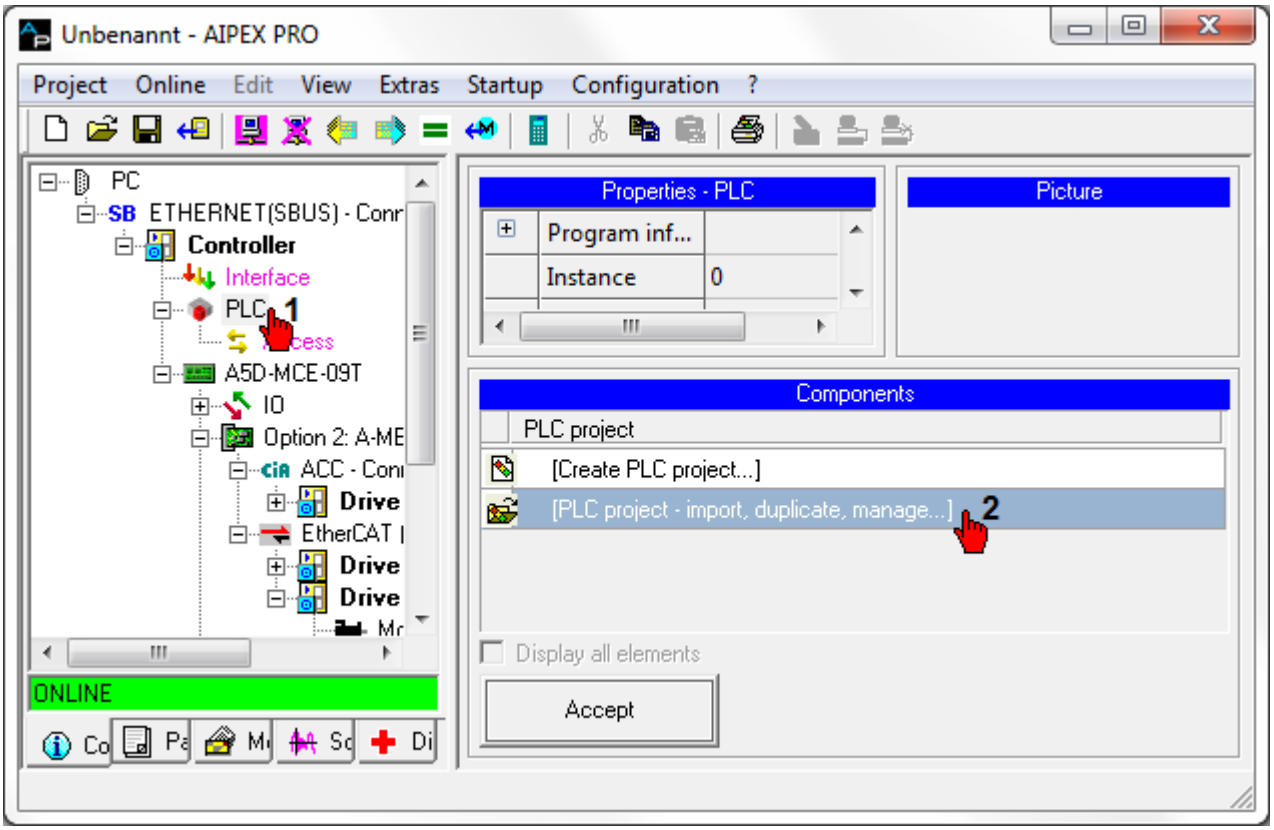


Check the wiring and the system statuses if devices are not shown.

Save the imported project under **"Save as..."**.

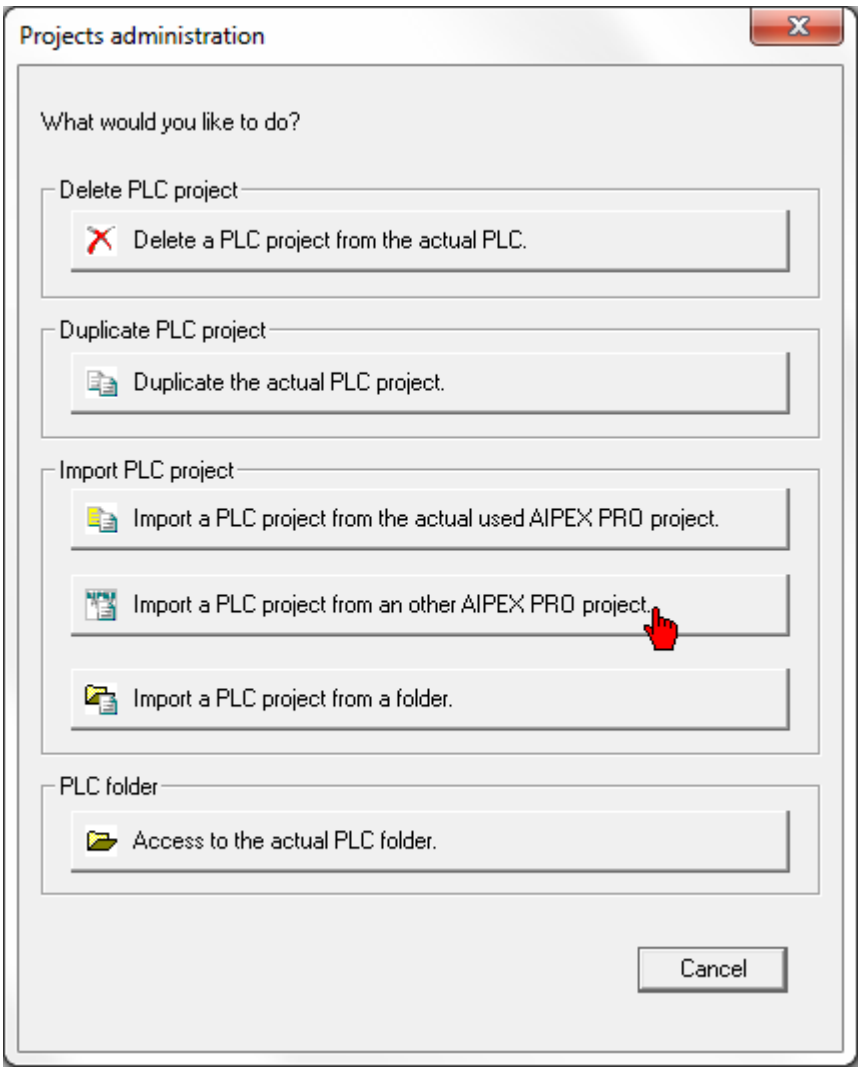


Click on the **'PLC'** icon in the device tree.
Click on **'PLC project - import, duplicate, manage ...'**

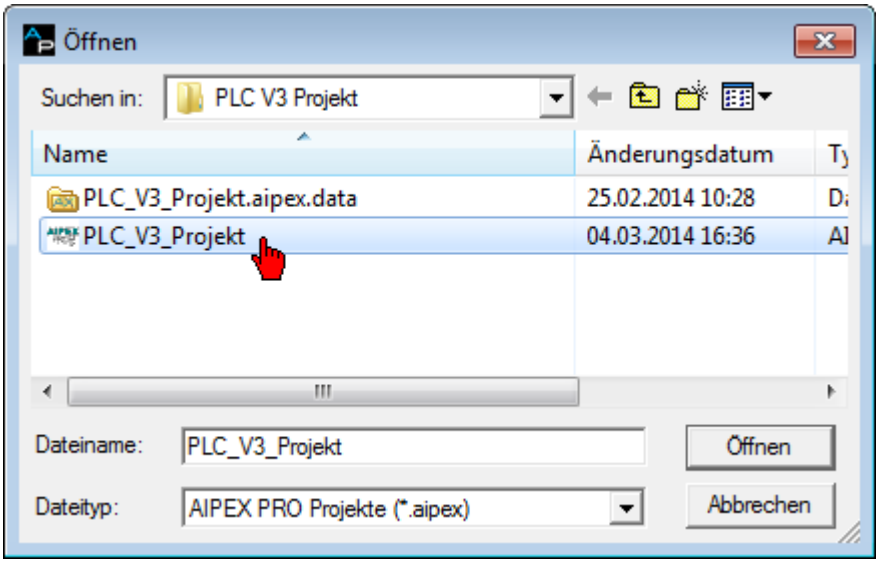


Icon '**PLC**' missing:
Right-click in the device tree. You can customize the view under 'Select view'.
Siehe 'Program overview - Display filter' auf Seite 39.

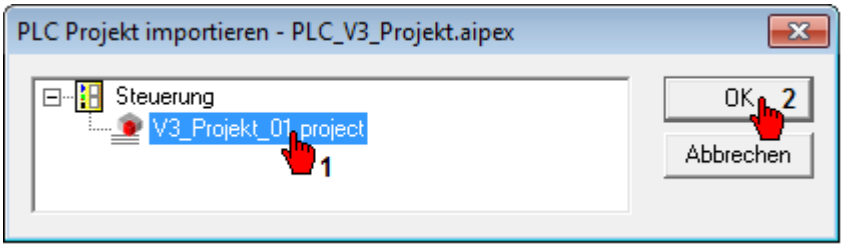
Click on **'Import a PLC project from another AIPEX PRO project'**.



Open the AIPEX PRO project with the integrated PLC project.



Select the PLC project. Confirm by pressing the **'OK'** button to start the CODESYS programming system.



Before the PLC program is transferred to the controller, it must first be translated without error and the message configuration must be generated.

Start this process under '**Configuration**' → '**Configuration create**'.

Siehe 'Creating PLC program and message configuration' auf Seite 663.

4.3.1.13 General information

4.3.1.13.1 Version control CODESYS V3

The different CODESYS V3 versions which are integrated in AIPEX PRO are not really compatible between each other. Therefore open an existing CODESYS V3 projects always with the CODESYS V3 version with the existing project was created.

If inside the existing project were used AFL Standard blocks or AFL Application blocks, the AFL AMK Function Library must be additionally installed. The installed AFL AMK Function Library must be compatible to the CODESYS V3 version.

The latest CODESYS V3 version is always recommended for new CODESYS V3 projects.

Only by AMK released CODESYS V3 versions may be used. The released CODESYS V3 versions are included in the AIPEX PRO setups.

Version overview

AIPEX PRO + integrated CODESYS V3 version + compatible AFL Version

AIPEX PRO version	CODESYS V3 version	CODESYS profile	compatible AFL version
3.04	3.5.10.4	CODESYS V3.5 **SP10** Patch 4 AIPEX PRO	AFL V4 Version 3.5.5.0 2015/41 (part-no. 206004)
	3.5.5.5	CODESYS V3.5 **SP5** Patch 5 AIPEX PRO	
	3.5.3.6	CODESYS V3.5 **SP3** Patch 6 AIPEX PRO	AFL V4 Version 3.5.3.0 2014/06 (part-no. 204786)
3.03	3.5.5.5	CODESYS V3.5 **SP5** Patch 5 AIPEX PRO	AFL V4 version 3.5.5.0 2015/41 (part-no. 206004)
3.02	3.5.3.6	CODESYS V3.5 **SP3** Patch 6 AIPEX PRO	AFL V4 version 3.5.3.0 2014/06 (part-no. 204786)
3.01			
3.00			

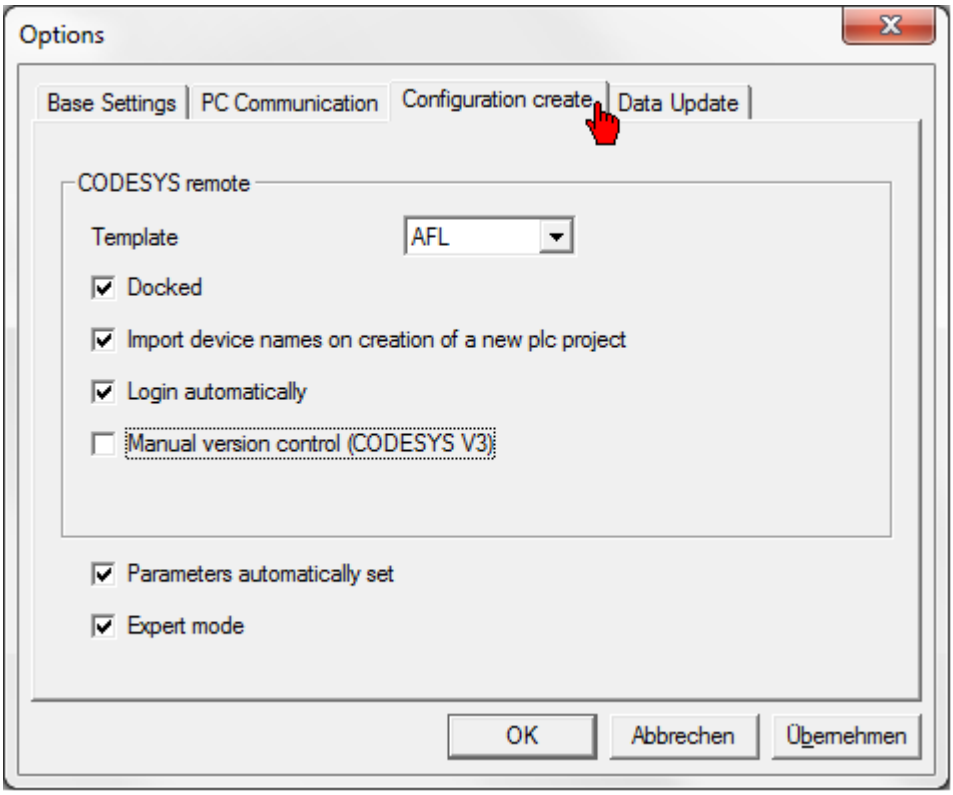
Steuerung	Firmware Version	CODESYS V3 Profil
iSA / A4	≥ 4.22	CODESYS V3.5 SP10 Patch 4 AIPEX PRO
iSA / A4	≤ 4.21	CODESYS V3.5 SP5 Patch 5 AIPEX PRO
A5 / A6	alle	CODESYS V3.5 SP10 Patch 4 AIPEX PRO (recommended) (with restrictions[1)])
		CODESYS V3.5 SP5 Patch 5 AIPEX PRO (with restrictions[1)])
		CODESYS V3.5 SP3 Patch 6 AIPEX PRO

1) New features that affect the runtime system are not supported.

Corresponding CODESYS version must also be installed when installing AIPEX PRO.

A AIPEX PRO setup includes only one CODESYS V3 version (see table: Version overview). Older CODESYS V3 versions must be installed separately with the corresponding AIPEX PRO setup.
The AFL AMK Function Library is not part of the PRO AIPEX setups and must be installed separately.

From AIPEX PRO version 3.03 a manual or automatic CODESYS V3 version control is offered.
AIPEX PRO menu 'Extras' → 'Options' → 'Configuration create' → 'Manuel version control (CODESYS V3)'



From AIPEX PRO version 3.03 a manual or automatic CODESYS V3 version control is offered.
Manual version control (CODESYS V3) 'disabled' (Default setting)

- Create a new CODESYS V3 project → A new CODESYS V3 project will always be created with the latest installed CODESYS V3 version.
- Open a existing CODESYS V3 projec → AIPEX PRO automatically detects with which CODESYS V3 version an existing CODESYS V3 project was created. AIPEX PRO automatically opens the matching CODESYS V3 version. Is the matching CODESYS V3 version not installed, the project will open with the latest installed CODESYS V3 version.

Manual version control (CODESYS V3) 'enabled'
Before opening CODESYS V3, AIPEX PRO opens a window to select manually the CODESYS V3 version.

- Create a new CODESYS V3 project → Users must select an installed CODESYS V3 version, with the new project is to be created.
- Open a existing CODESYS V3 projec → Users must select an installed CODESYS V3 version, with the existing project is to be opened.

Example with different versions:

You need the functionality of AIPEX PRO Version 3.03. But to program the PLC you need the previous CODESYS V3 Version 3.5.3.6.

Procedure:

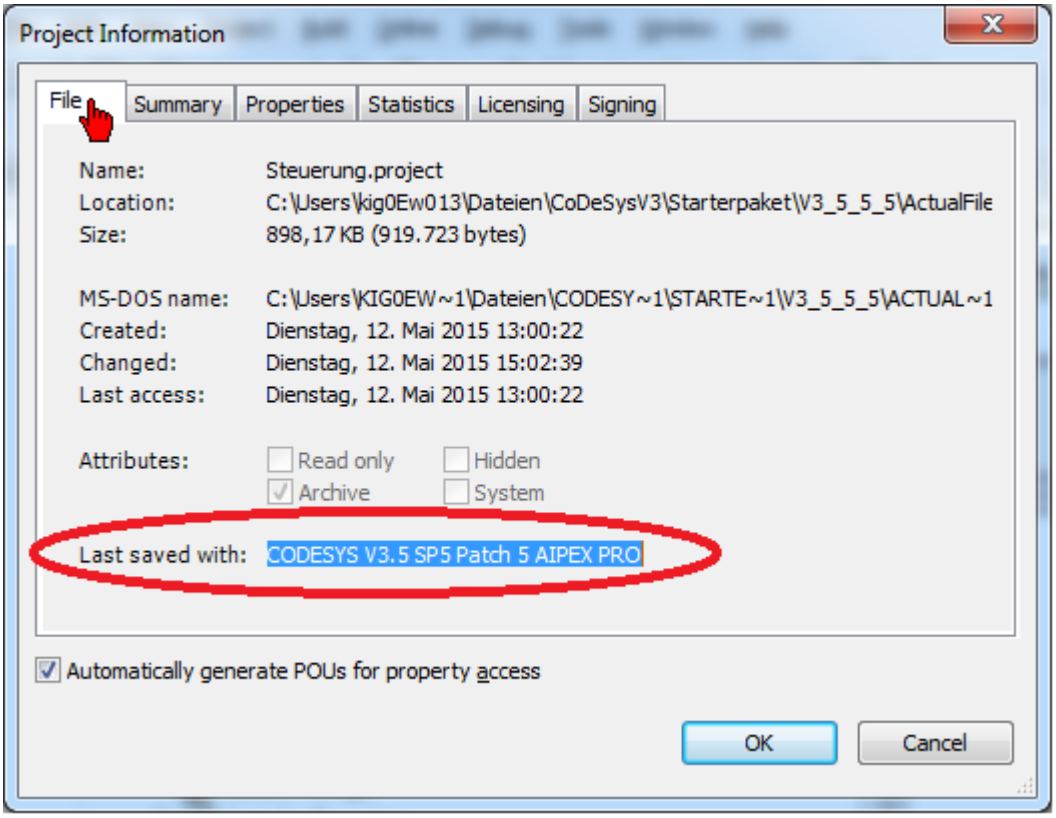
First install the AIPEX PRO Setup 3.02 (contains the CODESYS V3 version 3.5.3.6). Then, install the AIPEX PRO 3.03 setup.

Activate the AIPEX PRO menu 'Extras' → 'Configuration create' → 'Manuel version control (CODESYS V3)'

When opening the CODESYS V3 project you have to select the profile 'CODESYS V3.5 **SP3** Patch 6 AIPEX PRO' (CODESYS V3 Version 3.5.3.6).

Determine used CODESYS V3 version in the actual project

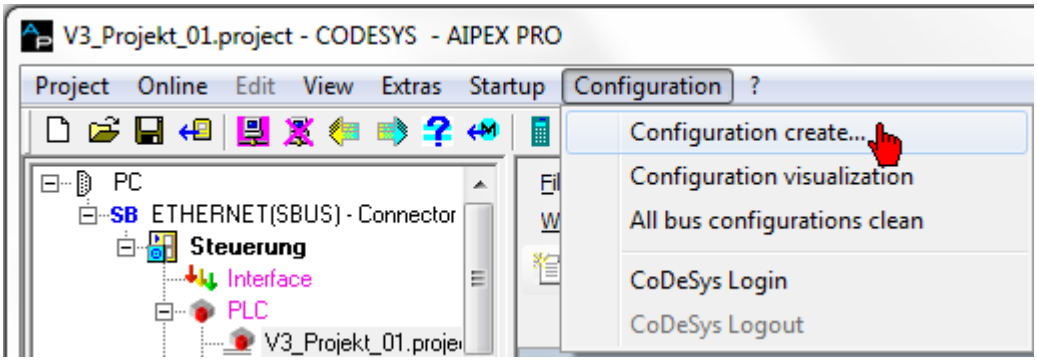
CODESYS V3 menue 'Project' → 'Project information'



4.3.1.13.2 Automatic bus configuration

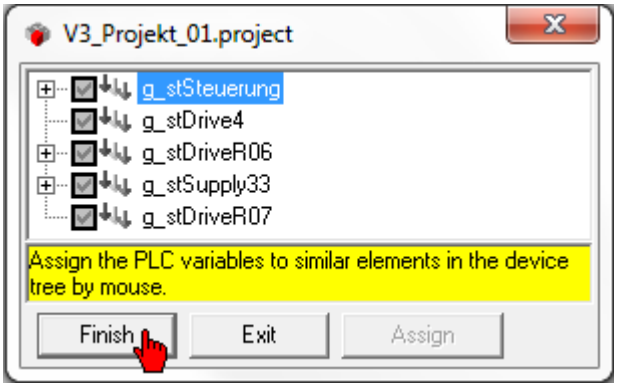
The allocation of the physical devices (AIPEX PRO) to the devices used in the PLC program, IO variables and PLC to PLC communication variables (described in the documentation as symbolic device names) takes place during the bus configuration procedure. The symbolic device names are created in the controller configuration. Siehe 'Control Configuration' auf Seite 732.

The automatic bus configuration is started with the **'Configuration create...'** menu item.



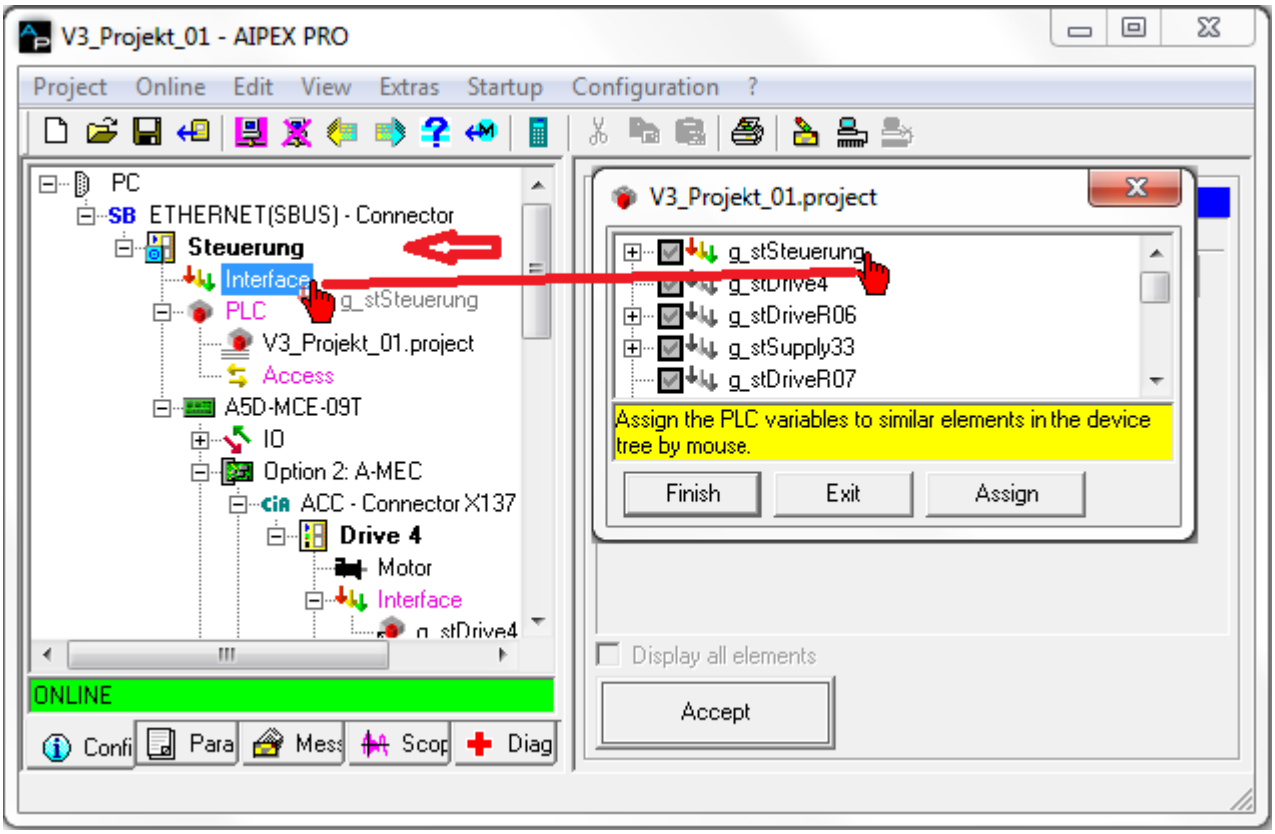
Here, the variables are analyzed, the linked symbolic device names determined and displayed in the '*.project' allocation window.

g_stSteuerung [Arrow grey] The symbolic device name is assigned to a device in the AIPEX PRO device tree



g_stSteuerung [Arrow colored] The symbolic device name is not assigned.

The symbolic device names can be assigned to the corresponding items of the device tree (interface or I/O sub-element) by mouse (drag & drop).



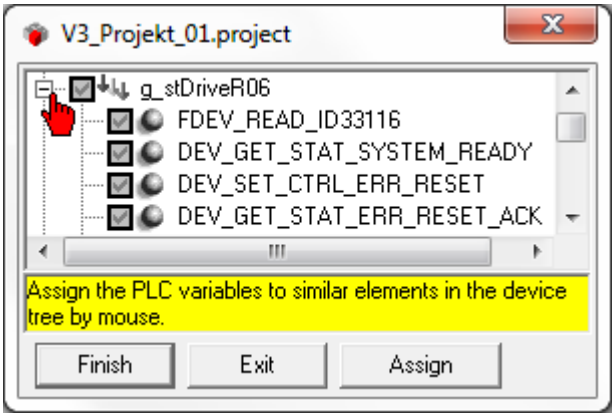
Additional information:

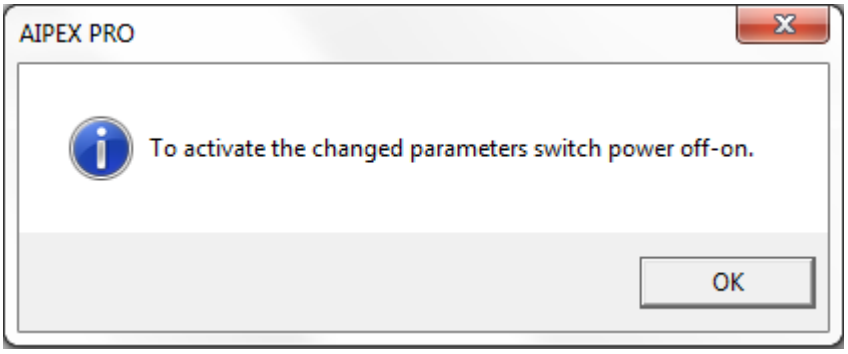
Click on the + icon. You will see which function blocks are assigned to a variable.

If activated, the configurator changes the relevant parameters automatically.

Activating/deactivating or function block specific changes:

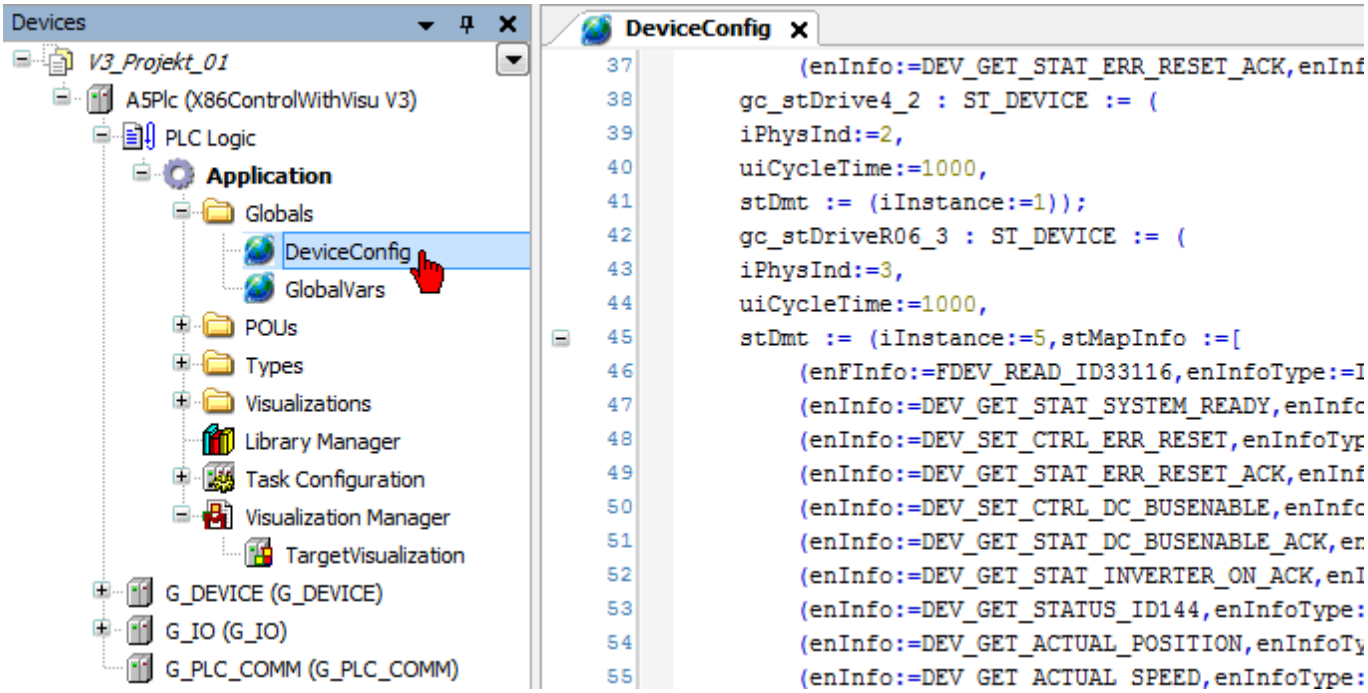
Siehe 'Project Settings - Configuration create (project specific)' auf Seite 108.





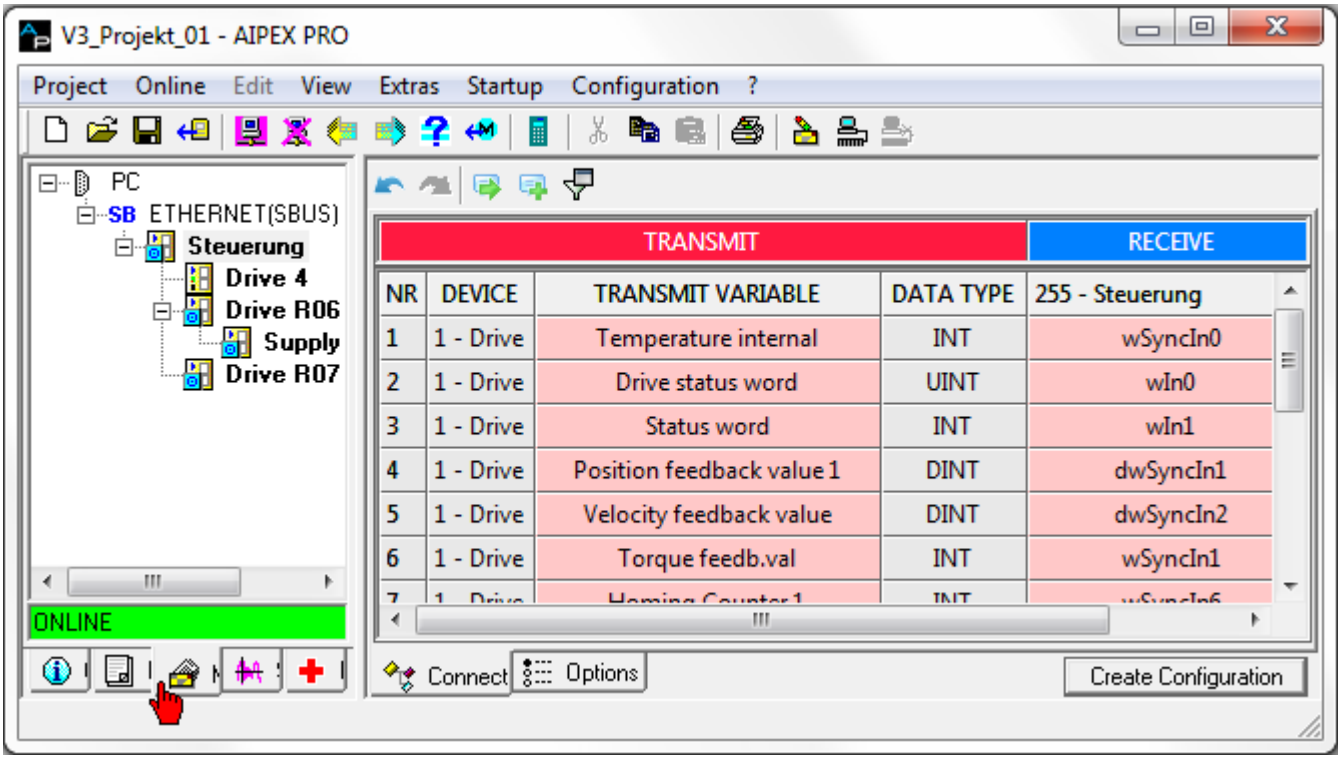
CODESYS V3

You can find the created bus configuration under 'Application' → 'Globals' → 'DeviceConfig'

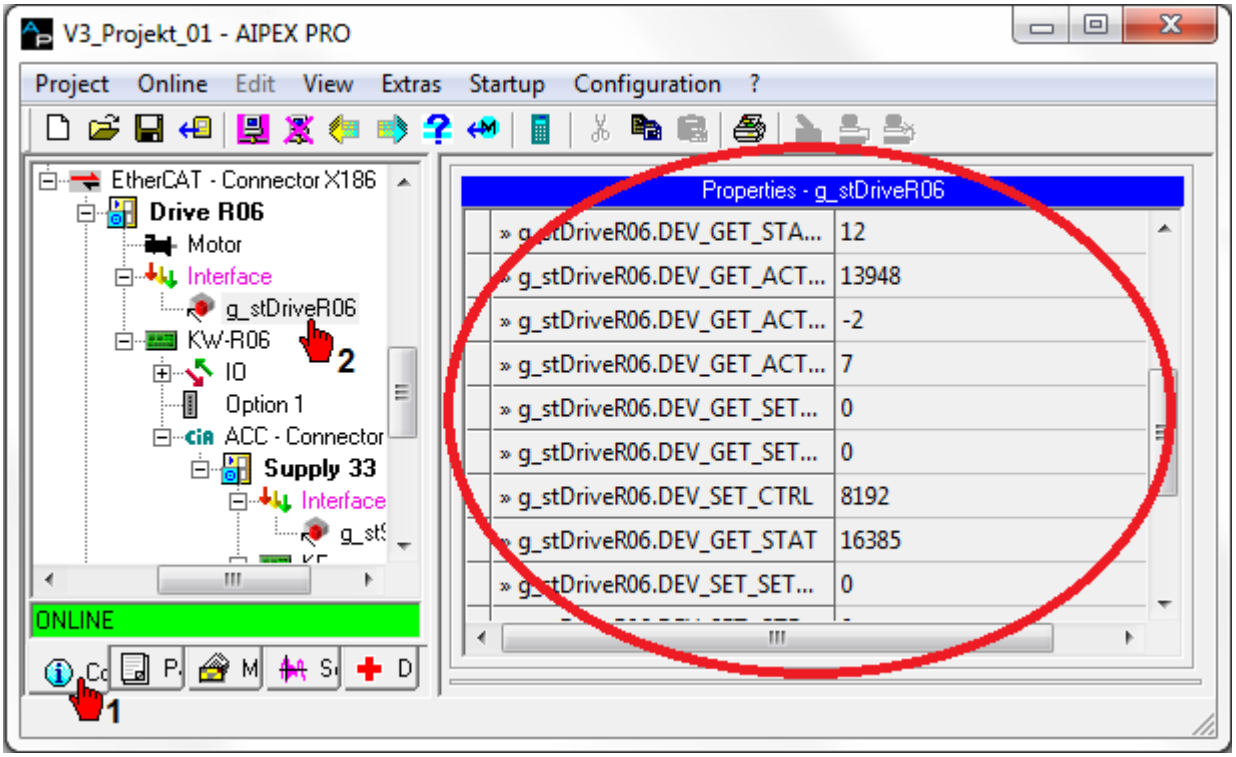


AIPEX PRO

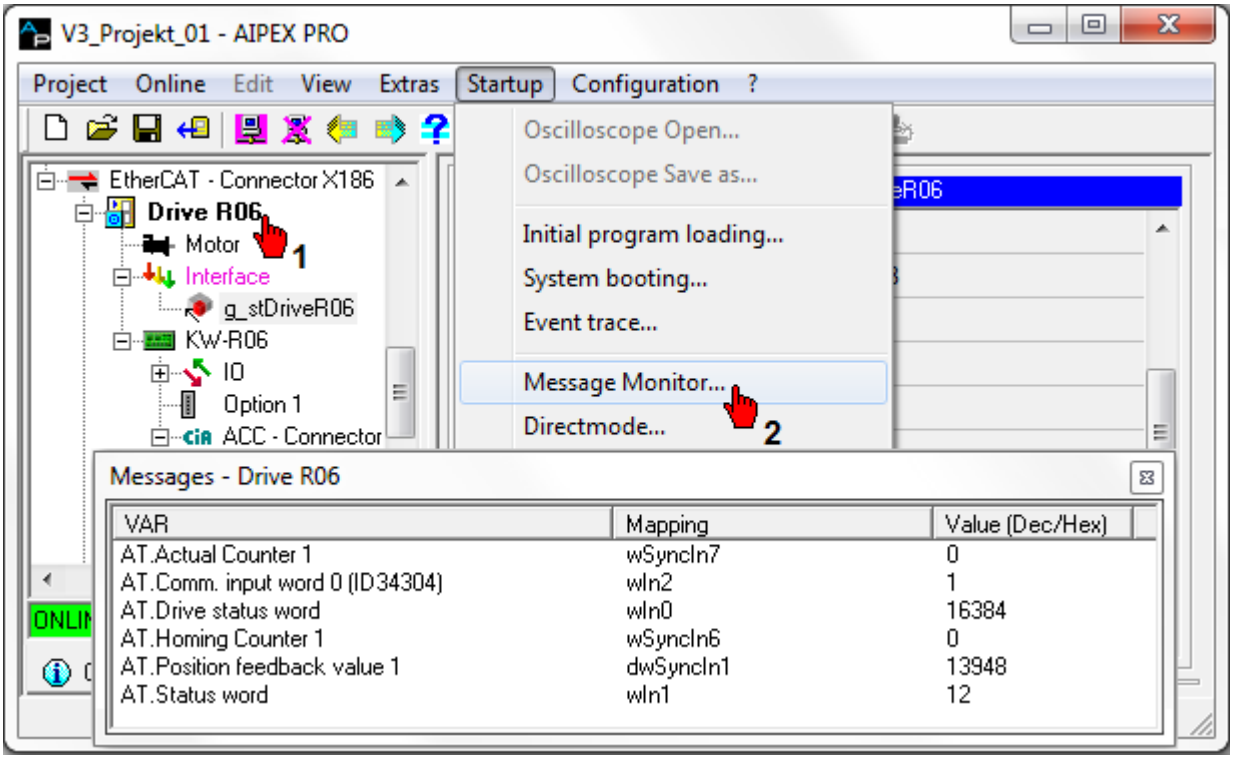
You can find the created message configuration file under the **'Messages'** tab



Online display of the function blocks used with current values



Online display of the mapped variables between slave and master.



4.3.1.13.2.1 Restriction of the automatic bus configuration

To determine the bus messages for the function blocks used in the CODESYS V3.5, a static source code analysis is carried out.

During this analysis, the linking of the symbolic device name is resolved with a function block. To resolve this link, the way (during the design time) must be clearly ascertainable as the symbolic device name is passed on.

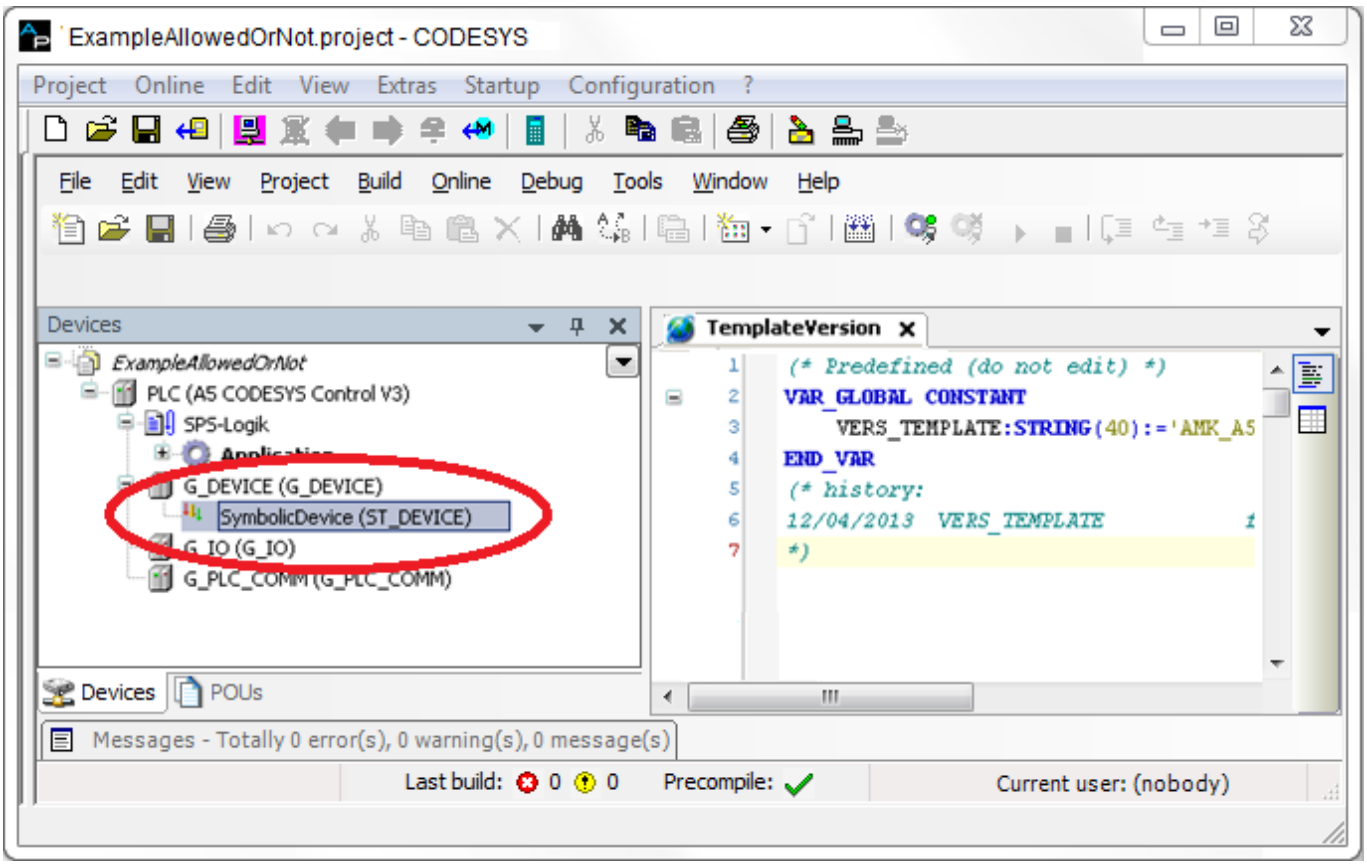
Relevant function blocks can be integrated into other modules for the automatic bus configuration. It should be borne in mind here that the relevant 'stDevice' variable for this is passed on directly. This variable must be declared in the 'VAR_IN_OUT' area of the higher-level module and have the 'ST_DEVICE' type.

If application modules containing relevant information for the automatic bus configuration are to be provided in a separate 'Internal CODESYS V3.5 library', then they have to be 'library' type modules. The 'compiled-library' type is not allowed!

Despite having the correct IEC61131-3 syntax, some constructs cannot be resolved for the configurator. This means that no bus messages are generated for the cases described. The PLC program itself, however, is functional. Only access to the bus message when calling up the corresponding basic function block (with 'boEnable := TRUE') will result in an error message (iErrID := 1).

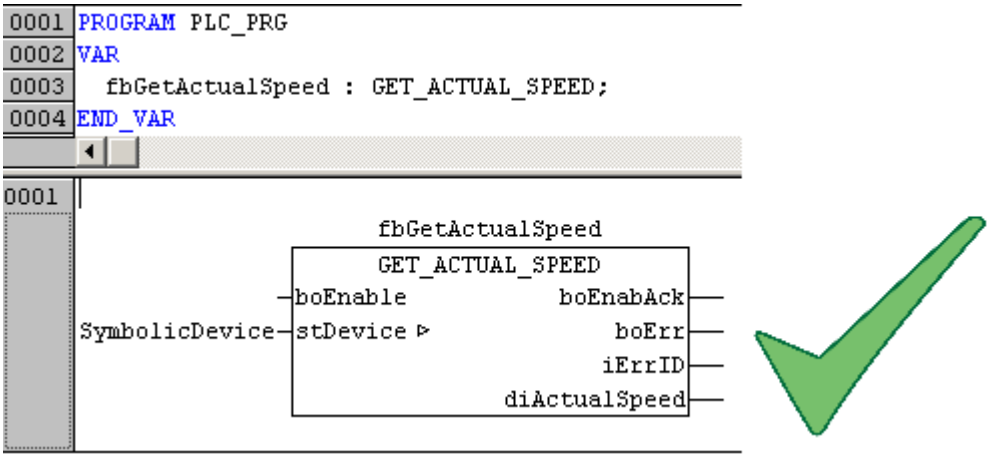
These restrictions are shown below by means of examples. The symbolic device for the following examples ('SymbolicDevice' variable) is defined in the next diagram.

Device definition in PLC controller configuration.



It is possible to determine bus messages correctly with the following variants

Variant 1: Calling of a function block in the cyclical program 'PLC_PRG' or 'FPLC_PRG' (standard module call)



Variant 2: Call encapsulated in its own function block (function block definition)

```
0001 FUNCTION_BLOCK OWN_FUNCTION_BLOCK
0002 VAR_INPUT
0003   boEnable : BOOL;
0004 END_VAR
0005 VAR_OUTPUT
0006   boEnableAck : BOOL;
0007   diSpeed : DINT;
0008 END_VAR
0009 VAR_IN_OUT
0010   stDevice : ST_DEVICE;
0011 END_VAR
```

```
0001
                    fbGetActualSpeed
                    GET_ACTUAL_SPEED
boEnable  ─boEnable              boEnabAck ──────── boEnableAck
stDevice  ─stDevice ▷              boErr ──
                                   iErrID ──
                          diActualSpeed ──diSpeed
```

Call of the superimposed module

```
0001 PROGRAM FPLC_PRG
0002 VAR
0003   fbOwnFunctionBlock : OWN_FUNCTION_BLOCK;
0004 END_VAR
```

```
0001
                    fbOwnFunctionBlock
                    OWN_FUNCTION_BLOCK
                  ─boEnable      boEnableAck ──
SymbolicDevice  ─stDevice ▷         diSpeed ──
```

Variant 3: Call of the function block or an action from another scope of application

```
0001 PROGRAM FPLC_PRG
0002 VAR
0003 END_VAR
0004
```

```
0001
                    PLC_PRG.fbGetActualSpeed
                    GET_ACTUAL_SPEED
                  ─boEnable            boEnabAck ──
SymbolicDevice  ─stDevice ▷              boErr ──
                                         iErrID ──
                                  diActualSpeed ──
```

Variant 4: Function block instance is defined globally

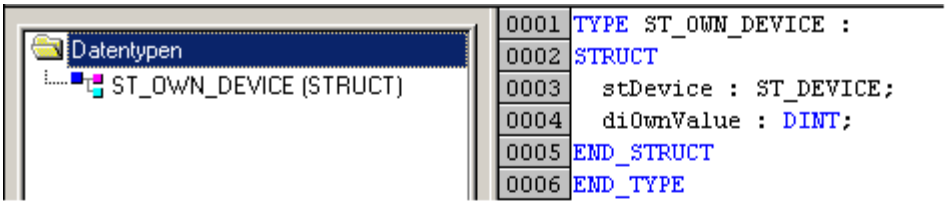
```
0001 VAR_GLOBAL
0002   fbGetActualSpeed : GET_ACTUAL_SPEED;
0003 END_VAR
0004
```

```
0001 PROGRAM FPLC_PRG
0002 VAR
0003 END_VAR
0004
```

```
0001
                        fbGetActualSpeed
                        GET_ACTUAL_SPEED
                    boEnable          boEnabAck
SymbolicDevice      stDevice ▷          boErr
                                        iErrID
                                    diActualSpeed
```

It is not possible to determine bus messages for the following variants

The symbolic device name is embedded in its own structure




```
0001 TYPE ST_OWN_DEVICE :
0002 STRUCT
0003   stDevice : ST_DEVICE;
0004   diOwnValue : DINT;
0005 END_STRUCT
0006 END_TYPE
```

```
0001 PROGRAM PLC_PRG
0002 VAR
0003   fbGetActualSpeed : GET_ACTUAL_SPEED;
0004 END_VAR
```

```
0001
stOwnDevice.stDevice———SymbolicDevice
```

```
0002
                        fbGetActualSpeed
                        GET_ACTUAL_SPEED
                    boEnable          boEnabAck
stOwnDevice.stDevice stDevice ▷         boErr
                                        iErrID
                                    diActualSpeed
```

4.3.1.13.3 Target system selection for CODESYS V3

For each controller variant, it is a suitable AMK target system (AMK-specific device description). In the target system selection must be differentiated between the built-in control processor type and the additional factory unlocked CODESYS options. The interfaces options are not required.

When creating a CODESYS project, the corresponding AMK target system is selected automatically depending on the selected controller variant.

The AMK target system must exactly match the existing controller variant (hardware). The 'Login' on the controller is not possible, when the selected AMK target system and the controller variant differ.

Example:

Controller type:	Display controller A5 with ATOM (X86 Intel) processor	
Unlocked factory options:	A5-VIS (Web visualization)	(Default display controller)
	A5-PCO (PLCopen (CODESYS 'SM_PLCopen.lib'))	(AMK part-no. O844)

Required AMK target system:
X86PLCopenControlWithVisu V3

4.3.1.13.3.1 Selection AMK target systems

Related controller, processor type and target system

Controller	Processor	AMK target system
A4	ARM processor	ARM...
A5	ATOM (X86 Intel) processor	X86...
A6	ATOM (X86 Intel)processor	X86...
iSA	ARM processor	ARM...

Available CODESYS options

Option	Meaning
A4-VIS [1)] A5-VIS [1)] A6-VIS [1)] iSA-VIS	Web visualization
A4-PCO A5-PCO A6-PCO iSA-PCO	PLCopen (CODESYS 'SM_PLCopen.lib')
A5-PNC A6-PNC iSA-PNC	Numerical Control Motion (A5-PCO integrated) for CODESYS V3

1) Ax-VIS unlocked from the factory for the A5/A6 display controllers

Additional information on the CODESYS options:
for AMKAMAC A-Serie: See document Product description Controllers A4 / A5 / A6 (Part no. 202975)
for AMKASMART iSA: See document Product description iSA (Part no. 205670)

A4 with ARM processor

Installed options	AMK target system for CODESYS V3	Description
-	X86Control V3	Controller A4 without additional options, no option VIS (visualization)
A4-VIS	X86ControlWithVisu V3	Controller A4 without additional options, with option VIS (visualization)
A4-PCO	X86PLCopenControl V3	Controller A4 with additional option PCO (PLCopen), no option VIS (visualization)
A4-VIS, A4-PCO	X86PLCopenControlWithVisu V3	Controller A4 with additional option PCO (PLCopen), with option VIS (visualization)
A4-PNC	X86PLCopenCncControl V3	Controller A4 with additional option PNC (PLCopen CNC), no option VIS (visualization)
A4-VIS, A4-PNC	X86PLCopenCncControlWithVisu V3	Controller A4 with additional option PNC (PLCopen CNC), with option VIS (visualization)

A5/A6 with ATOM (X86 Intel) processor

Installed options	AMK target system for CODESYS V3	Description
-	X86Control V3	Controller A5/A6 without additional options, no option VIS (visualization)
A5/A6-VIS	X86ControlWithVisu V3	Controller A5/A6 without additional options, with option VIS (visualization)
A5/A6-PCO	X86PLCopenControl V3	Controller A5/A6 with additional option PCO (PLCopen), no option VIS (visualization)
A5/A6-VIS, A5/A6-PCO	X86PLCopenControlWithVisu V3	Controller A5/A6 with additional option PCO (PLCopen), with option VIS (visualization)
A5/A6-PNC	X86PLCopenCncControl V3	Controller A5/A6 with additional option PNC (PLCopen CNC), no option VIS (visualization)
A5/A6-VIS, A5/A6-PNC	X86PLCopenCncControlWithVisu V3	Controller A5/A6 with additional option PNC (PLCopen CNC), with option VIS (visualization)

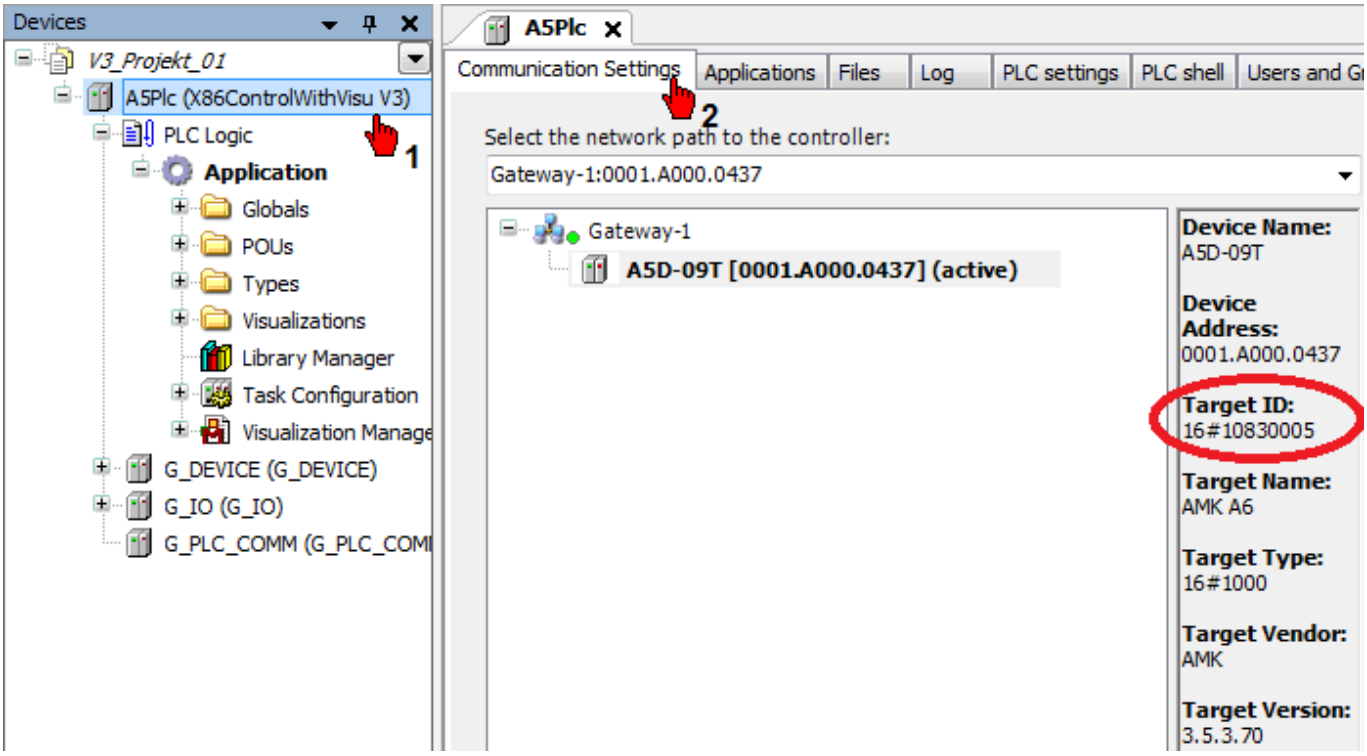
iSA with ARM processor

Installed options	AMK target system for CODESYS V3	Description
-	ArmControl V3	Controller iSA without additional options, no option VIS (visualization)
iSA-VIS	ArmControlWithVisu V3	Controller iSA without additional options, with option VIS (visualization)
iSA-PCO	ArmPLCopenControl V3	Controller iSA with additional option PCO (PLCopen), no option VIS (visualization)
iSA-VIS, iSA-PCO	ArmPLCopenControlWithVisu V3	Controller iSA with additional option PCO (PLCopen), with option VIS (visualization)
iSA-PNC	ArmPLCopenCncControl V3	Controller iSA with additional option PNC (PLCopen CNC), no option VIS (visualization)
iSA-VIS, iSA-PNC	ArmPLCopenCncControlWithVisu V3	Controller iSA with additional option PNC (PLCopen CNC), with option VIS (visualization)

4.3.1.13.3.2 Device identification

According to the following table the controllers (AMK target system) are identified based on their target ID and target system type.

AMK target system	Target ID	Target system type	
X86Control V3 X86ControlWithVisu V3	16#10830006 16#10830005	16#1000	Controller
X86PLCopenControl V3 X86PLCopenControlWithVisu V3	16#10830004 16#10830002	16#1006	Softmotion Controller
X86PLCopenCncControl V3 X86PLCopenCncControlWithVisu V3	16#10830003 16#10830001	16#1006	Softmotion Controller
ArmControl V3 ArmControlWithVisu V3	16#10830016 16#10830015	16#1000	Controller
ArmPLCopenControl V3 ArmPLCopenControlWithVisu V3	16#10830014 16#10830012	16#1006	Softmotion Controller
ArmPLCopenCncControl V3 ArmPLCopenCncControlWithVisu V3	16#10830013 16#10830011	16#1006	Softmotion Controller



4.3.1.13.3.3 AIPEX PRO CODESYS projects with CODESYS options

The term CODESYS options means factory unlocked options like PLCopen, Numerical Control Motion or web visualization.

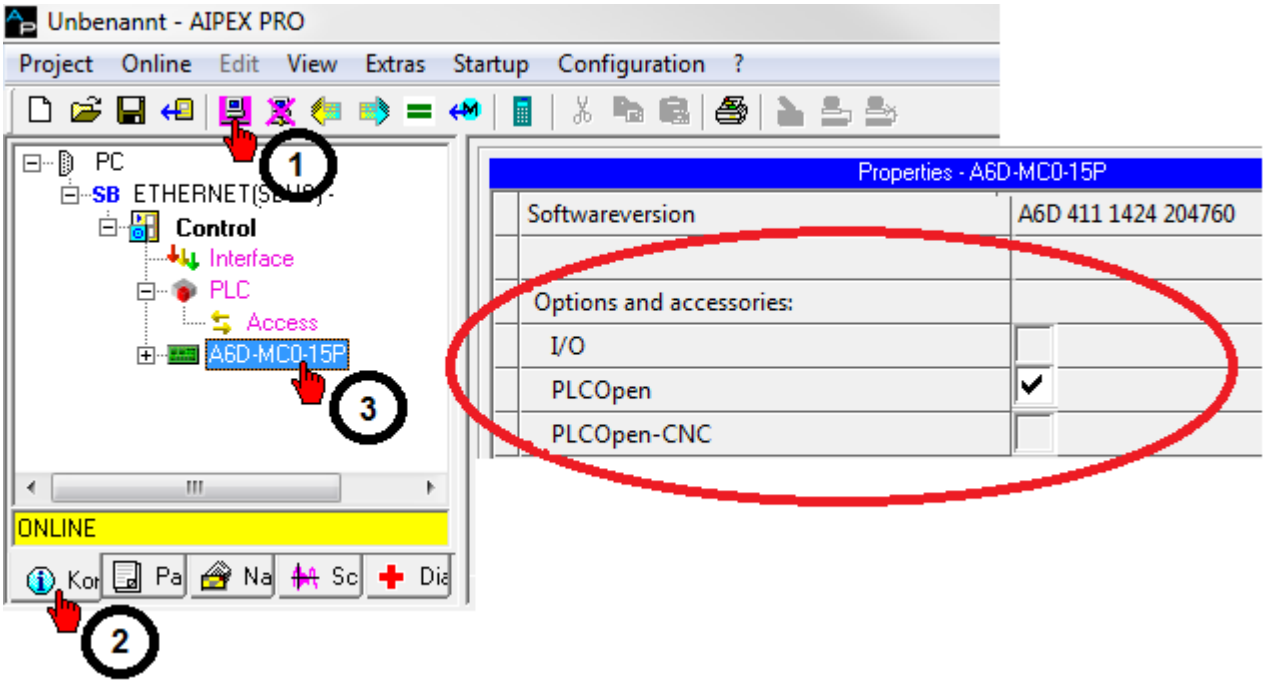
The controller variant in AIPEX PRO device tree is the base for a new PLC project. The template and the AMK target system is automatically adapted to this control type and enabled CODESYS options.

Subsequent controller variant changes has no effect on the CODESYS project. Create a new AIPEX PRO project for a new controller varaint. Afterwards you can take over the existing program code in the new CODESYS project.

The AMK target system must exactly match the existing controller variant (hardware). The 'Login' on the controller is not possible, when the selected AMK target system and the controller variant differ.

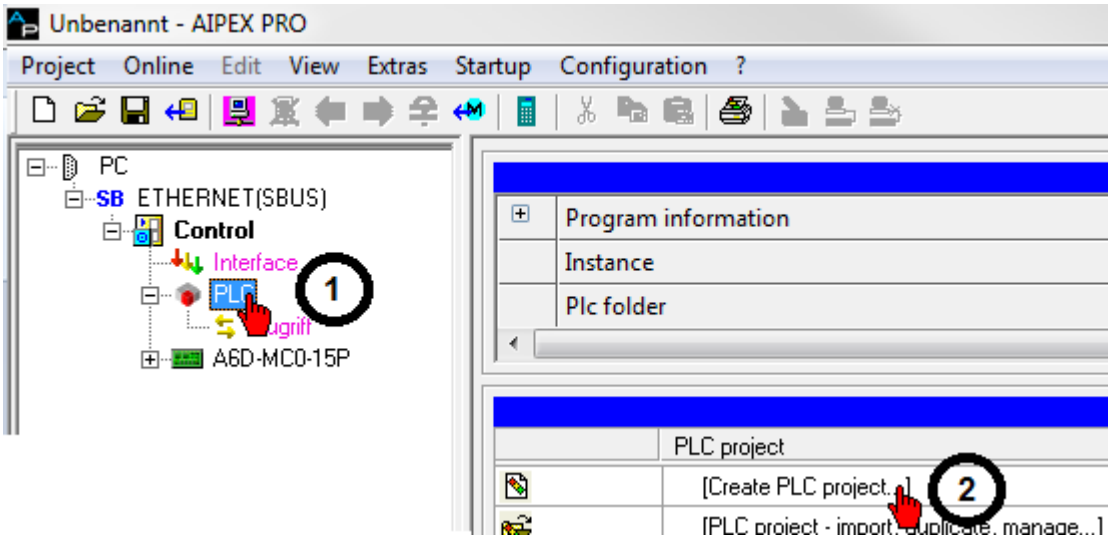
New 'Online - Project'

During 'Online' - Scan with AIPEX PRO, active and factory unlocked CODESYS options will be automatically taken over in the AIPEX PRO project.



'Create new PLC project'

Templates and target settings are automatically adjusted to the imported controller variant.

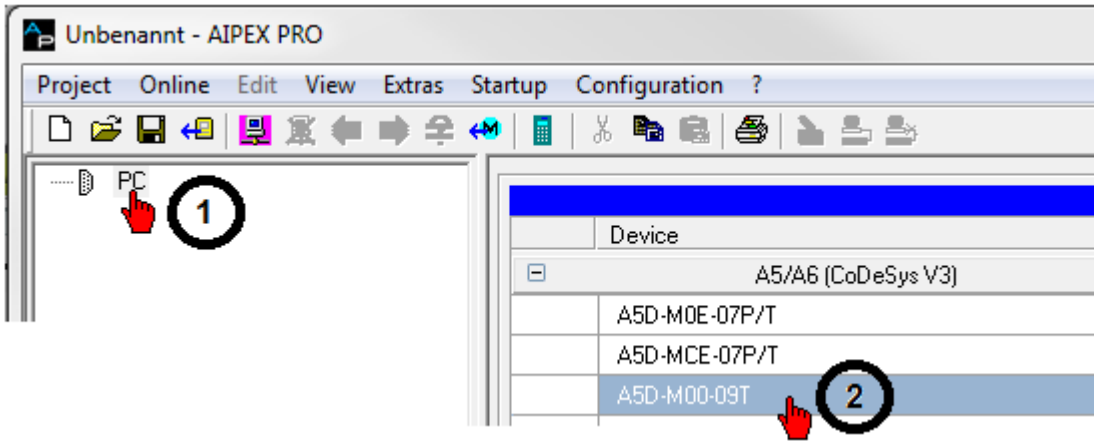


New 'Offline - Project'

When you use a 'Offline-Project', the selected controller type and the selected CODESYS options must necessarily match with the controller variant (hardware).The 'Login' on the controller is not possible, when the selected selected controller type and the selected CODESYS options and the controller variant differ.

Procedure for a new 'Offline - Project':

Add the PLC controller



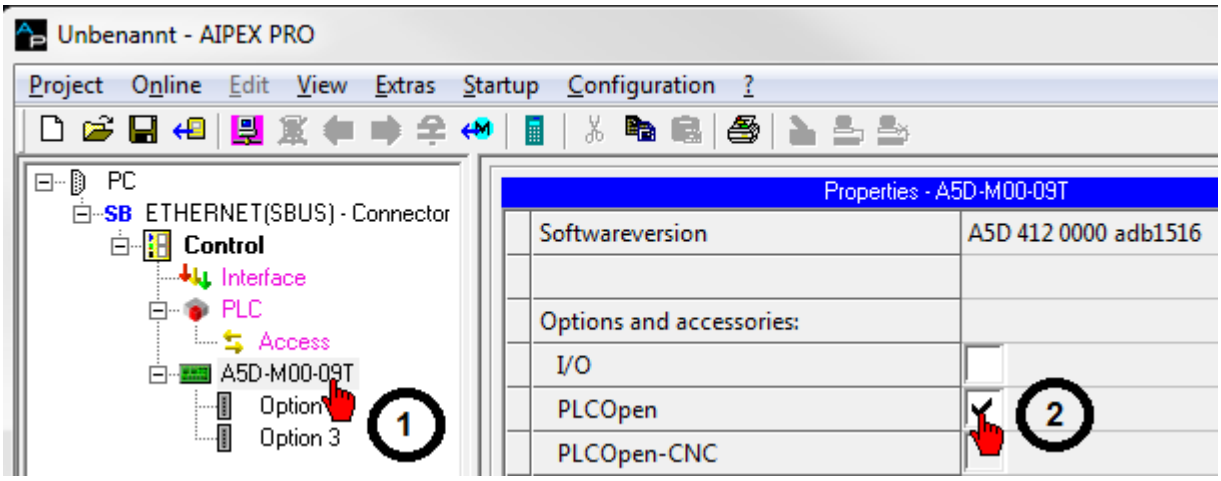
Select CODESYS options

Select all CODESYS options that in the controller variant (hardware) are available

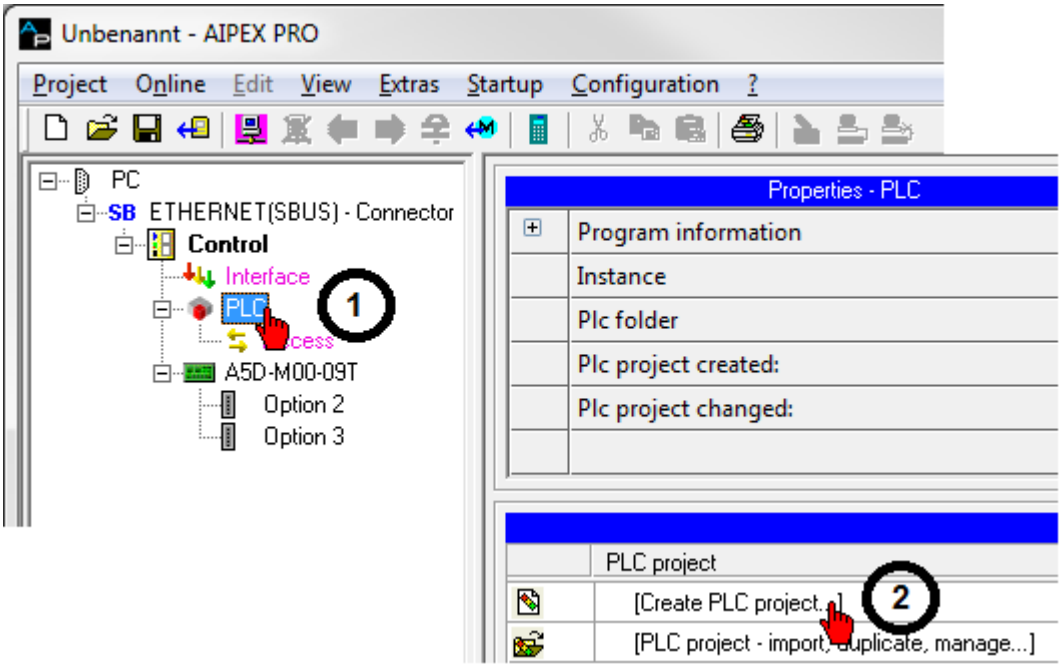
After the PLC project is generated (see next step), subsequent changes has no longer affect to the template and the target system.

Only CODESYS options can be used, which were considered in the controller order and are unlocked from the AMK factory.

Example: Controller A5D-M00-09T with CODESYS option PLCopen (CODESYS 'SM_PLCopen.lib')

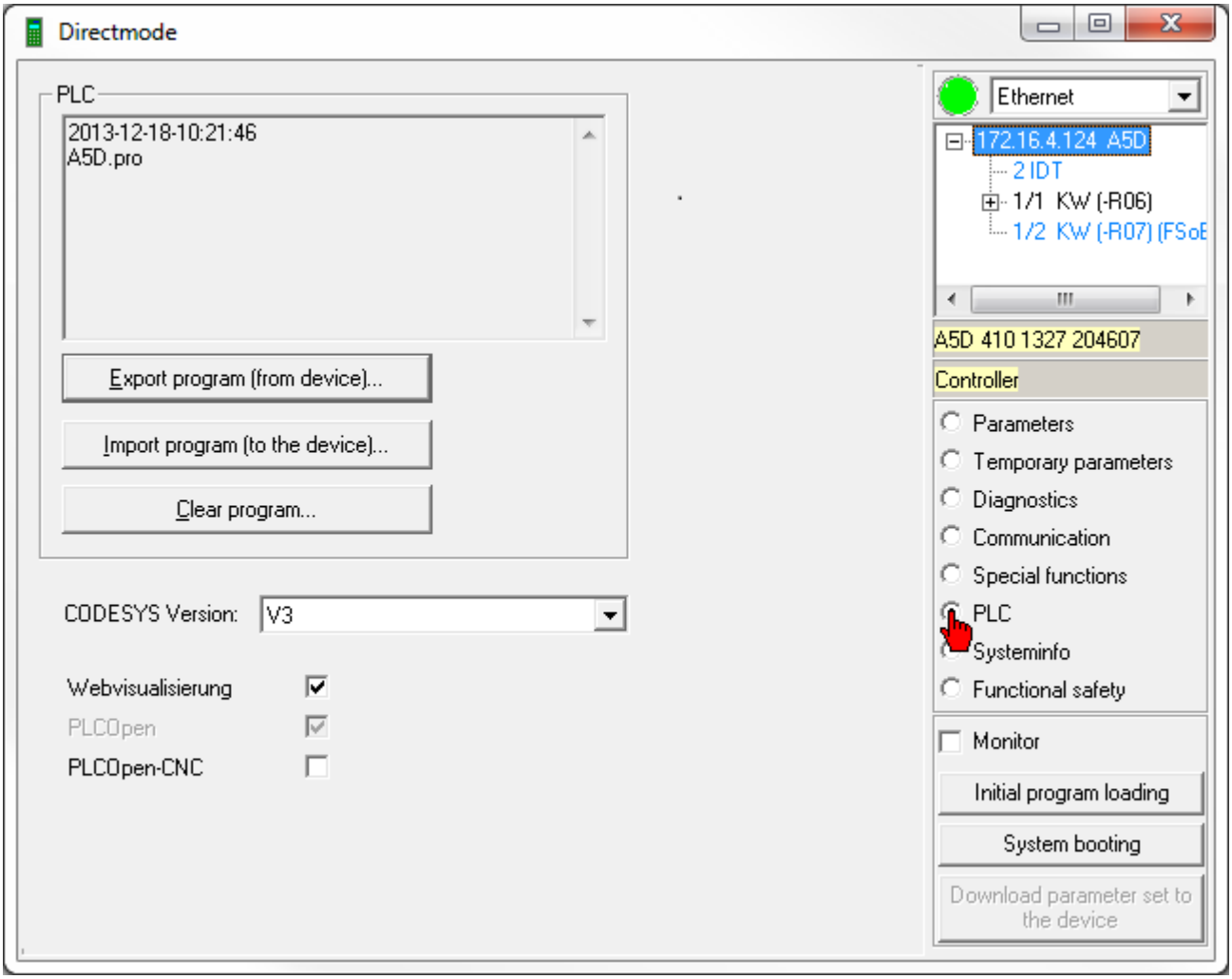


'Create new PLC project'
Templates and target settings are automatically adjusted to the selected controller type and CODESYS options.



4.3.1.13.3.4 Disable CODESYS Options

The factory unlocked CODESYS options can be disabled by the user and activated when needed.
To do this, open the AIPEX PRO 'Direct Mode' menu 'PLC'



The installed controller options at factory setting can be disabled and enabled by using the checkboxes.

After dis- or enabling the options, the controller must be restarted (24 VDC OFF / ON).

Example:

For an ISA controller with additional option PCO (PLCopen) and option VIS (visualization) uses the target system: ArmPLCopenControlWithVisu V3.

After disabling the checkbox 'PLCopen' the device will be a iSA controller with additional option VIS (visualization) with target system: ArmControlWithVisu V3.

The CODESYS option selection in AIPEX PRO 'Direct Mode' menu 'PLC' accesses the ID34175 'Controller settings' bits 8..10.

The following table shows how by setting the bits 8..10, a originally type of controller variant and the associated AMK target system will be changed to a new controller variant with new associated AMK target system.

Example:

Factory unlocked CODESYS Option PLCopen (CODESYS 'SM_PLCopen.lib')

Bit9 = 1 = Option PLCopen disabled

Bit9 = 0 = Option PLCopen enabled

Installed options	AMK target system for CODESYS on delivery	A5/A6 -VIS	A5/A6 -PCO	A5/A6 -PNC	AMK target system for CODESYS after disabling options
-	X86Control V3	-	-	-	X86Control V3
A5/A6-VIS	X86ControlWithVisu V3	Bit 8 = 1	-	-	X86Control V3
A5/A6-PCO	X86PLCopenControl V3	-	Bit 9 = 1	-	X86Control V3
A5/A6-VIS, A5/A6-PCO	X86PLCopenControlWithVisu V3	Bit 8 = 1	-	-	X86PLCopenControl V3
		-	Bit 9 = 1	-	X86ControlWithVisu V3
		Bit 8 = 1	Bit 9 = 1	-	X86Control V3
A5/A6-PNC	X86PLCopenCncControl V3	-	-	Bit 10 = 1	X86PLCopenControl V3
		-	Bit 9 = 1	Bit 10 = 1	X86Control V3
A5/A6-VIS, A5/A6-PNC	X86PLCopenCncControlWithVisu V3	Bit 8 = 1	-	-	X86PLCopenCncControl V3
		-	-	Bit 10 = 1	X86PLCopenControlWithVisu V3
		Bit 8 = 1	-	Bit 10 = 1	X86PLCopenControl V3
		-	Bit 9 = 1	Bit 10 = 1	X86ControlWithVisu V3
		Bit 8 = 1	Bit 9 = 1	Bit 10 = 1	X86Control V3

Installed options	AMK target system for CODESYS on delivery	iSA -VIS	iSA -PCO	iSA -PNC	AMK target system for CODESYS after disabling options
-	ArmControl V3	-	-	-	ArmControl V3
iSA-VIS	ArmControlWithVisu V3	Bit 8 = 1	-	-	ArmControl V3
iSA-PCO	ArmPLCopenControl V3	-	Bit 9 = 1	-	ArmControl V3
iSA-VIS, iSA-PCO	ArmPLCopenControlWithVisu V3	Bit 8 = 1	-	-	ArmPLCopenControl V3
		-	Bit 9 = 1	-	ArmControlWithVisu V3
		Bit 8 = 1	Bit 9 = 1	-	ArmControl V3
iSA-PNC	ArmPLCopenCncControl V3	-	-	Bit 10 = 1	ArmPLCopenControl V3
		-	Bit 9 = 1	Bit 10 = 1	ArmControl V3
iSA-VIS, iSA-PNC	ArmPLCopenCncControlWithVisu V3	Bit 8 = 1	-	-	ArmPLCopenCncControl V3
		-	-	Bit 10 = 1	ArmPLCopenControlWithVisu V3
		Bit 8 = 1	-	Bit 10 = 1	ArmPLCopenControl V3
		-	Bit 9 = 1	Bit 10 = 1	ArmControlWithVisu V3
		Bit 8 = 1	Bit 9 = 1	Bit 10 = 1	ArmControl V3

4.3.1.13.3.5 AMK target system (controller variant) change

Service information!

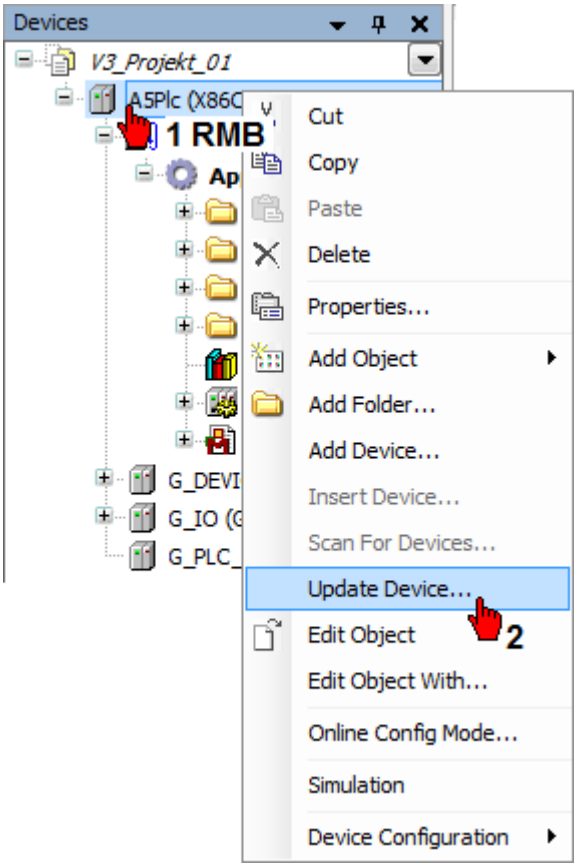
The example below shows how you can change an existing project to another controller variant.

Additional adjustments have to be made, depending on the new controller variant. Attention controller-specific templates are not updated.

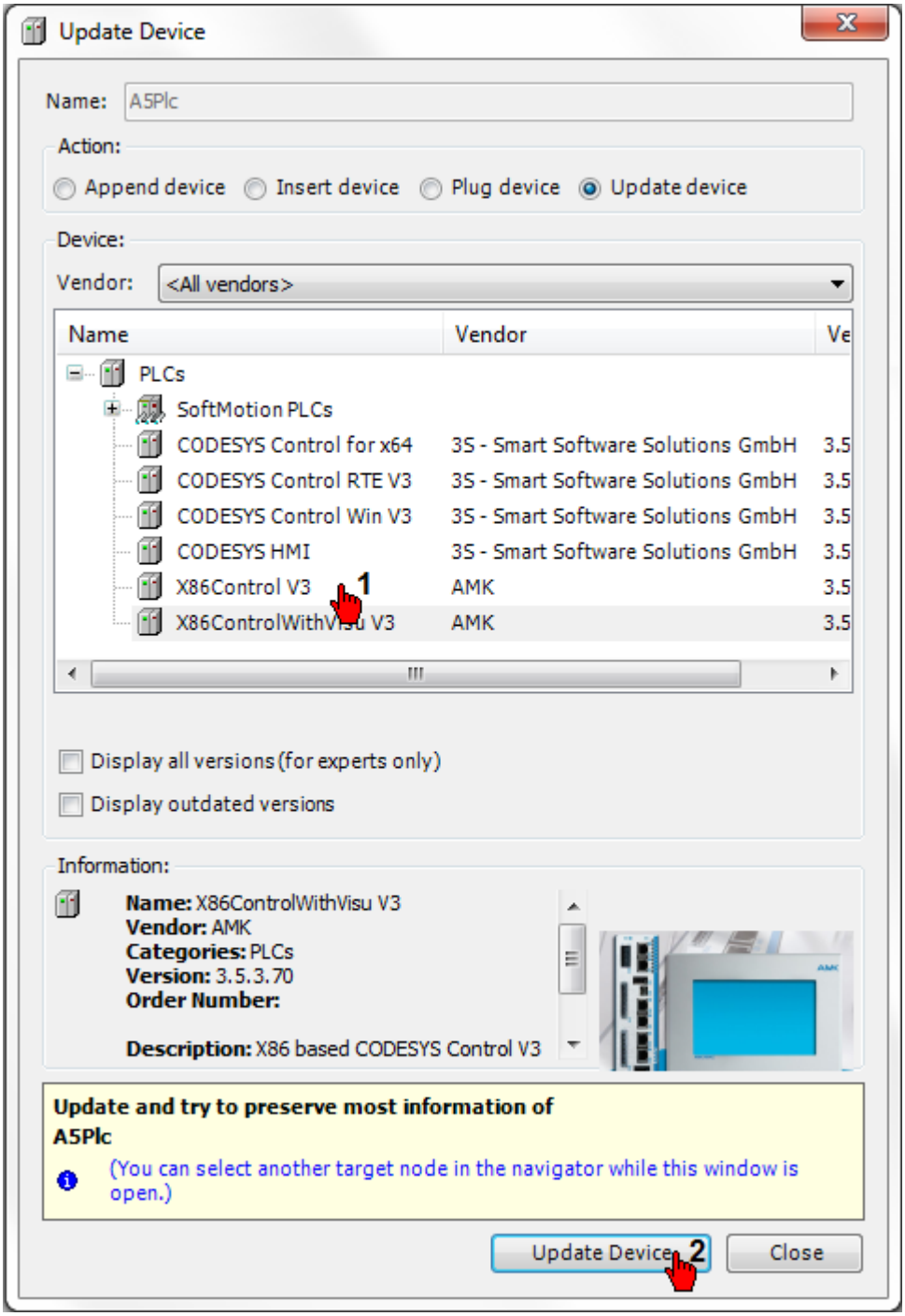
A subsequent change of the target system is not being considered. Create in this case a new AIPEX PRO project for the new controller variant. Afterwards you can take over the existing program code in the new CODESYS project.

Change the device description (controller variant)

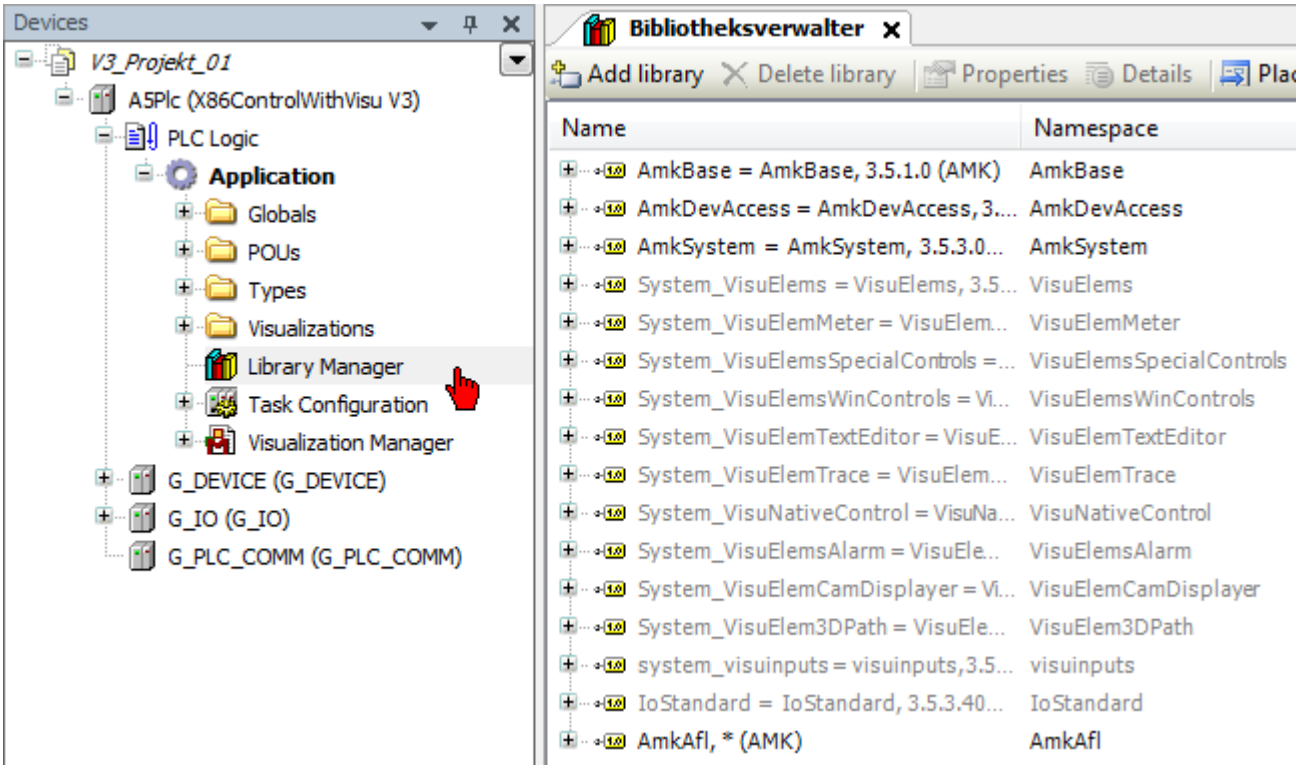
The example below shows how you can change an existing project to another controller variant. Additional adjustments have to be made, depending on the new control variant.



Select the new control type in the selection list. Confirm with '**Update device**'.



4.3.1.13.4 Library Administrator



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.

Library overview of the AMK basic modules

Topic	<Name>.library	Version	Impl.[1)]	Lib.[2)]	Place [3)]	Note
Basic	AmkBase	3.5.1.0	E	C	G	Base functionality
	AmkFile	3.5.1.0	E	C	G	File functions
	AmkSystem	3.5.3.0	I	C	B	System functionality
Communication	AmkCom	3.5.1.0	E	C	G	Communication functionality
	AmkSocket	3.5.1.0	E	C	G	Ethernet Socket functions
	AmkTcp	3.5.3.0	I	C	B	TCP communication interface
	AmkUdp	3.5.3.0	I	C	B	UDP communication interface
Device	AmkDevAccBase	3.5.3.0	I	S	B	Base device access functionality
	AmkDevAccess	3.5.3.0	I	S	B	Device access functionality
	AmkEasyDev	3.5.3.0	I	S	B	Simplified AMK device interface

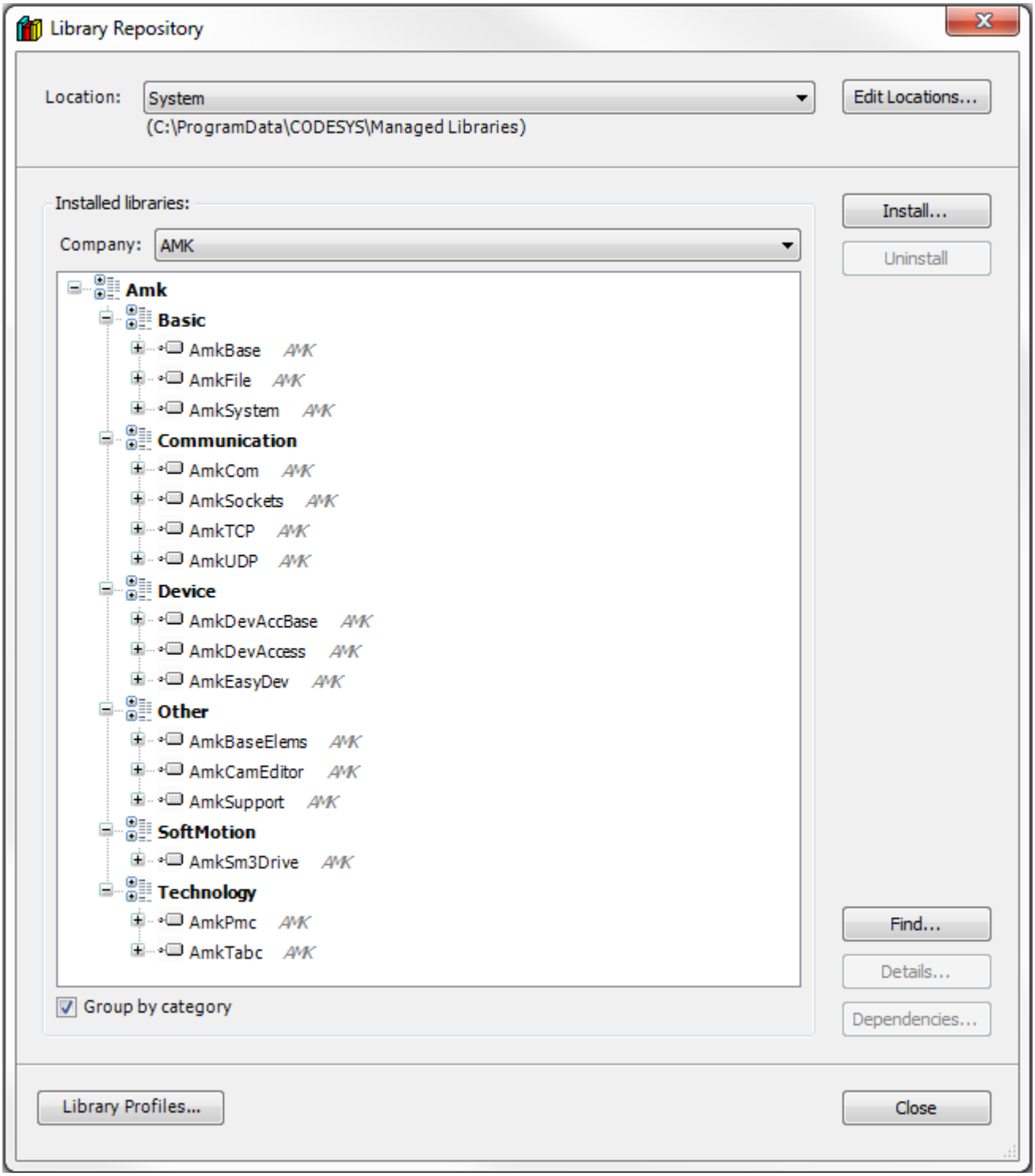
Topic	<Name>.library	Version	Impl.[1)]	Lib.[2)]	Place [3)]	Note
Other	AmkBaseElems	3.5.3.0	I	C	B	Basic visualization elements
	AmkCamEditor	3.5.3.0	I	S	B	CamEditor specific type definitions
	AmkSupport	3.5.3.0	I	C	B	Support of special hardware/technologies
SoftMotion	AmkSm3Drive	3.5.3.0	I	S	B	AMK Softmotion drive interface
Technology	AmkPmc	3.5.3.0	I	C	B	Register mark controller functionality
	AmkTabc	3.5.3.0	I	C	B	Spreadsheet modules

1) Implementation: E = external / I = internal
   - External: Implemented in the form of a system software component, programmed in 'C'..
   - Internal: Implemented in the form of an IEC program.
2) Library implementation: C = as 'compiled library'/ S = as 'source library'

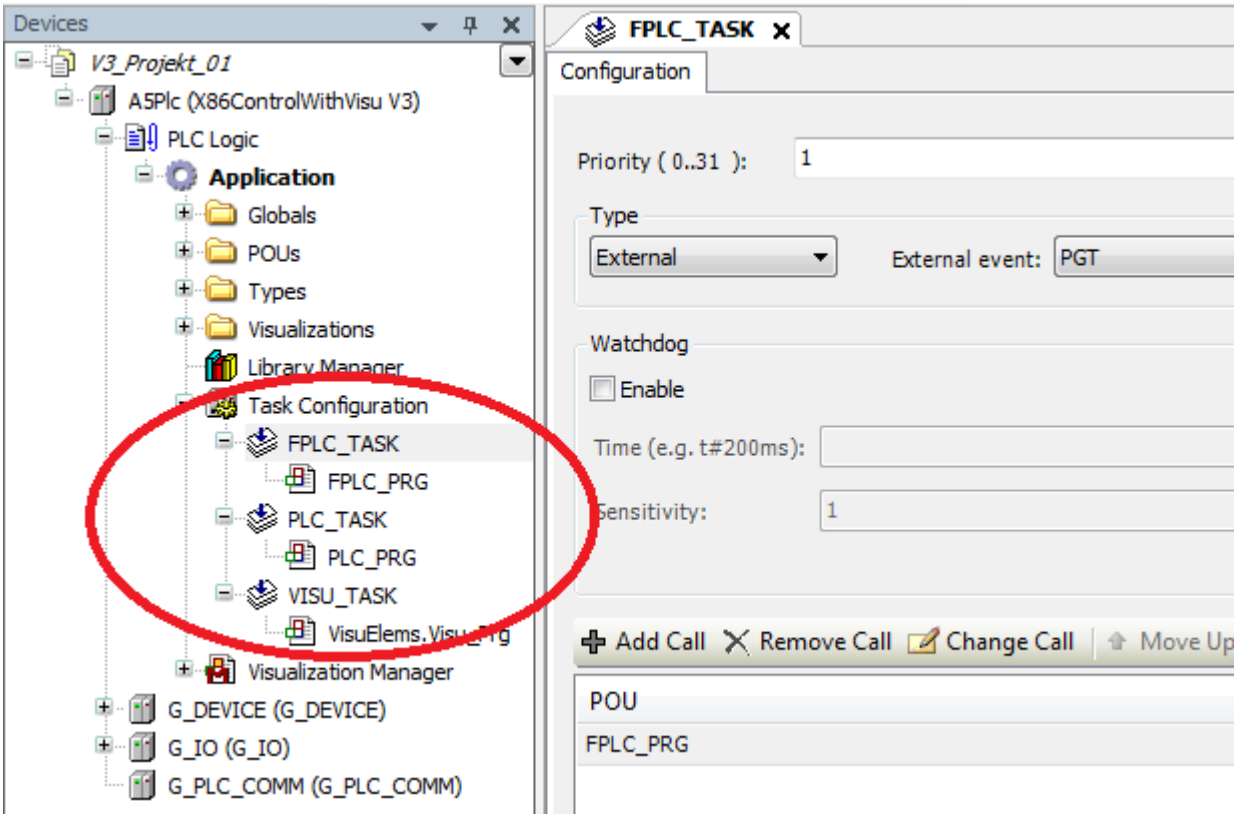
   A library implementation in the form of a 'compiled library' or source text library.
   - Compiled libraries are more code efficient but cannot be analyzed in the source text. For this reason, it is not possible to 'step' into the libraries for test purposes.
   - Source text libraries can be analyzed like the program in test mode (breakpoints, single-step mode,..).
3) Placeholder implementation: G = by means of device description, B = by means of library profile

   One placeholder implementation based on a device description or by means of a library profile.
   - With the 'Placeholder implementation based on a device description', the version resolution of the library takes place within the device description of the controller (controller version).
   - With the 'Placeholder implementation based on a library description', the version resolution of the library takes place in CODESYS depending on the compiler version.

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



4.3.1.13.5 Default task configuration



POU	Typ	Priority
FPLC_PRG	external, PGT event-controlled PGT cycle time= ID2 'SERCOS cycle time'	1
PLC_PRG	free-running	5
VISU_TASK	cyclic, interval 100 ms	10

4.3.1.13.6 Diagnostic information

Diagnostic information is displayed with the boErr (error signal) and iErrID (error identity number) variables.

```
fbSET_PLCVAR_SYNC_DINT_B(
    boEnable:= TRUE,
    diInVal:= 250,
    boEnabAck=> ,
    boErr=> ,
    iErrID=> ,
    stPlcVar:= g_stPLCVAR_B);
```

The following applies here:

iErrID = 0 no error or no warning
iErrID <> 0 Warning, if boErr = FALSE
Error if boErr = TRUE

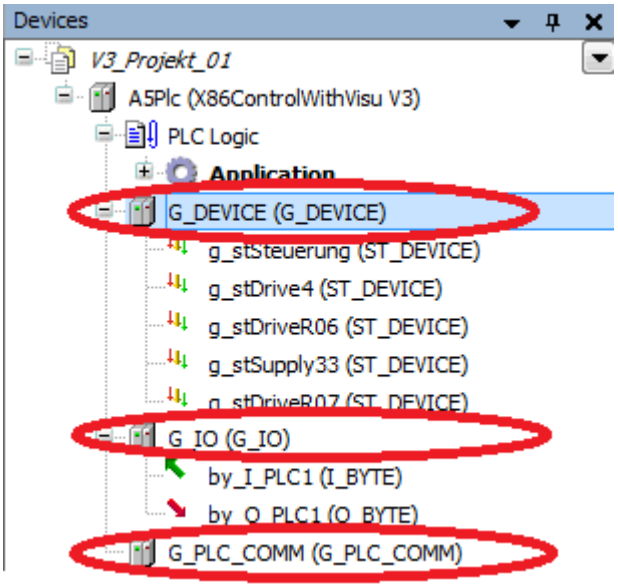
iErrID > 0 local diagnostic information according to the relevant module specification
iErrID < 0 global diagnostic information according to the subsequent

global diagnostic information (interlibrary)

Error identification number (iErrID)	Meaning
- 8	enMode: Mode unacceptable
- 9	ID2: Cycle time unacceptable
- 10	Serial interface is assigned
- 20	Function is not supported by this target system

4.3.1.13.7 Control Configuration

The 'Control configuration' maps the target hardware in the programming system.



The control configuration distinguishes between the following configuration options:

G_DEVICE: global devices (e.g. for access to device information of drives, power supply modules or controls). The global devices are referred to as 'Handle' in the field of computer science.

G_IO: global IO variables (e.g. for access to binary inputs/outputs)

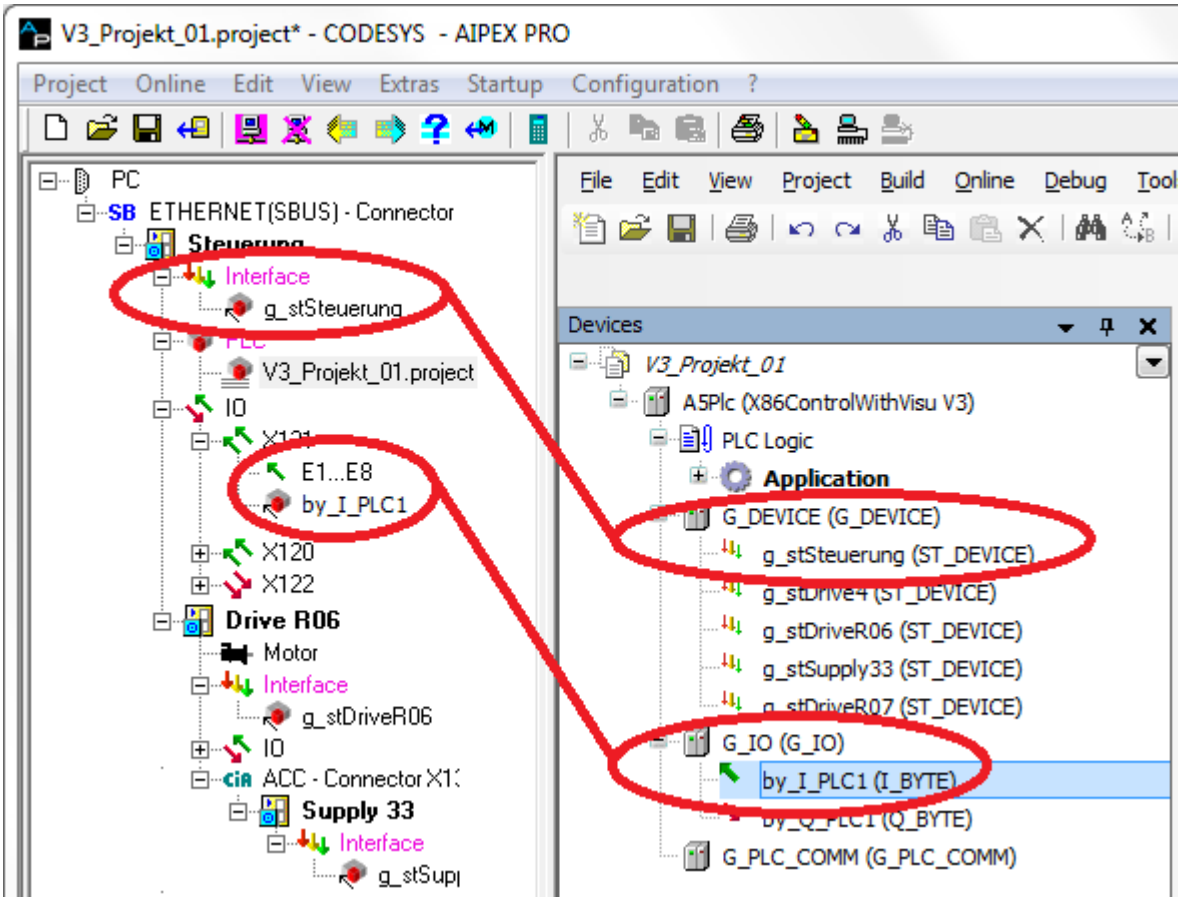
G_PLC_COMM: global PLC to PLC communication variable (for communication between PLCs). The global PLC to PLC communication variables are referred to as 'Handle' in the field of computer science.

Create new symbolic device names:

Click with the RMB on **G_DEVICE**, **G_IO**, **G_PLC_COMM**. Then select the 'attach device' menu item.

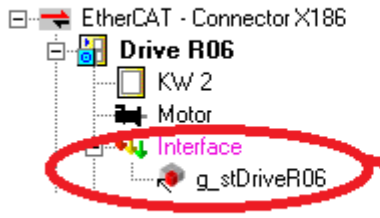
The devices created in the control configuration are referred to as 'symbolic device names' in the AMK documentation.

The physical devices are allocated to the symbolic device names during the bus configuration procedure.



A PLC function block is allocated to a physically existing device via the 'stDevice' variable. To do this, link the 'stDevice' variable with the symbolic device name.

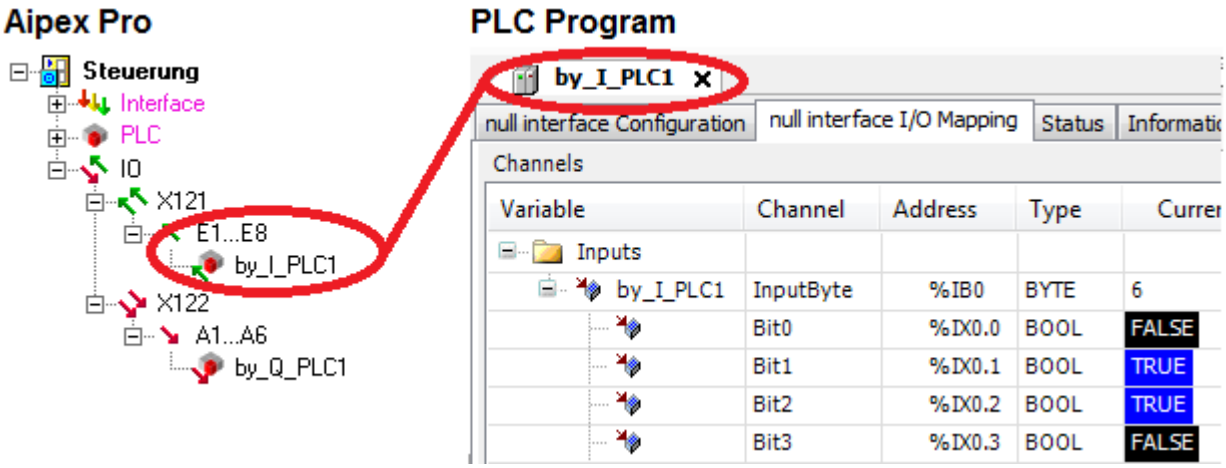
Aipex Pro



PLC Program

```
fbTempInternal_R06(
    boEnable:= g_boEnable,
    boEnabAck=> ,
    boErr=> ,
    iErrID=> ,
    ReadId33116=>
    stDevice:= g_stDriveR06 );
    (*Symbolic Device*)
```

Example IOs:
The physical devices are allocated to the symbolic device names (IO variables)during the bus configuration procedure.



4.3.2 FIRST STEPS with CODESYS V2 with AFL Standard blocks

'FIRST STEPS' describes how:

- to do the AIPEX PRO 'Basic adjustment'
- to generate a new PLC project with AIPEX PRO
- to import AFL Standard Function Blocks and instance for a controller, input and axis
- to access the axis structures (actual and setpoint values)
- to transfer the PLC program

The chapter 'Quick start' contains further examples with the following topics:

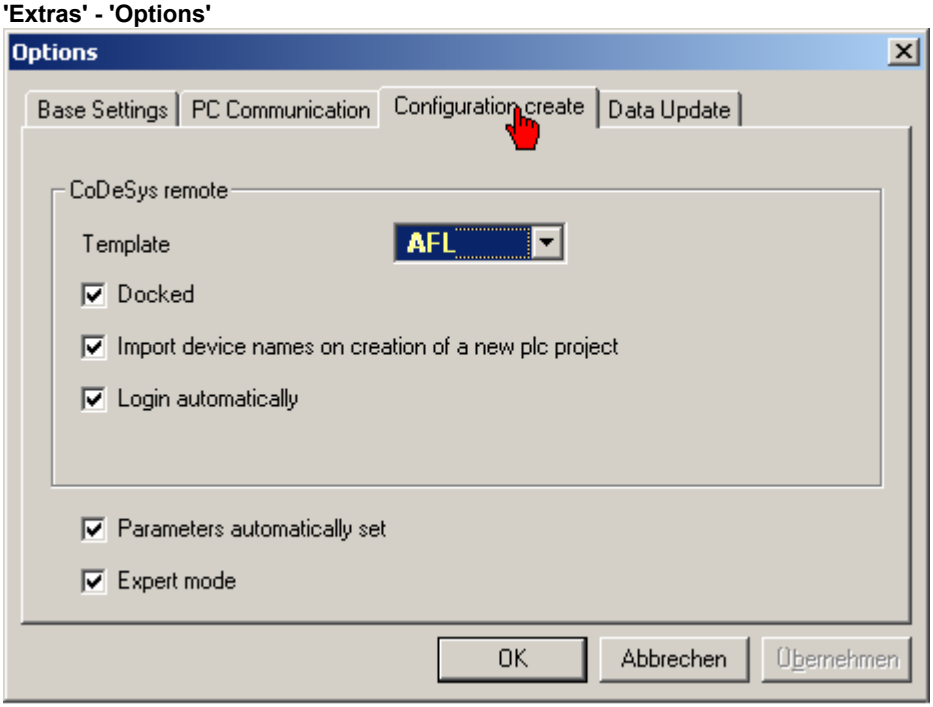
- Visualising a diagnostic array (arstDiagnosis)
- IO access
- Additional variable access
- Creating a cam
- Importing a PLC project from AIPEXPRO
- Importing a PLC project
- Expanding status bits
- Access to PLC IOs

4.3.2.1 Basic adjustment

4.3.2.1.1 Configuration create

The template must be on 'AFL' stand before the CODESYS project is generated.

'Extras' - 'Options'



Template

You define with Template the start template of a new CODESYS project.

You can choose between ST structured text FBD function block diagram or if installed AFL library.

ST is the default adjustment.

Docked

The PLC editor CoDeSys is incorporated directly in the AIPEX PRO interface. If this option is not set, then the PLC editor CoDeSys will be started as an independent application.

Login automatically

If a configuration is created without faults, the CoDeSys function "Logon" is invoked.

Import device names on creation of a new plc project

Devices that are physically available are automatically copied into the PLC project by the call up of "Creating a PLC project". You will find the symbolic device names at **Resources** -> **Controller configuration** -> **PLC configuration**.

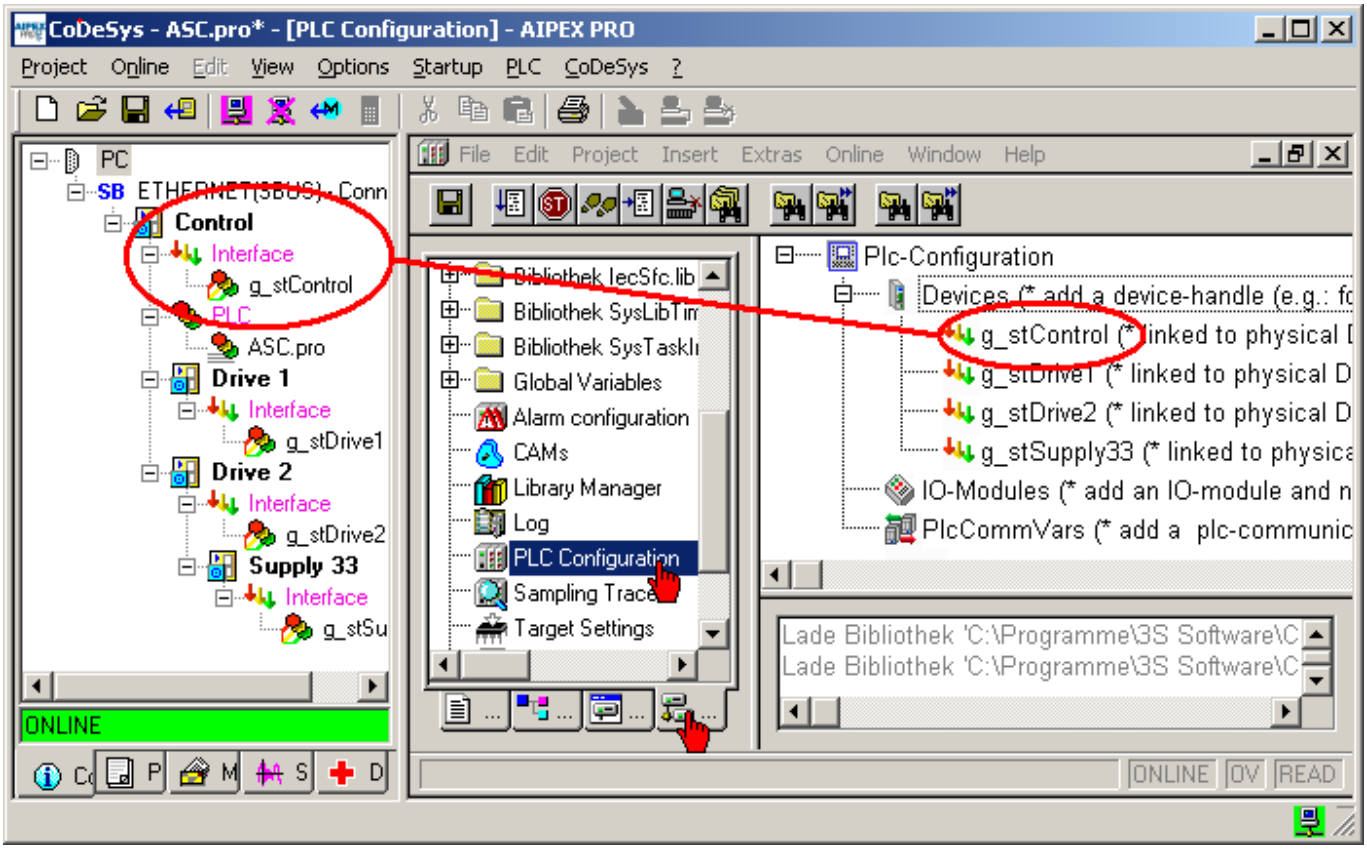
A later changing of the device name is not permitted.

Example CODESYS V2

You will find the symbolic device names at **'Resources'** -> **'Controller configuration'** -> **'PLC configuration'**.

In the device tree from AIPEX PRO is the PLC device handle name linked with the attendant device icon 'Interface' automatically.

The automatic message configuration use this information to create a message configuration file.



Parameters automatically set

The device parameters which are needed from the CoDeSys bibliotheca are set automatically.

This global adjustment can be individual overwrite via the menu Extras -> Project Settings -> Configuration create -> Parameters automatically set.

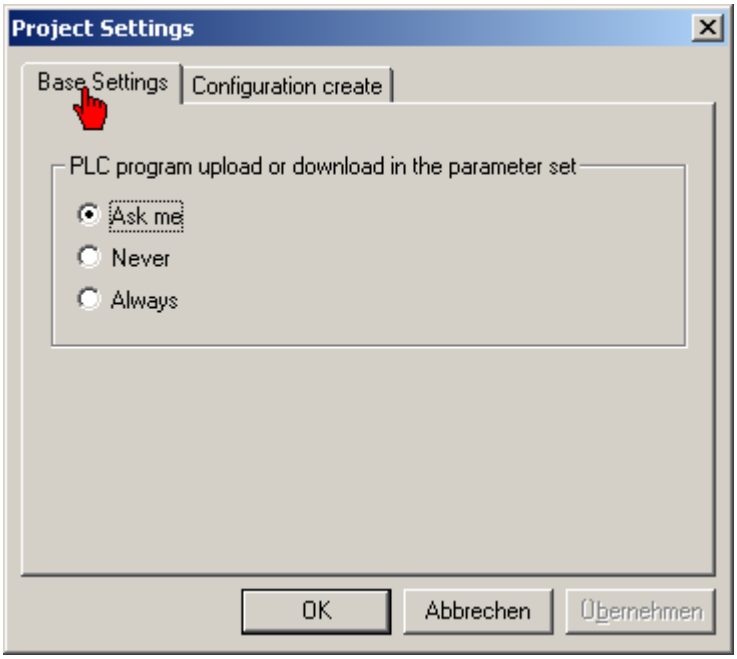
Expert mode

No new "bus configuration" is created under **CoDeSys** -> **Logon**.

Changes in the PLC program are ignored that would have a new bus configuration as a consequence.

4.3.2.1.2 Base Settings

'Extras' - 'Project Settings'

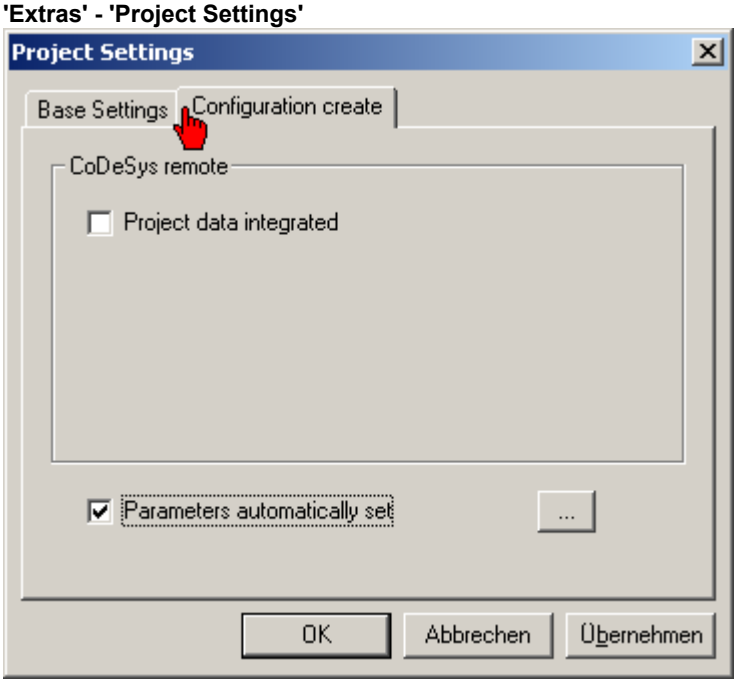


An existing PLC project will be stored inside ID34159 PLC project.

The **Project Settings** have affect to the menu points:
Online -> **Upload parameter set**
Online -> **Download project parameter set**

4.3.2.1.3 Configuration create (project specific)

'Extras' - 'Project Settings'



Project data is integrated
The CoDeSys projects are integrated in the AIPEX project file.
If this option is not set, the CoDeSys projects are saved in a separate directory next to the AIPEX project.

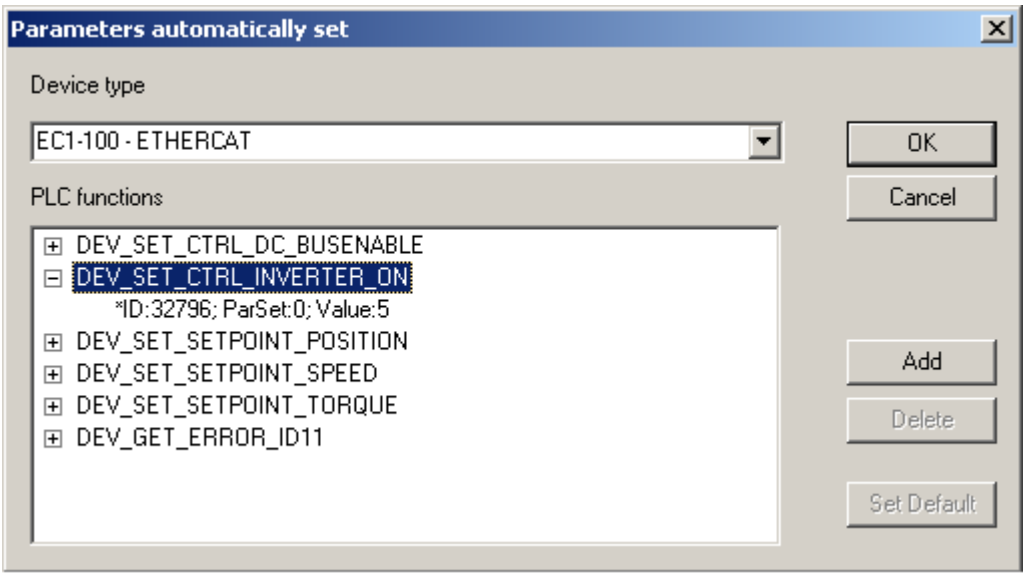
Parameter automatically set

The device parameters needed for the CoDeSys library functions are set automatically.

Example:

If the AMK function block 'SET_CTRL_INVERTER_ON_x_RF' is used in CoDeSys, AIPEX PRO sets the value of the ID32796 source controller enable to 5 (importance: Signal controller enable is set by the PLC).

The IDs and their significance are described in the parameter description.



If there is no data set for a concrete device type, the general ("other") data set is used.

Using the button **Add** you can assign own device parameters to the PLC functions.

Using the button **Delete** you can delete own device parameters.

Pressing the **Specification** button resets to the original AMK conditions.

4.3.2.2 Creating a PLC project

This section describes how to create a new PLC project.

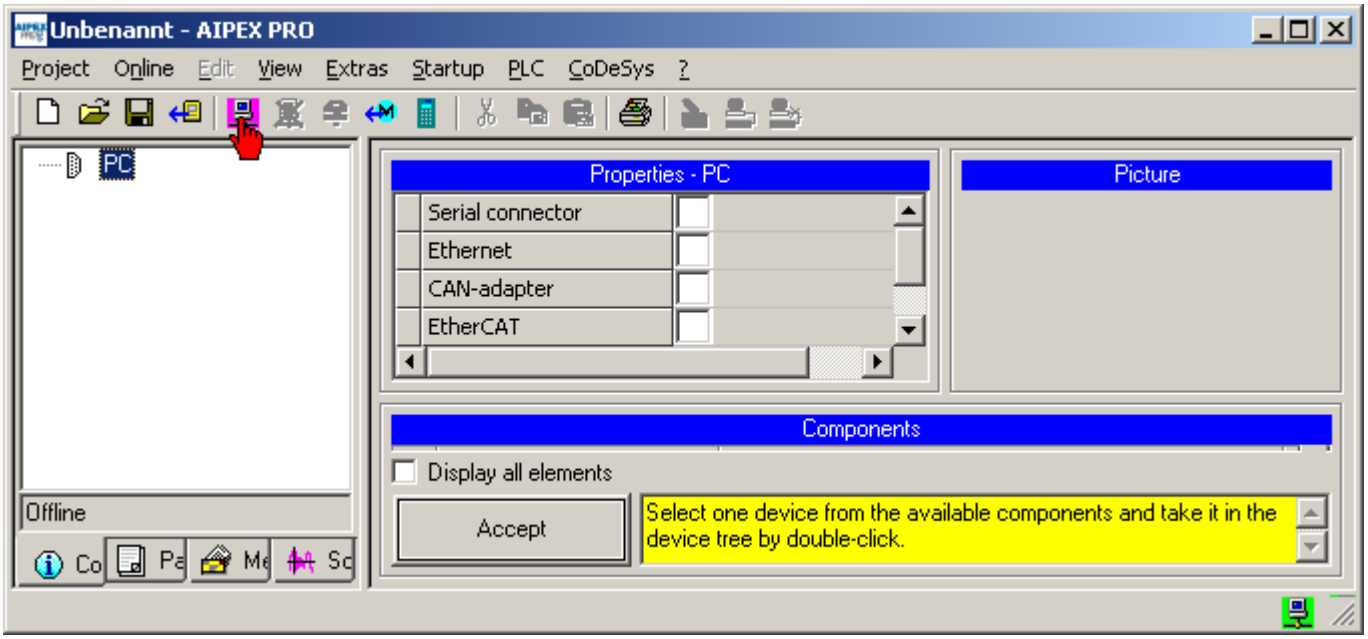
Requirement:

Icon: State 'green' (interface ready)

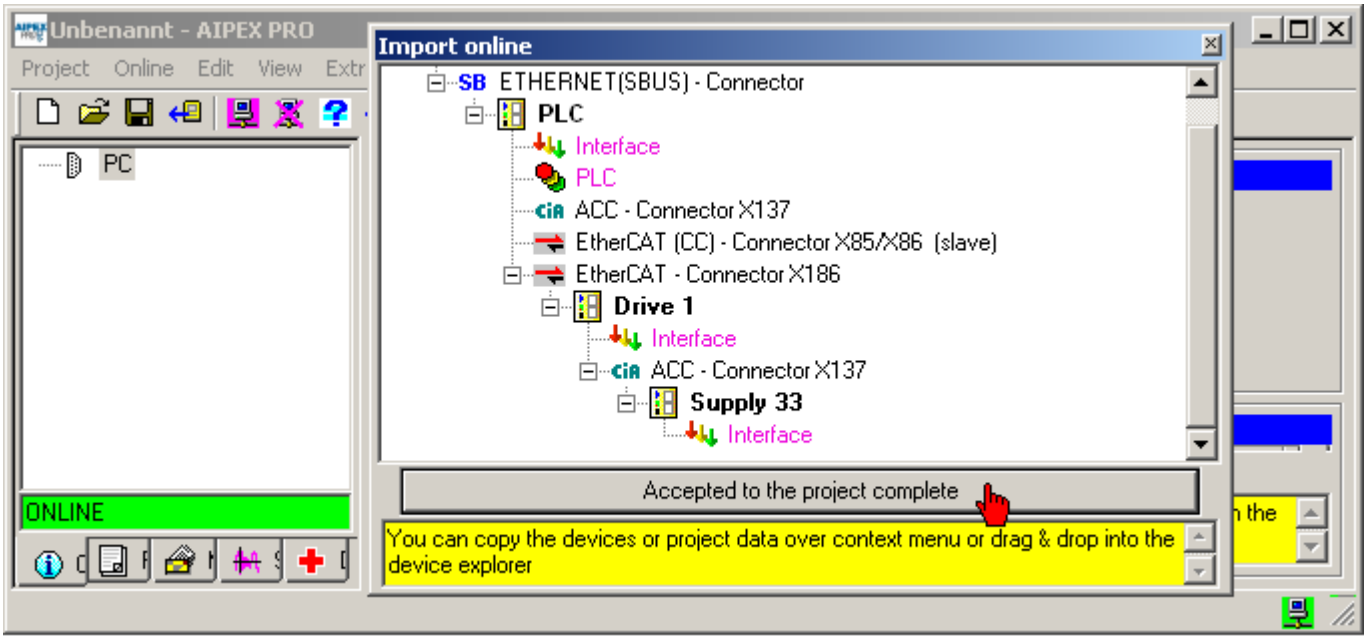
Icon: State 'red' or. no icon available, check next points:

- Initial startup fieldbus done: See documentation KE/KW Initial startup (AMK part no. 204539) chapter 'initial startup fieldbus' e.g. documentation iC/iX/idt5 Initial startup (AMK part no. 204737) chapter 'initial startup fieldbus'.
- Direct connection via Ethernet is running: See product description Controller A-Series (AMK part no. 202975) chapter 'direct connection via Ethernet'.

Press **'Login'** to create an online connection with the connected modules.

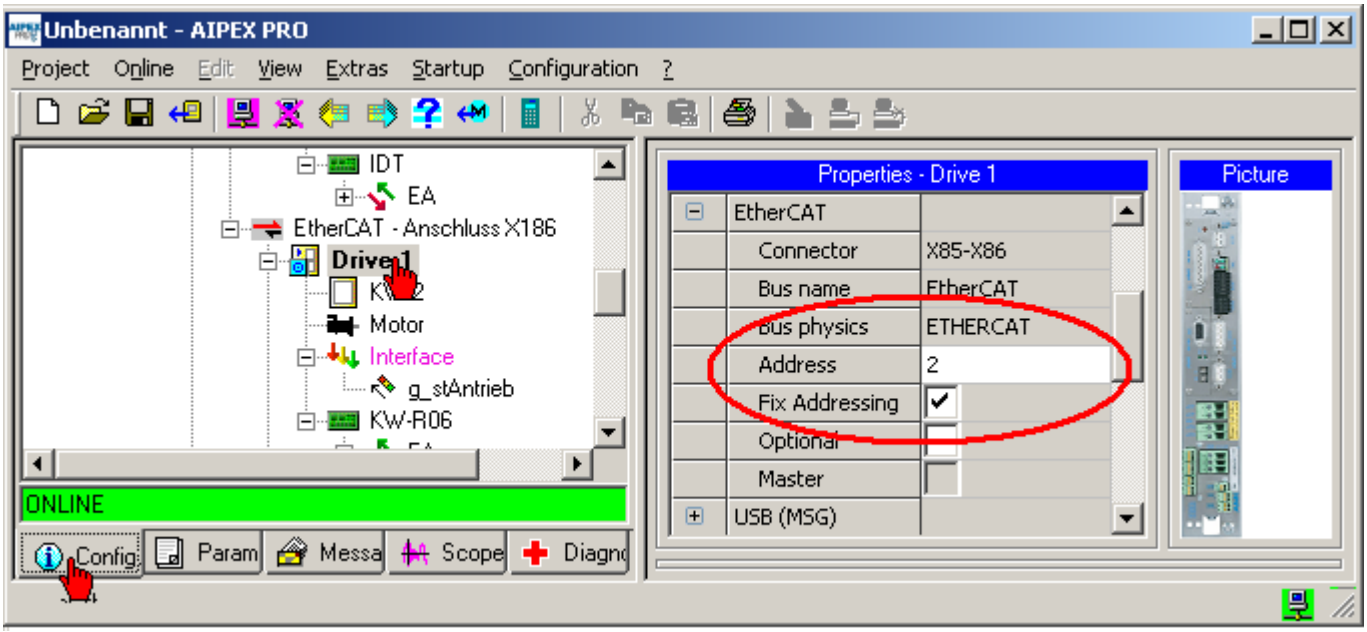


In the **'Import online'** window, check whether all physical devices were detected.
Press **'Accepted to the project complete'** to import the device data.

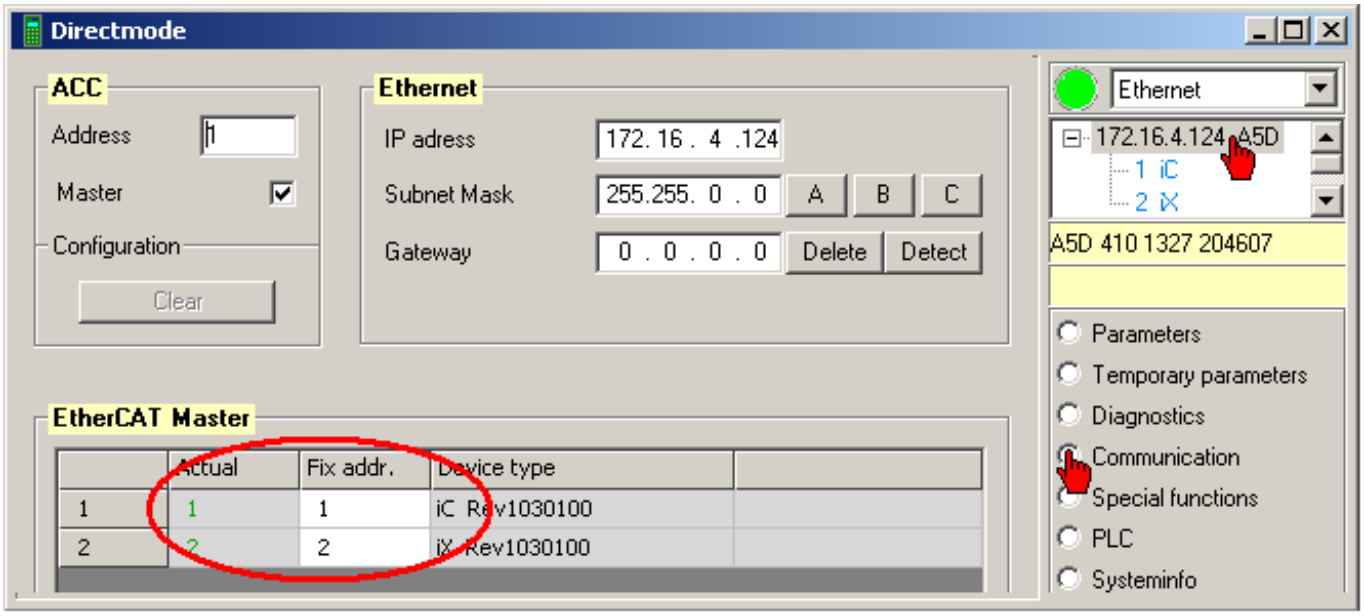


Check the wiring and the system statuses if devices are not shown.

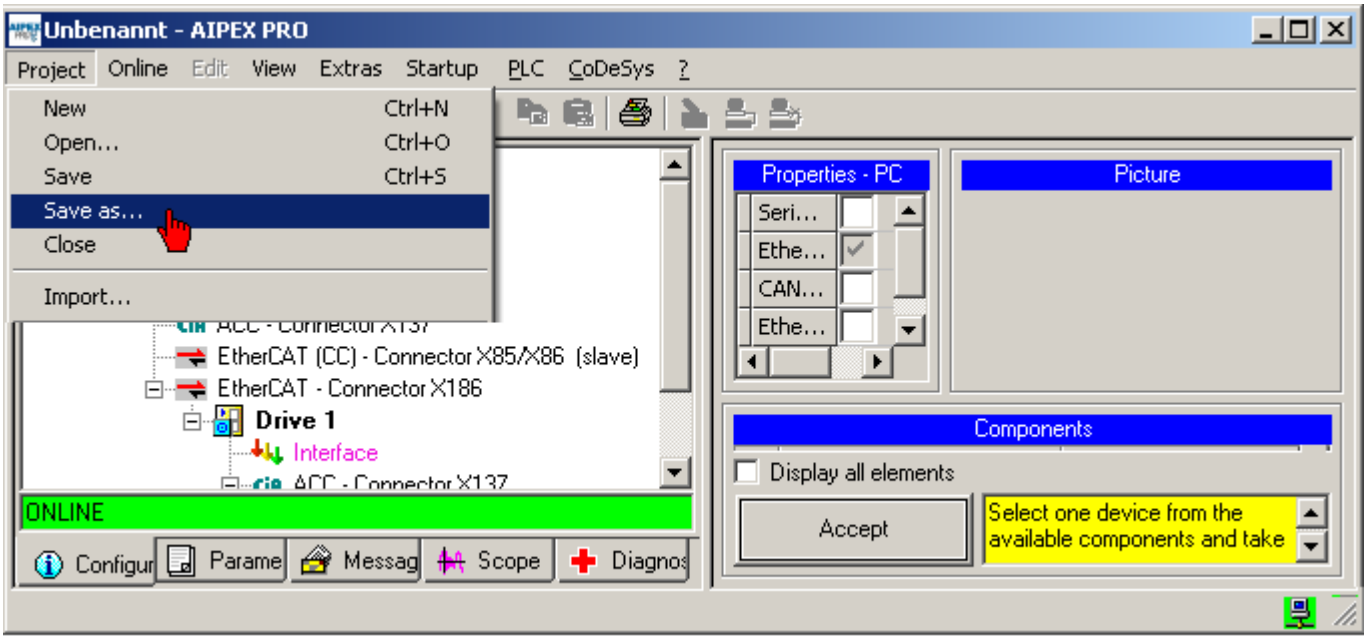
Open the tab **'Configuration'** to set fix addresses to each EtherCAT drive. Following you have to restart the system.



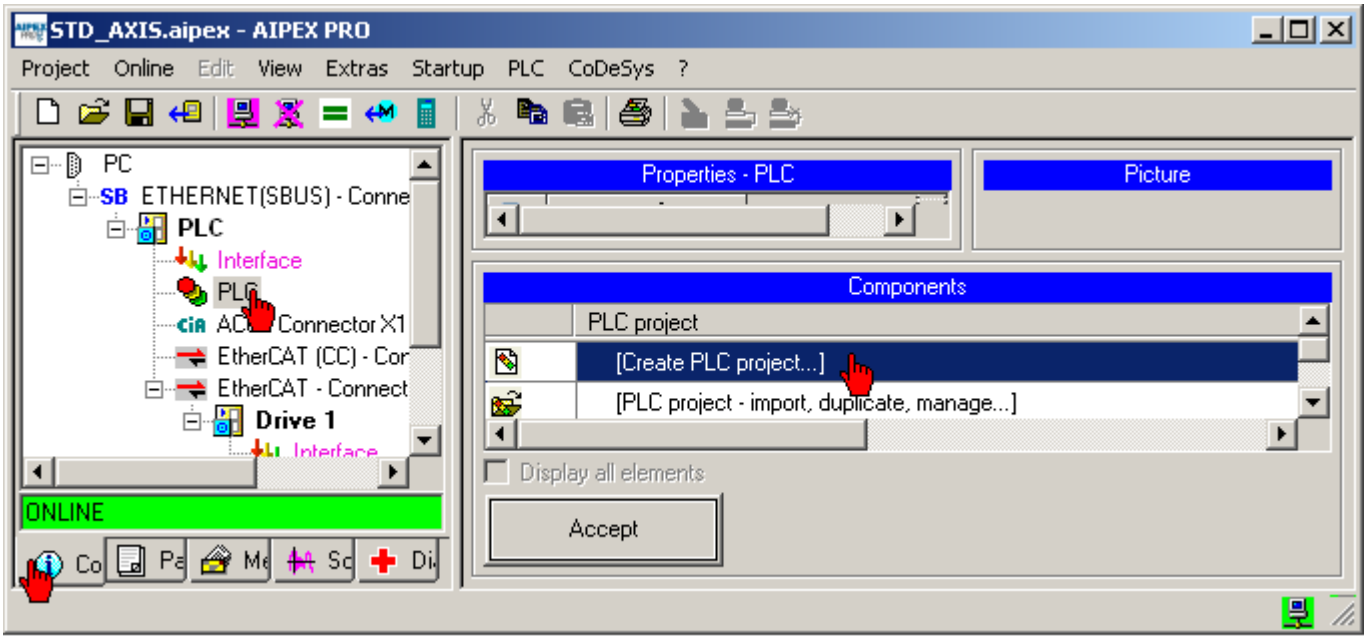
After restart, open the AIPEX PRO menu **'Directmode'** – **'Communication'** to check the actual position and the fix address of the EtherCAT slaves.



Save the imported project under **'Project' 'Save as...'**. You can assign a different name to the PLC project.

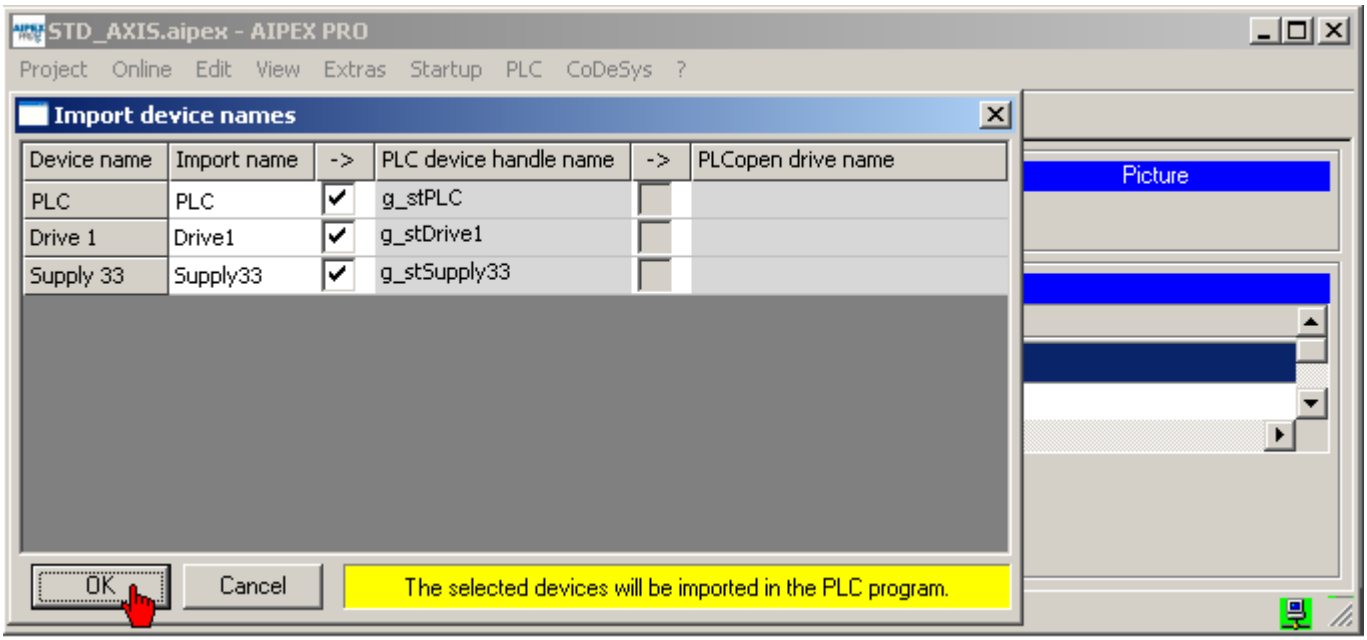


Select the **'PLC'** in the device tree. You can then launch CODESYS by selecting **'Create PLC project'**.



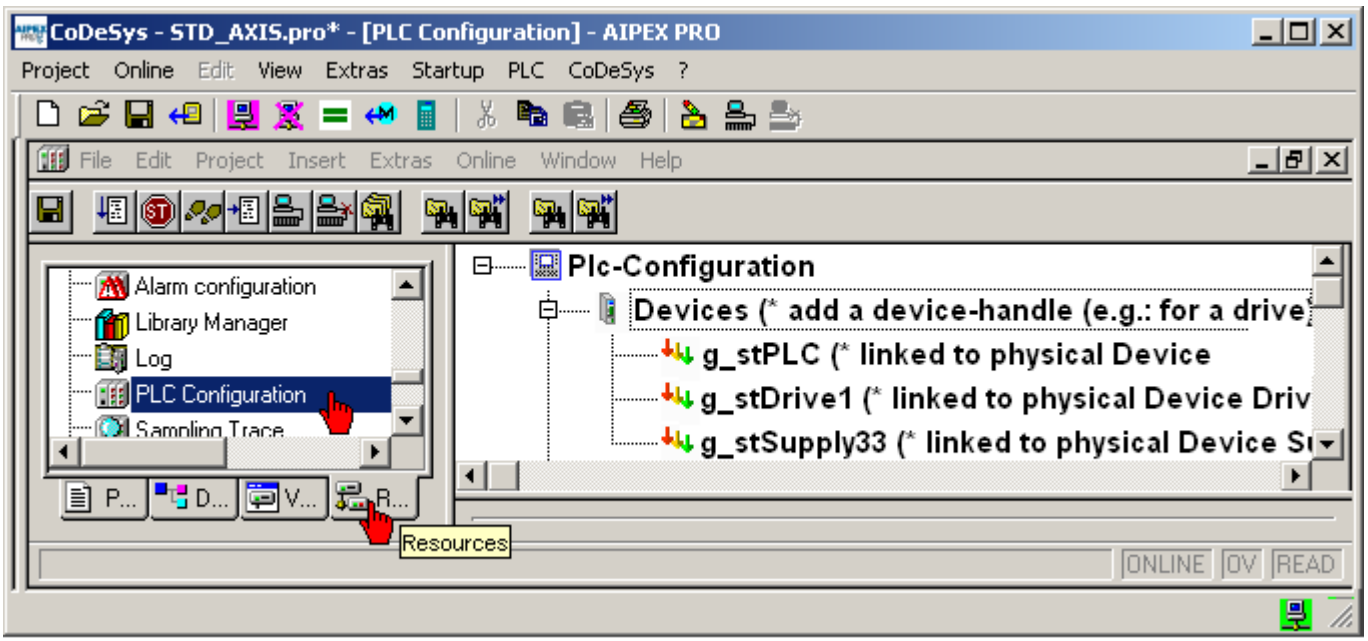
PLC missing:
Right-click in the device tree. You can customize the view under **'Select view'**.

The physical device names are automatically imported as symbolic device names (variables) into the PLC project on the **'Import device names'** window.



The **'Import device names'** window can be opened manually in the tab **'Configuration (PLC selected)' 'PLC properties'**.
The function can be enabled and disabled under **'Extras' 'Options' Configuration create'**.

The symbolic device names (variables) can be found under **'Resources' 'Controller configuration'**.

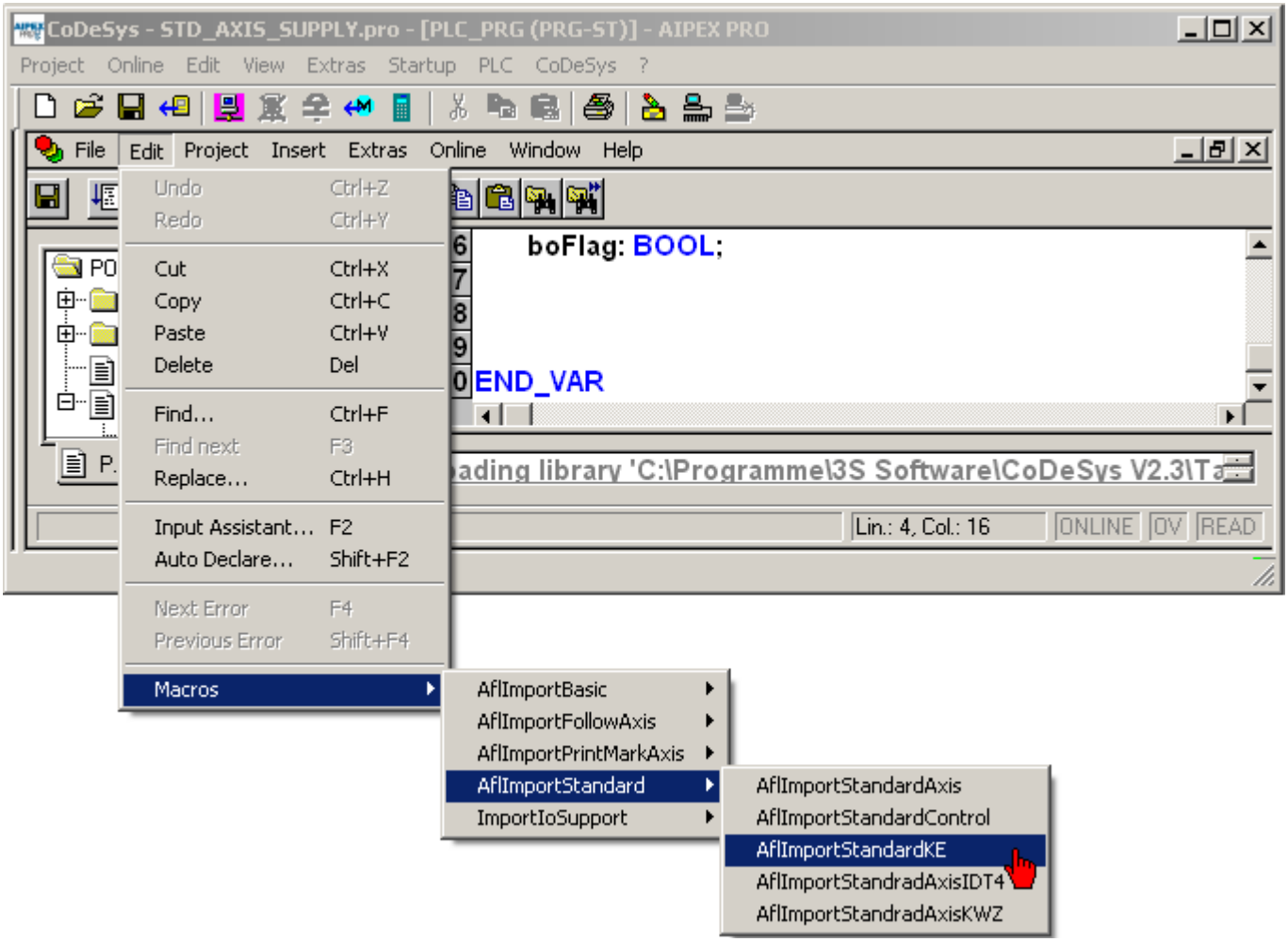


4.3.2.3 Importing a standard input

Description of standard input: see function block 'STANDARD_KE' starting on page 1

Only if you use power supply module KE/KEN/KES (exception KEN 05-xx).
The devices AMKASMART iC und AMKASYN KEN 05-xx charge automatically the DC-Bus to supply the connected inverters.

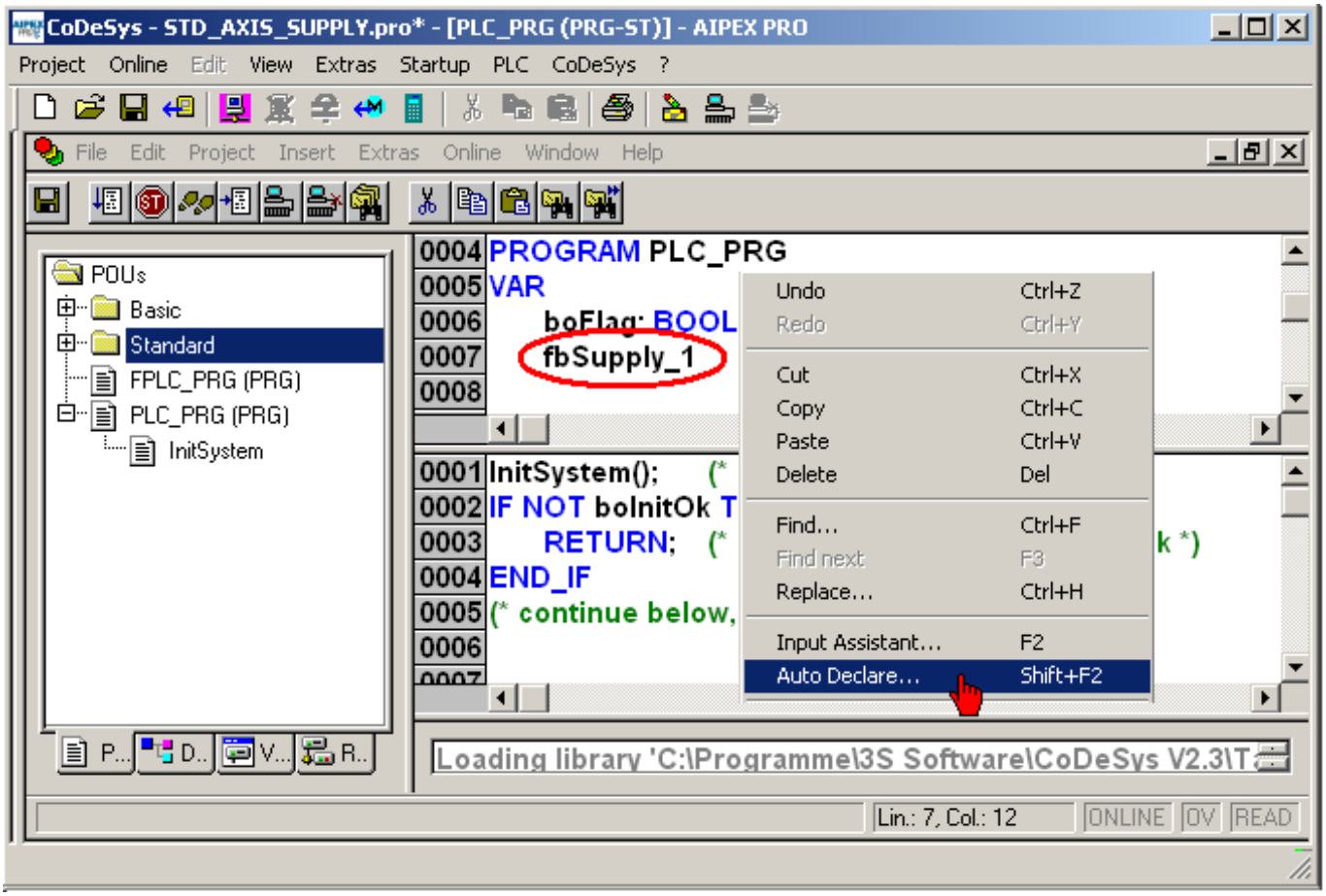
Import the function **'AflImportStandardKE'** via the menu **'Edit' 'Macros' 'AflImportStandard'**.



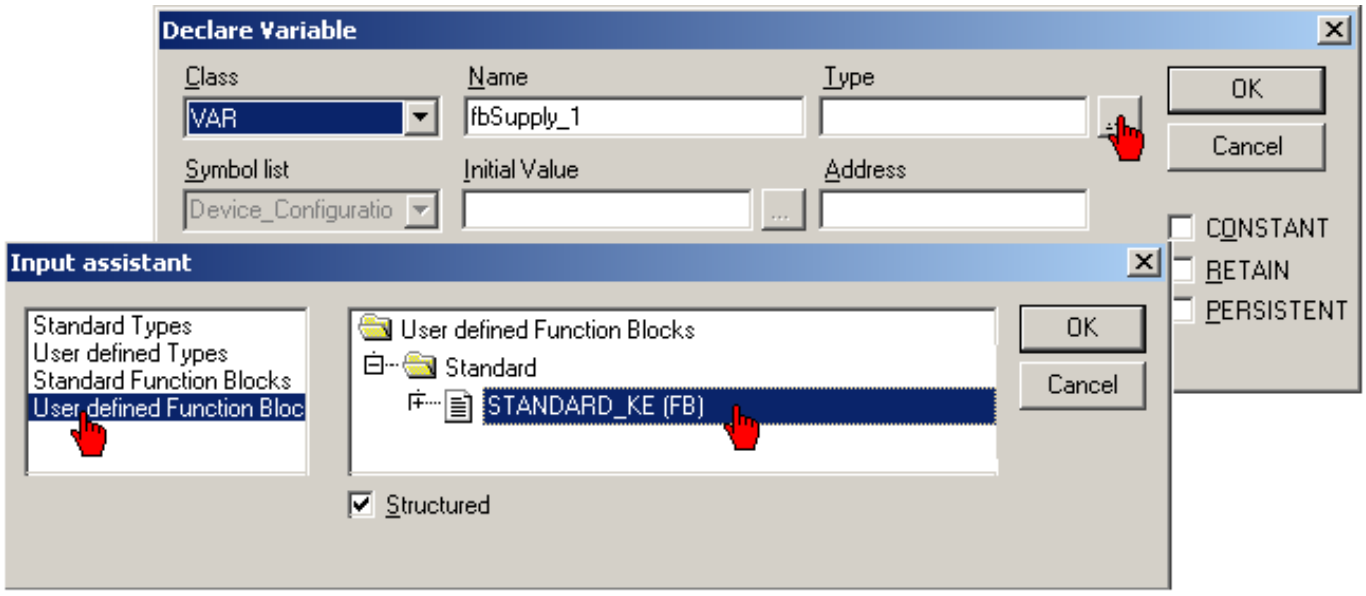
4.3.2.4 Instancing a standard input

Only if you use power supply module KE/KEN/KES (exception KEN 05-xx).
The devices AMKASMART iC und AMKASYN KEN 05-xx charge automatically the DC-Bus to supply the connected inverters.

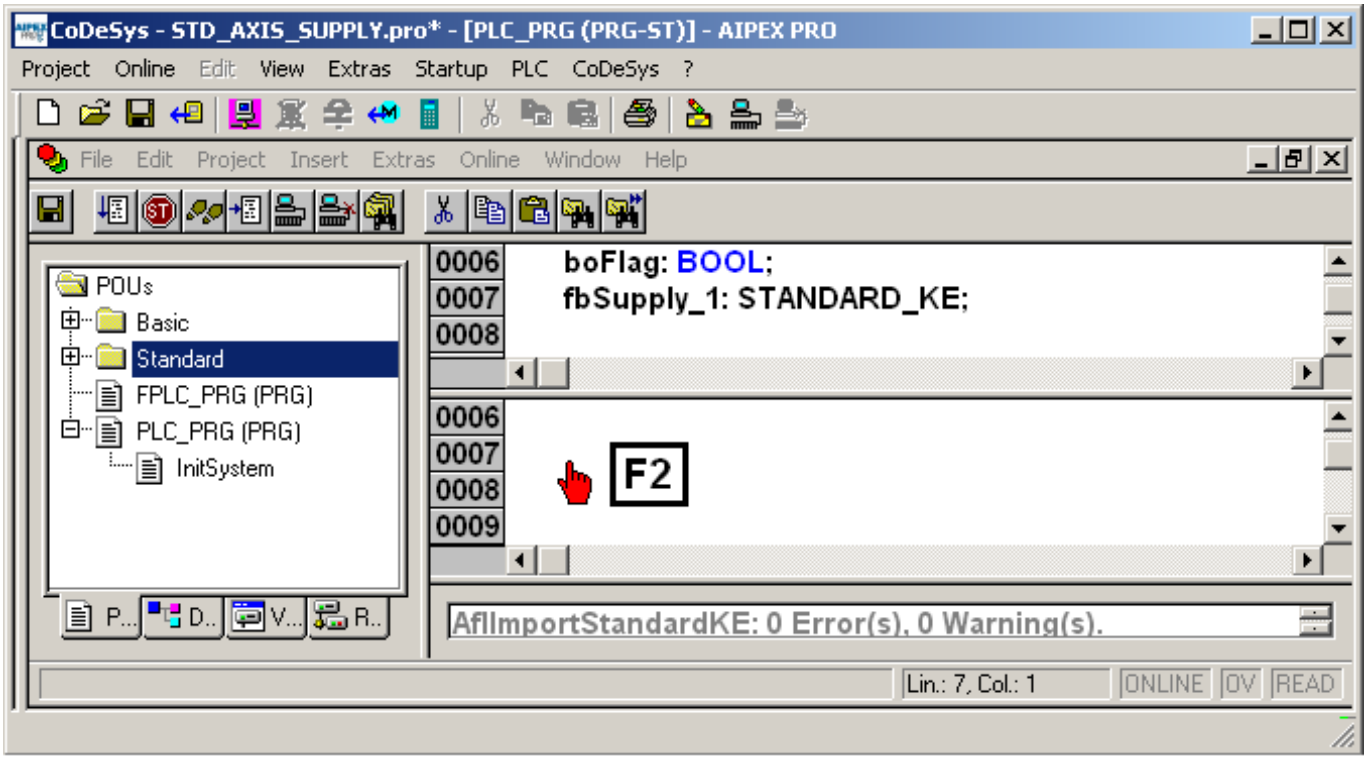
In the PLC_PRG Declare Variable editor, create an instance of type 'STANDARD_KE' (in this example, fbSupply_1) for each physical axis.
Right-click on the variable. You can then open the **'Declare Variable' window** (CTRL + F2).



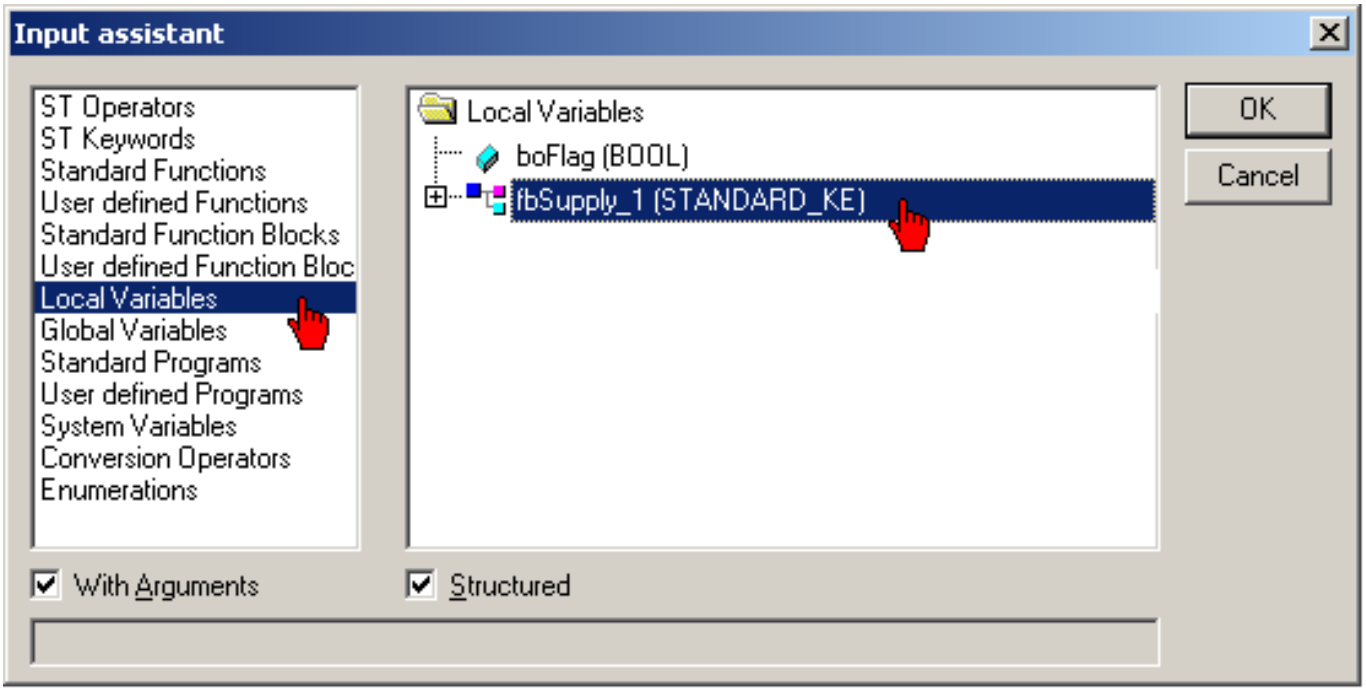
The function block 'STANDARD_KE' can be found in the Input assistant under **'User defined Function Blocks' 'Standard'**.



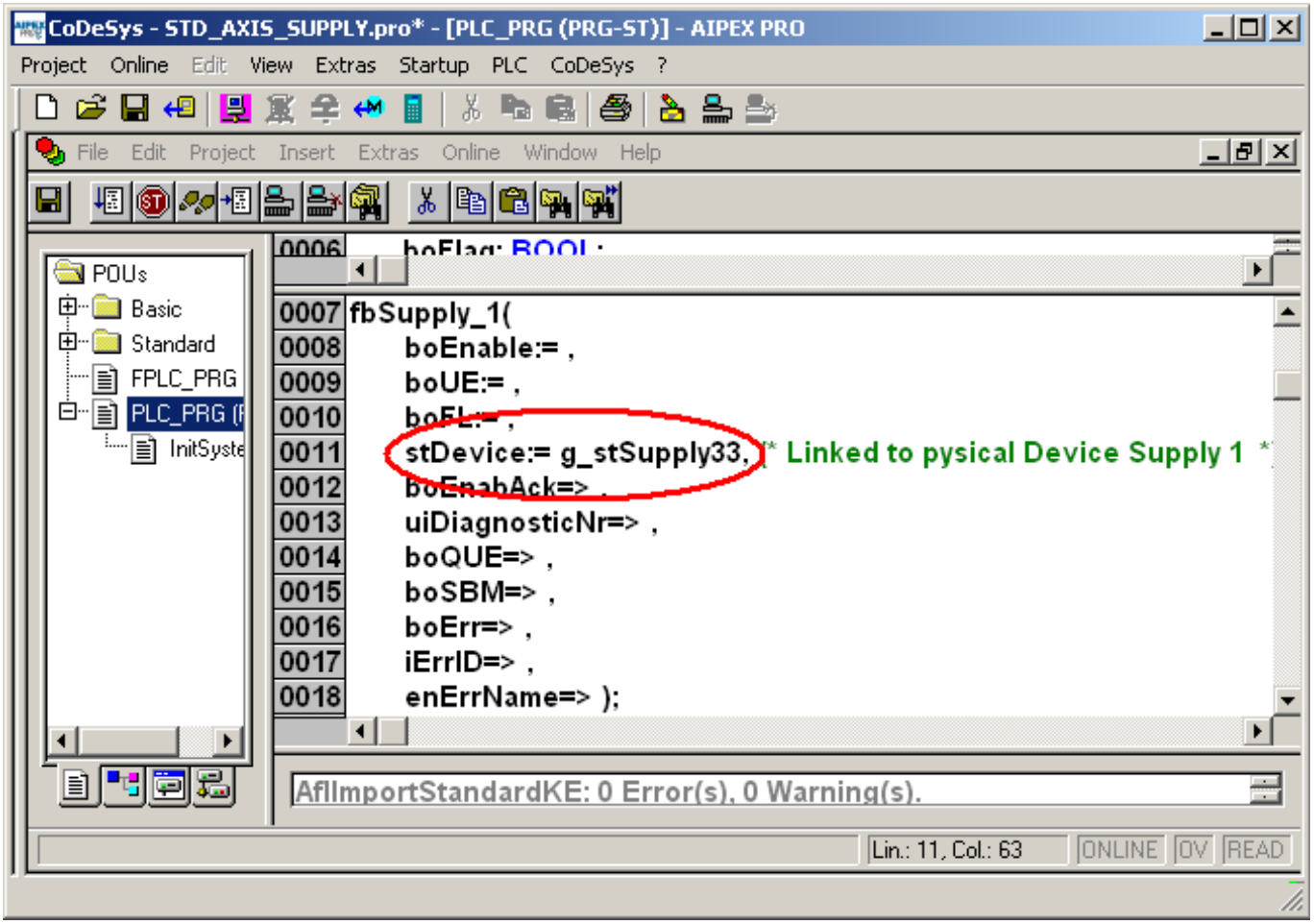
Add its function block ' fbSupply_1' in the program editor.
To do so, click the program editor. Press **'F2'** to open the **'Input assistant**.



The function block **'fbSupply_1'** can be found in the Input assistant under **'Local Variables'**.



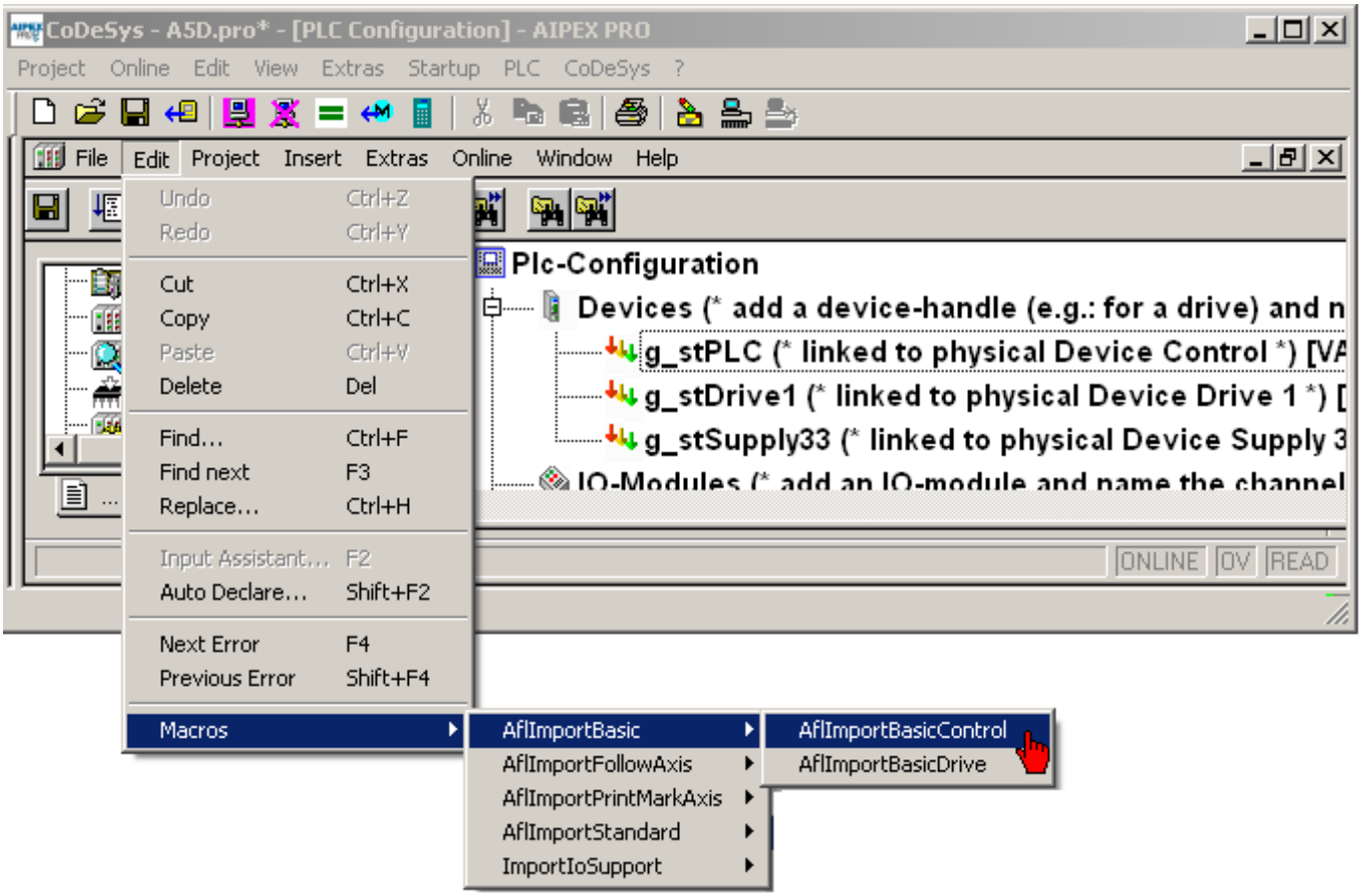
The IN_OUT variable 'stDevice' must always be assigned.



4.3.2.5 Importing a basic control

'AflImportBasicControl' only needs to be added once for each project.

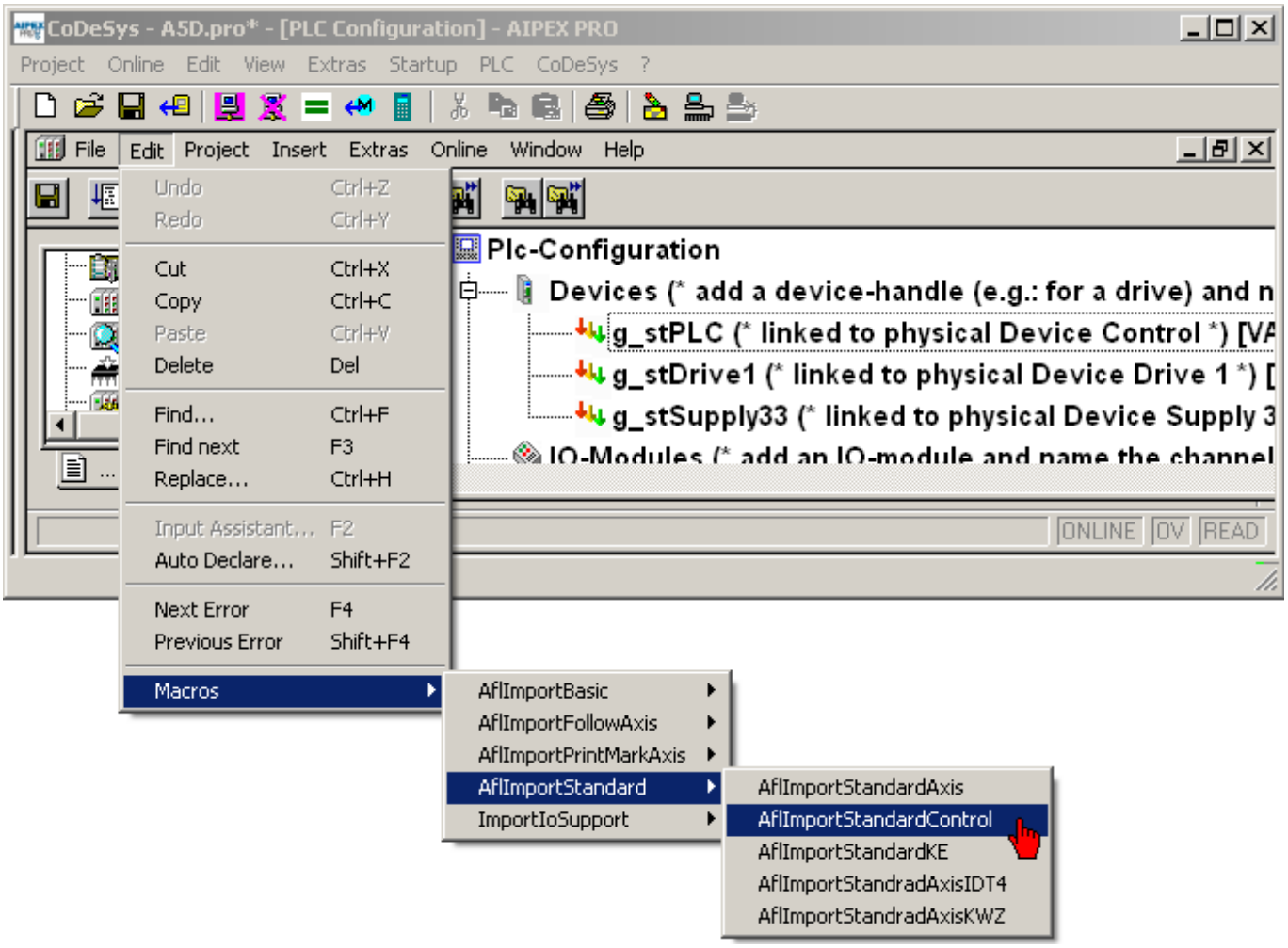
Import the function **'AflImportBasicControl'** via the menu **'Edit' 'Macros' 'AflImportBasic'**.



4.3.2.6 Importing a standard control

Description of standard control: Siehe 'STANDARD_CONTROL (FB)' auf Seite 195.

Import the function **'AflImportStandardAxis'** via the menu **'Edit' 'Macros' 'AflImportStandard'**.

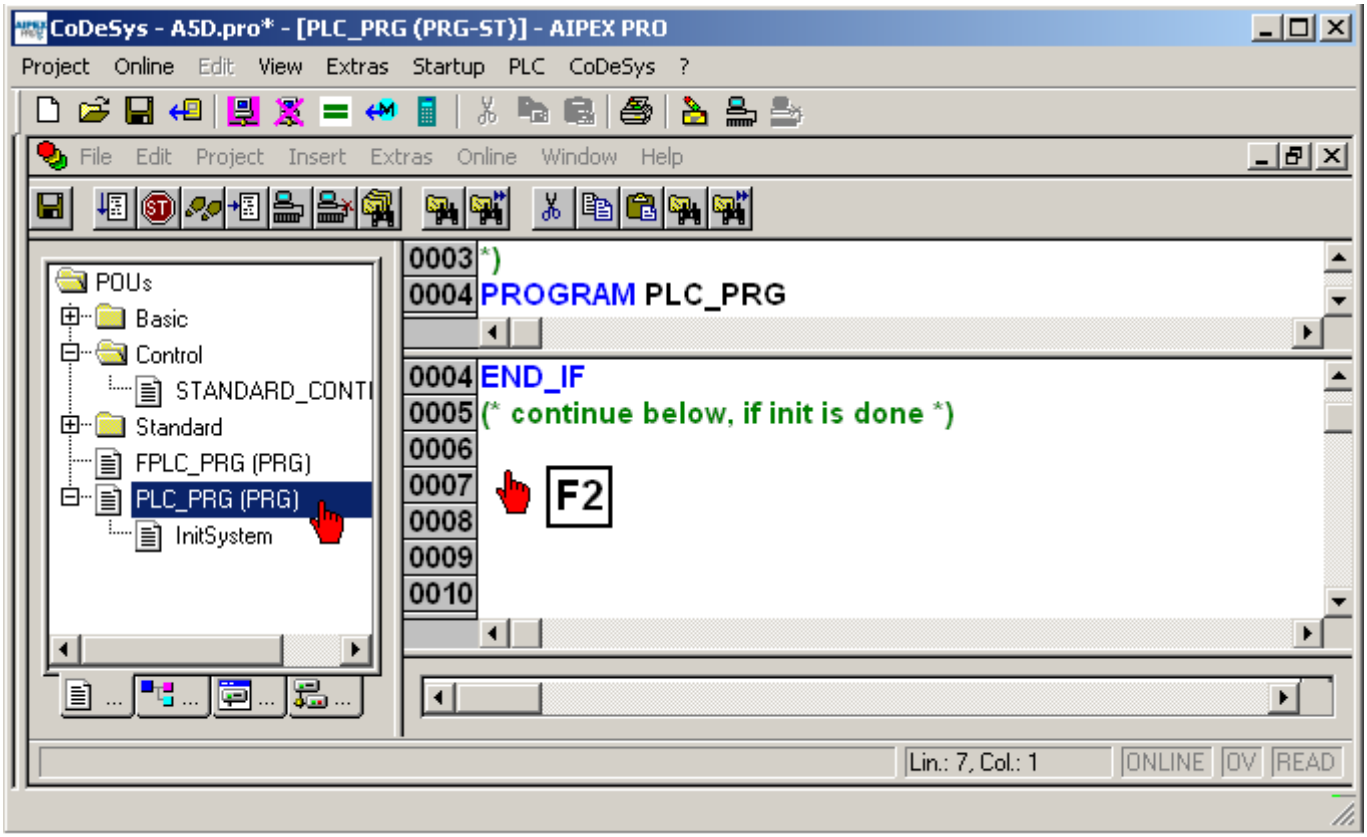


4.3.2.7 Instancing the Standard controller

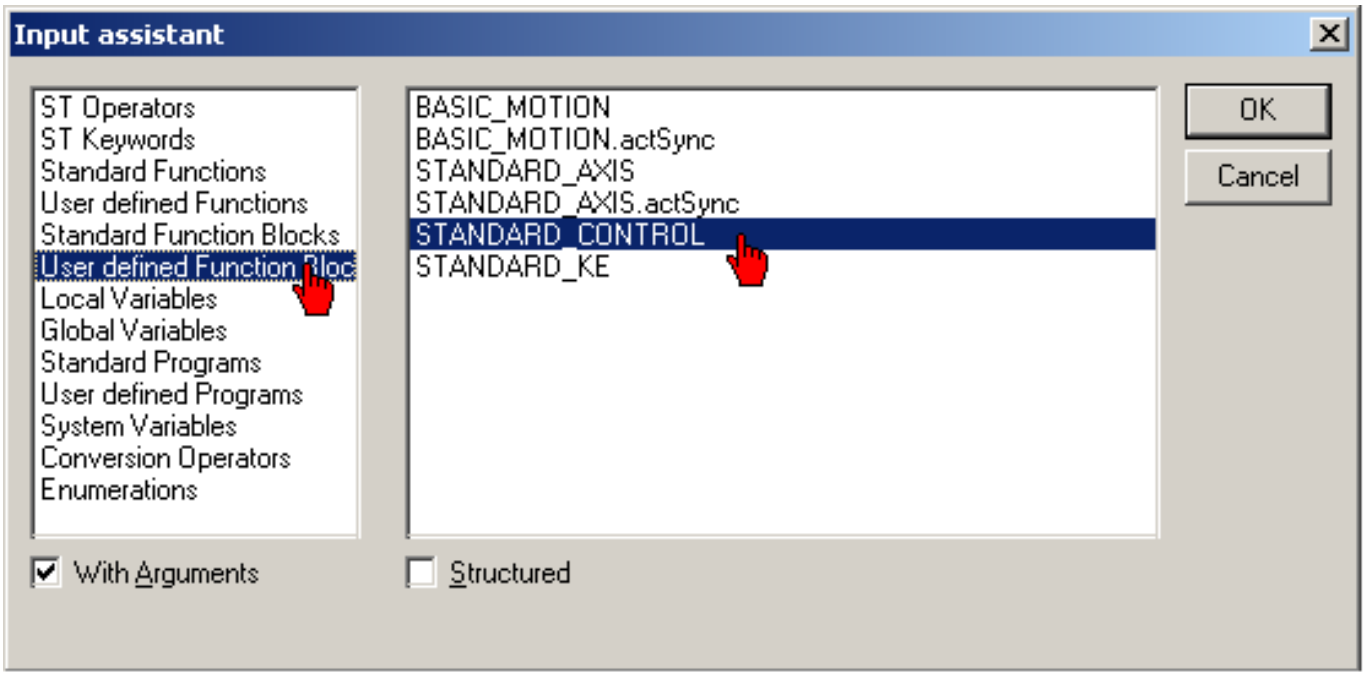
(instancing of blocks - version 1)

Add the function block 'STANDARD_CONTROL' in the program editor.

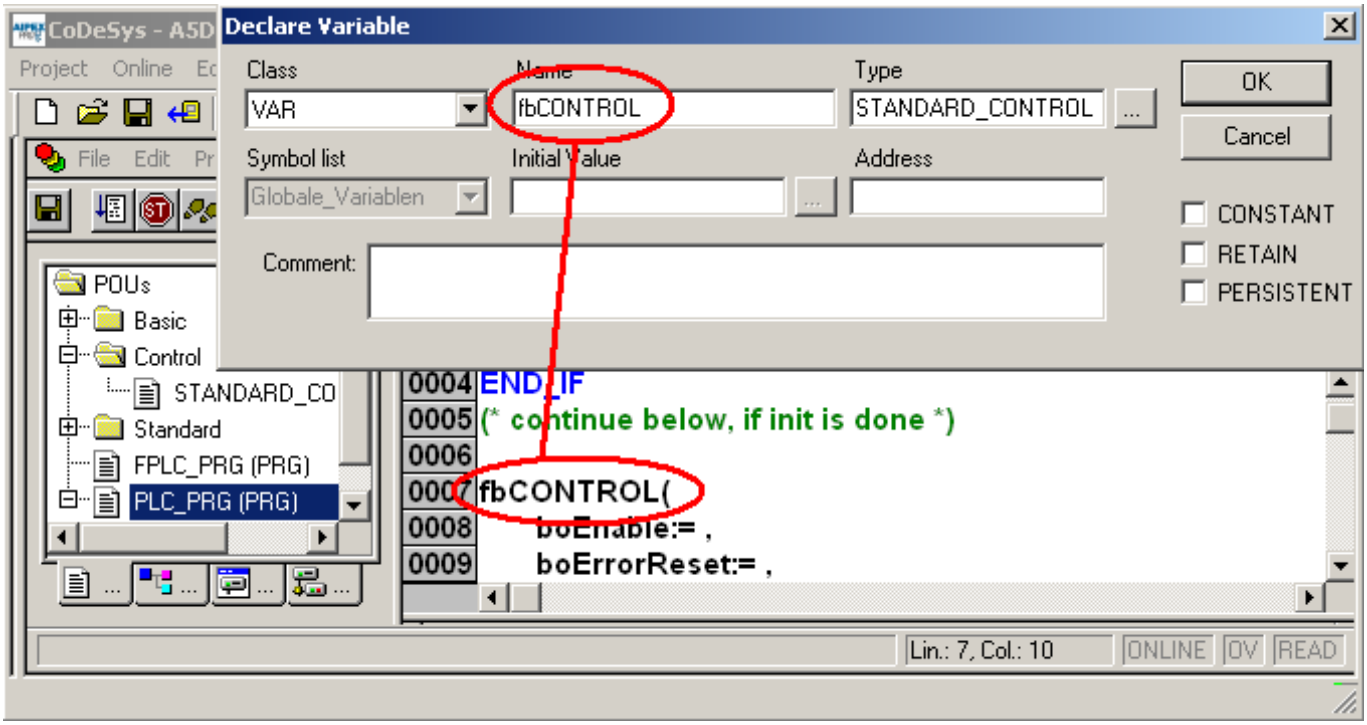
To do so, click the program editor. Press **'F2'** to open the **'Input assistant**.



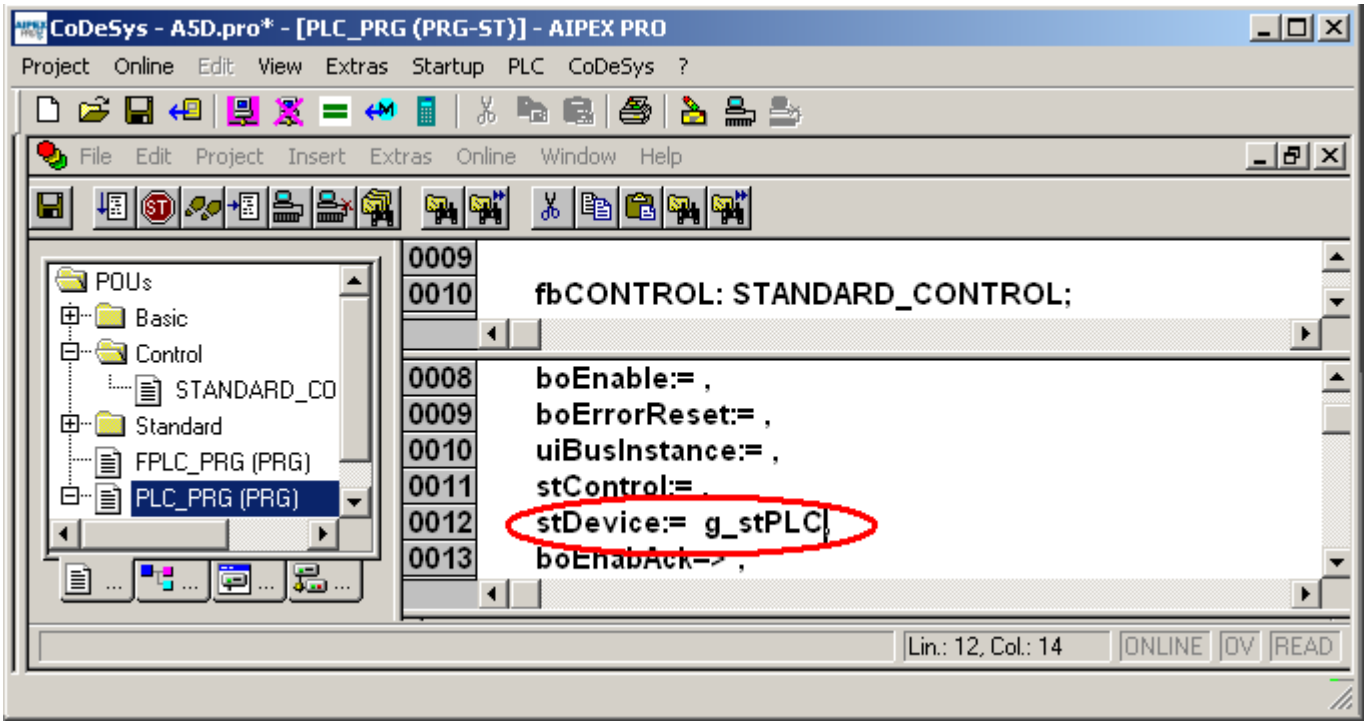
The function block 'STANDARD_CONTROL' can be found in the Input assistant under **'User defined Function Blocks'**.



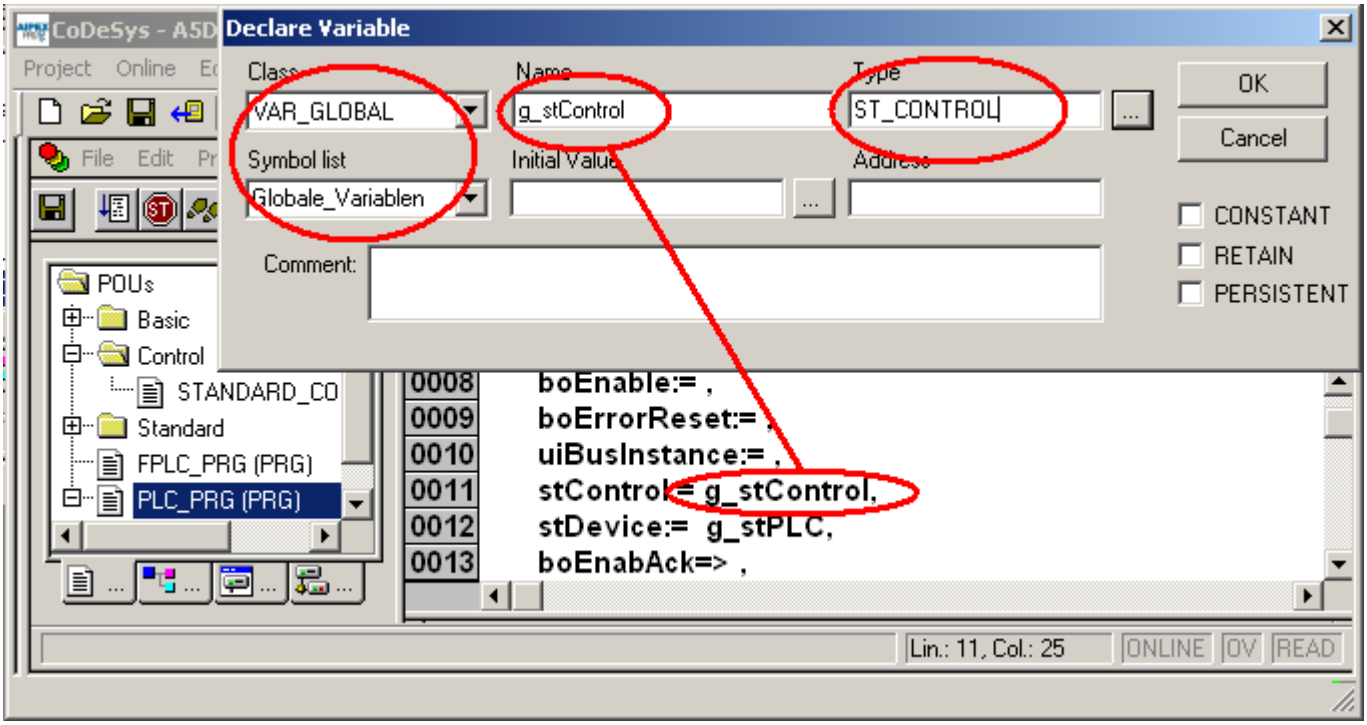
Create an instance (copy) of **FB** 'STANDARD_CONTROL' by assigning it a separate name. In this example: fbControl.
Click on the program editor to open the **'Declare Variables' window**.
Type = Original block name 'STANDARD_CONTROL'



The IN_OUT variables 'stControl' and 'stDevice' must always be assigned.
stDevice:= Link to symbolic device name (in this example: g_stPLC)



Create a global structure variable. In this example: g_stControl.
Click on the program editor to open the **'Declare Variables' window**.
Class = VAR_GLOBAL
Symbol list = Globale_Variablen
Type = Structure 'ST_CONTROL'

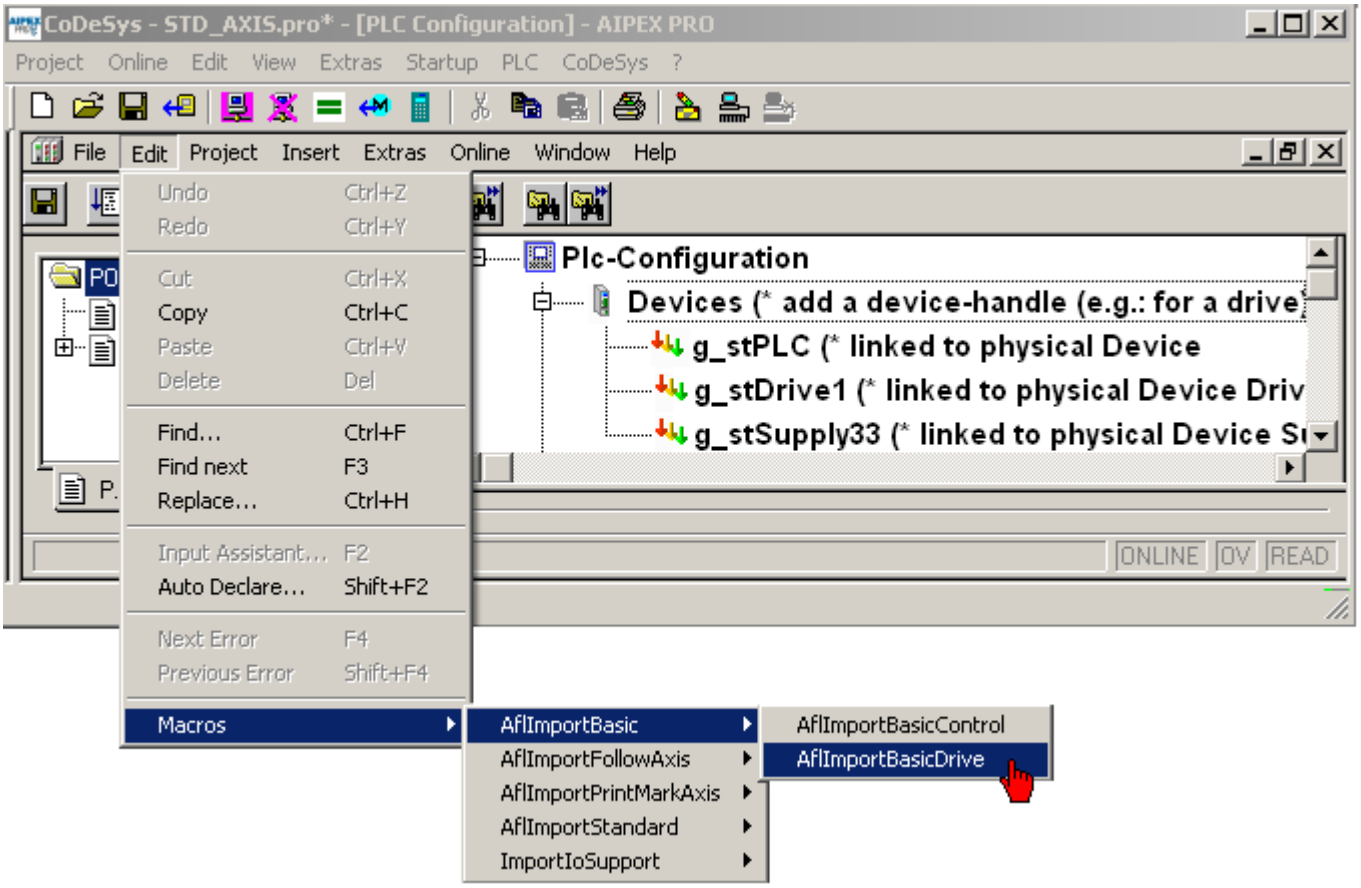


4.3.2.8 Importing a basic drive

Description of the basic drive: Siehe 'Description of basic drive' auf Seite 753.

'AflImportBasicDrive' only needs to be added once for each project.

Import the function **'AflImportBasicDrive'** via the menu **'Edit' 'Macros' 'AflImportBasic'**.



4.3.2.8.1 Description of basic drive

The basic drive must be added once to each PLC project.
It forms the basis for all subsequent Standard, print and following drives.

FB BASIC_MOTION is added with the basic drive.

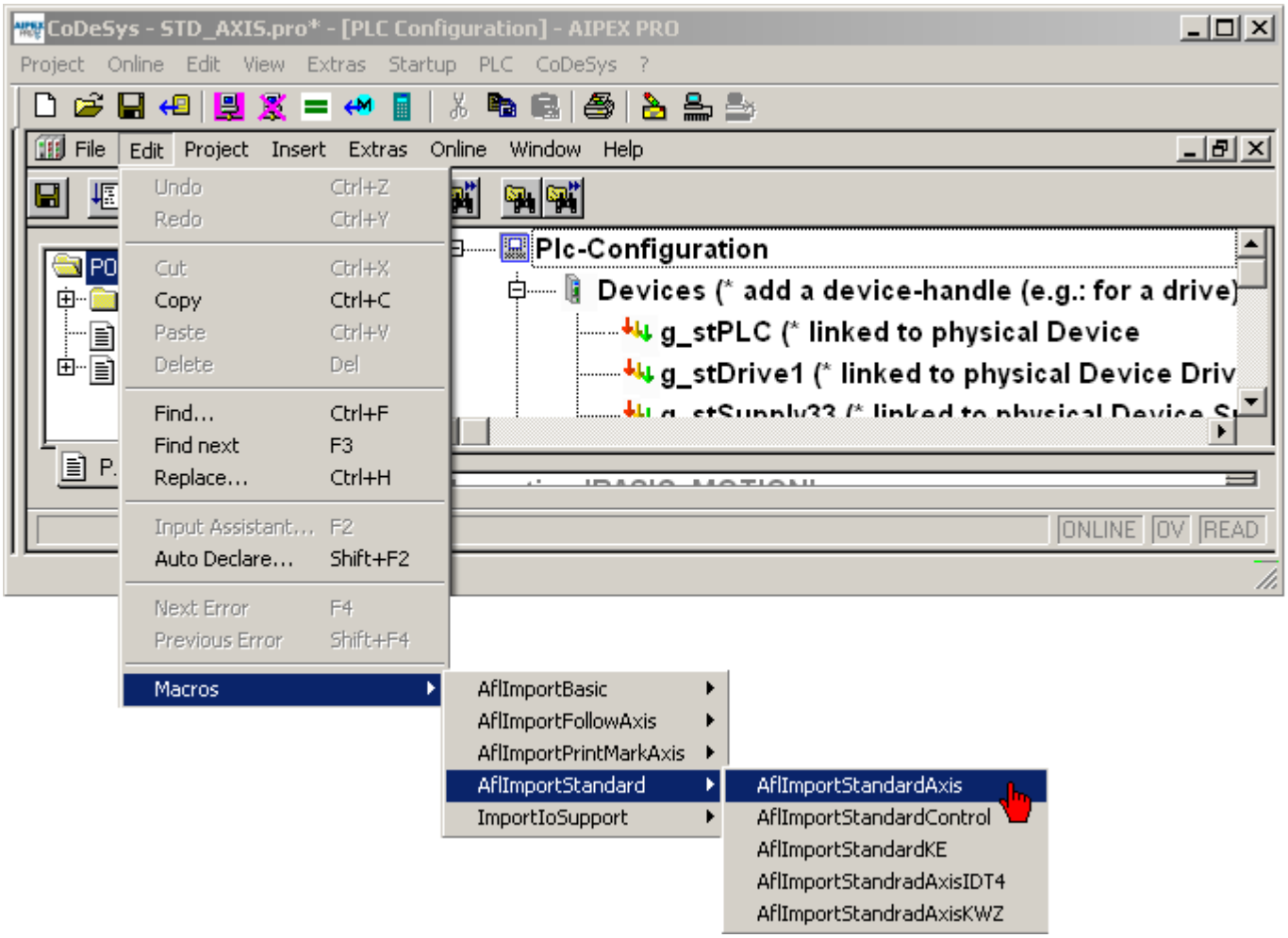
FB BASIC_MOTION functions

- Homing cycle
- Absolute positioning
- Relative positioning
- Speed control
- Jog mode

4.3.2.9 Declaring an axis structure

Description of standard axis: Siehe 'STANDARD_AXIS (FB) (STANDARD_AXIS_KWZ, STANDARD_AXIS_IDT4, STANDARD_AXIS_ihX)' auf Seite 201.

Import the function **'AflImportStandardAxis'** via the menu **'Edit' 'Macros' 'AflImportStandard'**.



4.3.2.10 Declaring an axis structure

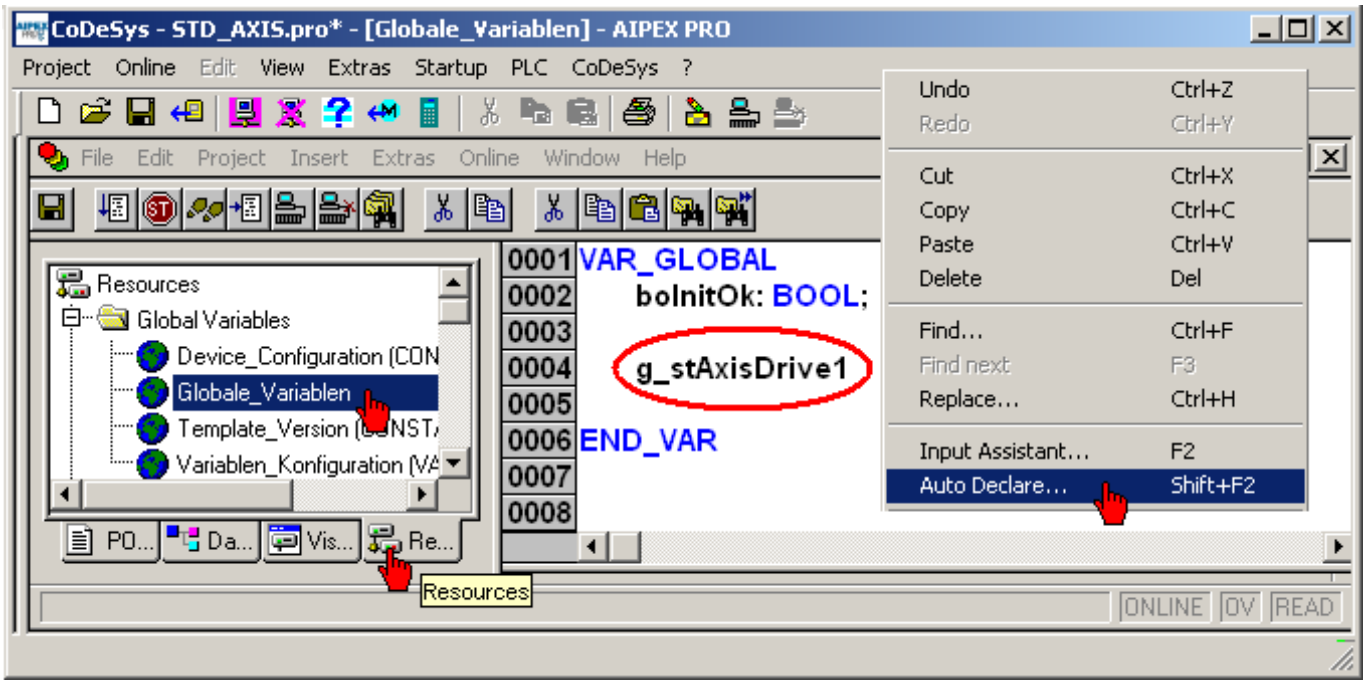
Description of the axis structure: Siehe 'ST_AXIS_DRIVE (ST)' auf Seite 191.

Create a global structure variable of type 'ST_AXIS_DRIVE' (in this example, g_stAxisDrive1) for each physical axis in the Declare Variable editor.

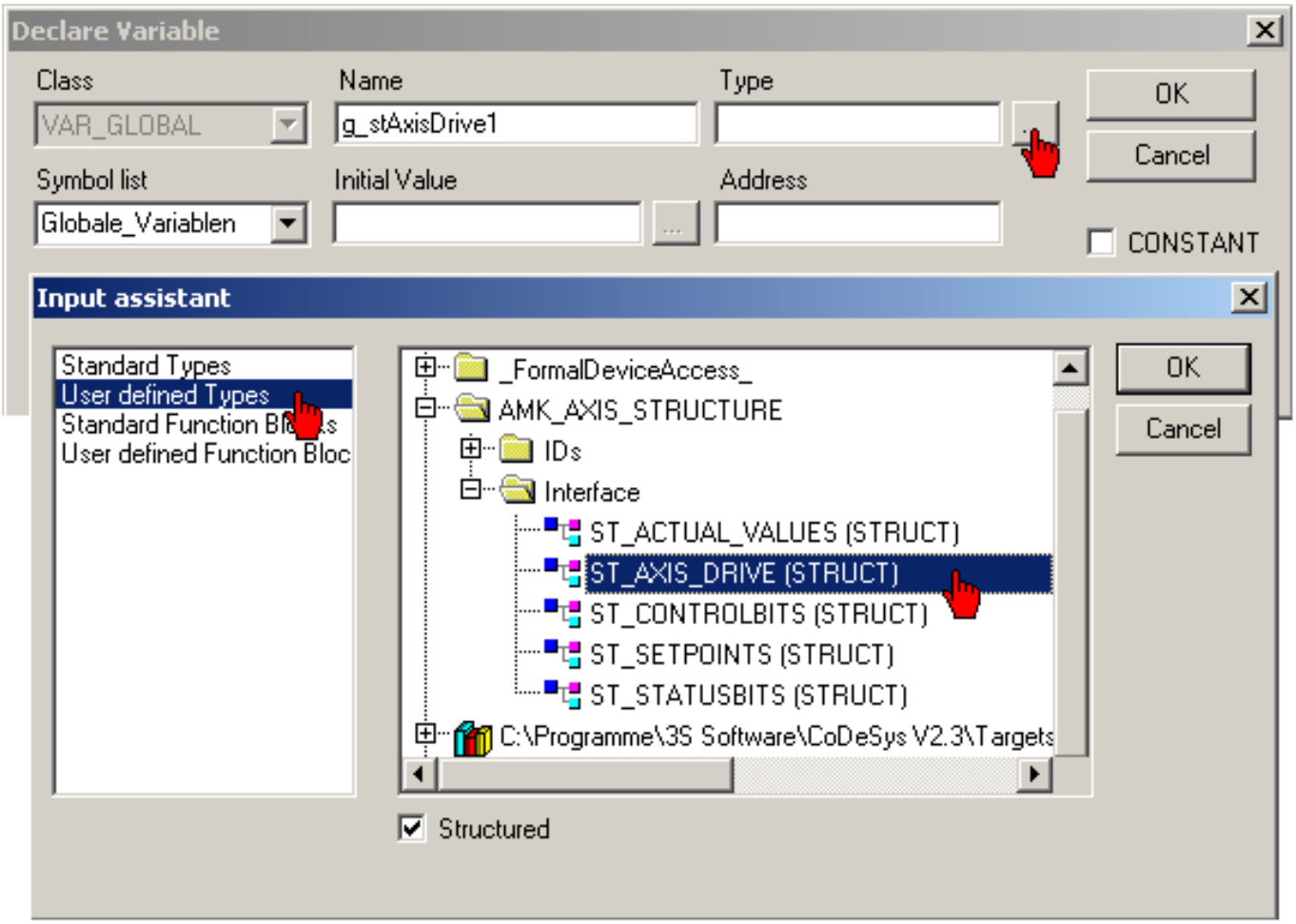
The Declare Variable editor for global variables can be found under **'Resources' 'Global Variables' 'Global_Variables'**.

Right-click on the structure variable. You can then open the **'Declare Variable' window** (CTRL + F2).

The declared structure variable allows you to access all axis-specific data. Siehe 'ST_AXIS_DRIVE (ST)' auf Seite 191.



You can find the structure 'ST_AXIS_DRIVE' in the Help Manager under **'Defined Types' 'AMK_AXIS_STRUCTURE' 'Interface'.**

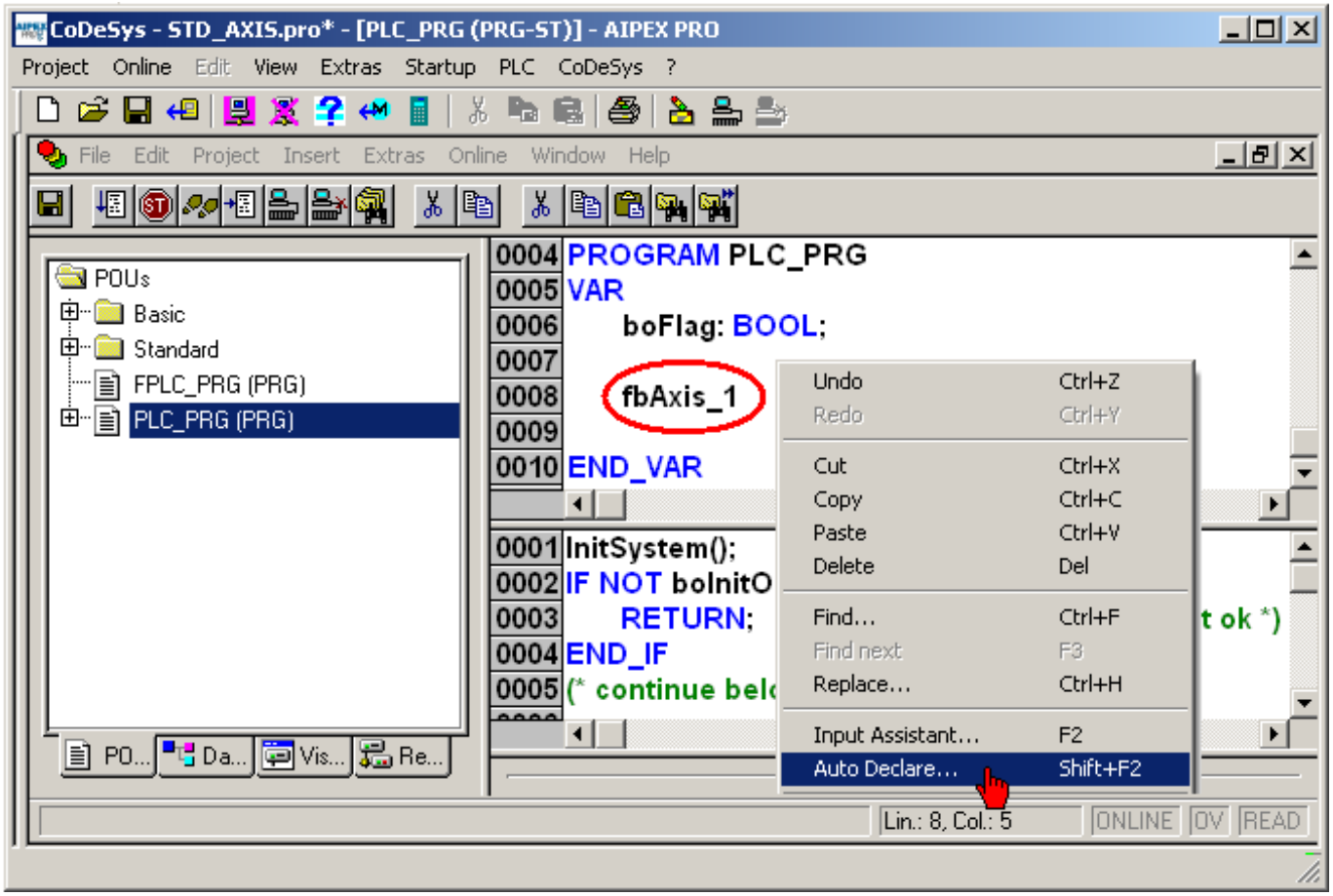


4.3.2.11 Instancing a standard axis

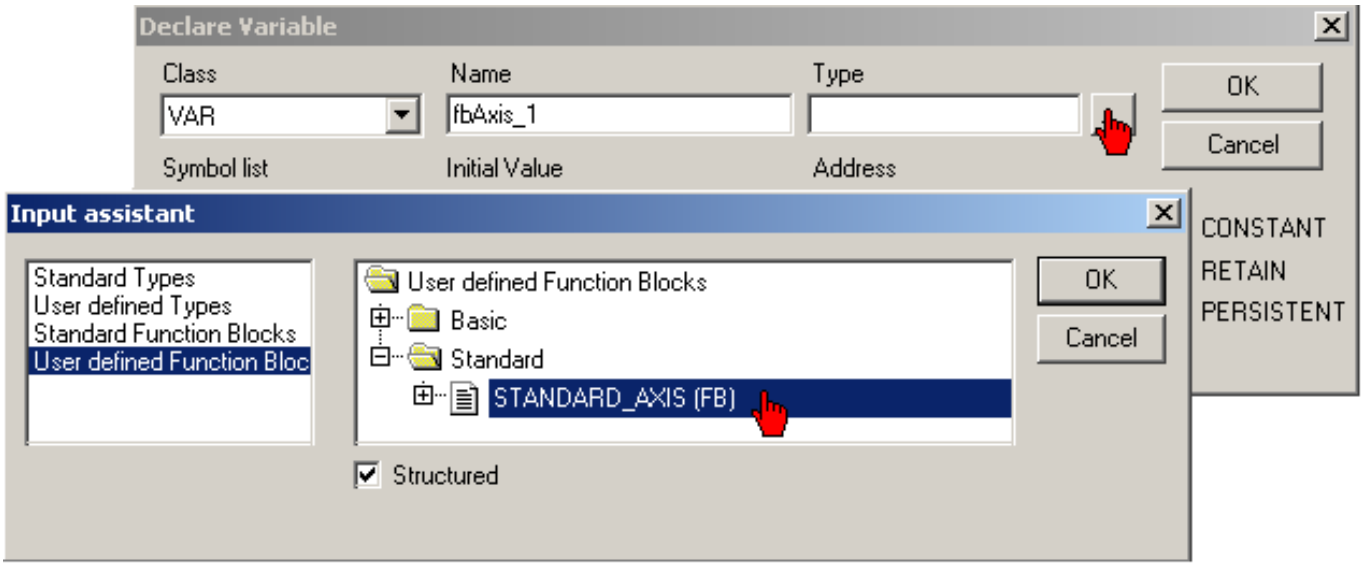
(instancing of blocks - version 2)

In the PLC_PRG Declare Variable editor, create an instance of type 'STANDARD_AXIS' (in this example, fbAxis_1) for each physical axis.

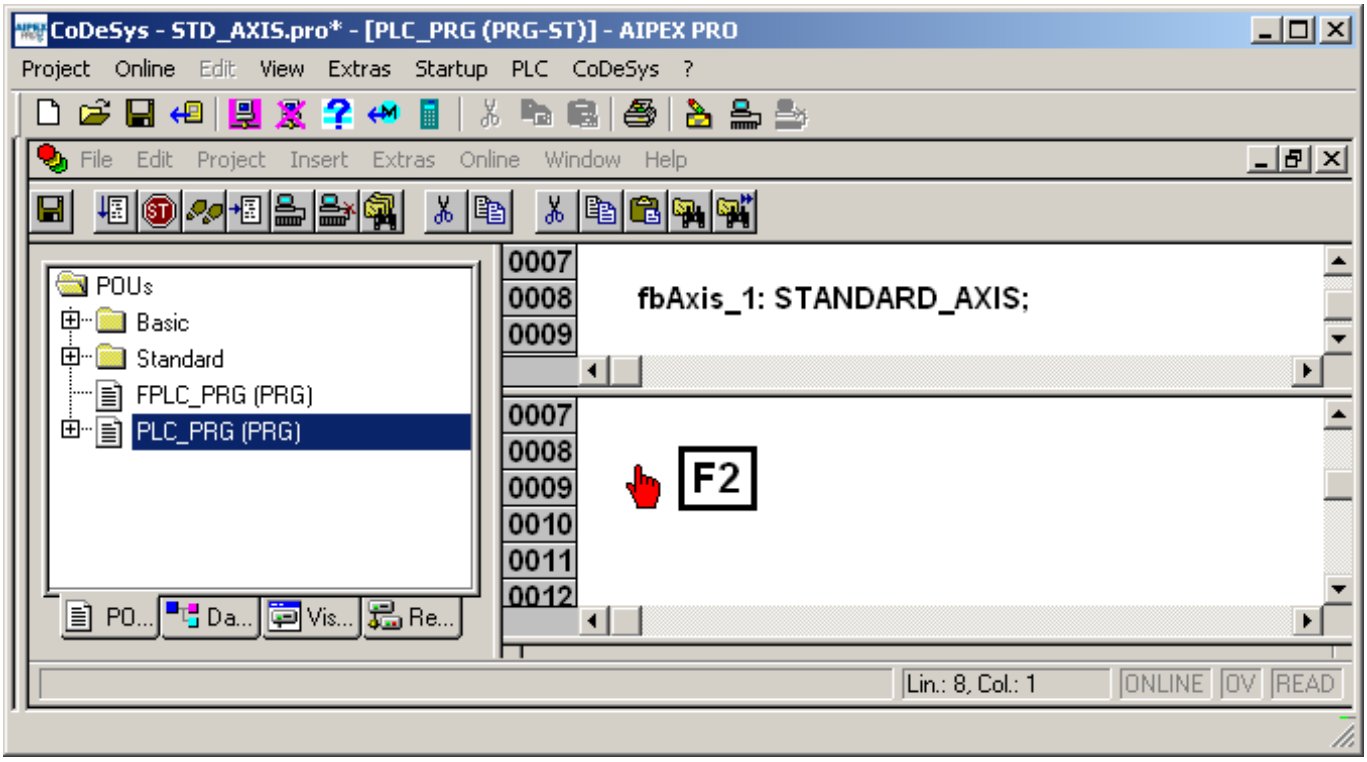
Right-click on the variable. You can then open the **'Declare Variable' window** (CTRL + F2).



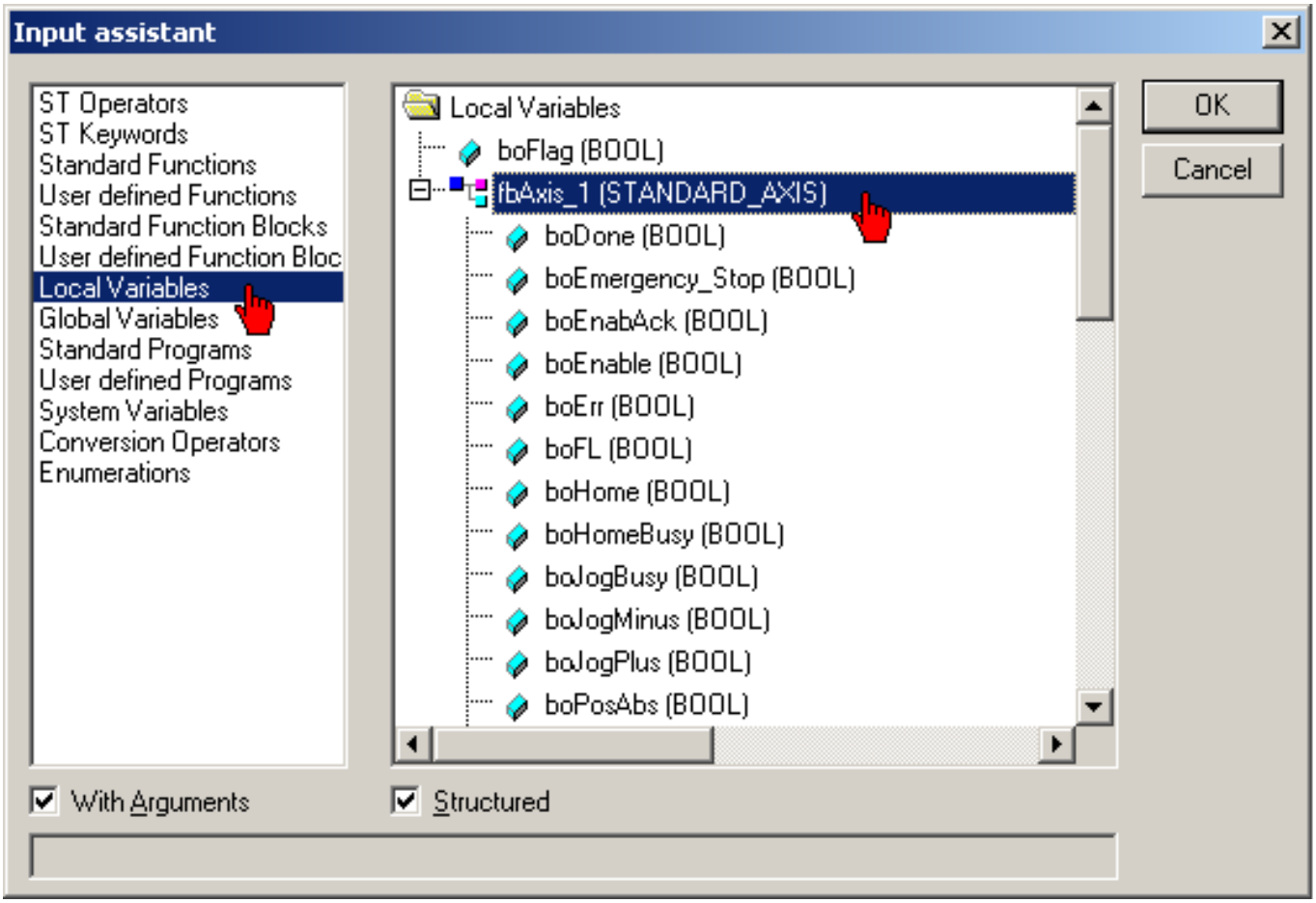
The function block 'STANDARD_AXIS' can be found in the Input assistant under **'User defined Function Blocks' 'Standard'**.

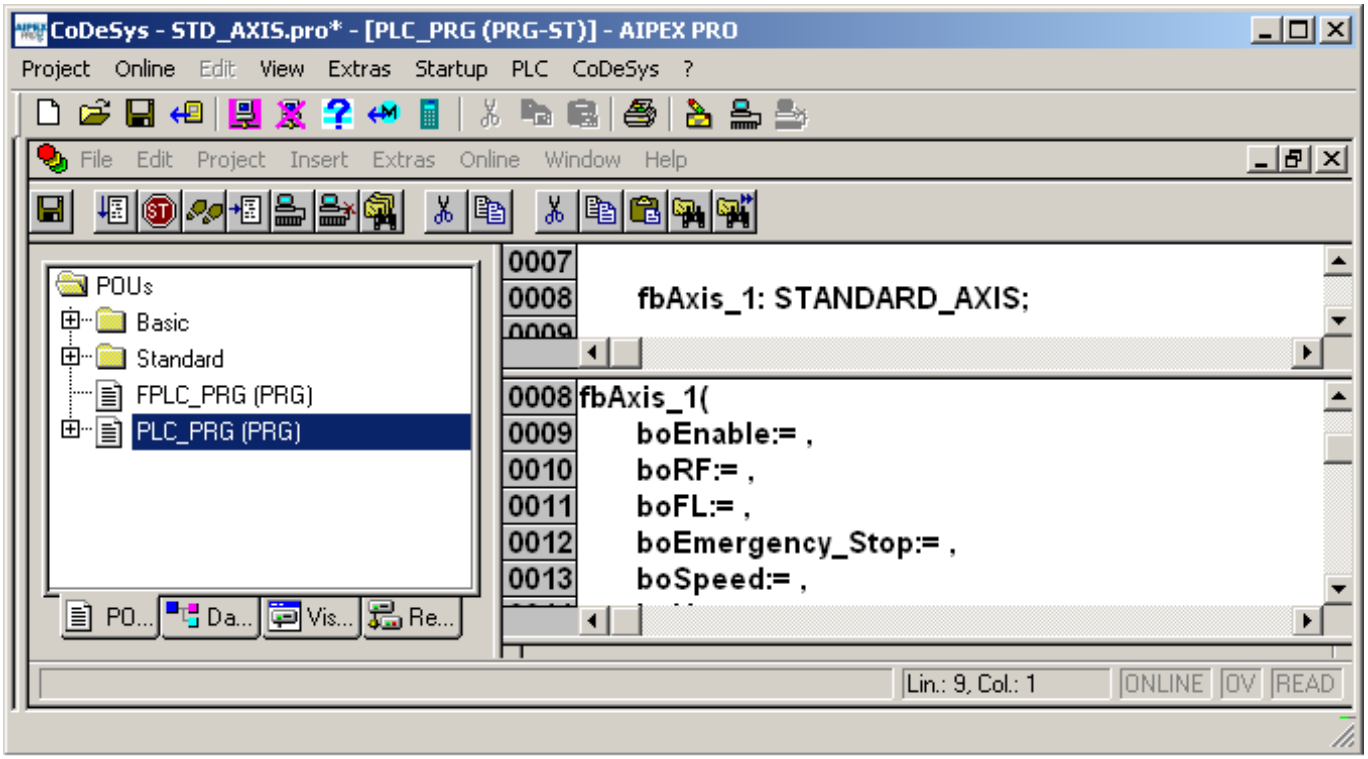


Add its function block ' fbAxis_1' in the program editor.
To do so, click the program editor. Press **'F2'** to open the **'Input assistant**.

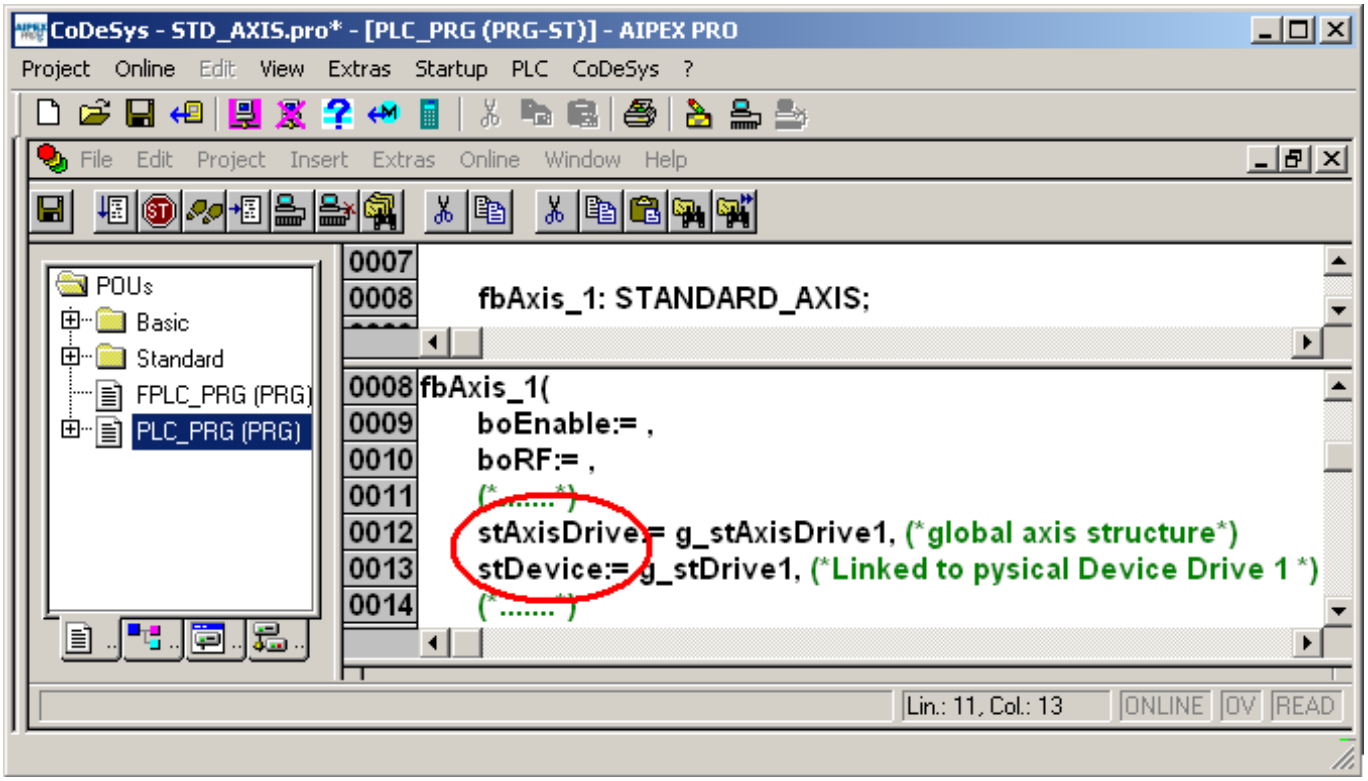


The function block **'fbAxis_1'** can be found in the Input assistant under **'Local Variables'**.





The IN_OUT variables stAxisDrive and stDevice must always be assigned.



Instancing a standard axis

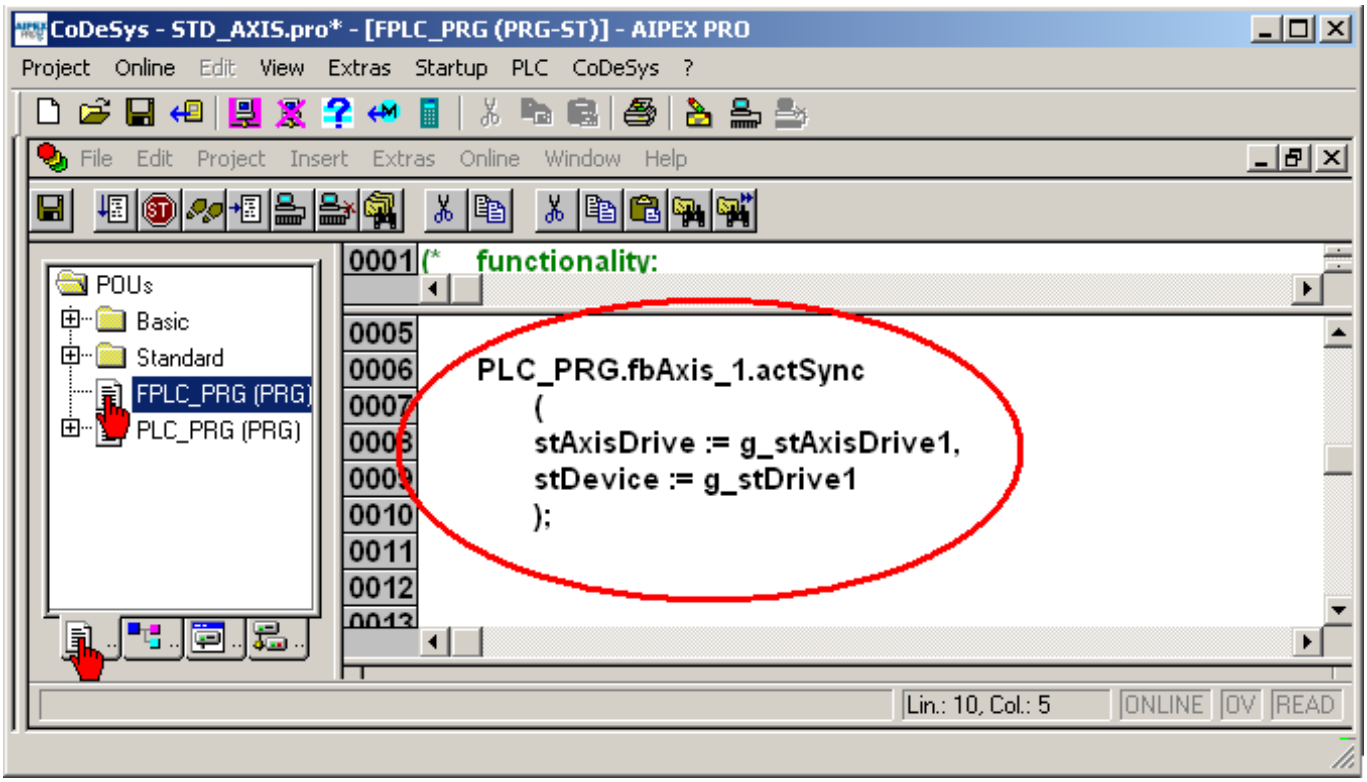
Alternative you can create a own function block to integrate and instance the AMK function block 'STANDARD_AXIS'.

4.3.2.12 Synchronous operations

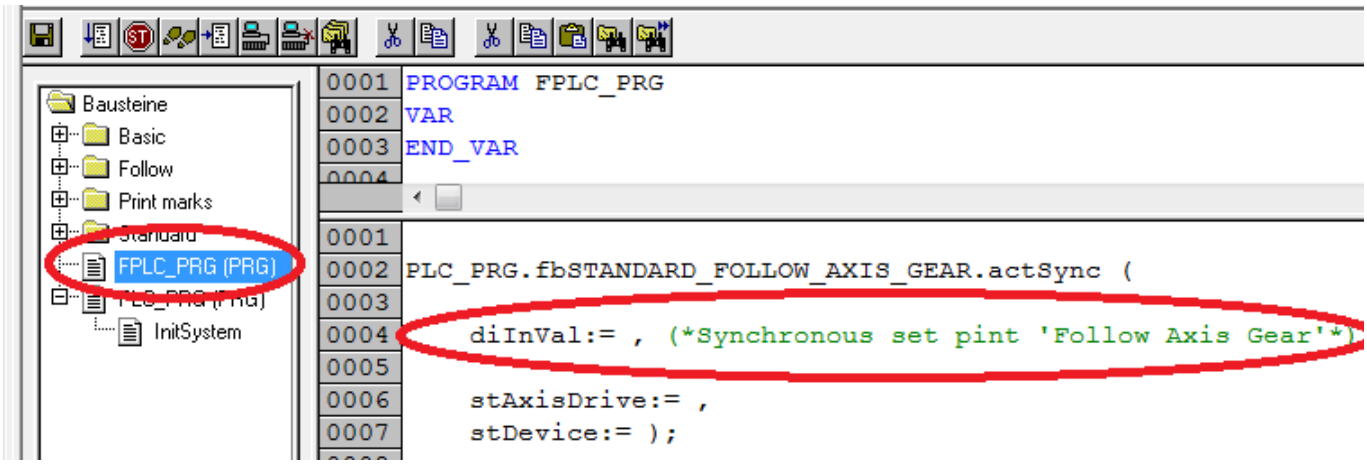
Function block 'STANDARD_AXIS' contains synchronous program sections that are synchronised with the drive.
The synchronous operation must be called up in **'FPLC_PRG'**. If this is not called up, the FB cannot be activated.

Create the following program code for this purpose.

```
PLC_PRG.fbAxis_1.actSync
(
stAxisDrive := (* axis structure in example g_stAxisDrive1 *),
stDevice := (* symbolic device name in example g_stDrive 1*)
);
```



Synchronous set point using as example STANDARD_FOLLOW_AXIS_GEAR (FB)
The synchronous setpoint is specified in FPLC_PRG / *.actSync call.
Add the Input Variable 'diInVal' (Setpoint specification for follow operation) as transfer variable.



4.3.2.13 Importing actual values

The structure 'ST_ACTUAL_VALUES' is a substructure of 'ST_AXIS_DRIVE'.

The actual axis values can be accessed via 'ST_AXIS_DRIVE.ST_ACTUAL_VALUES'.

Contents:

Name	Description	Unit
diID00040_Velocity_feedback_value	Actual speed value	0.0001 rpm
diID00051_Position_feedback_value	Actual position value	Increments
diID00084_Torque_feedb_val	Actual torque value	0.1 % Mn
diID00130_Probe1_val_p_edge	Probe value 1 positive edge 1)	Increments
diID00131_Probe1_val_n_edge	Probe value 1 negative edge 1)	Increments
diID00189_Following_dist	Following distance error 1)	Increments
diRectangularPulsInput_X132	Input pulses of square-wave pulse input	Increments

1) Only available for EtherCAT!

The 'Probe values' and the 'Following distance error' are inside the plc program commented out. This is the reason while the values are not updated.

Commented out parts inside plc program see:

- Declaration editor Standard Axis: fbGetStatusActVal
- Program code editor : Evaluation variable 'boErr'
- Program code editor : Evaluation variable 'boErrID'
- Action actSync: PLC Program code editor: part 'Read status'

Storage location:

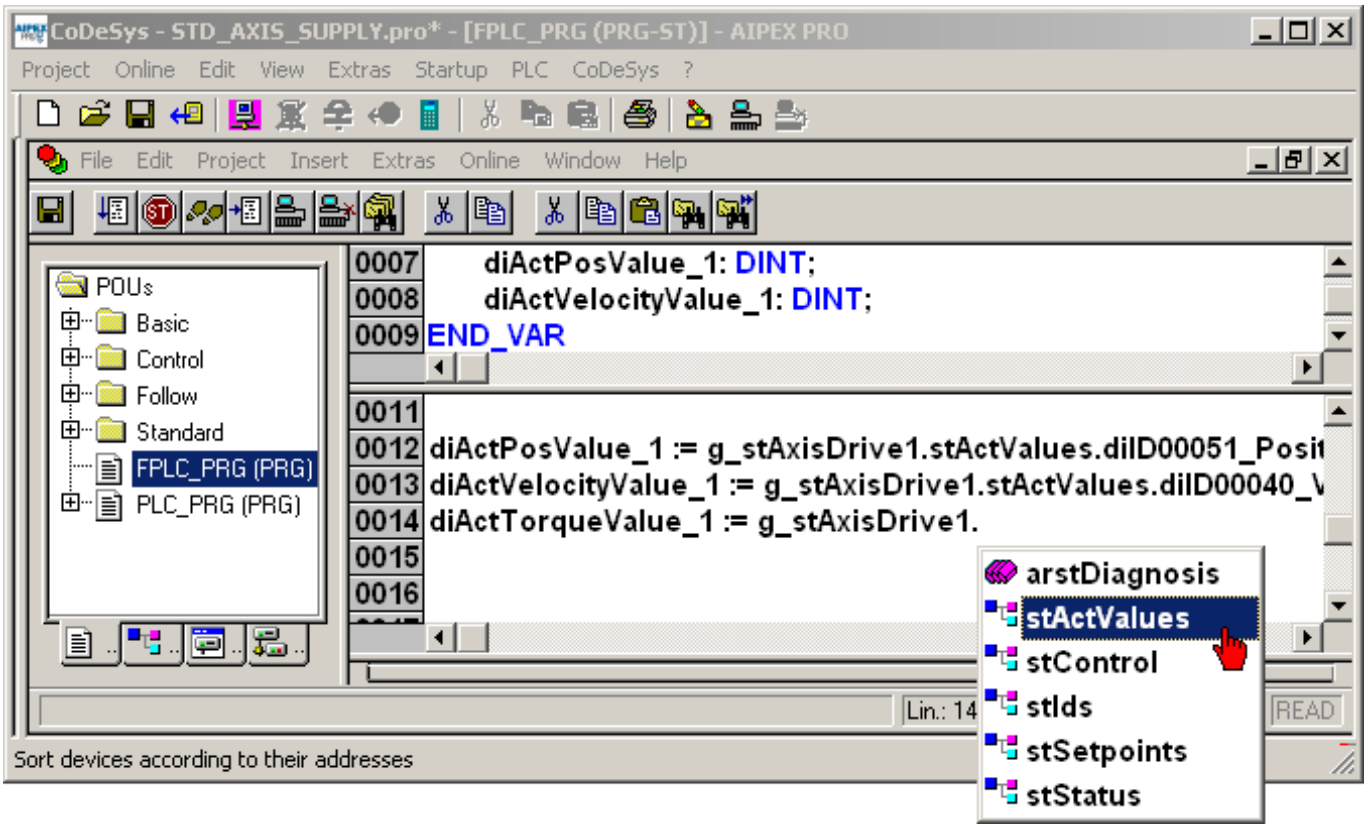
CODESYS V2

'Data types' → 'AMK_AXIS_STRUCTURE' → 'Interface'

CODESYS V3

'POUs' → 'Control' → 'Standard' → 'Types' → 'AMK_AXIS_STRUCTURE' → 'Interface'

Example: The actual values are copied to local variables.



4.3.2.14 Reading status bits

The structure 'ST_STATUSBITS' is a substructure of 'ST_AXIS_DRIVE'.

The status bits of the axis can be accessed via 'ST_AXIS_DRIVE.ST_STATUSBITS'.

Contents:

Name	Description	Unit
boID330_VelocityWindow	Code 330 nact = nset ID157 Velocity window ("To velocity")	-
boID331_ZeroVelocityWindow	Code 331 nact < nmin ID124 Zero velocity window	-
boID332_VelocityWindowLimit	Code 332 nact < nx ID125 Velocity limit nx	-
boID333_TorqueLimit	Code 333 Md>=Mdx ID126 Torque limit Mdx	-
boID334_NegTorqueLimit	Code 334 Mset >Mlimit Set torque, Limit torque (ID82/83)	-
boID335_NegVelocityLimit	335 nset >= mlimit velocity setpoint. Limit speed (ID38/39)	-
boID336_InPosition	In position [1)]	-
boID33029_SBM	System ready message	-
boID33030_QUE	Acknowledgement DC bus ON	-
boID33031_QRF	Acknowledgement controller enable	-
boID33036_RFP_known	Code 33036 Home position known	-
boID33074_Warning_active	Code 33074 Warning active group warning	-
boDiagOk	Diagnosis OK	-
boSetPosEnabAck	Secondary operation mode1 release position specifications	-
boSetVelocityEnabAck	Secondary operation mode2 release speed	-
boSetTorqueEnabAck	Secondary operation mode3 release torque setpoint	-
boResidDistDone	Residual distance cleared	-

1) The 'In position' bit will evaluate only with controller cards with internal interpolator. If you use an Ethernet controller card, the bit will only evaluate if you use the commands absolute or relative positioning (boPosAbs / boPosRel).

Storage location:

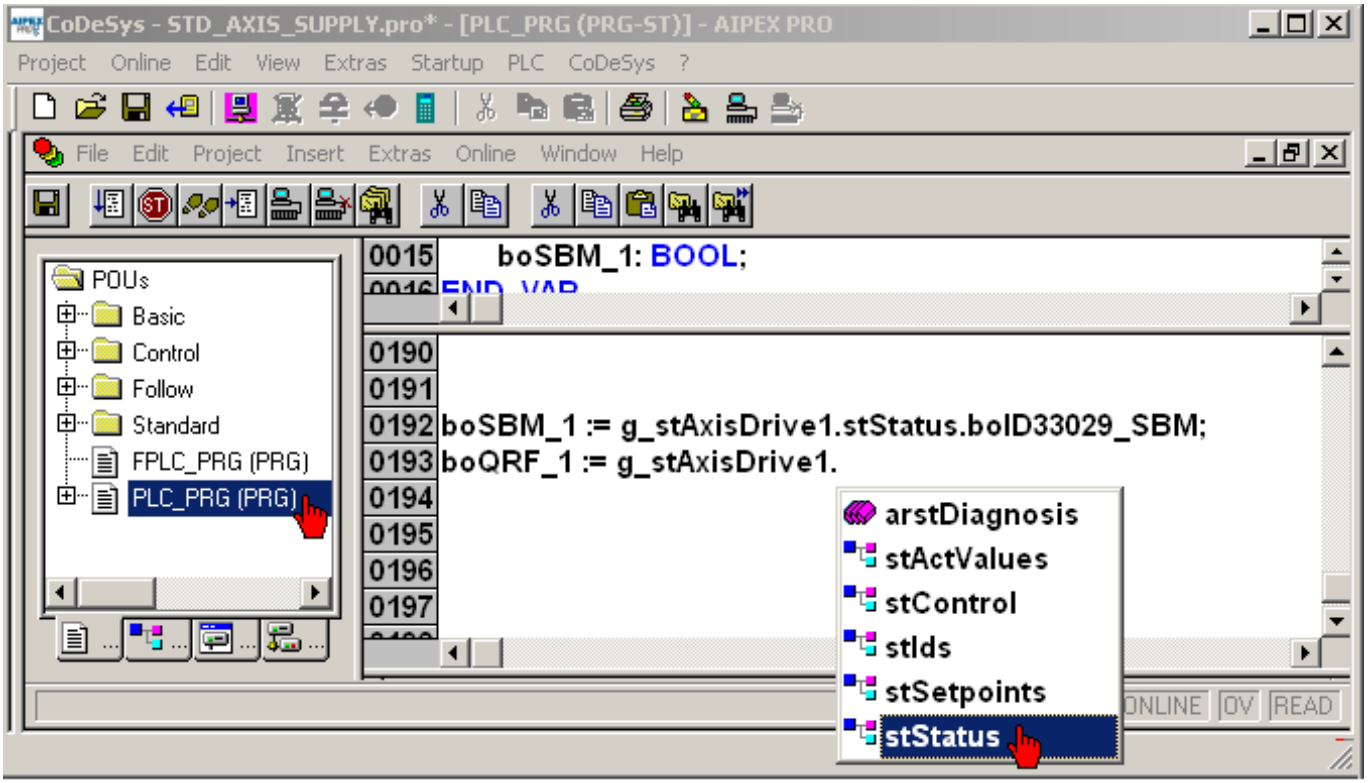
CODESYS V2

'Data types' → 'AMK_AXIS_STRUCTURE' → 'Interface'

CODESYS V3

'POUs' → 'Control' → 'Standard' → 'Types' → 'AMK_AXIS_STRUCTURE' → 'Interface'

Example: The status bits are copied to local variables.



Expand status bit description: Siehe Expanding standard bits auf Seite 800.

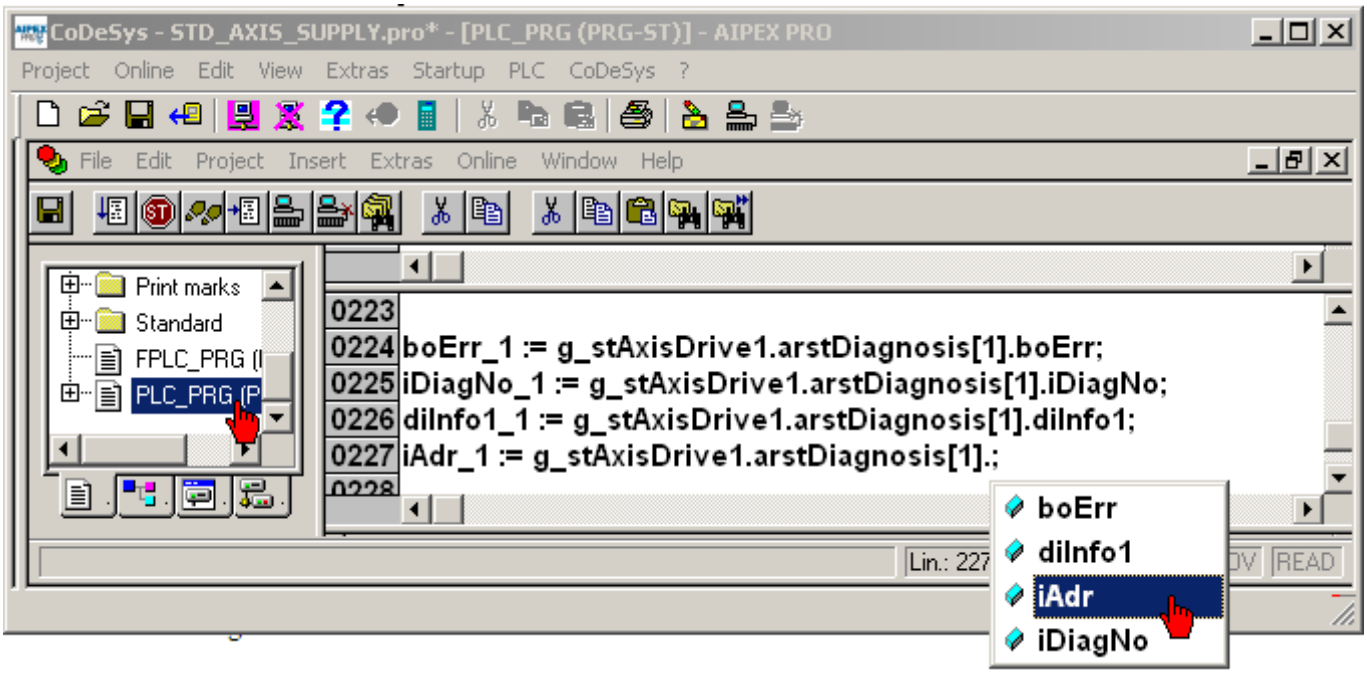
4.3.2.15 Reading a diagnosis

The array 'arstDiagnosis' is part of the structure of 'ST_AXIS_DRIVE' and 'ST_CONTROL'.
The diagnostic messages can be accessed via 'ST_AXIS_DRIVE.arstDiagnosis [1-10]', for example.
The diagnostic array contains up to 10 diagnostic messages which are structured as follows:

Name	Description	Unit
arstDiagnosis [1]	boErr: Error bit for error 1 iDiagNo: Diagnostic number for error 1 diInfo1: Additional 1 for error 1 iAdr: Device address*	-
arstDiagnosis [2]	boErr: Error bit for error 2 iDiagNo: Diagnostic number for error 2 diInfo1: Additional 1 for error 2 iAdr: Device address*	-
...	...	
arstDiagnosis [10]	boErr: Error bit for error 10 iDiagNo: Diagnostic number for error 10 diInfo1: Additional 1 for error 10 iAdr: Device address*	-

* iAdr: Device address
An ACC-bus controller displays the device address when a slave error occurs; in all other cases, 0 (local error). EtherCAT always 0.

Example: The diagnostic message is copied to local variables.



4.3.2.16 Visualisation

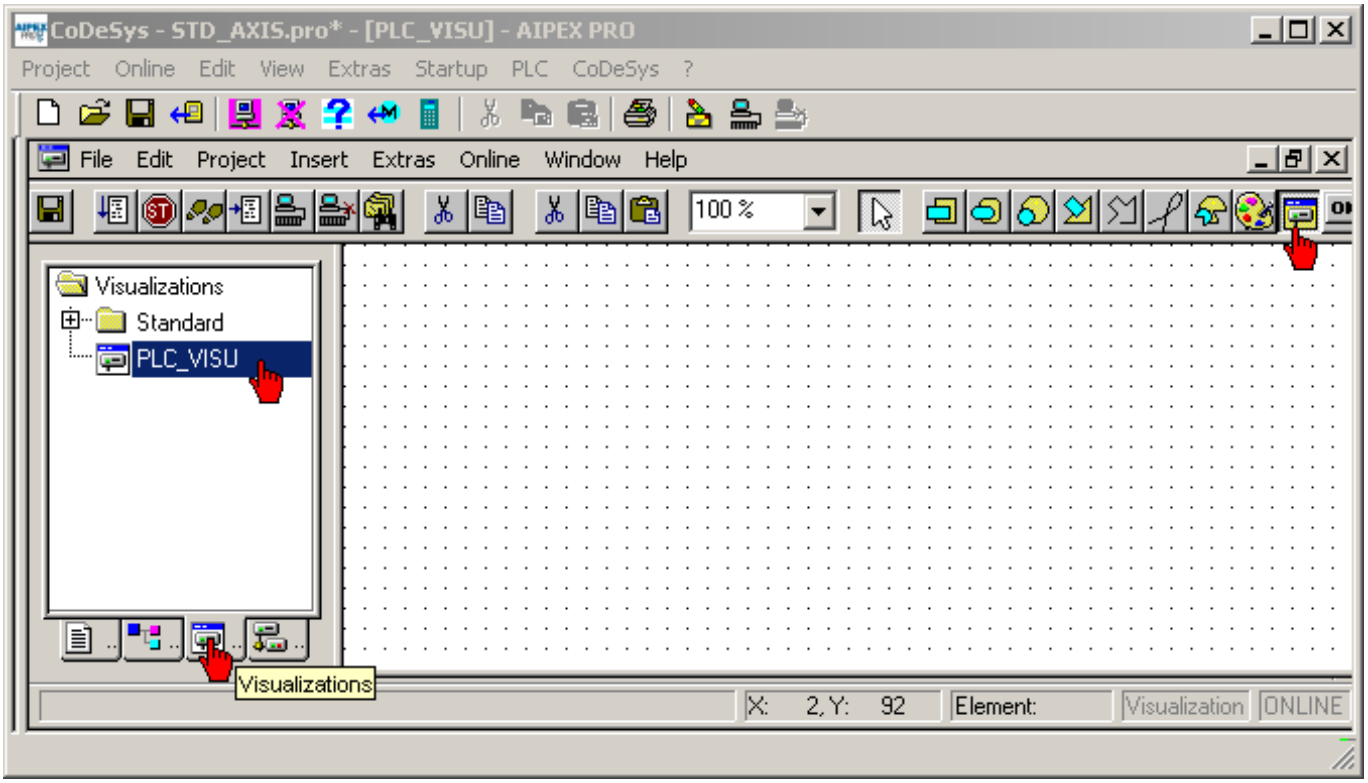
CODESYS contains an integrated visualisation editor. With the visualisation editor, graphic interfaces are generated with which the user can control and monitor the PLC program.

For AMK function blocks there are preconfigured visualisation elements.

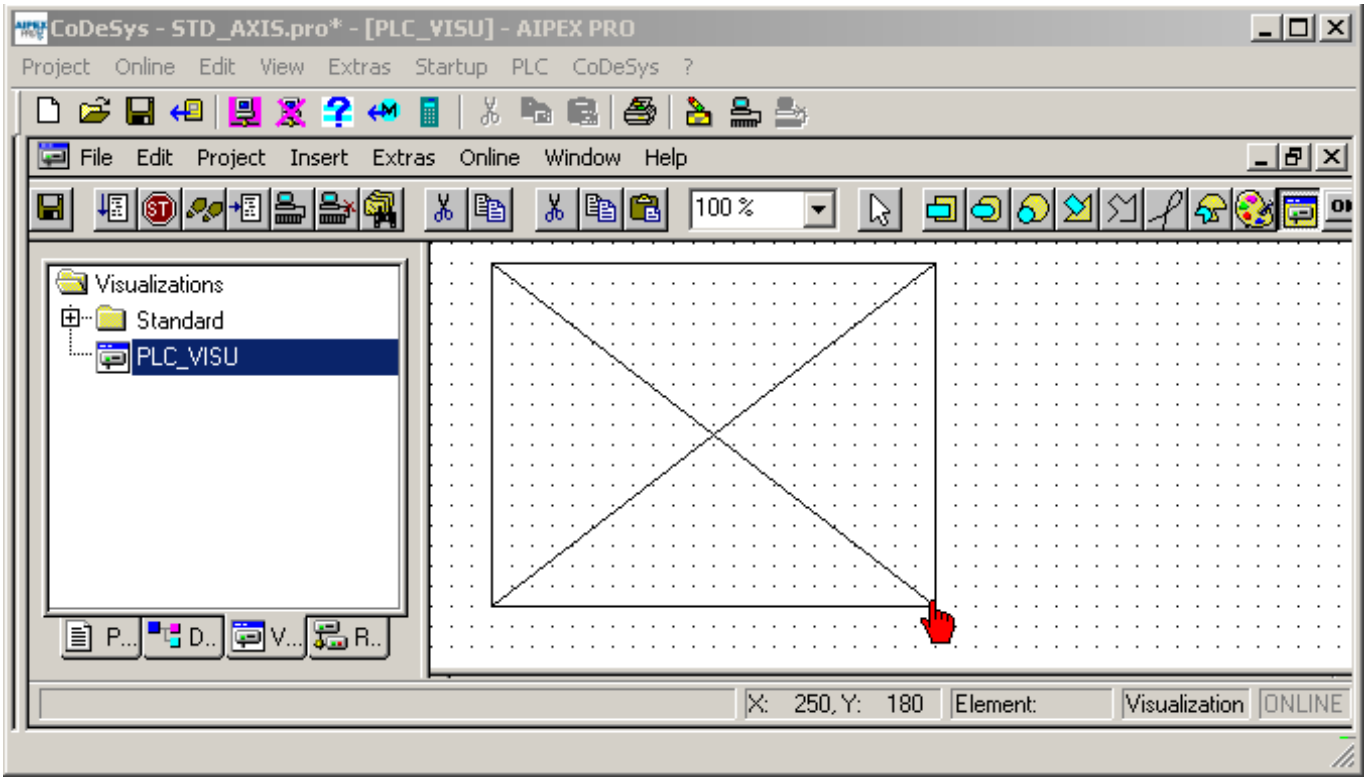
A description of how to link a standard axis to a visualisation element is provided in the steps to follow.

PLC_VISU is the start page and may not be renamed.

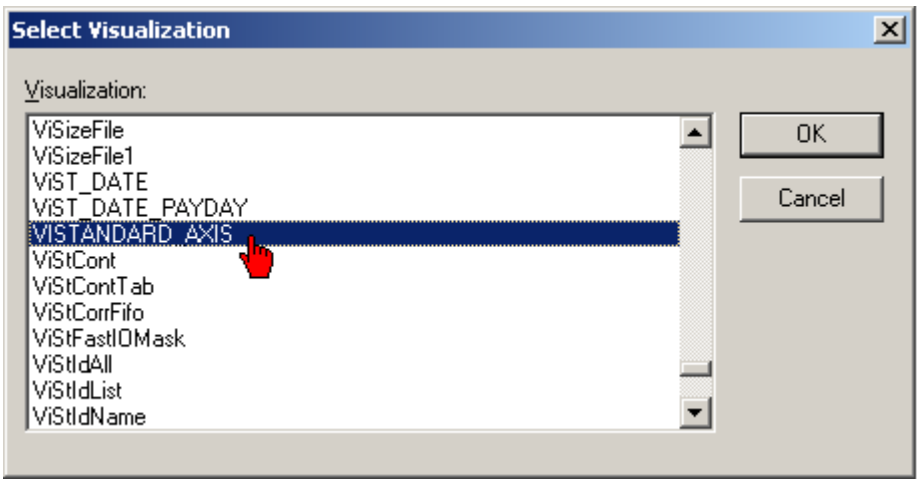
Switch to the **'Visualisations'** tab. Then double-click on **'PLC_VISU'** to open the editor window.
Select **'Visualisations'** in the menu.



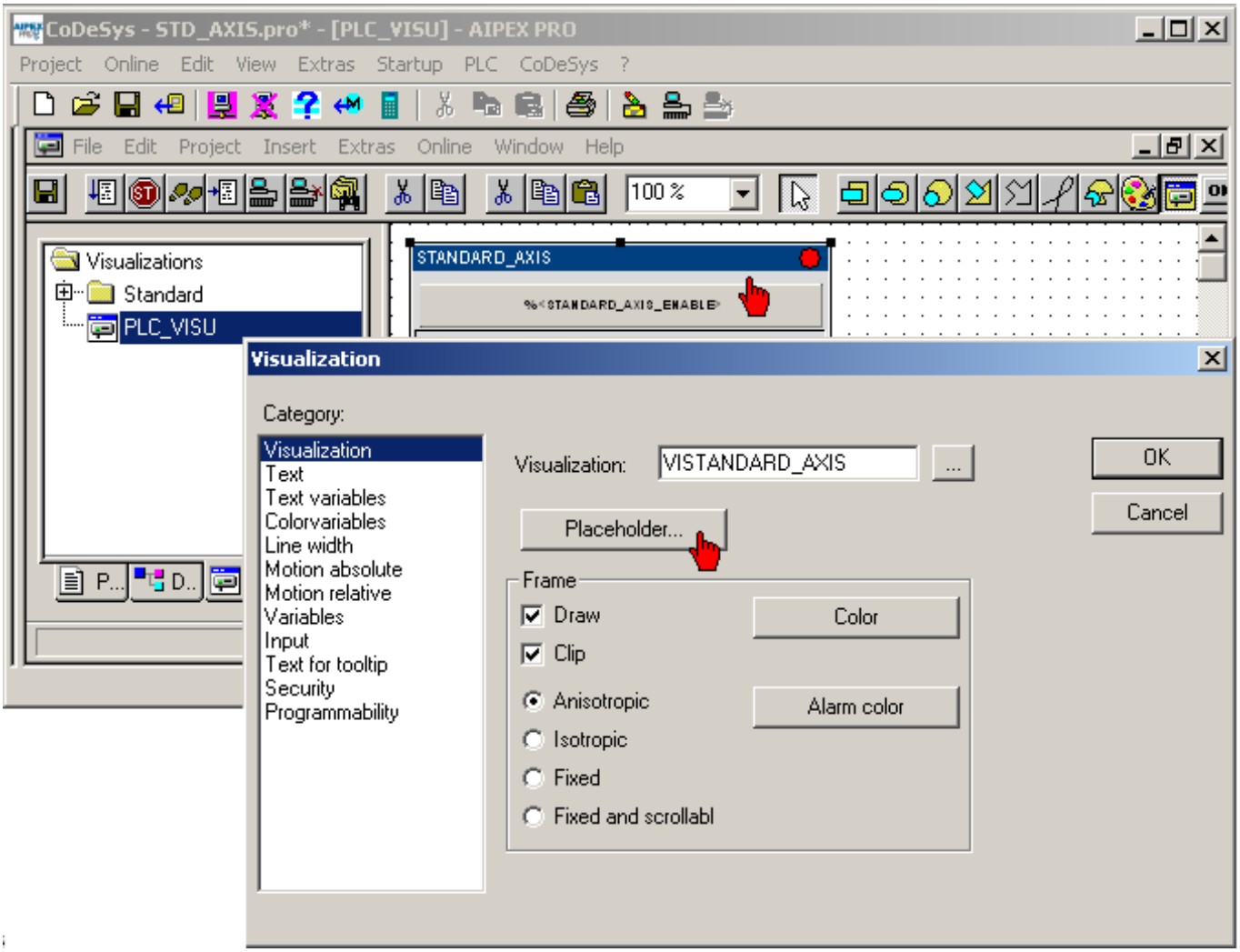
With the left mouse button held down, you can drag a visualisation element to any location. The exact size can be modified.



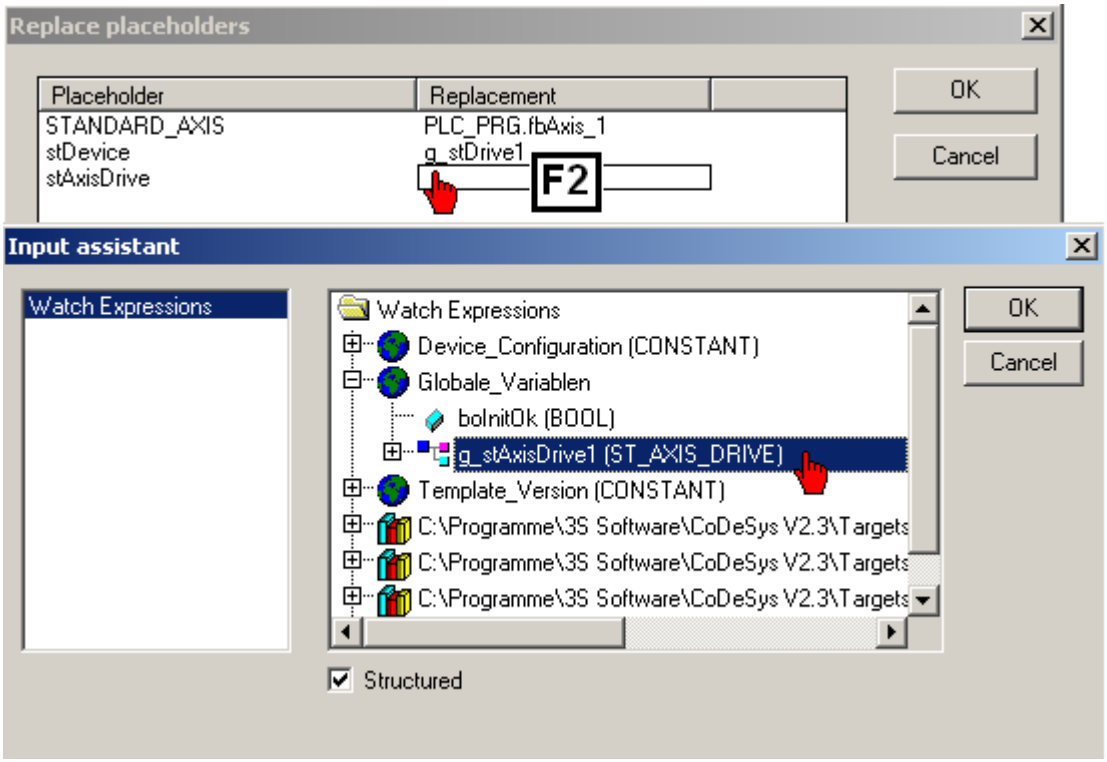
Select the element **'VISTANDARD_AXIS'** in the dialog window.



To do so, double-click the visualisation element. The **'Visualisation'** dialog box then opens.
Click on the **'Placeholder'** button. Here you can set the connections to the program code.

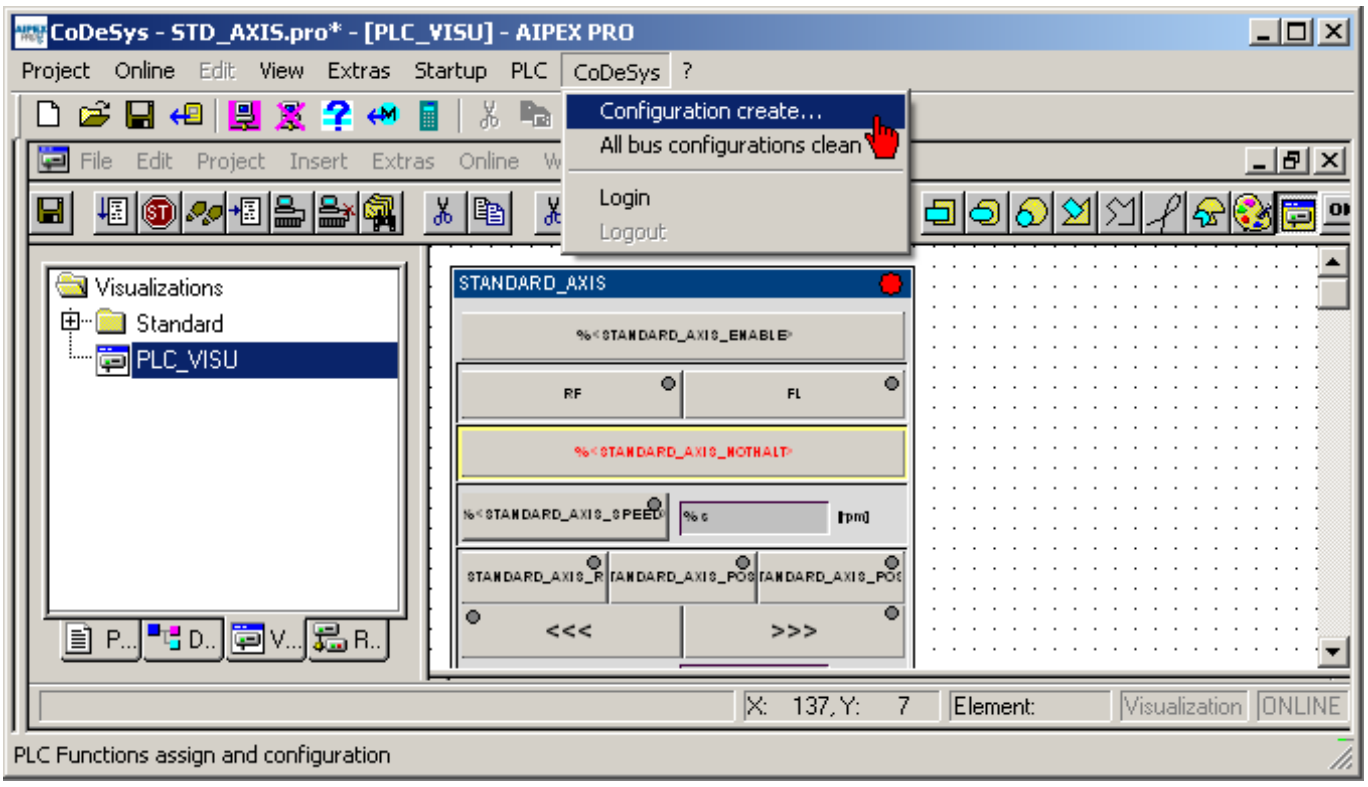


STANDARD_AXIS: Connection to function block in PLC_PRG.
stDevice: Connection to axis structure
stAxisDrive: Connection to symbolic device name

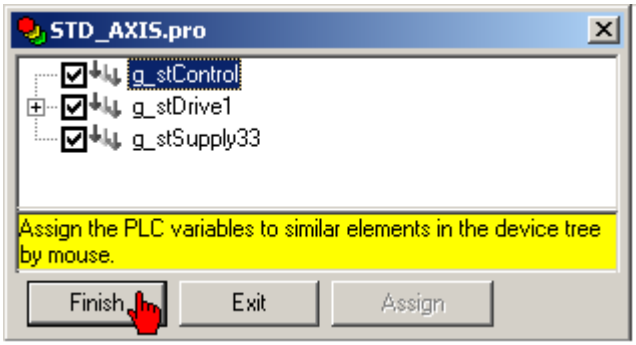


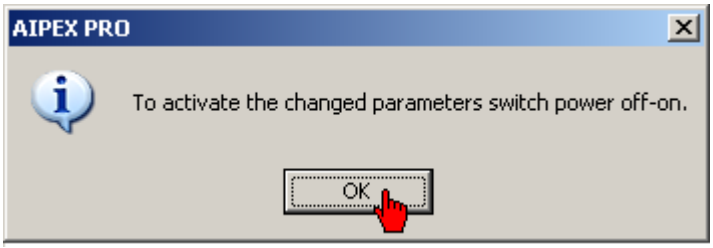
4.3.2.17 Transferring a program

Before the program can be transferred, it must first be translated without error and the message configuration generated.
Start this process under **'CODESYS' 'Configuration create'**.

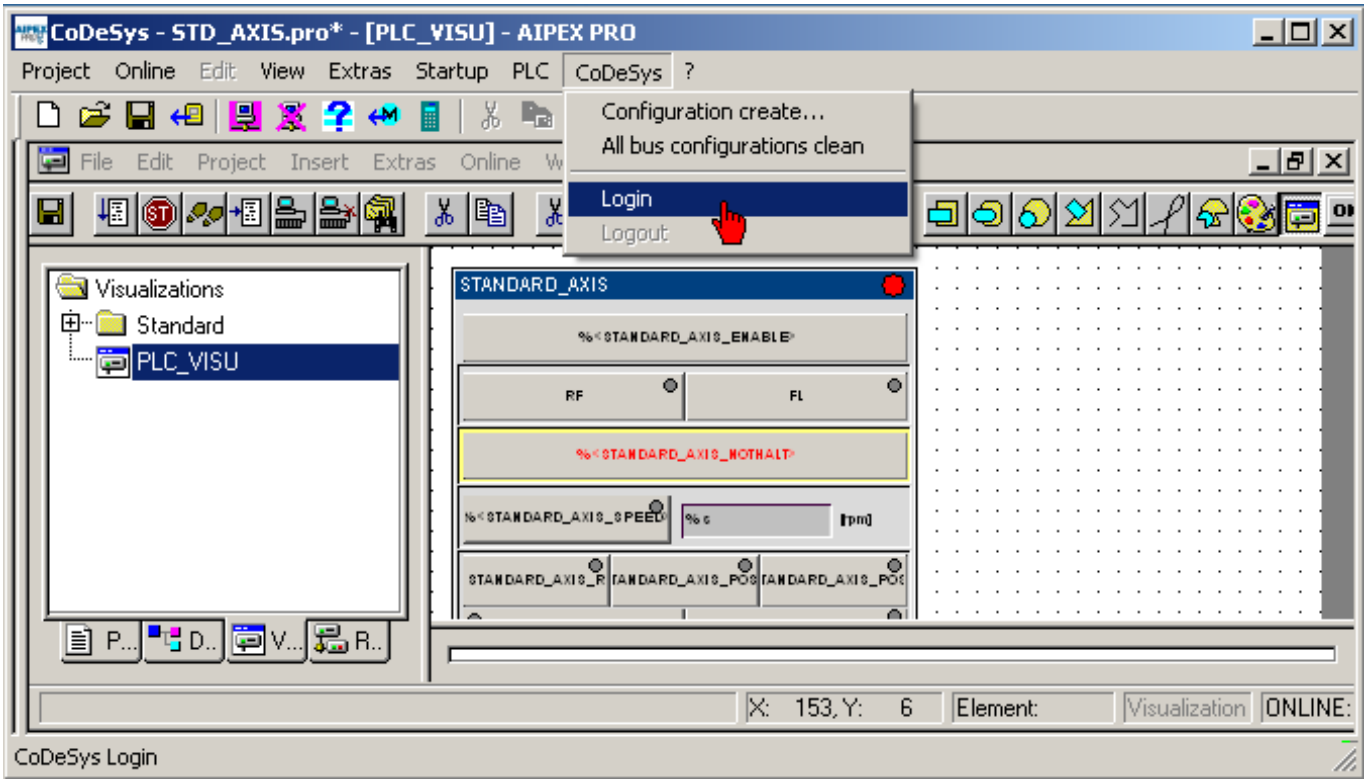


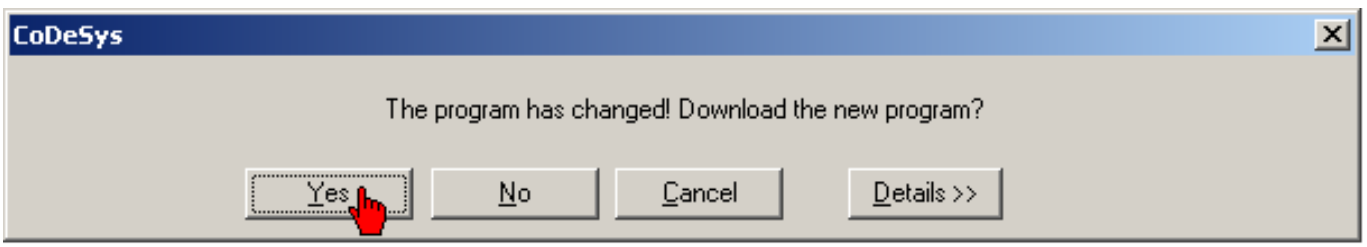
Press the **Finish** button. AIPEX PRO automatically generates the bus configuration and modifies the relevant parameters.
(Automatic parameterisation: See AIPEX PRO **'Menu' 'Extras' 'Configuration create'**)



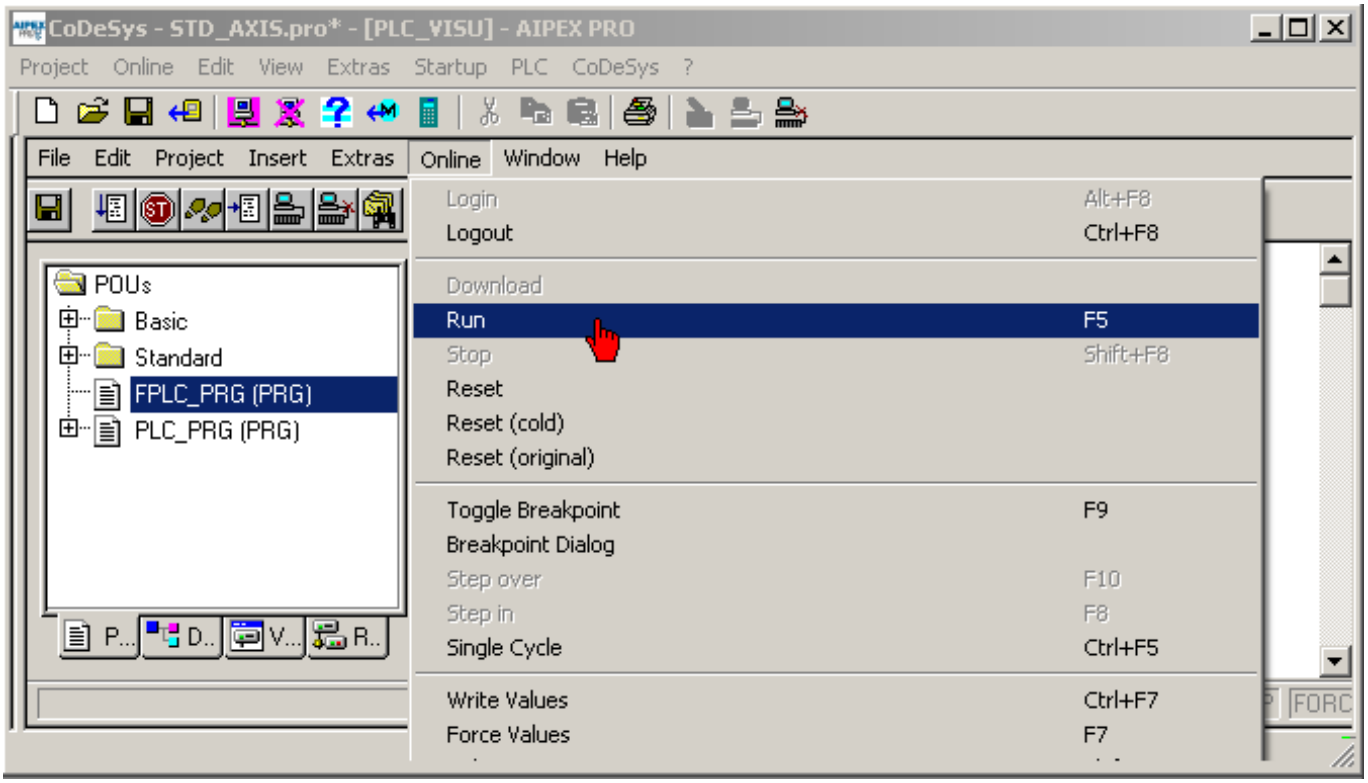


AIPEX PRO must be working online before you can login under **'CODESYS' 'Login'** onto the PLC.

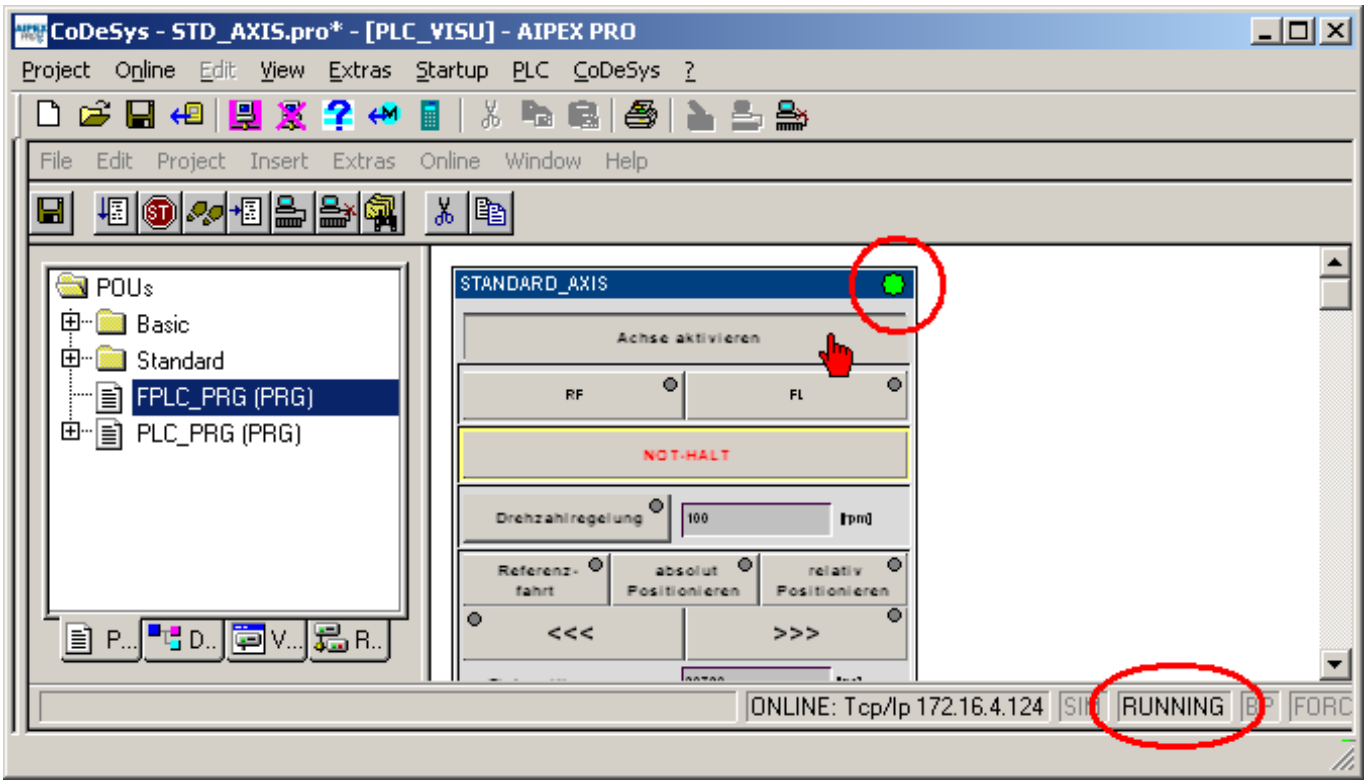




The PLC is launched under **'Online' 'Run'**.



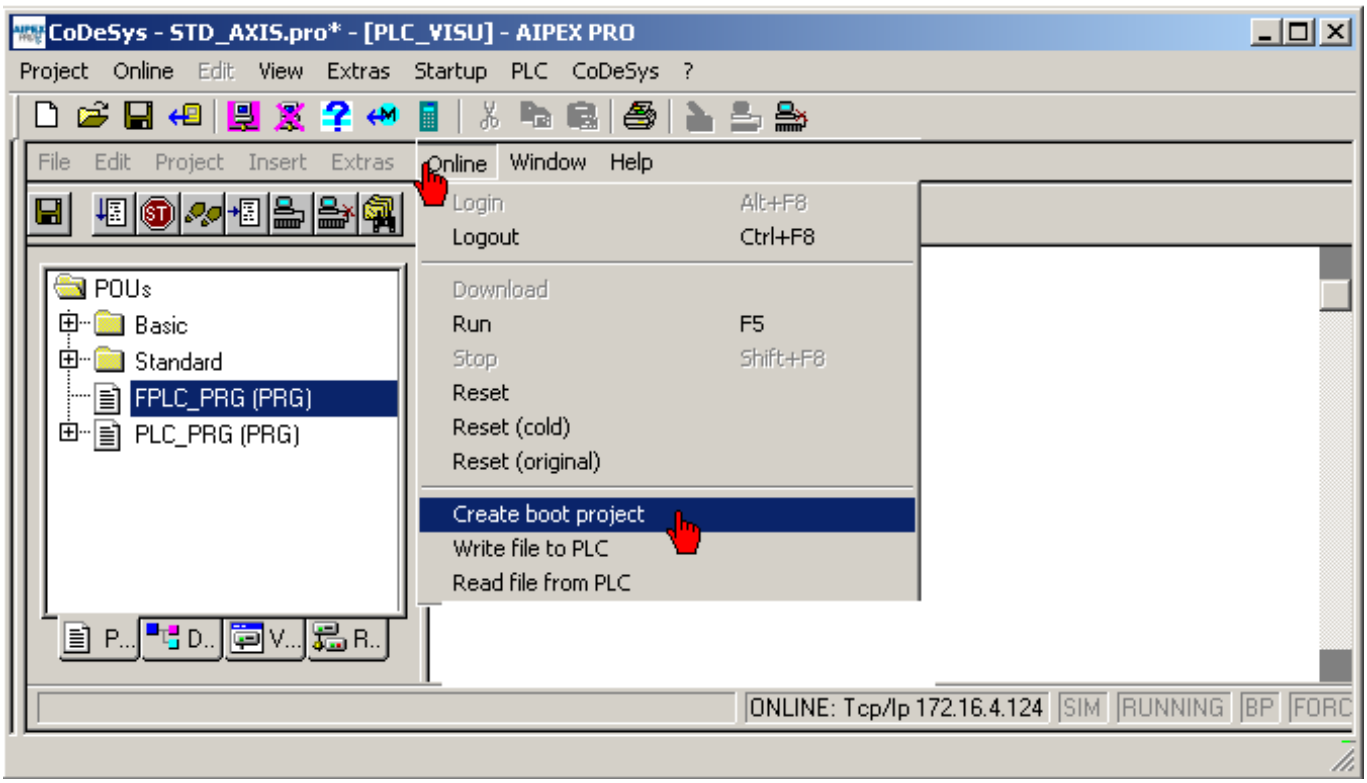
You can now control your drive system with the visualisation that you have created.



AMKASYN KE/KW - Switch-on and -off flow chart: See documentation KE/KW Initial startup (AMK part no. 204539) chapter 'Switch-on and -off flow chart

AMKASMART iC/iX/iDT5 - Switch-on and -off flow chart: See documentation iC/iX/iDT5 Initial startup (AMK part no. 204737) chapter 'Switch-on and -off flow chart.

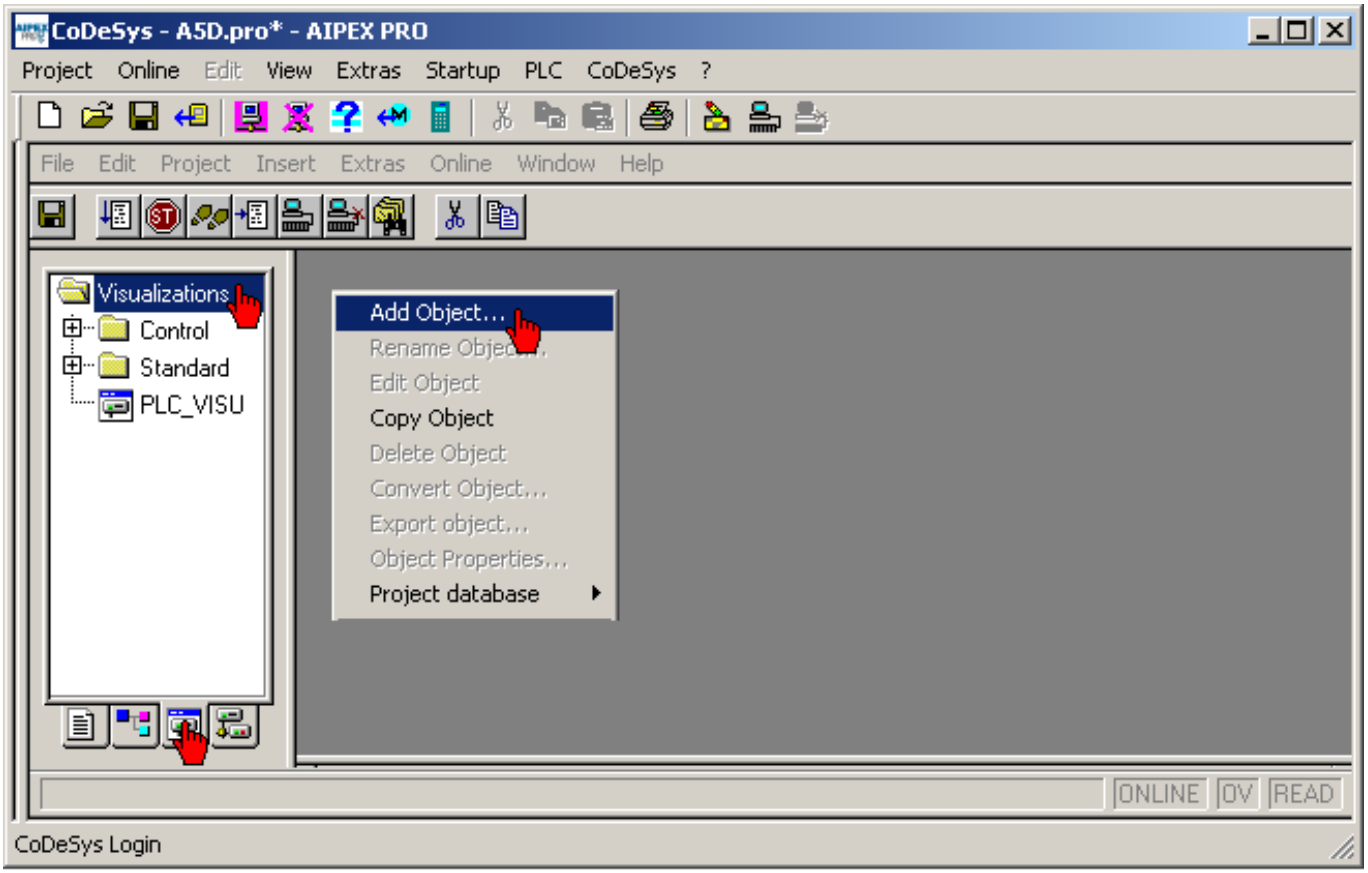
If the PLC program is to be saved remanently, you need to create a boot project. PLC editor - **Menu online -> Create boot project**.



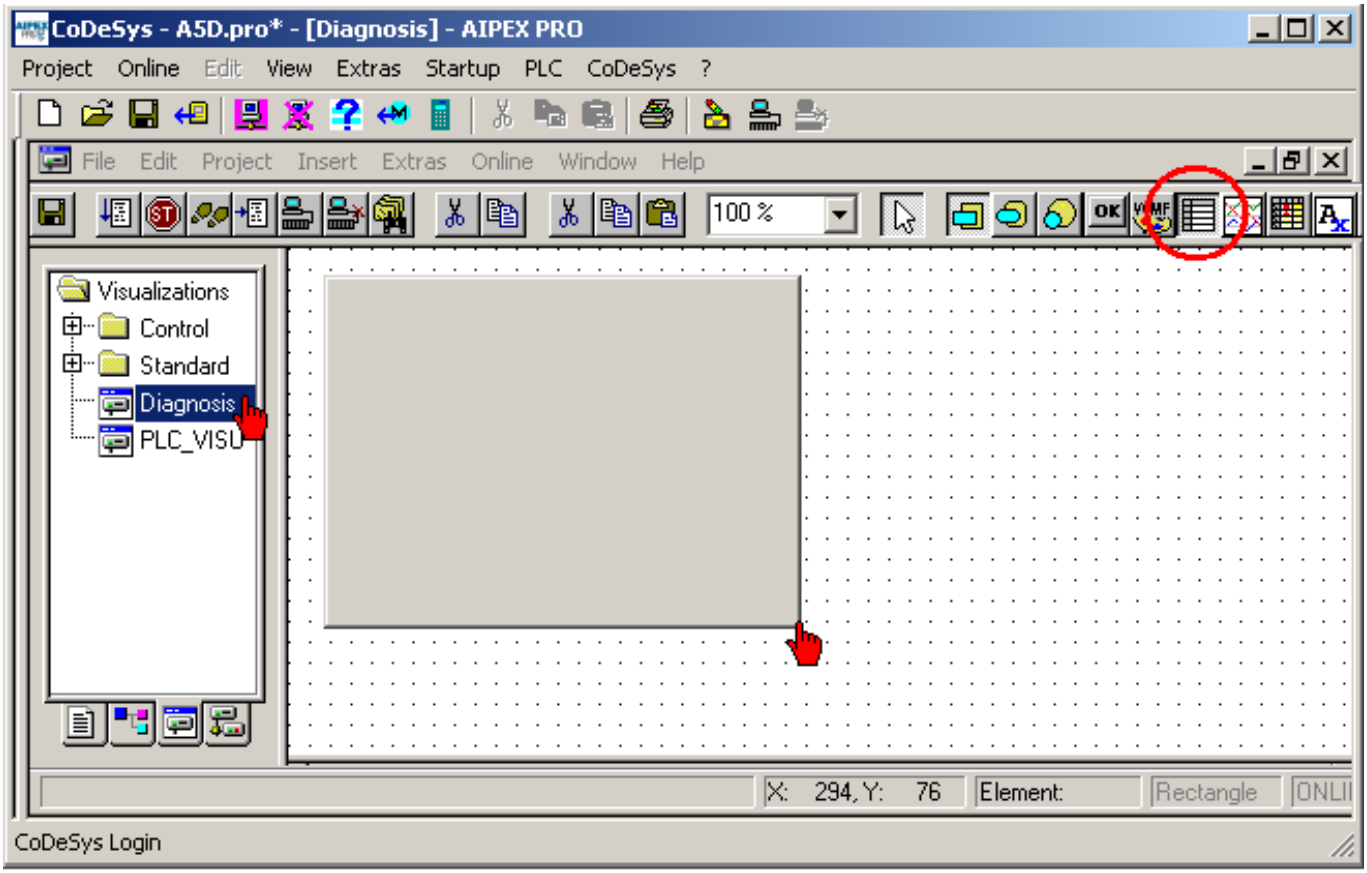
4.3.2.18 Quick start

4.3.2.18.1 Visualising a diagnostic array (arstDiagnosis)

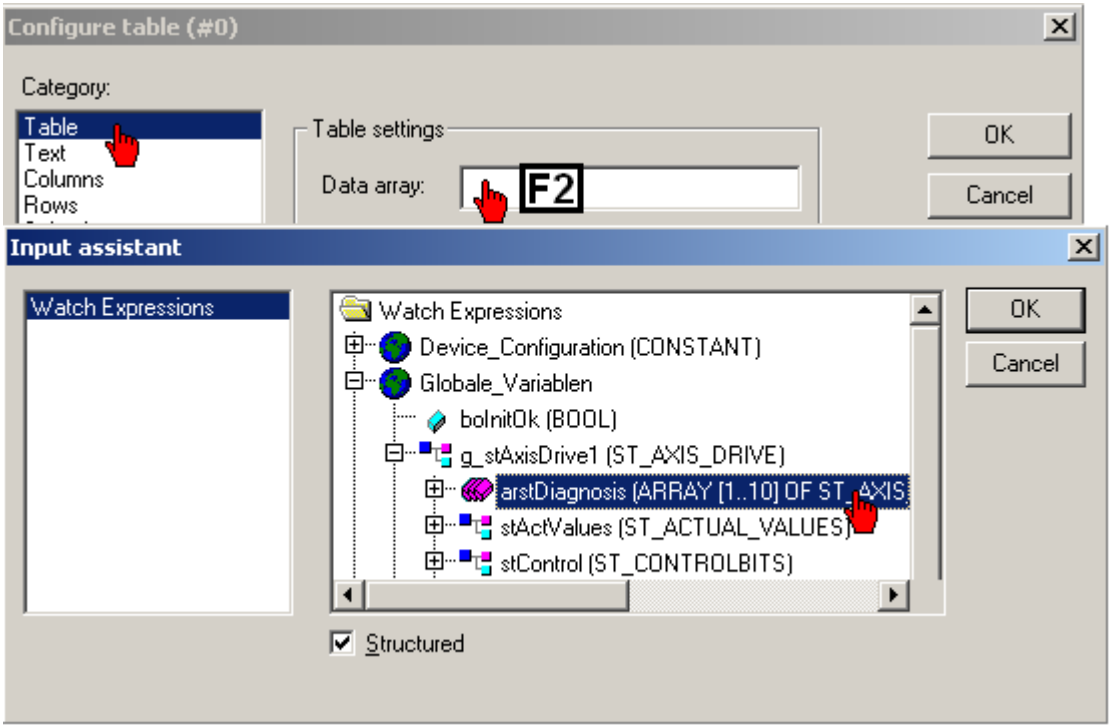
Switch to the Visualisation tab. Right-click to add a new page.



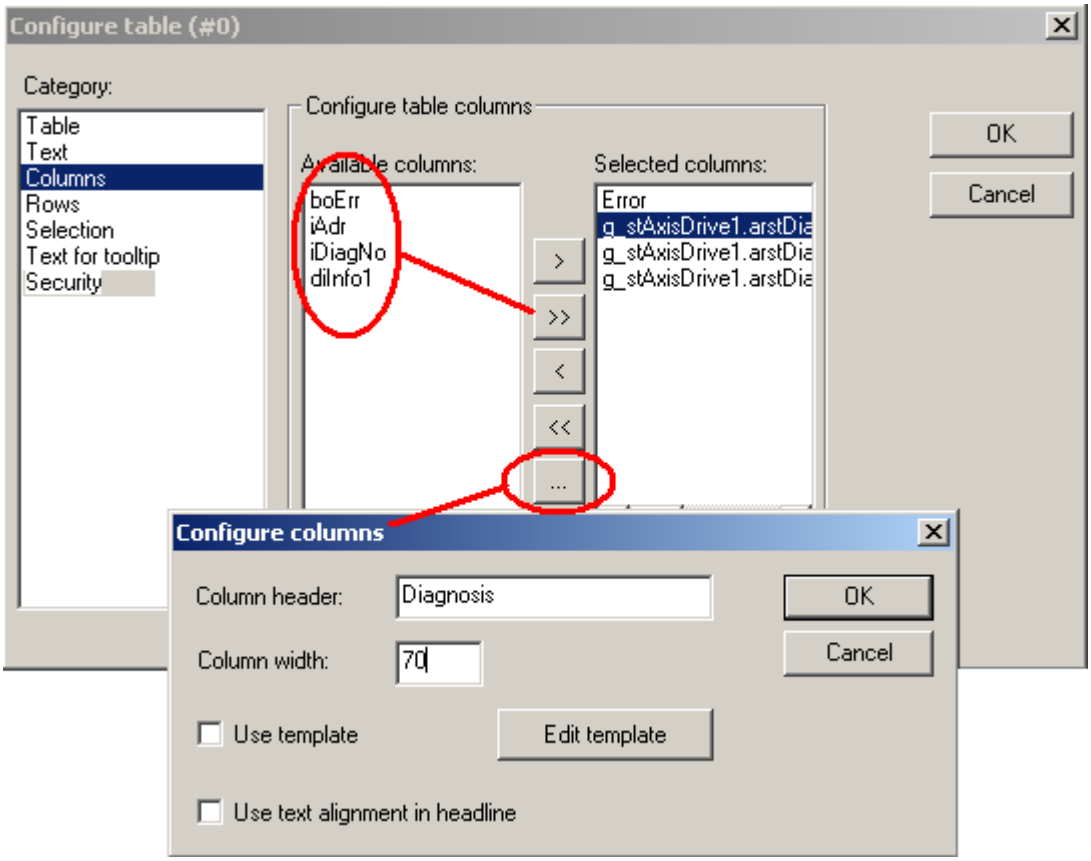
Select **'Table'** in the menu.
With the left mouse button held down, you can drag the table to any location. The exact size can be modified.



Define the connection under **'Data array'**. You can find the diagnostic array 'arstDiagnosis' in the structure 'ST_AXIS_DRIVE'.

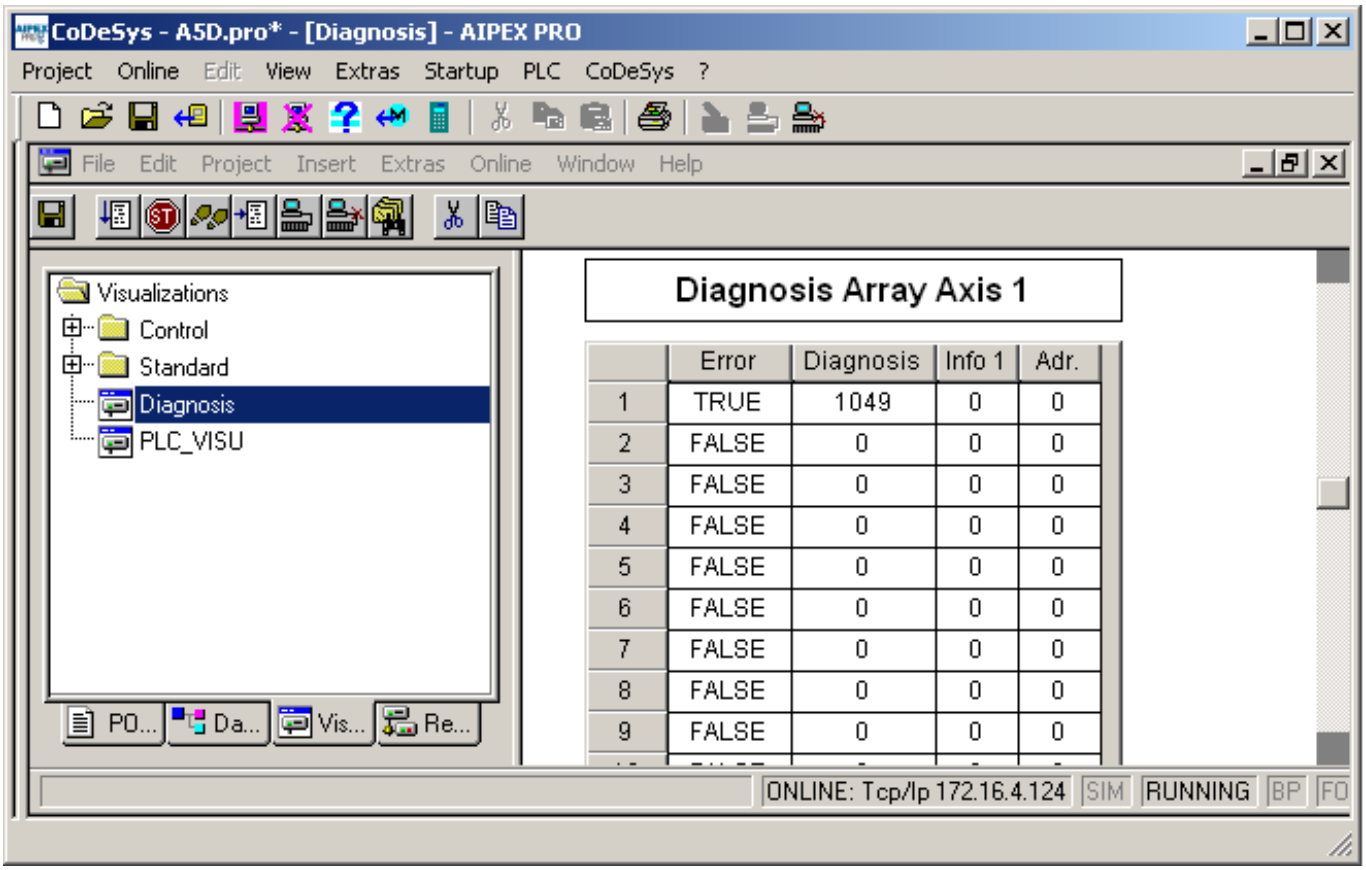


Import the variable you wish to display in the table under **'Configure columns'**.
Then select the variables one by one to customise the table header and width.



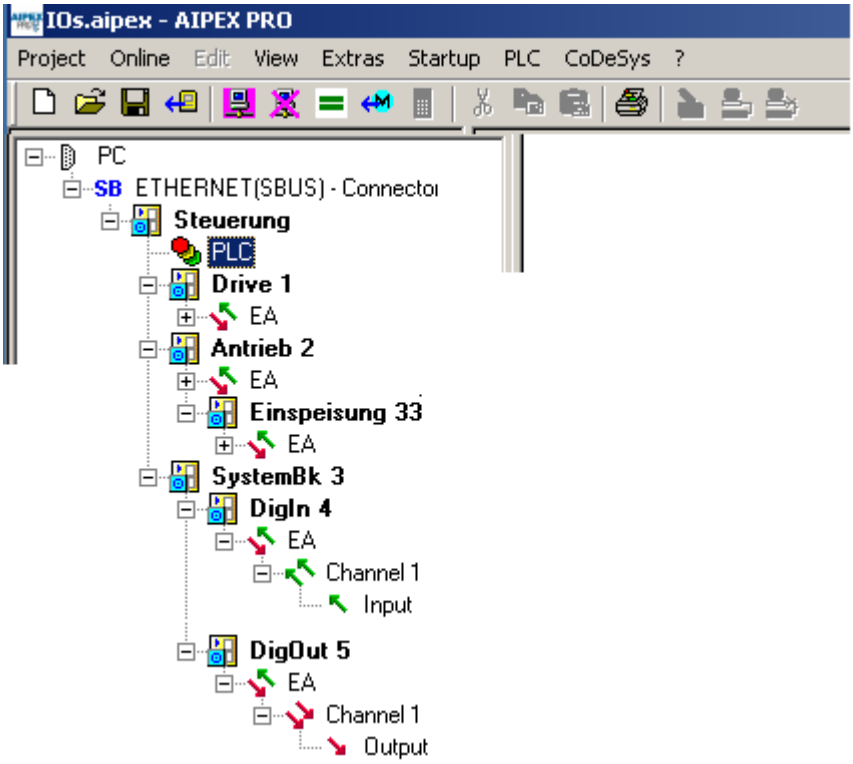
Diagnostic number 1: Main message

Diagnostic number 2 – 10: Possible resulting error



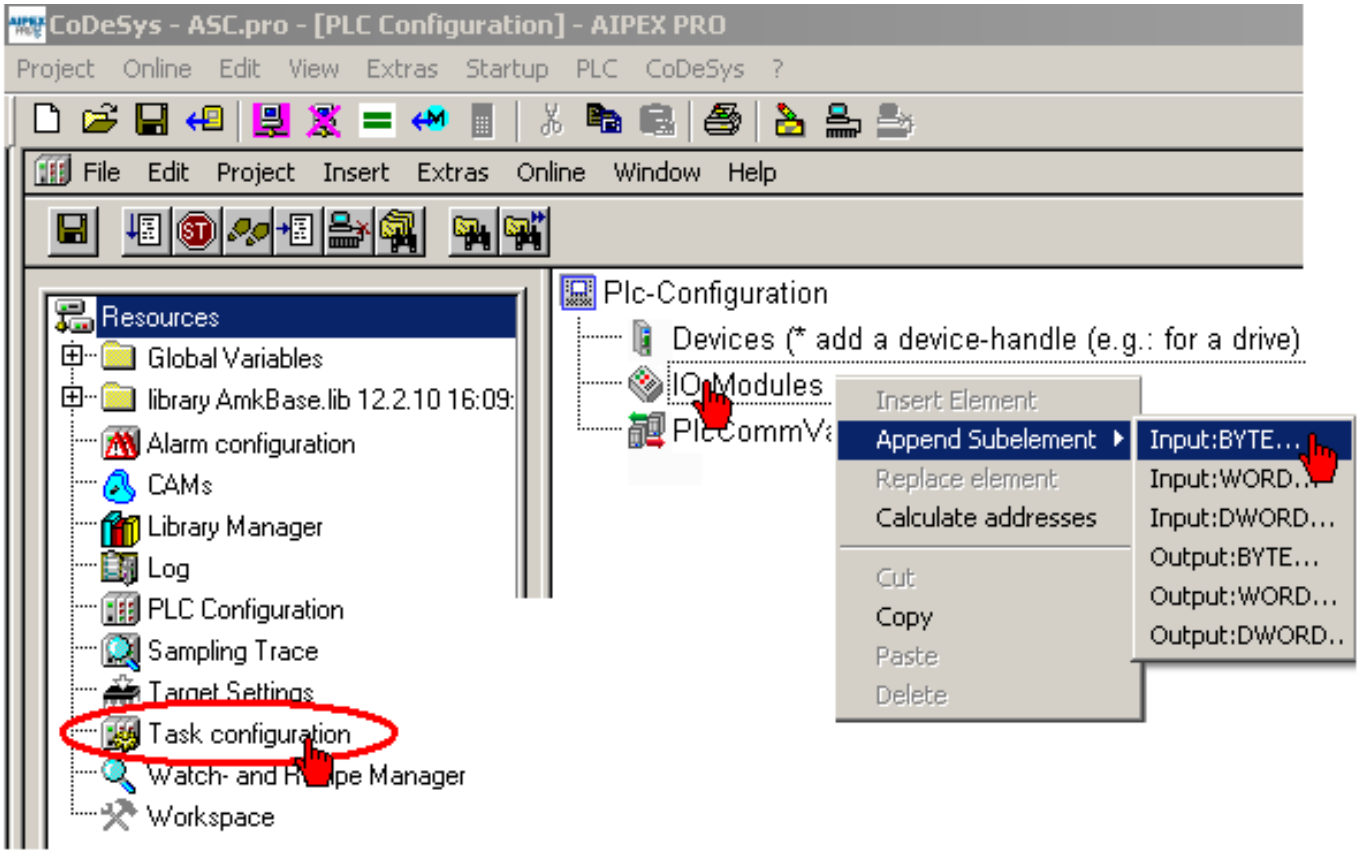
4.3.2.18.2 External asynchronous IO terminal

ntegration of an external I/O terminal. The example shows an EtherCAT I/O terminal (symbolic name SystemBK3) with 8 inputs and 8 outputs. The EtherCAT terminal is detected automatically by AIPEX PRO.

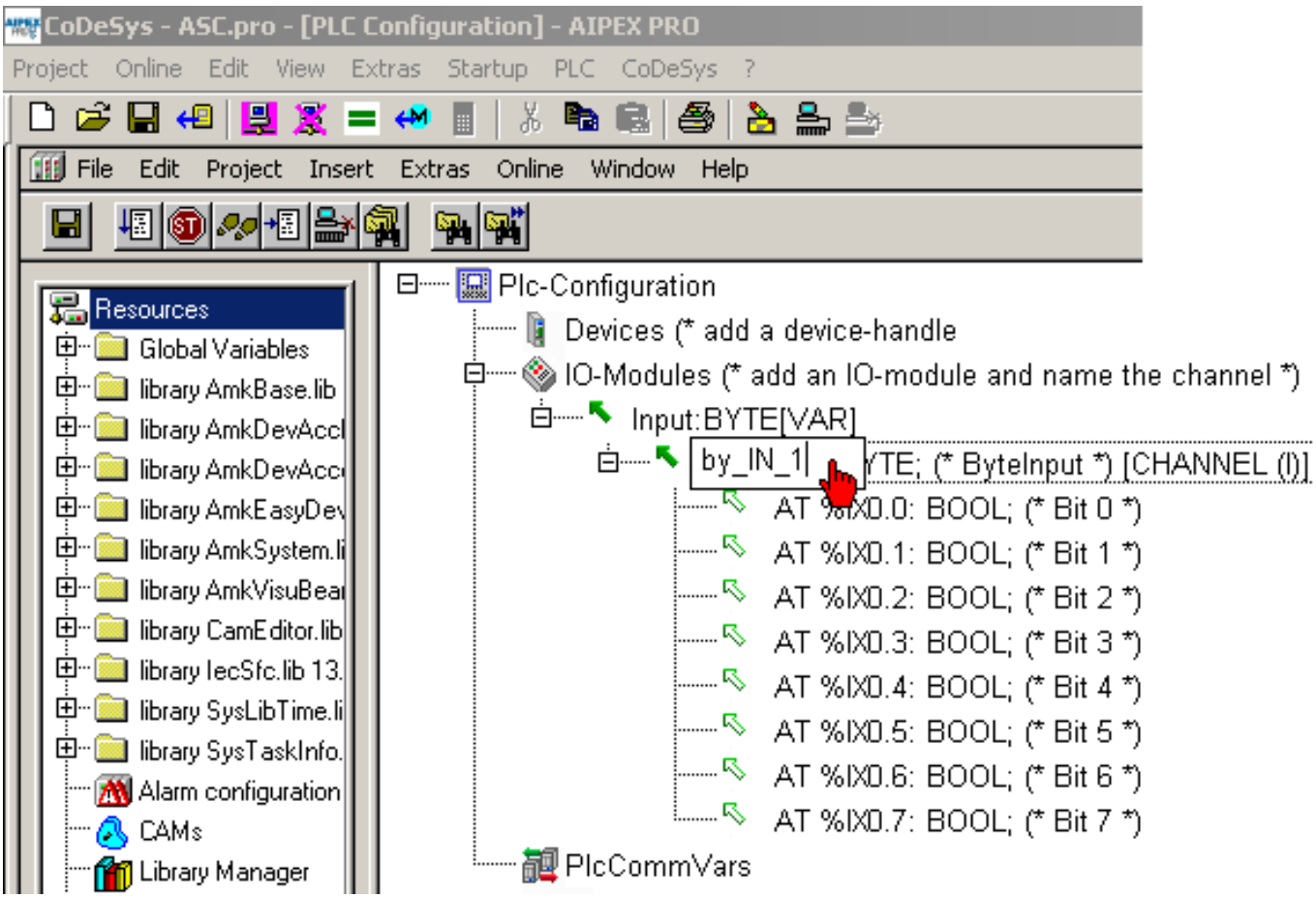


For each in- and output block, one variable needs to be set in CoDeSys.
Open the **Controller configuration** for that in your CoDeSys project under **Resources**.
Right click on the I/O modules insert new variables.

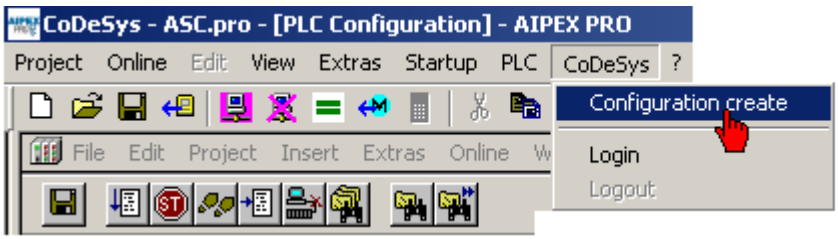
Input: Data is read in from the drives to the PLC.
Output: Data is written from PLC to the drives.



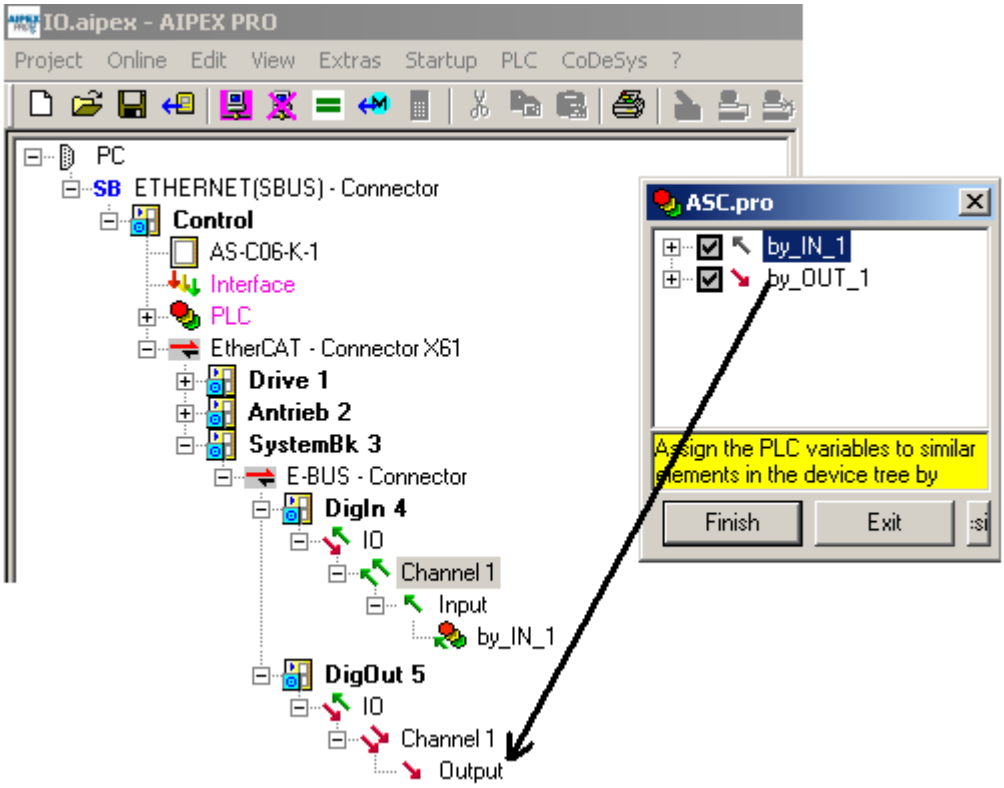
For each physically existing module, a symbolic variable needs to be created.



Afterwards, start the automatic message configuration creation.

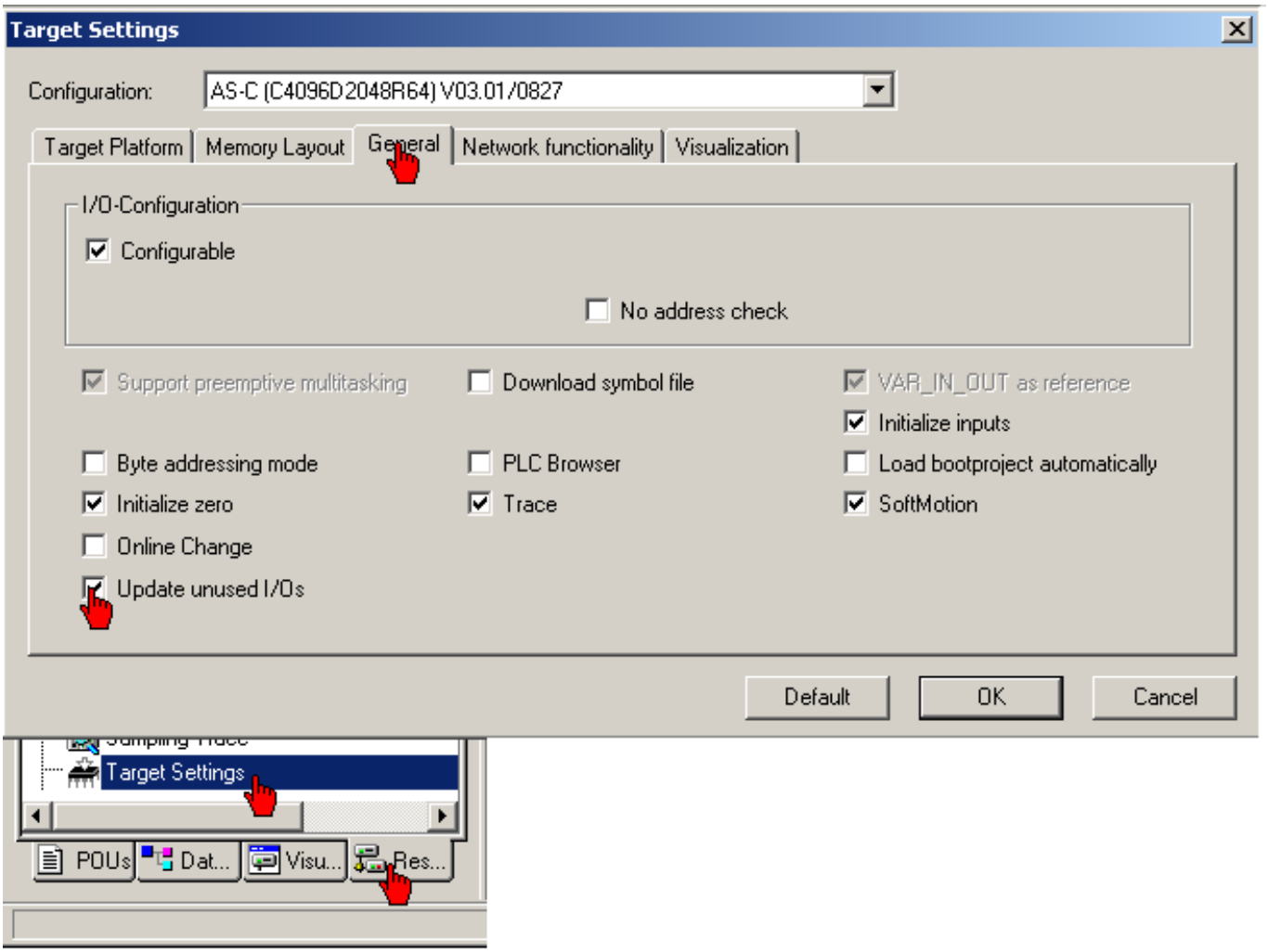


Assign the symbolic variables to the physically existing in- and outputs.
After the message configuration has been created, the system needs to be rebooted.



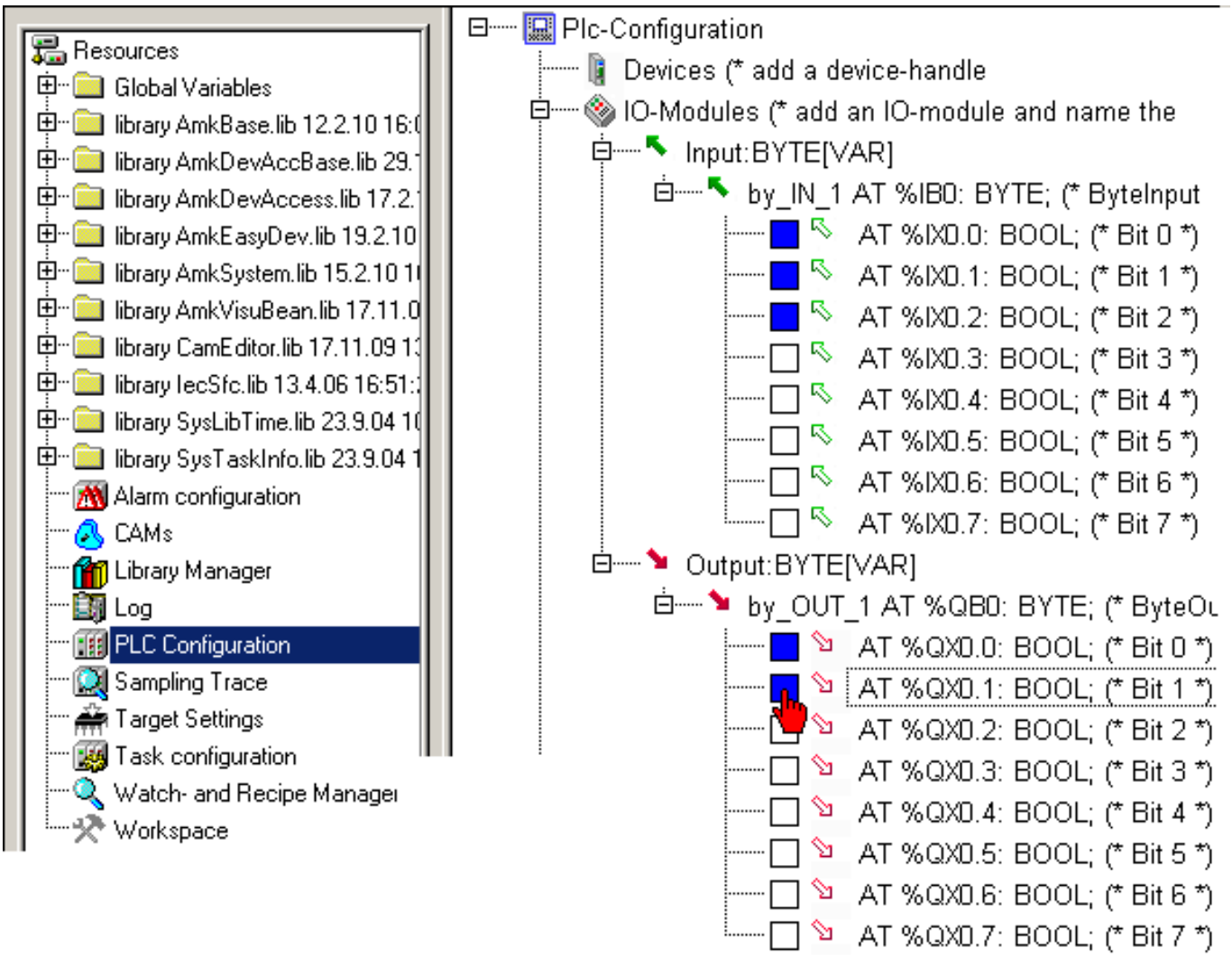
Function test

Activate the checkbox **Update unused I/Os** under **Resources** -> **Target setting** -> **General**
or assign a symbolic name for each I/O.

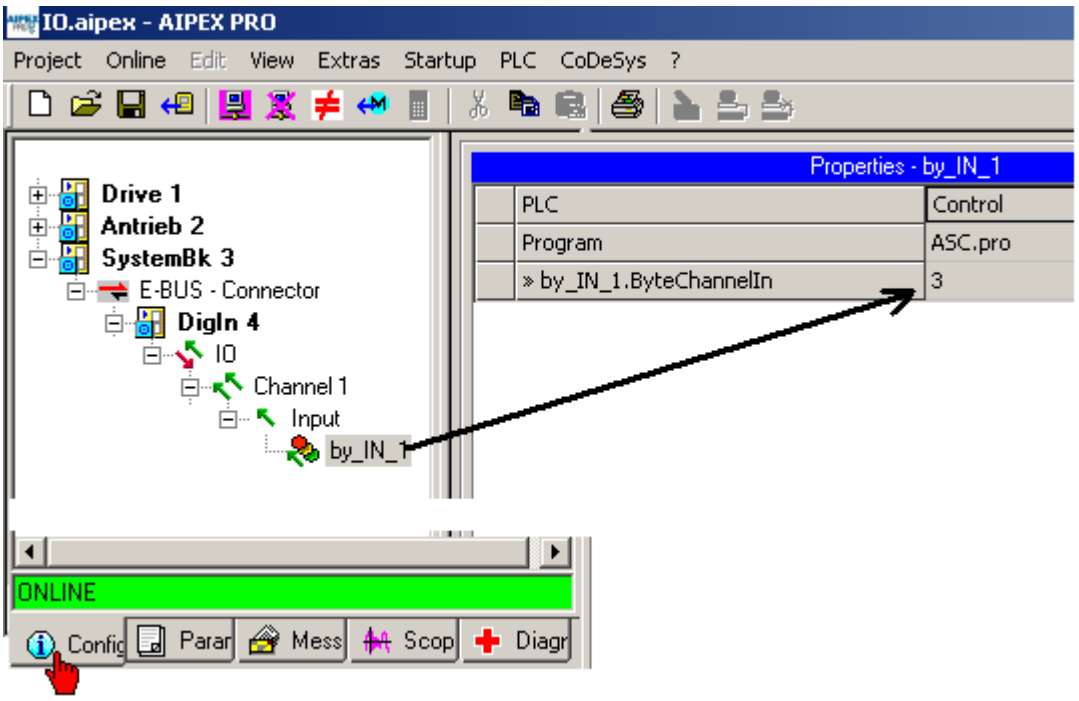


Function test: Prerequisite: AIPEX PRO logged in, CoDeSys logged in, PLC running.

Output variables can be set by double clicking.



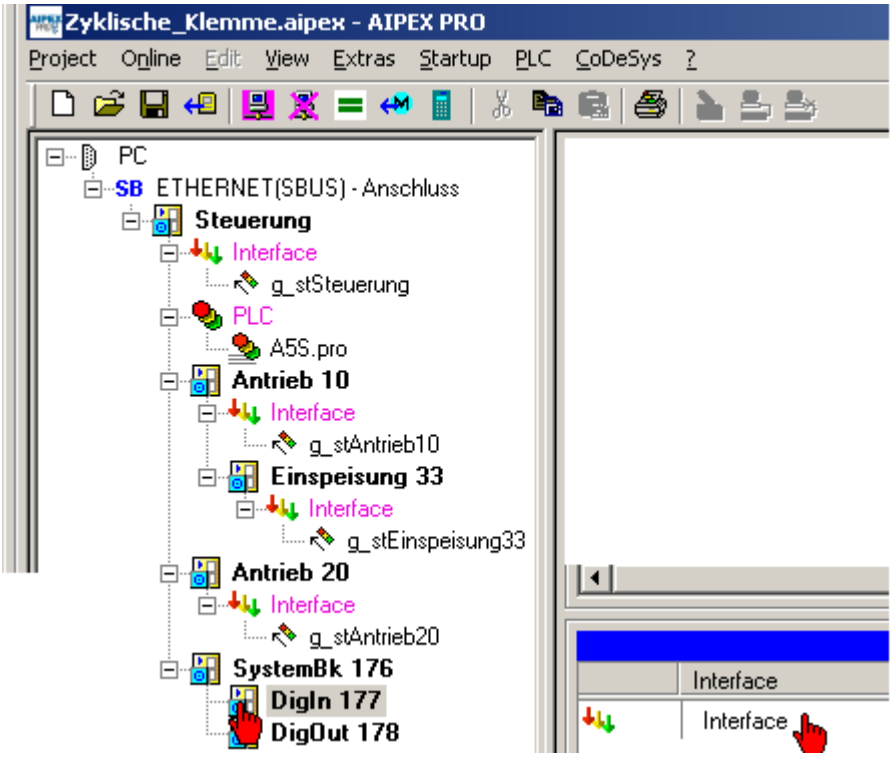
The current value is displayed decimally in the AIPEX PRO display under Configuration.



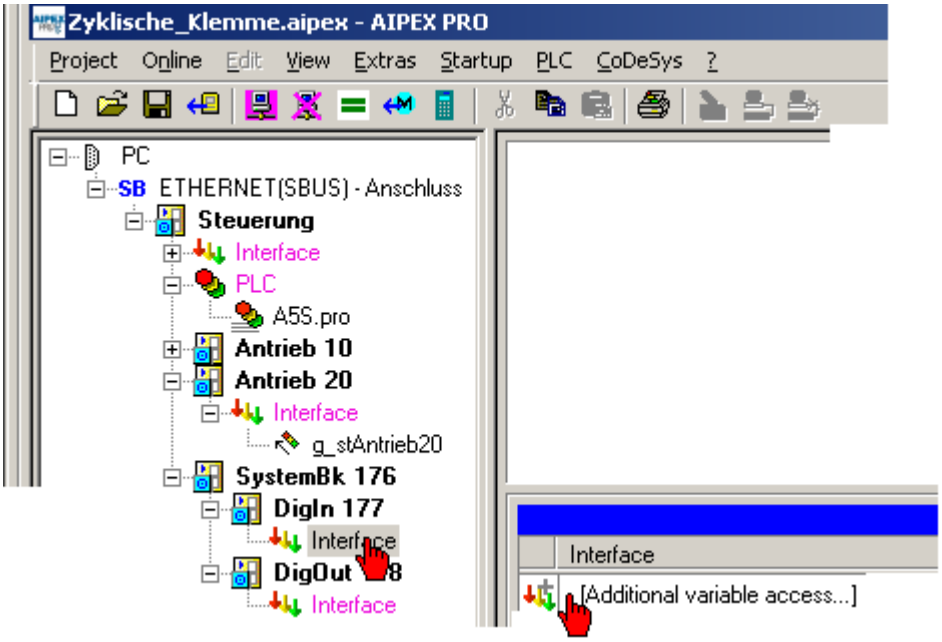
4.3.2.18.3 External cyclical IO terminal

The cyclical reading and writing of an IO mapping takes place via the device interface.

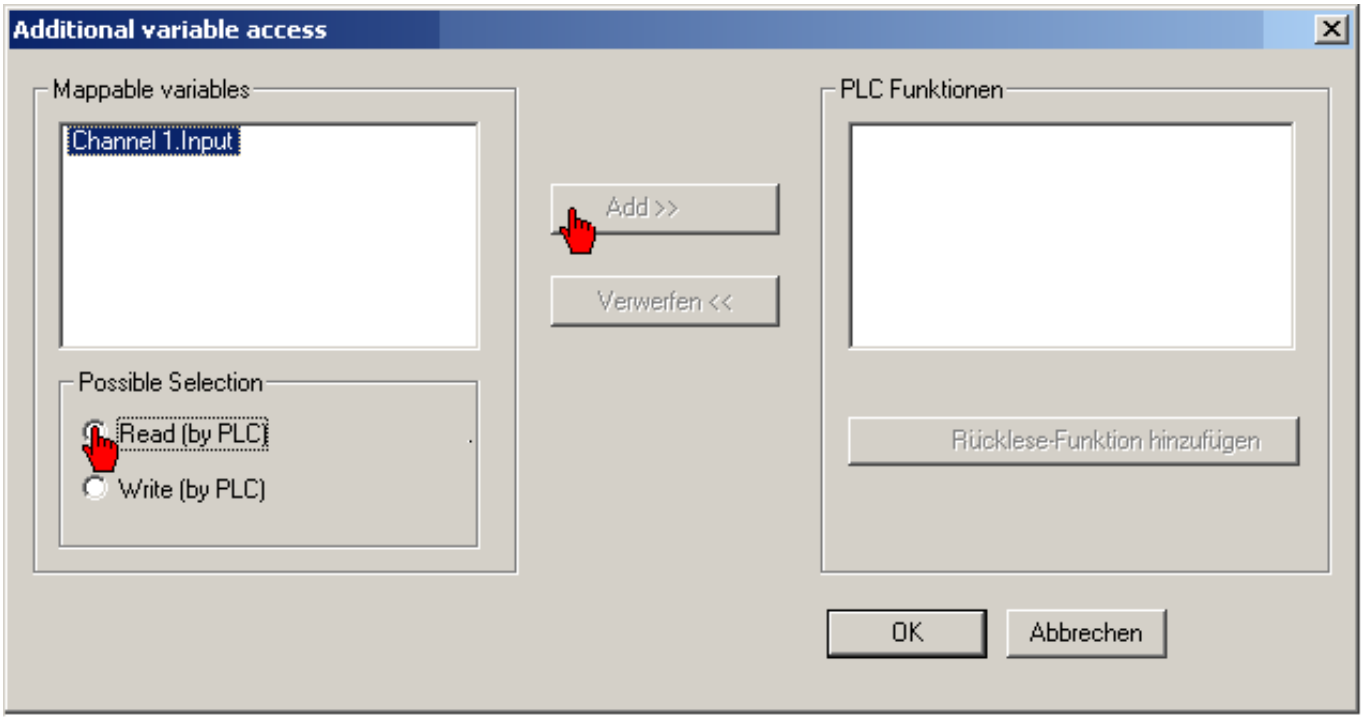
Apply therefore an interface to your terminal.

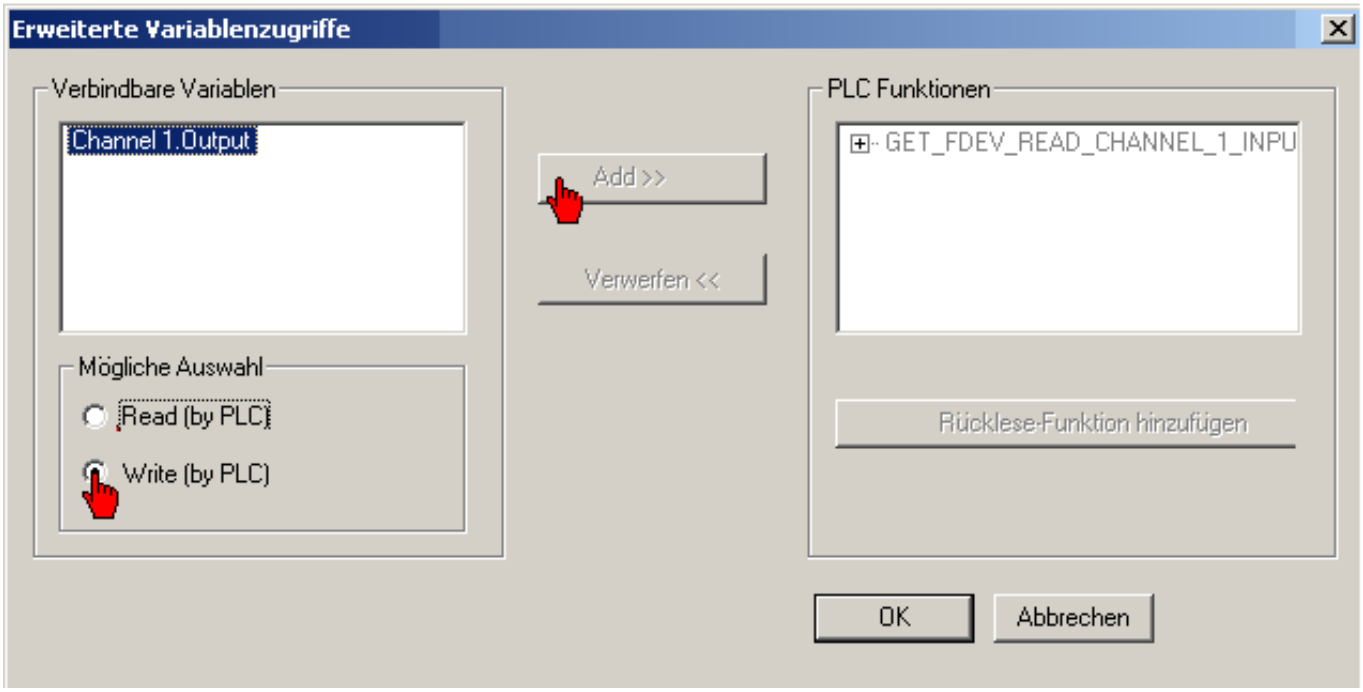


Link the interface to a formal function block. Call up the **Extended variable accesses** menu item for that.



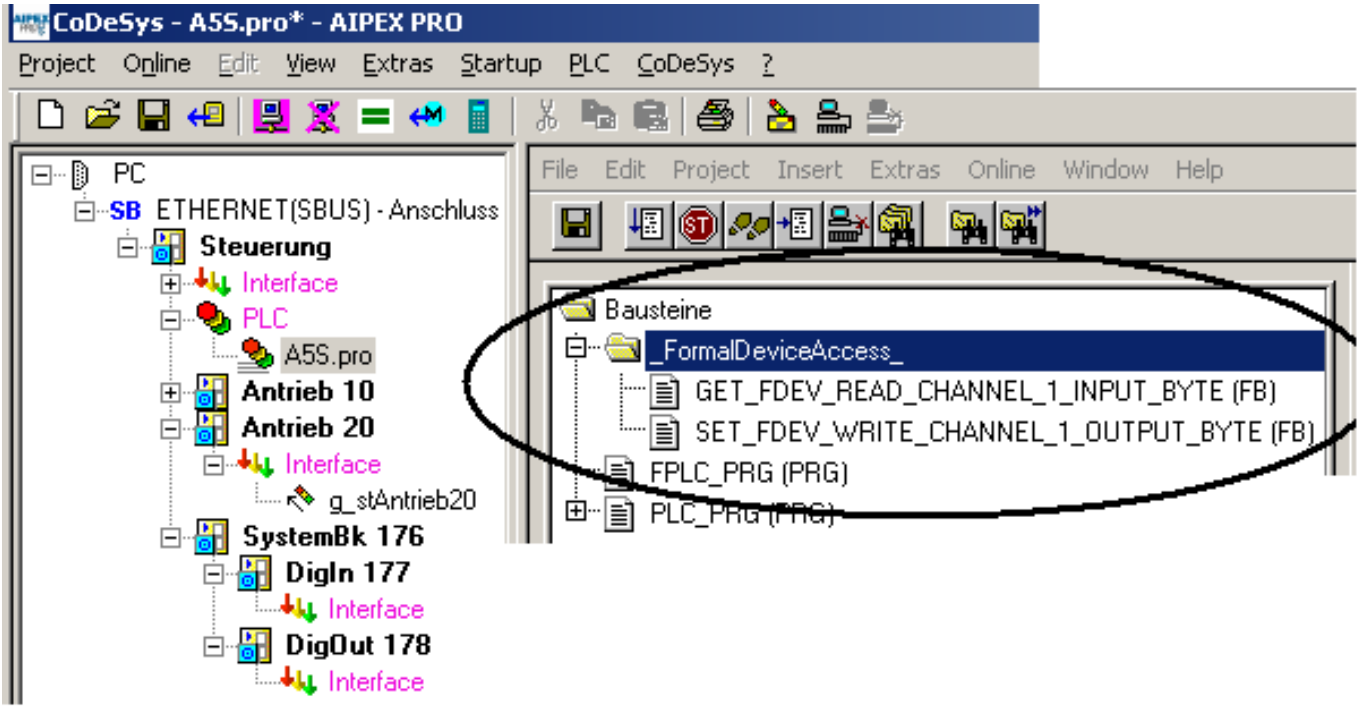
Choose Read (read through the PLC) for inputs.
Choose Write (write through the PLC) for outputs.
All values that are available can be found in the variable list.
The variable selection is accepted by pressing the Add button.





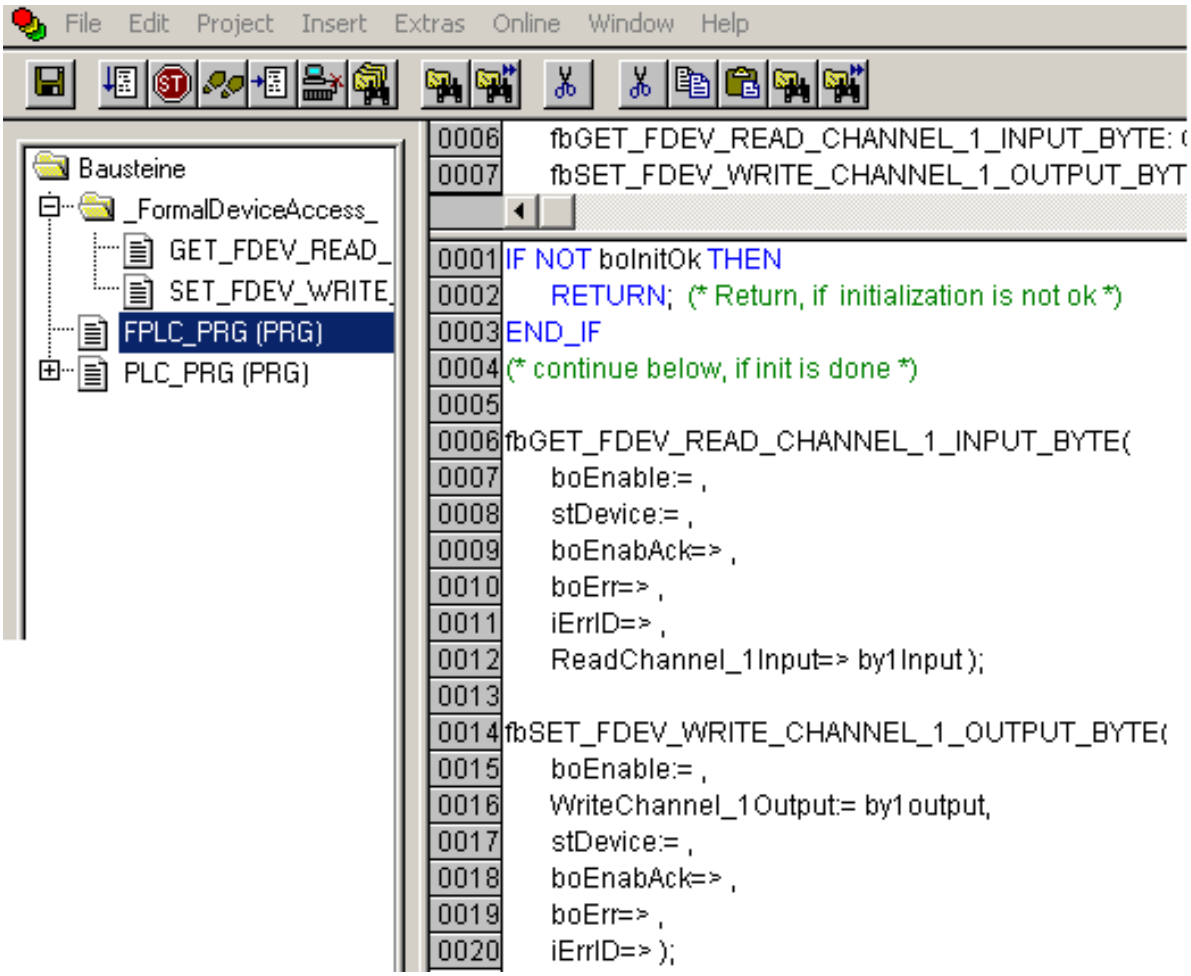
Open your CoDeSys project.

In the **_FormalDeviceAccess_** directory, you will find the newly created function blocks.

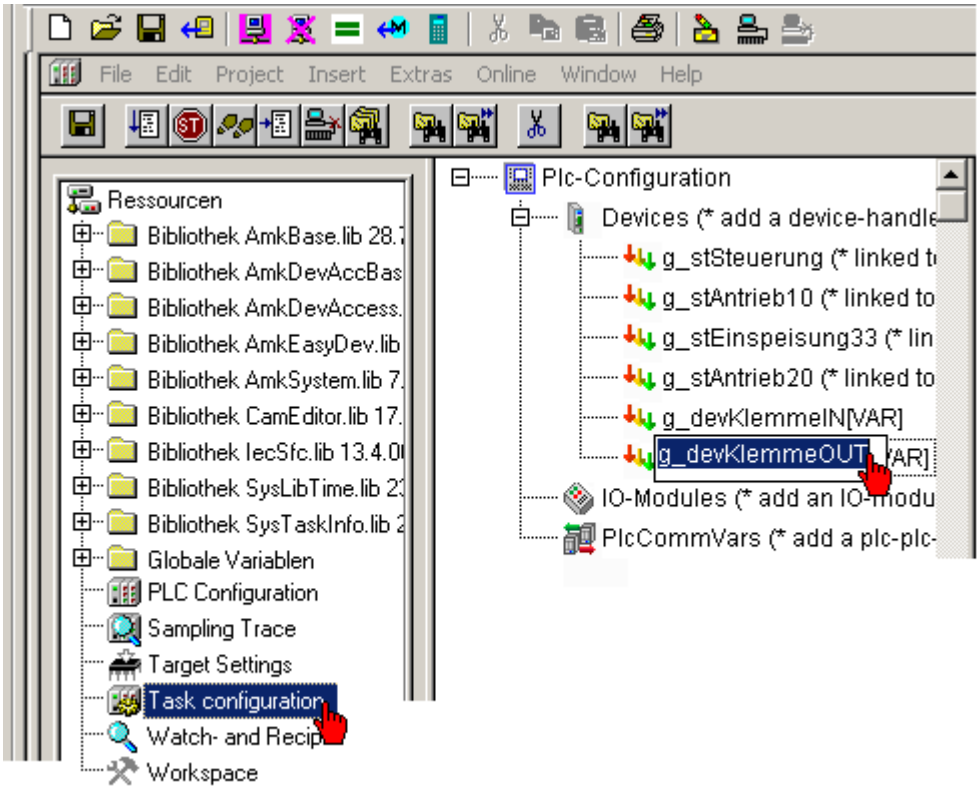


Insert the function blocks into the real-time level FPLC_PRG.

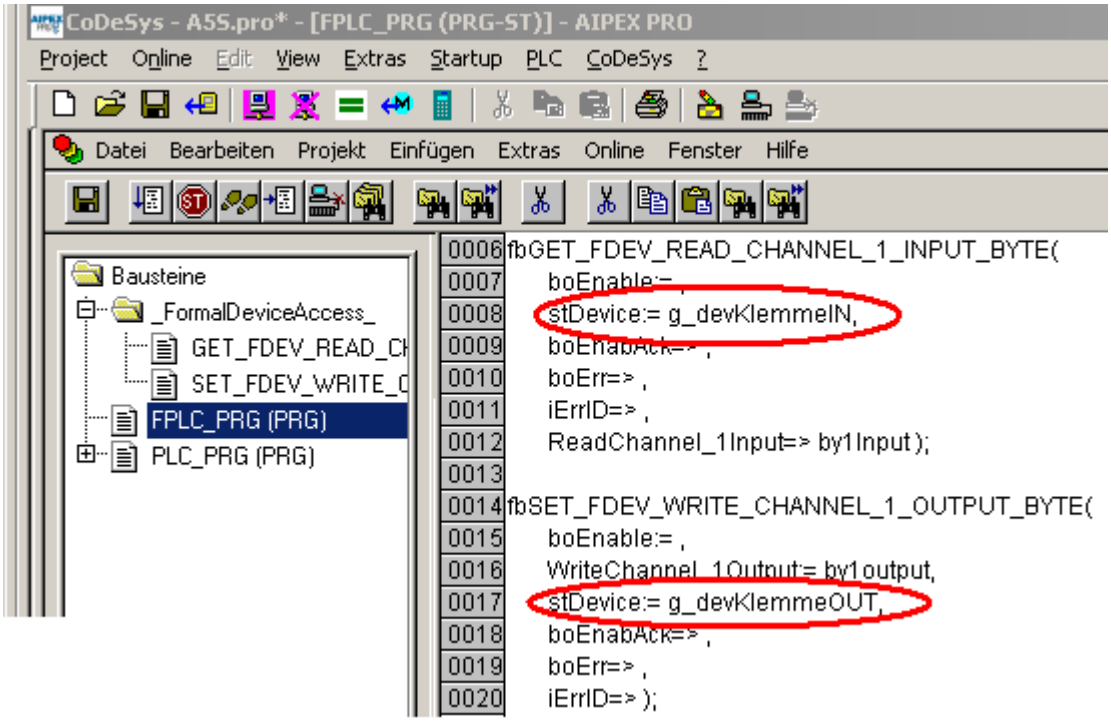
Create an instance of the module.



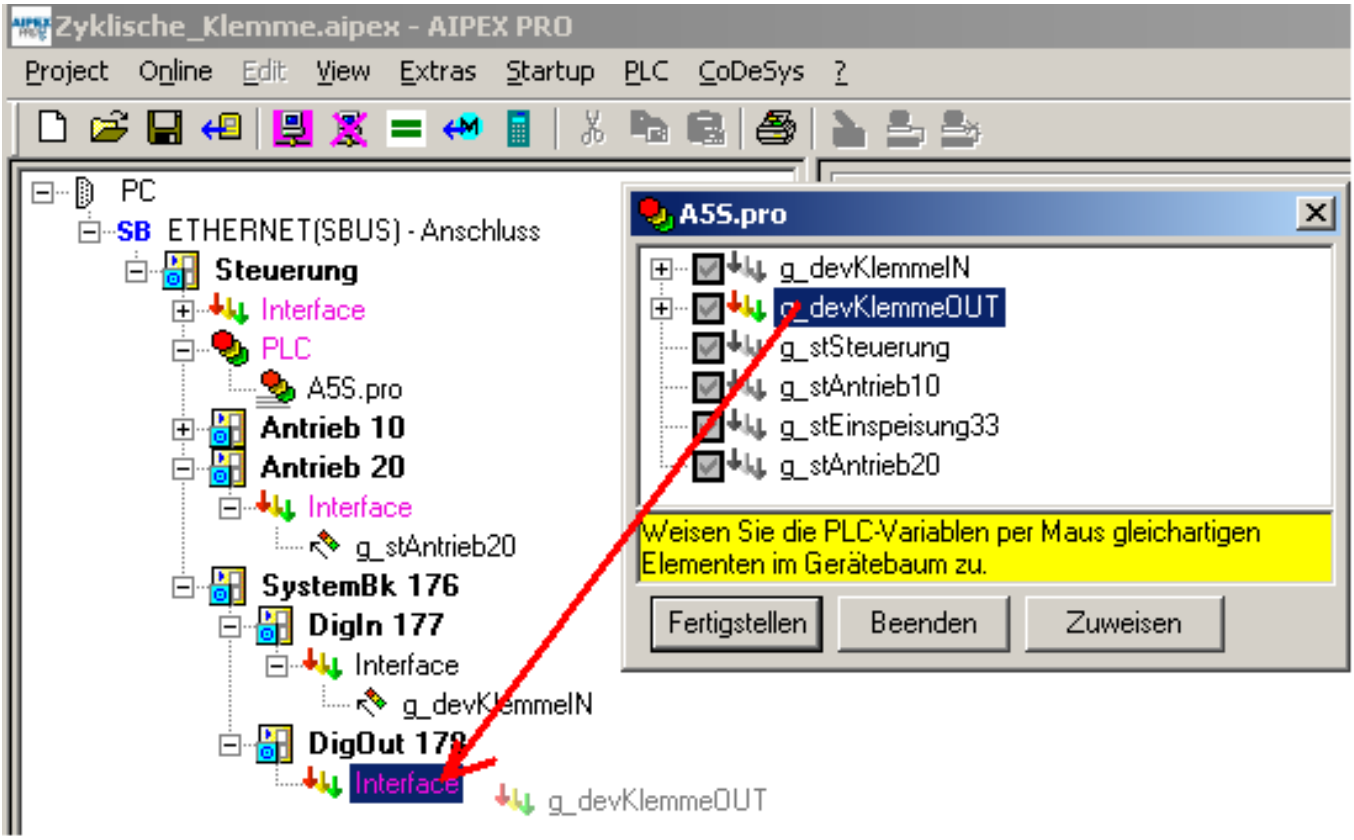
'stDevice' always has to be assigned a symbolic name. It is linked with the device interface of the external terminal.
Save symbolic names for the terminals in the **Controller configuration** under **Devices**.



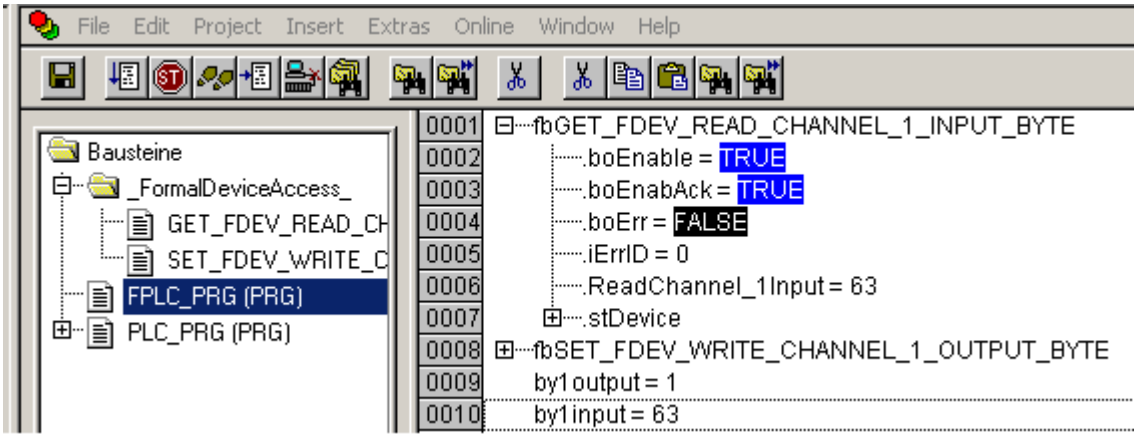
Link 'ST_DEVICE' with the symbolic device names.



Activate the automatic message configuration, menu item **Generate CoDeSys message configuration**.
Drag and drop the symbolic device name then onto the corresponding device interface.



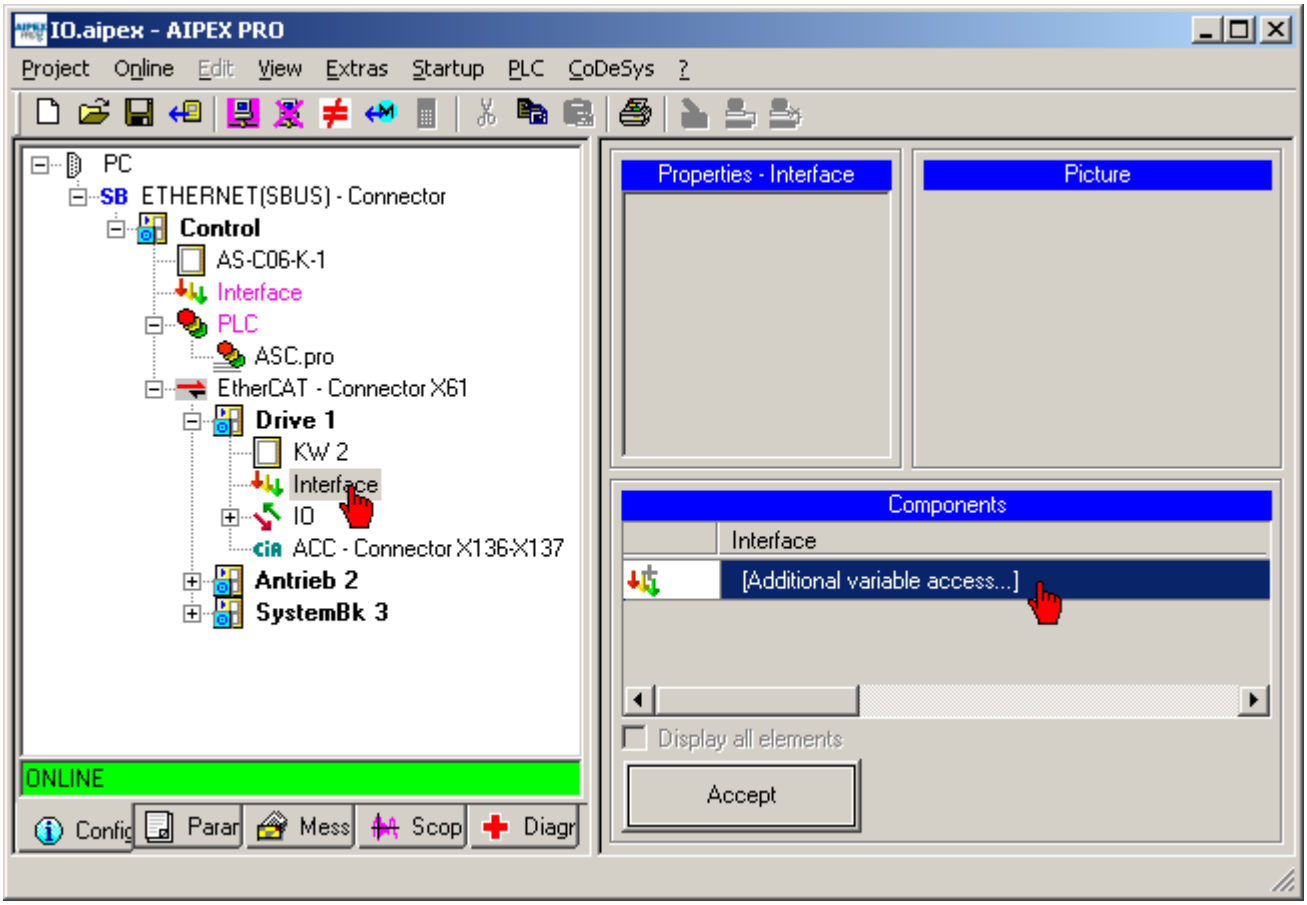
Set 'boEnable' = TRUE to start the function block.



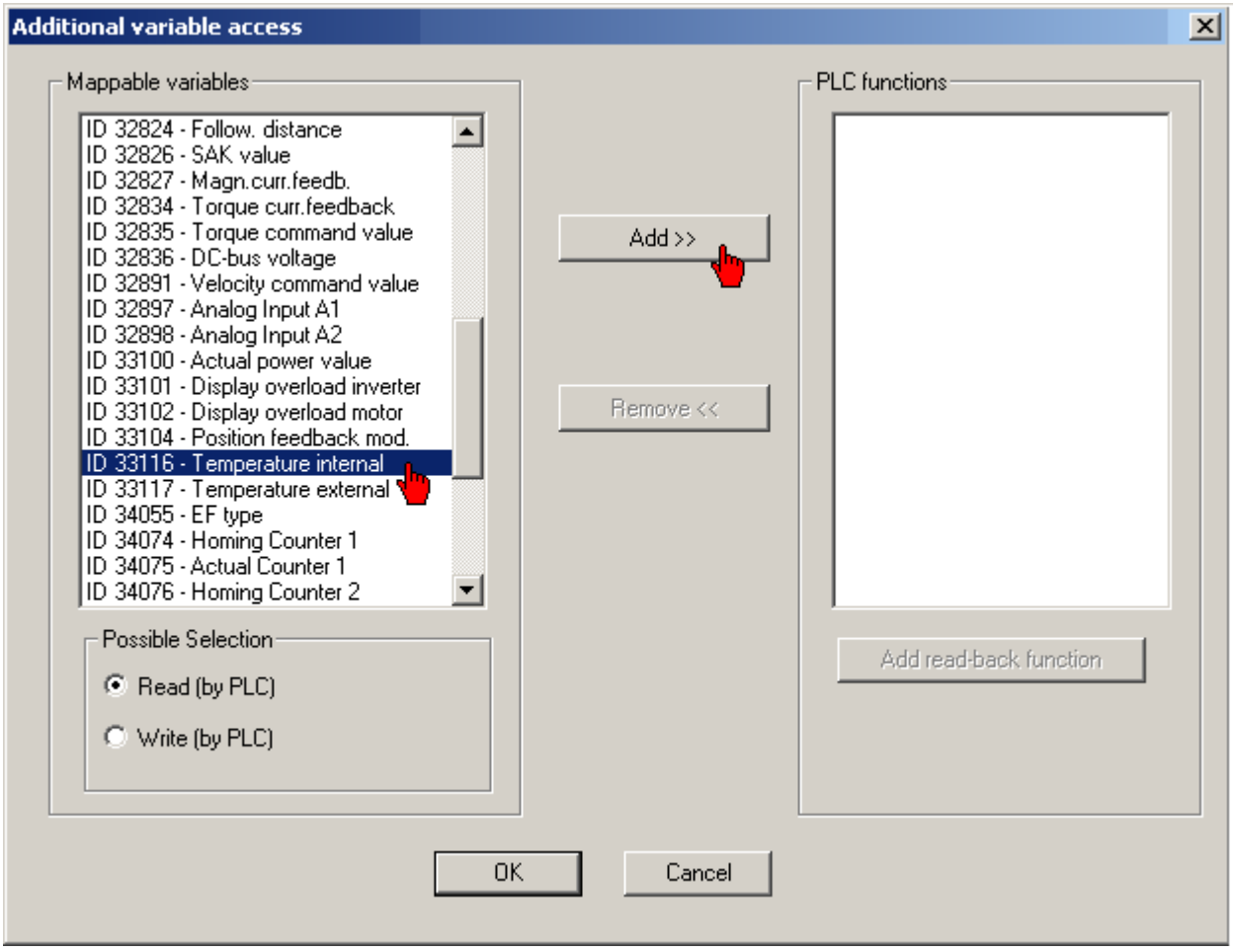
4.3.2.18.4 Additional variable access

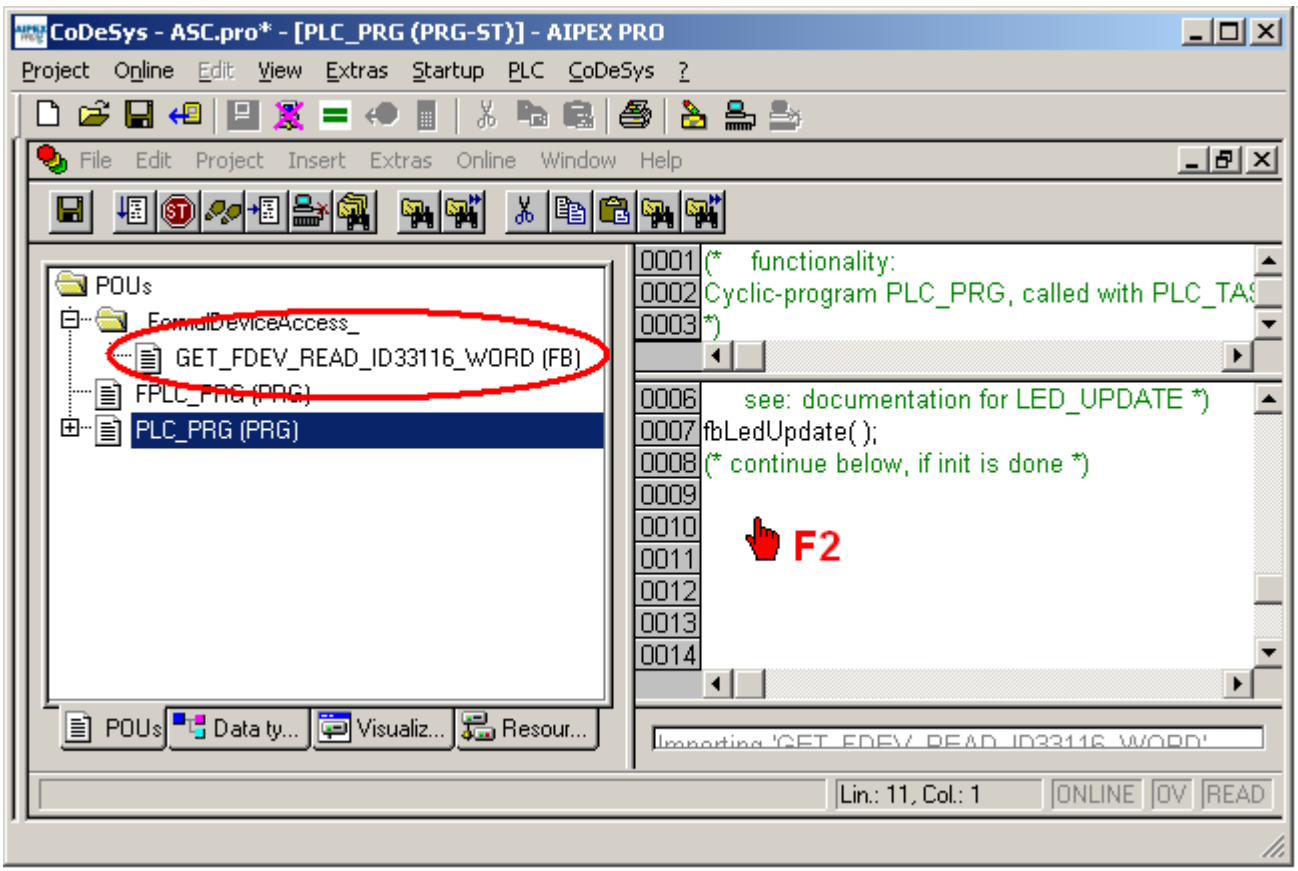
The AMK CoDeSys libraries provide a selection of function modules to exchange real-time bit signals, actual value, setpoints, etc., between drive and PLC. Non-existing (formal) modules can be created using the option extended variable accesses.

Select for that the drive (interface) for which a new function module should be created.

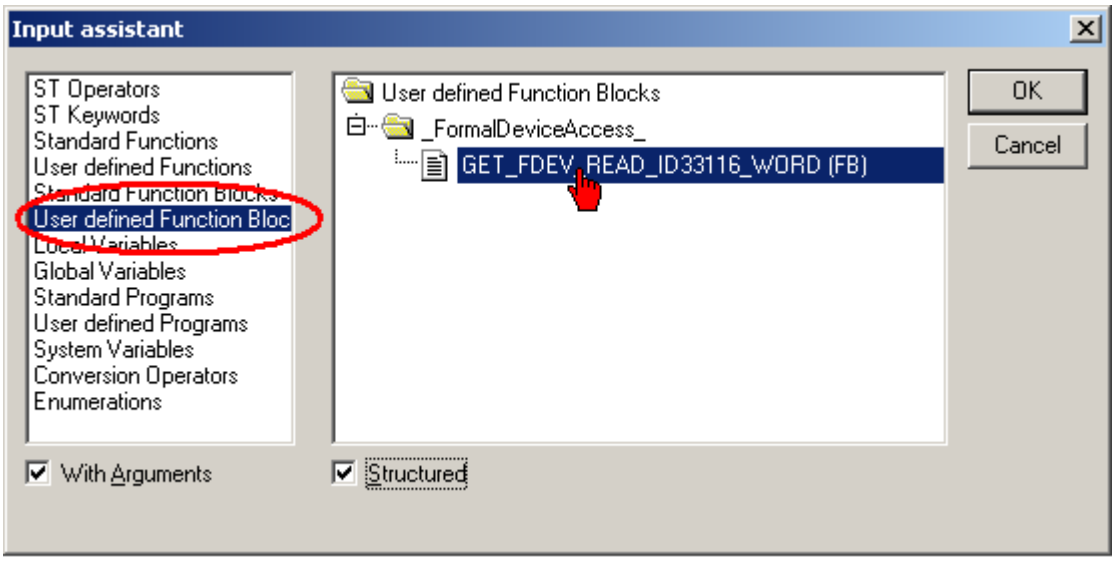


In the first step, choose whether the PLC should have reading or writing access to the drive.
All values that are available in the drive can be found in the variable list.
The variable selection is accepted by pressing the **Add** button.

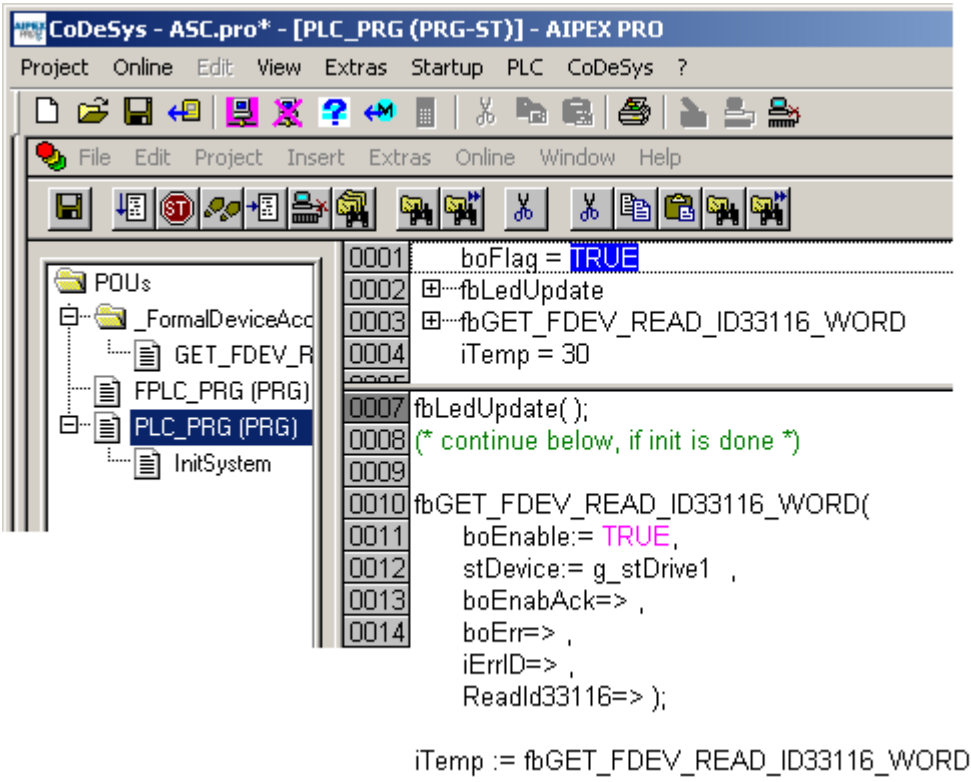




Using the Help Manager F2, the function blocks can be tied into the PLC project and be instanced.



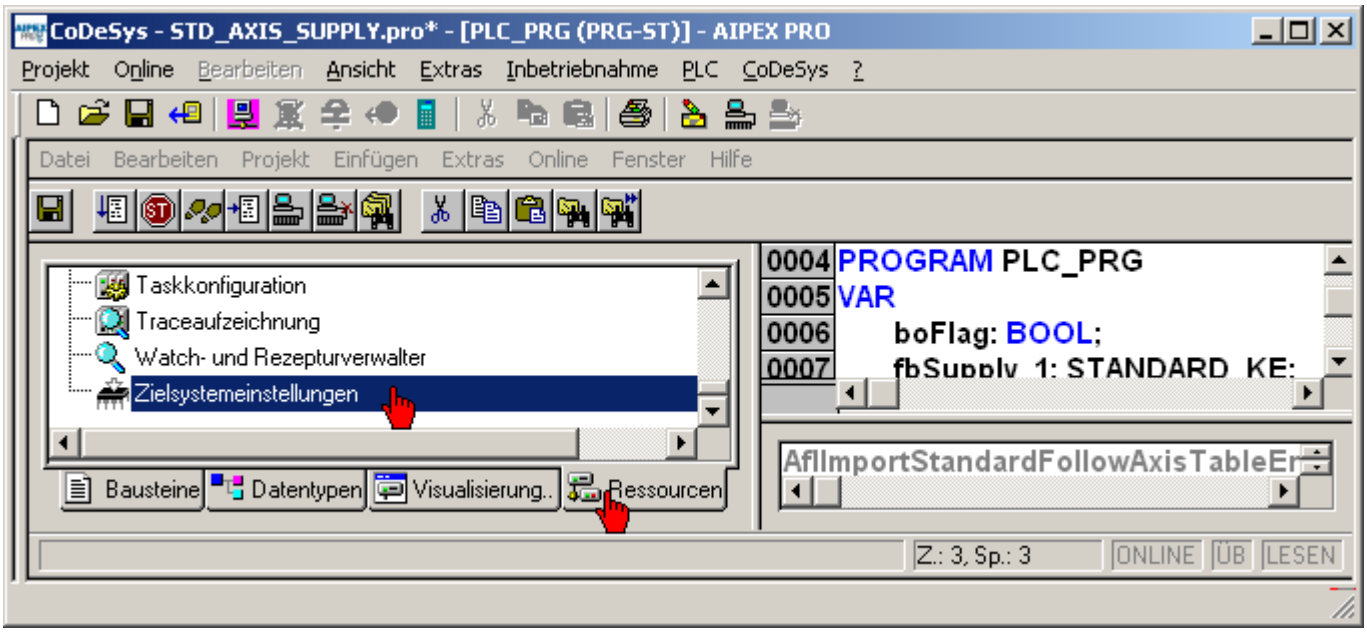
Observe the scaling respectively of the various units. Factor 10 in the example.



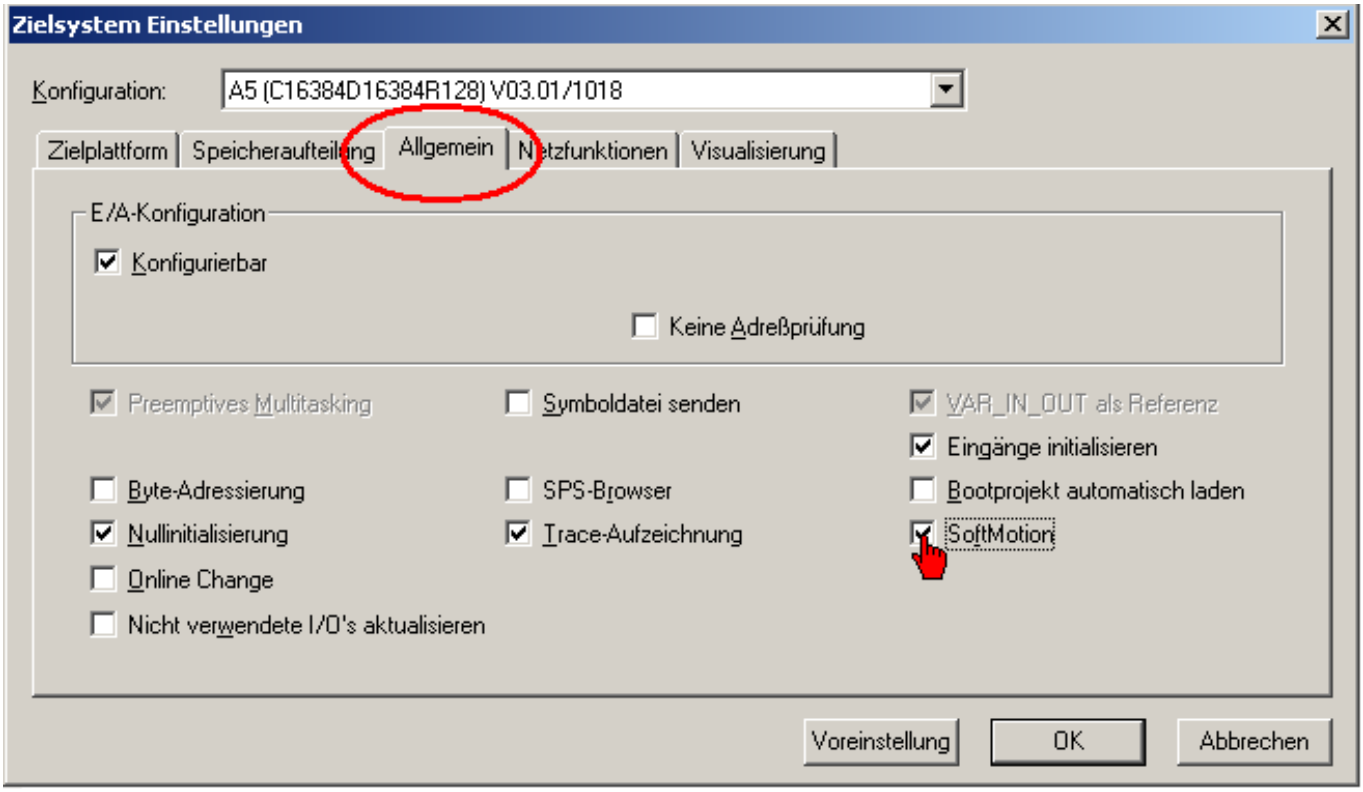
4.3.2.18.5 Creating a cam

This example demonstrates how to create a cam and link this to an FB of the Standard axes.

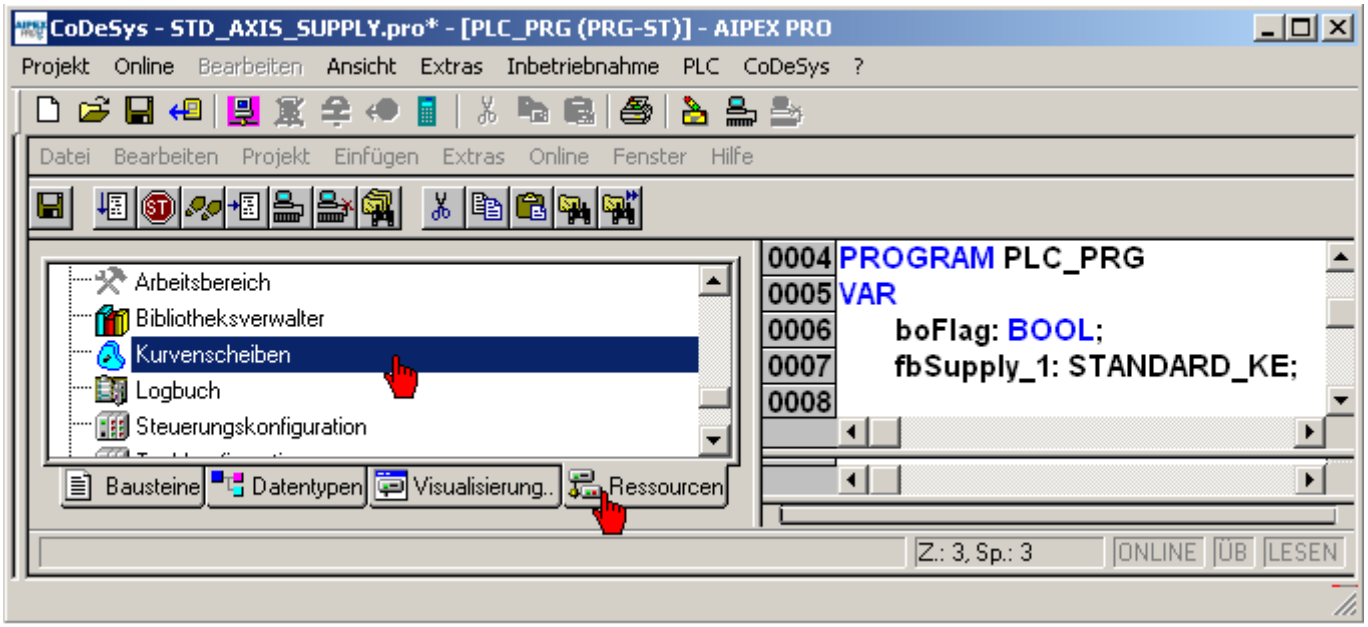
Firstly, open the window **'Target System Settings'** under **'Resources'**.



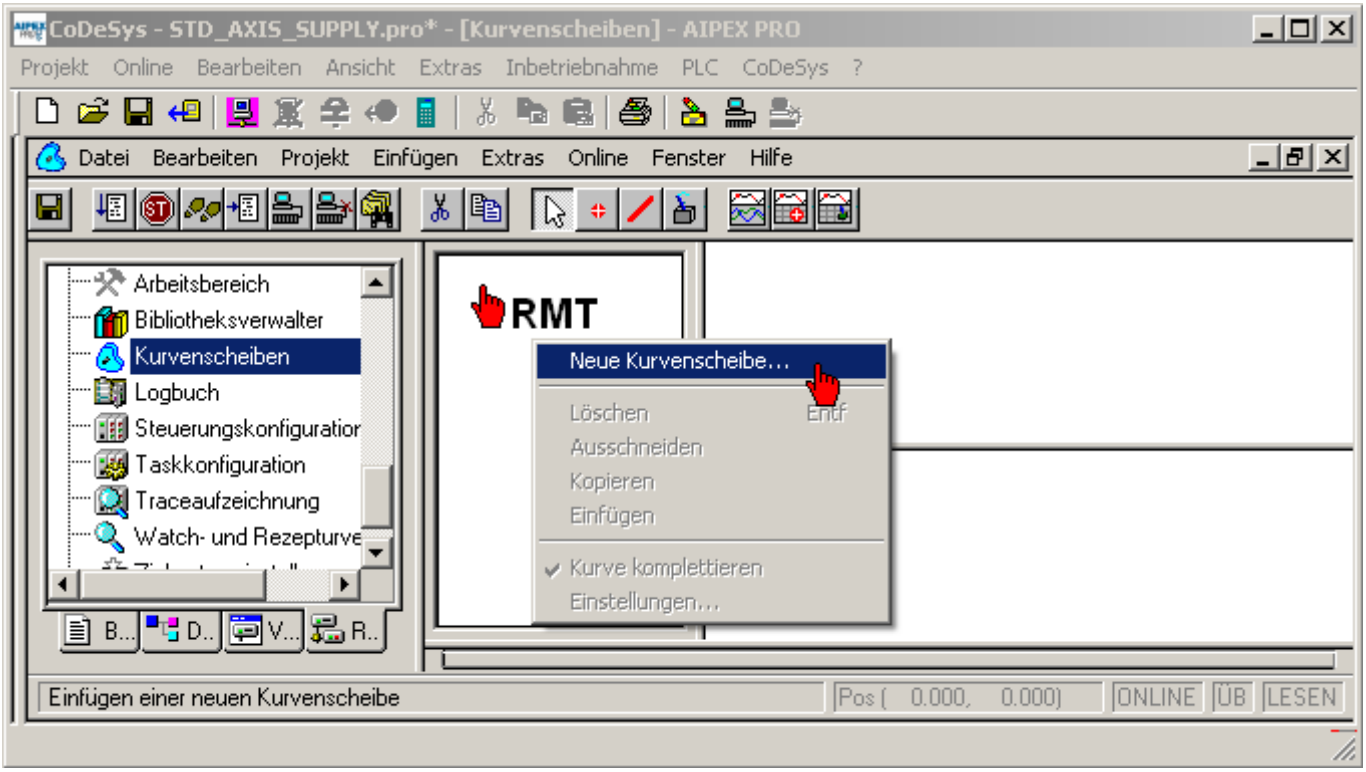
Enable **'SoftMotion'** under **'General'**.



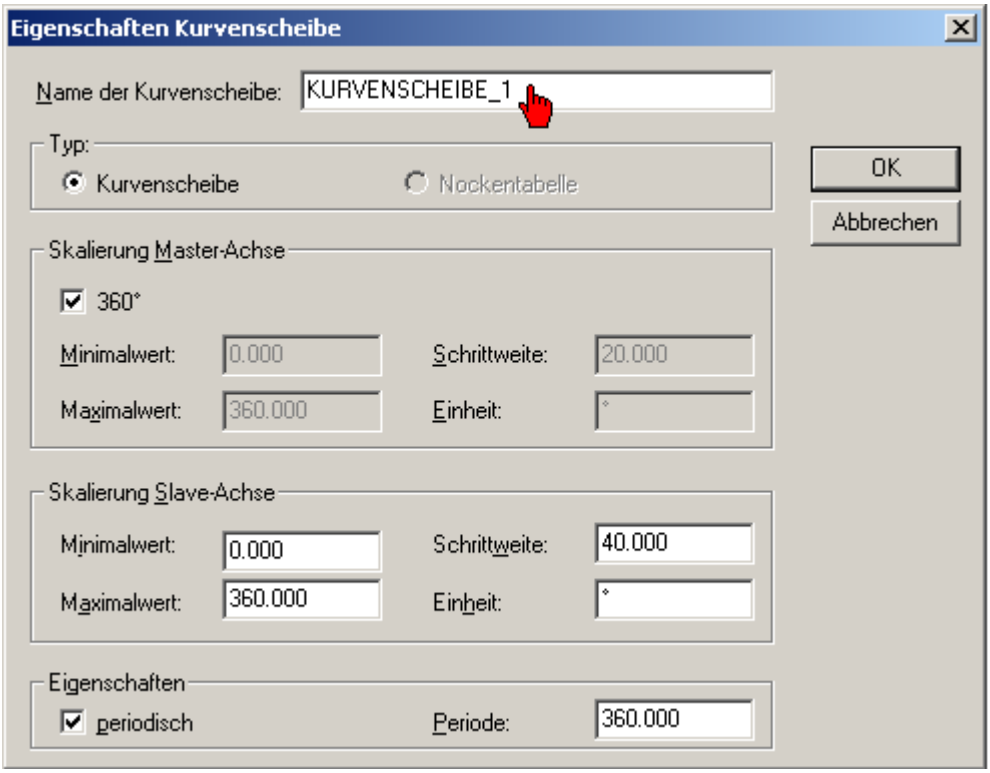
You will now see the menu **'Cam'** under **'Resources'**.

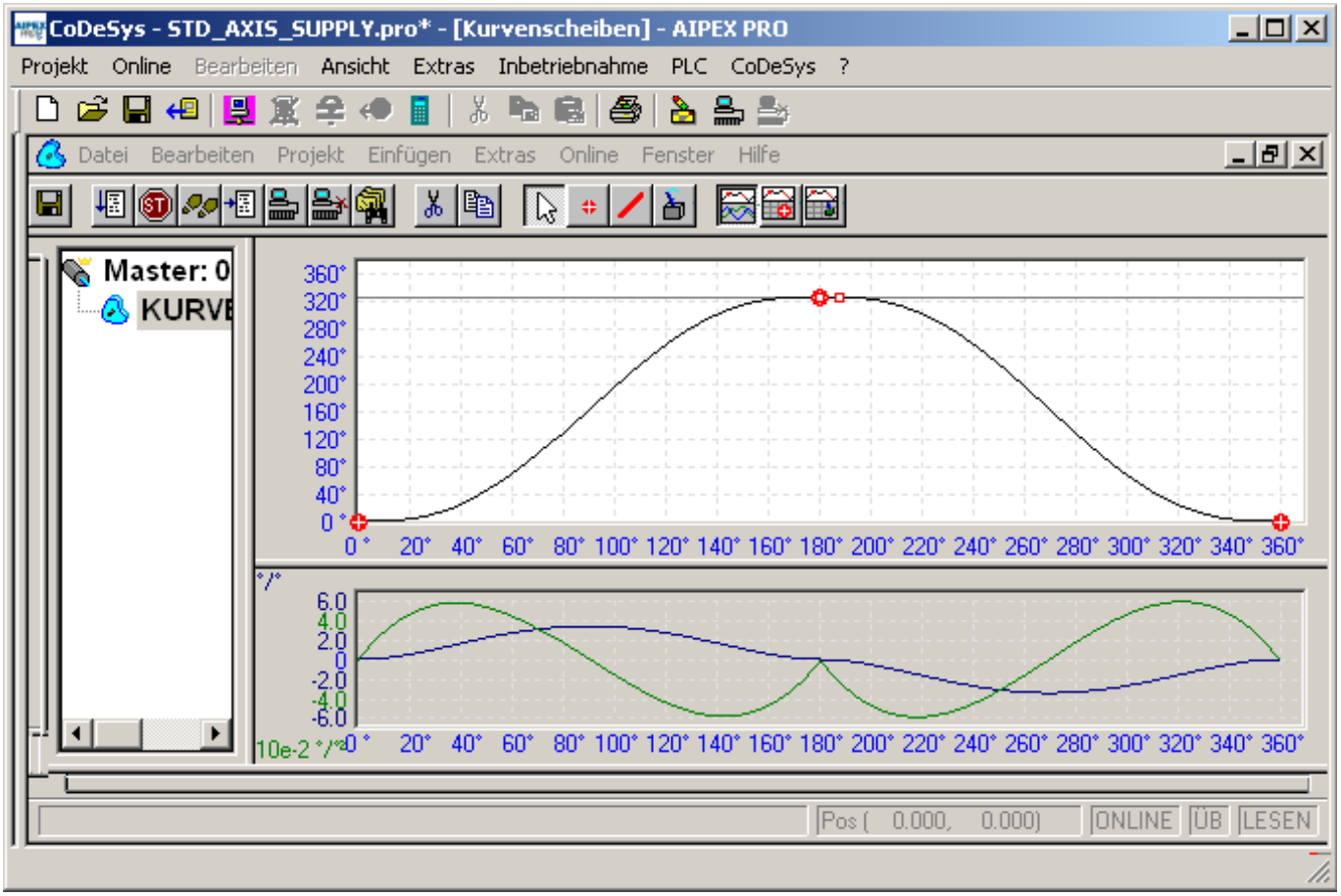


Right-click in the middle field to insert a new cam.

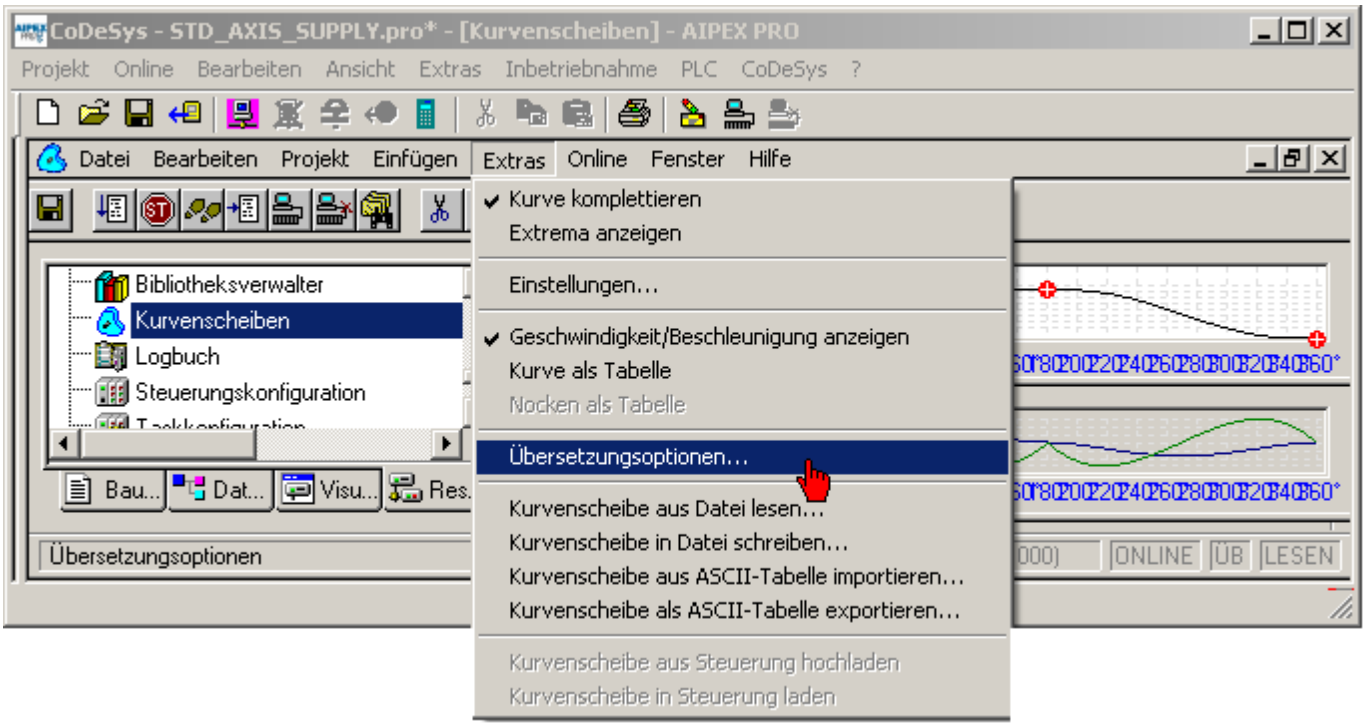


Under CODESYS Help, you will find detailed information on cams and the acconpanying settings. Please use these.



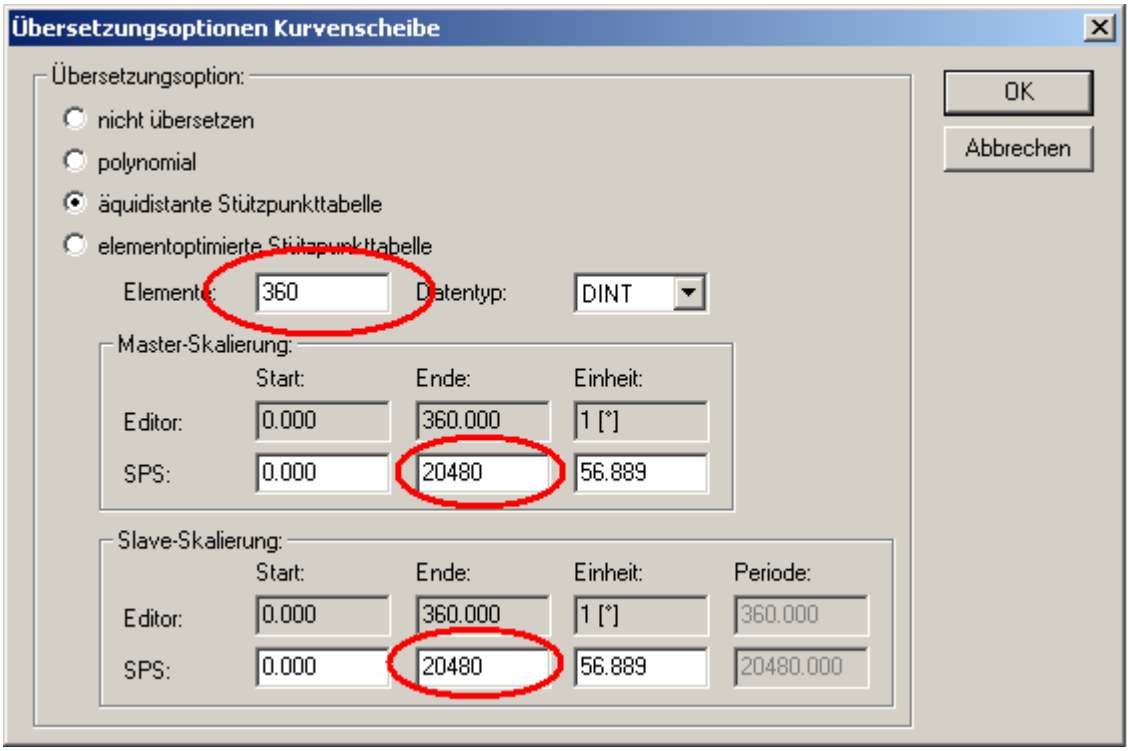


In the **'Translation options'** you can enter the number of support points and the scaling (motor encoder resolution).

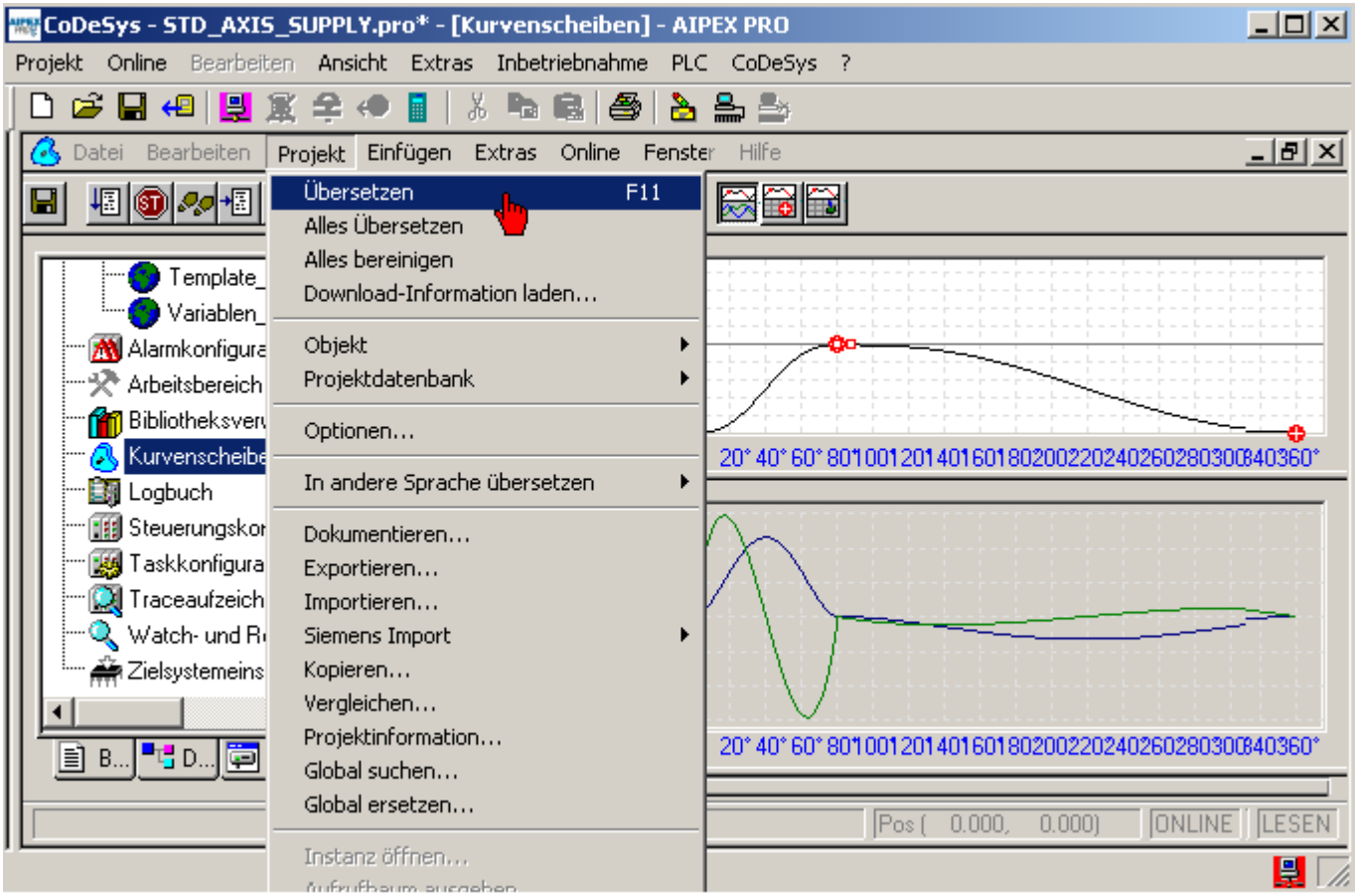


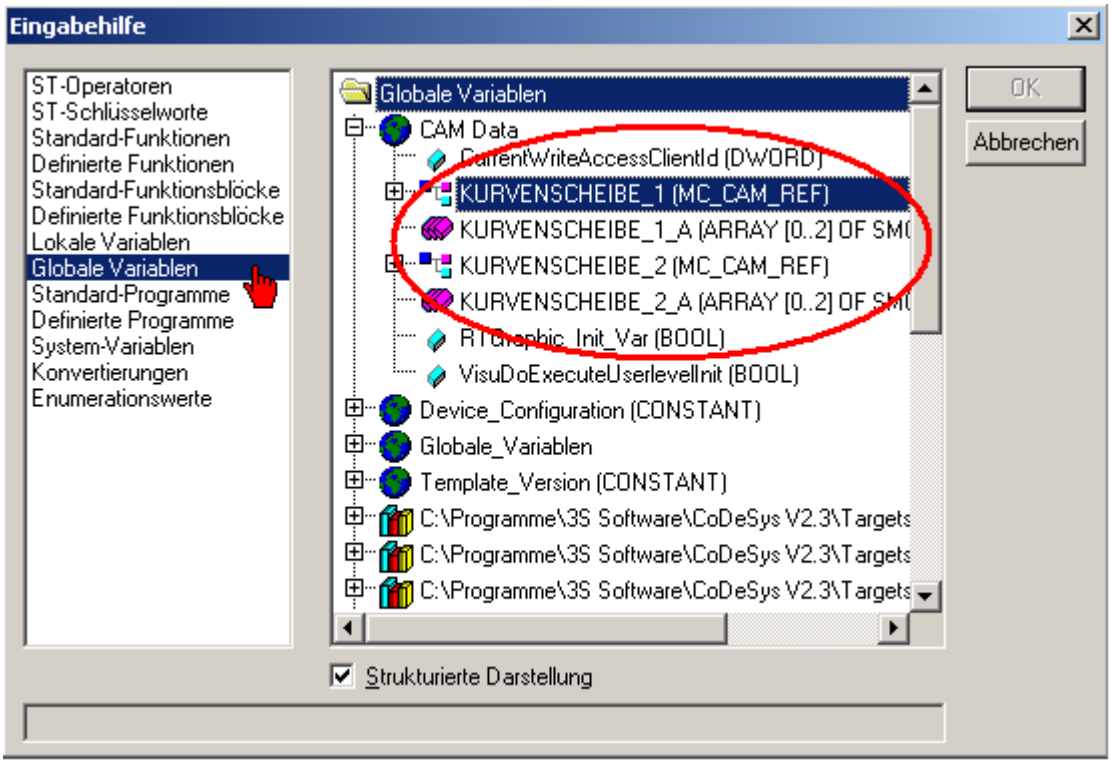
Equidistant support point table, max. 361 DINT values
Element-optimized support point table, max. 181 DINT values

Scaling: 360° corresponds to ID116 'Resolution motor encoder'

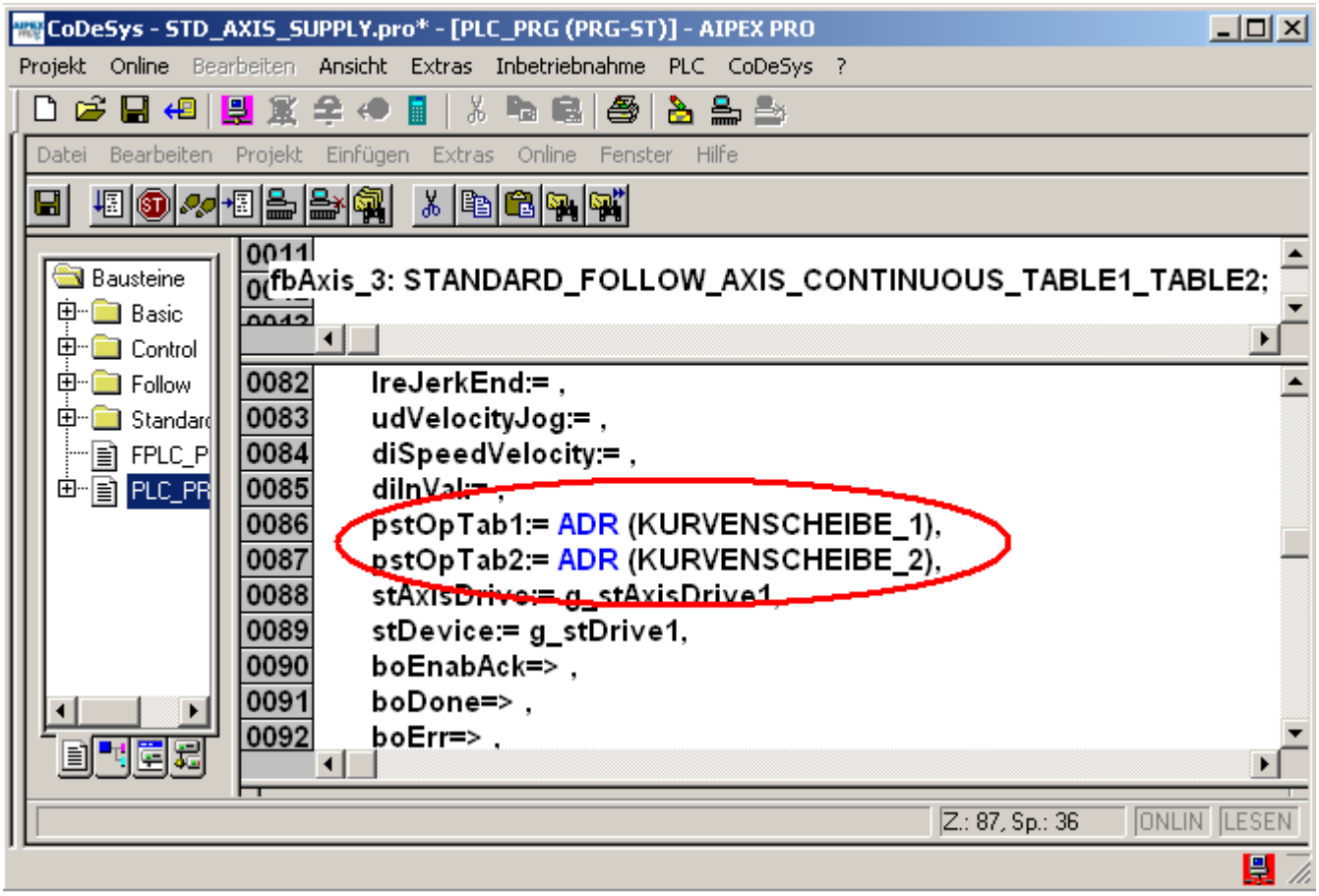


After translation, you can find the cam data under **'Global Variables'**.





Use the address function **'ADR'** to link the cam to your block.

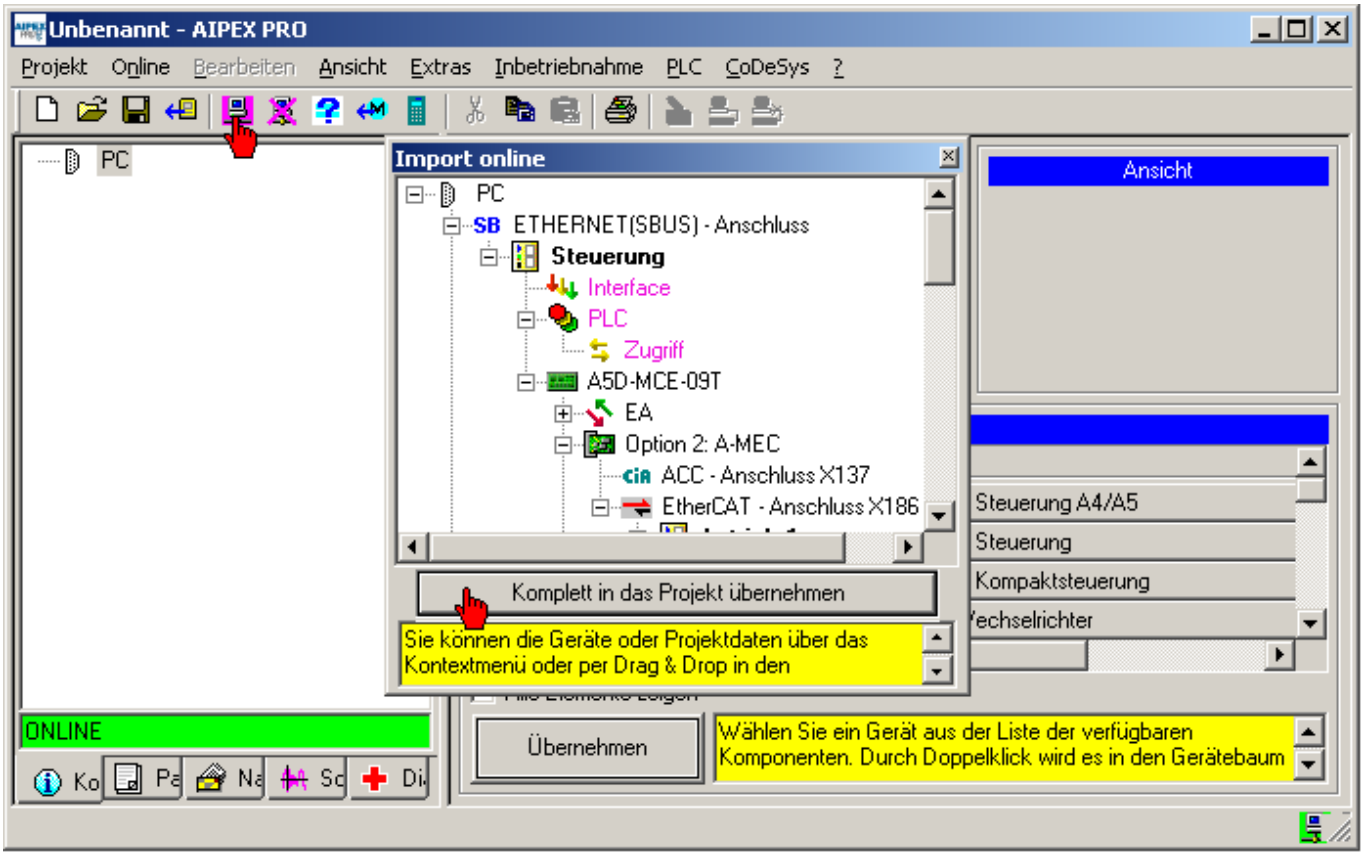


4.3.2.18.6 Importing a PLC project from AIPEXPRO

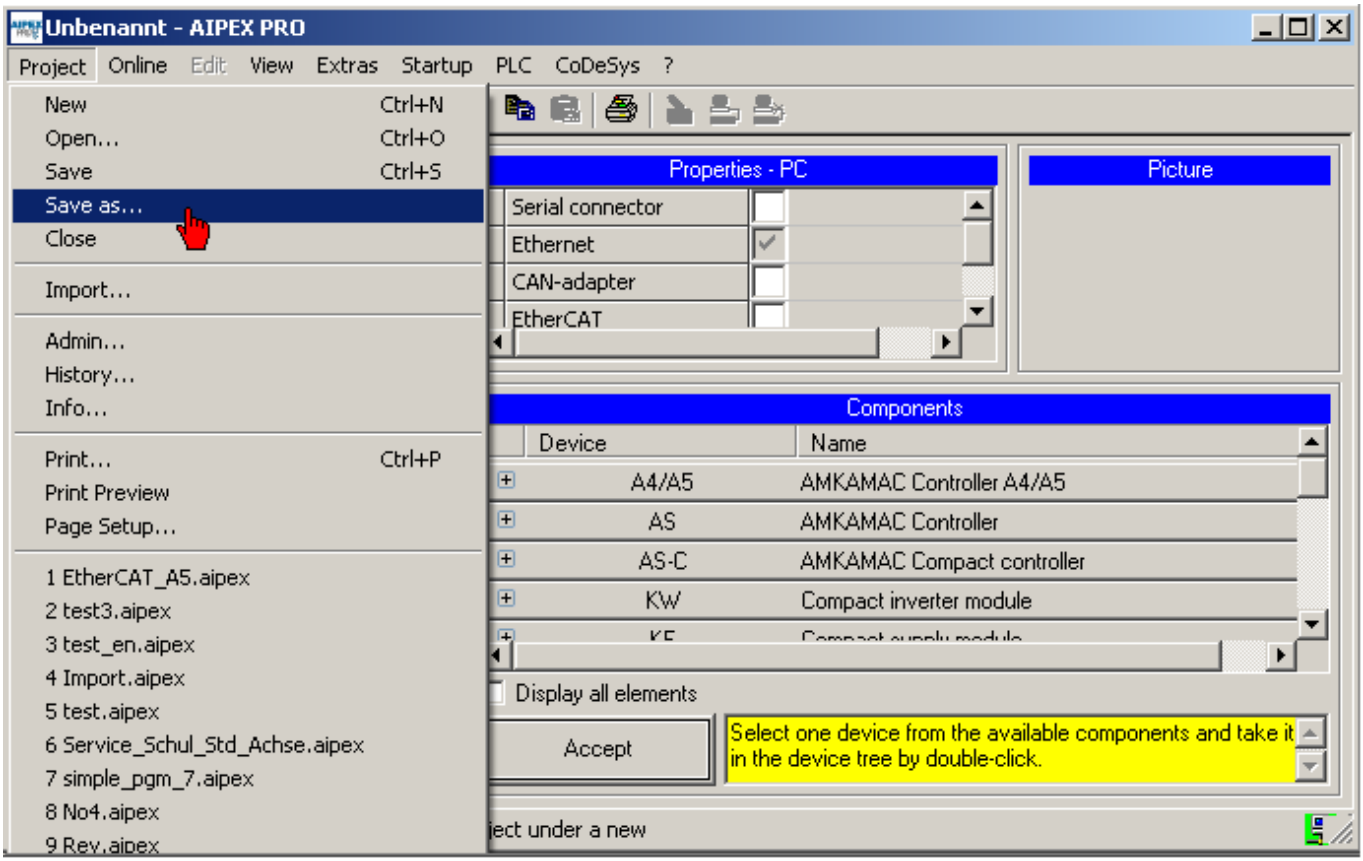
Open an empty AIPEX PRO project under **Project - New**.

Press the **Logon** button.

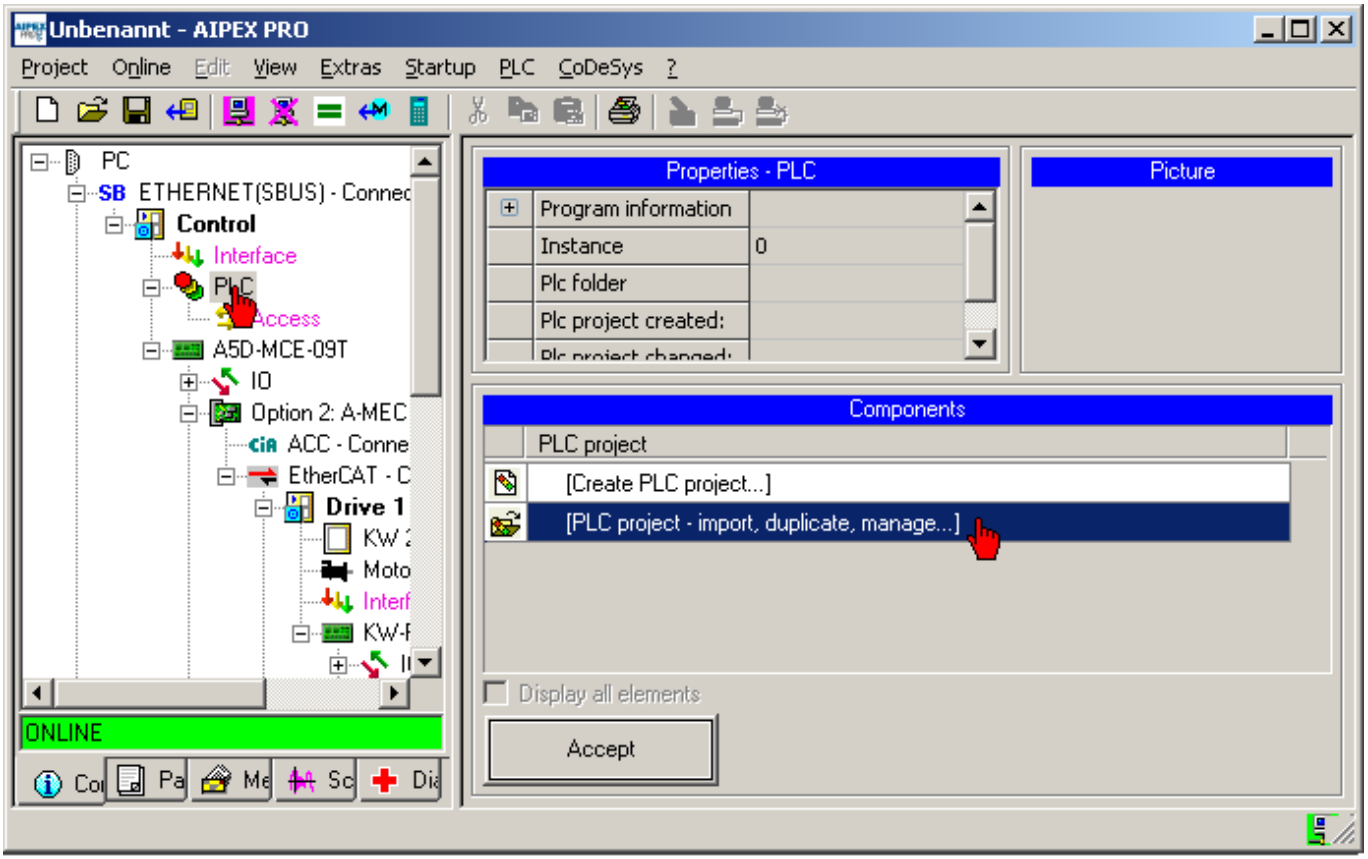
Take over the connected devices in your AIPEX PRO file.



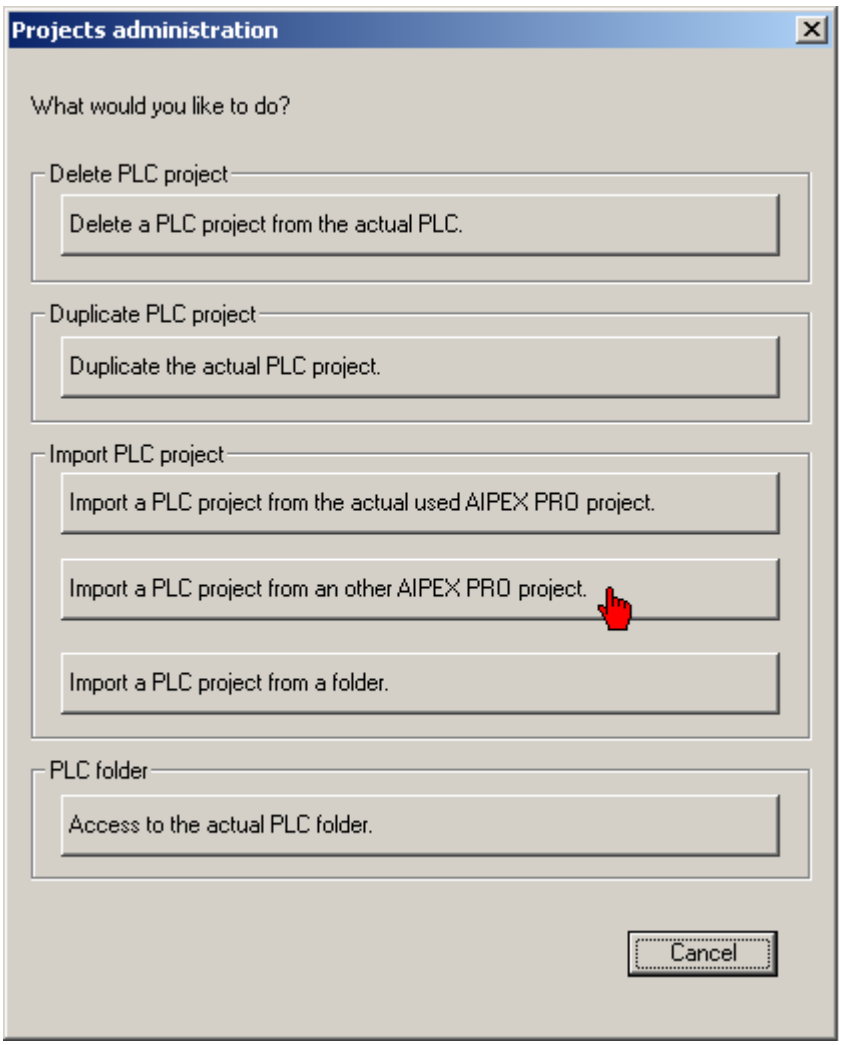
Save the read-out device data.



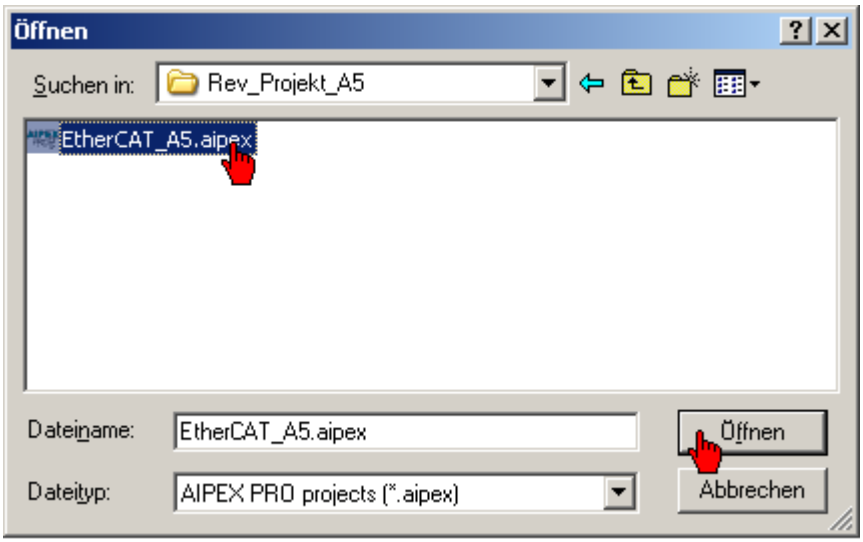
Select the **PLC** in the device tree.
Afterwards, the selection **[PLC project - import, duplicate, manage ...]**



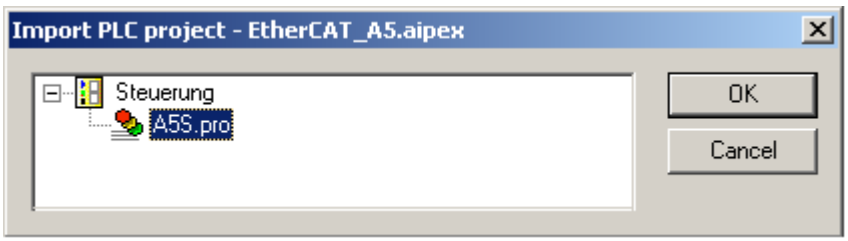
Select **Import a PLC project from an other AIPEX PRO project**.



Open the desired file that contains the PLC program.



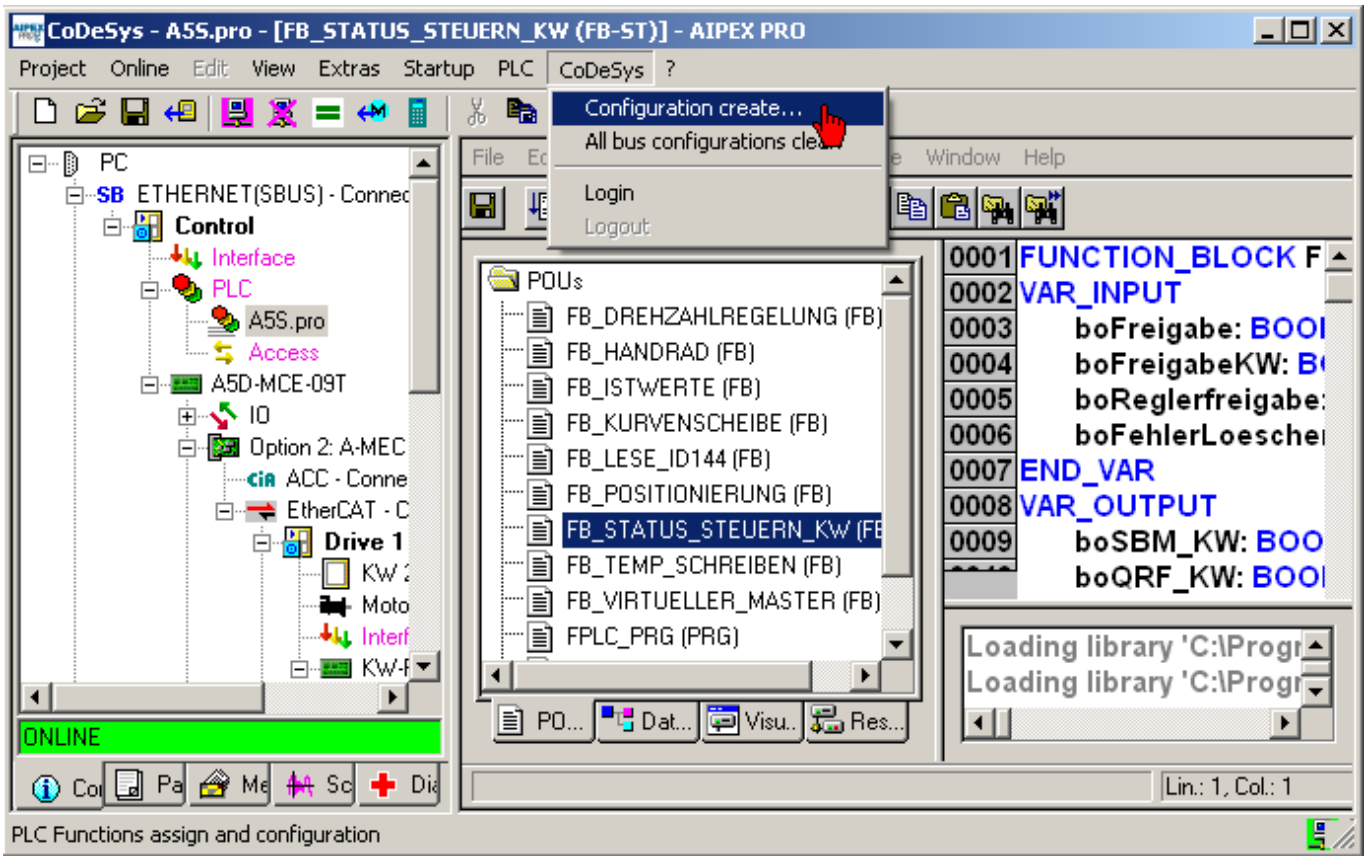
Open the desired PLC program [*.pro]



Align possibly differing addresses.

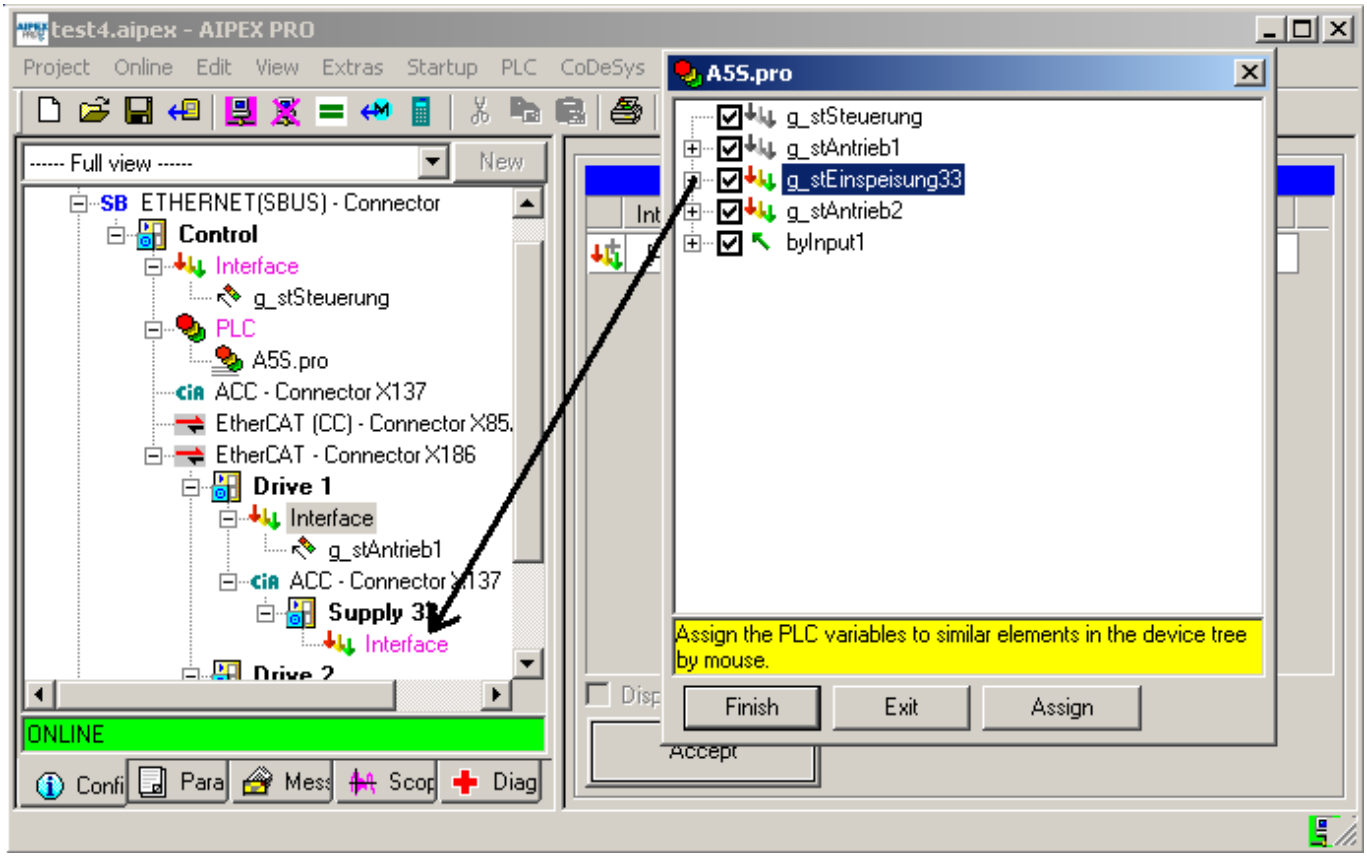
CoDeSys opens automatically.

Generate the new message configuration under **CoDeSys - Generate configuration**.

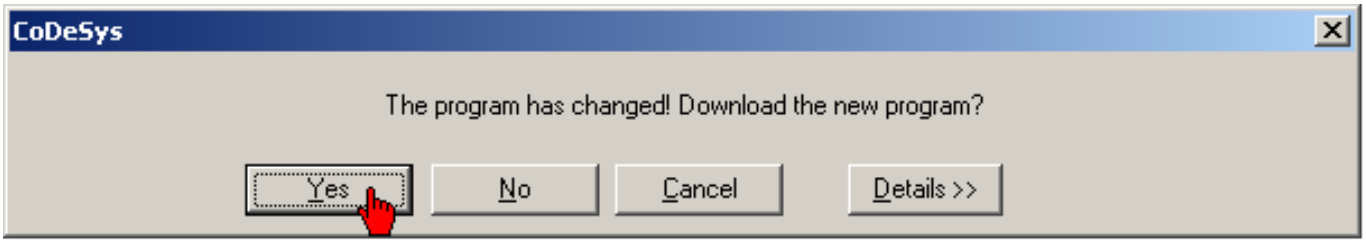


Click (and hold) your symbolic device name with the left mouse button. Now pull it onto the physically present device in the AIPEX PRO device tree (interface).

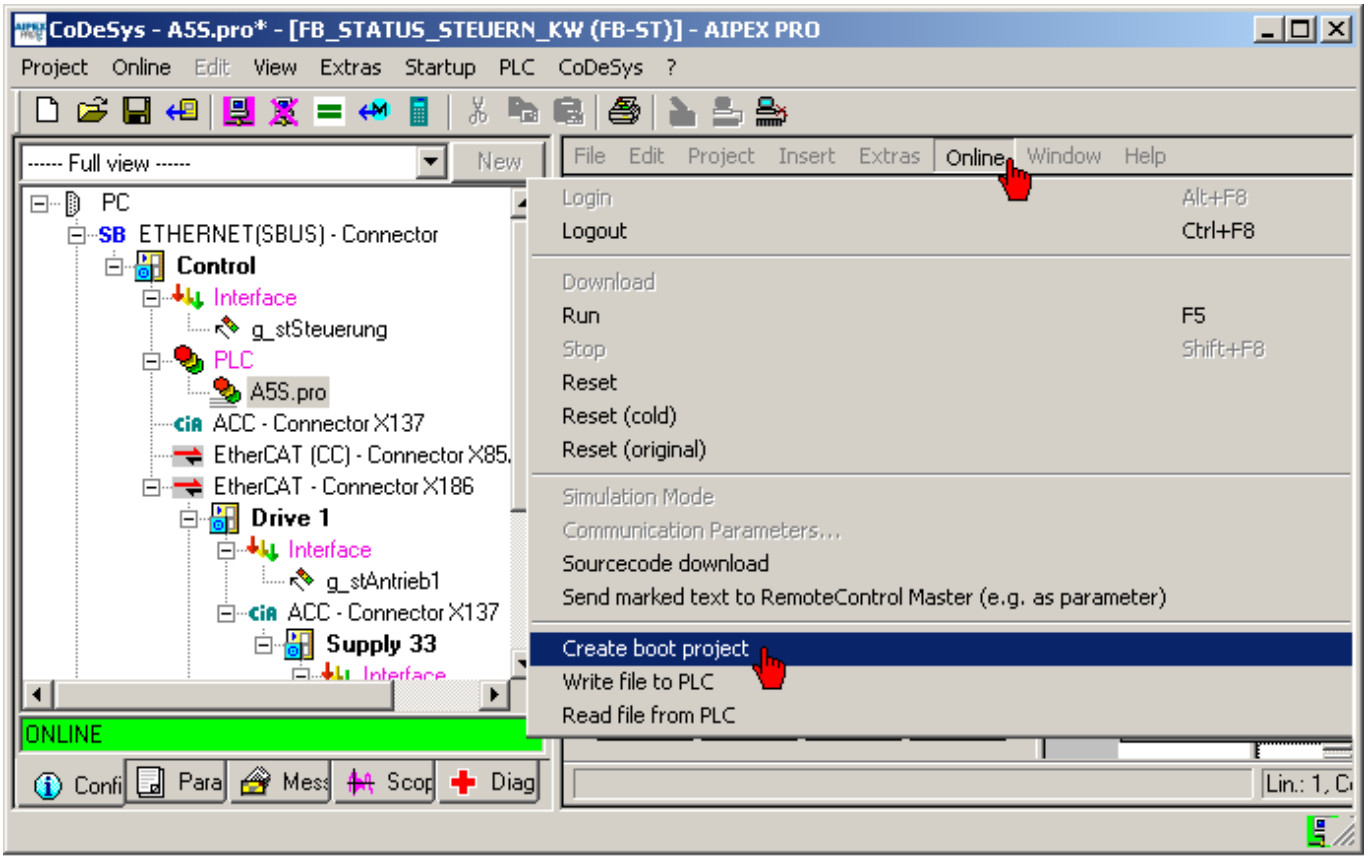
Afterwards, press the **Finish** button.



Load the PLC program onto the controller.



Generate a boot project with **Online - Generate boot project**.
Then restart the devices.



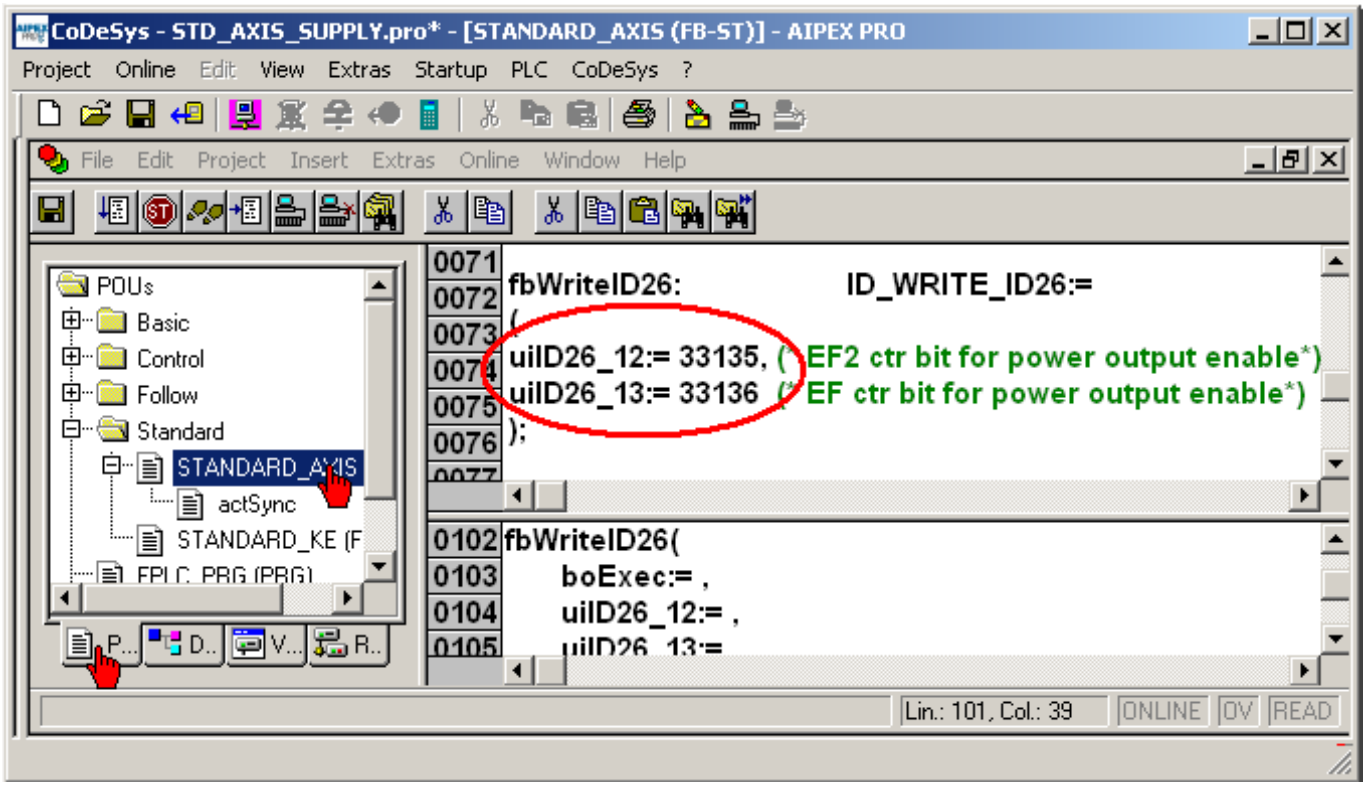
4.3.2.18.7 Expanding standard bits

Status bits are configured with the ID26 'Configuration status bits' on the device.
The Standard axis uses the function block 'ID_WRITE_ID26' to transfer a configuration from the PLC to the device.

The list elements 2 - 11 are pre-assigned as follows: Siehe 'ST_STATUSBITS (ST)' auf Seite 192.
For the list elements 12 - 17, additional codes can be selected freely. Refer to the device description for the codes.

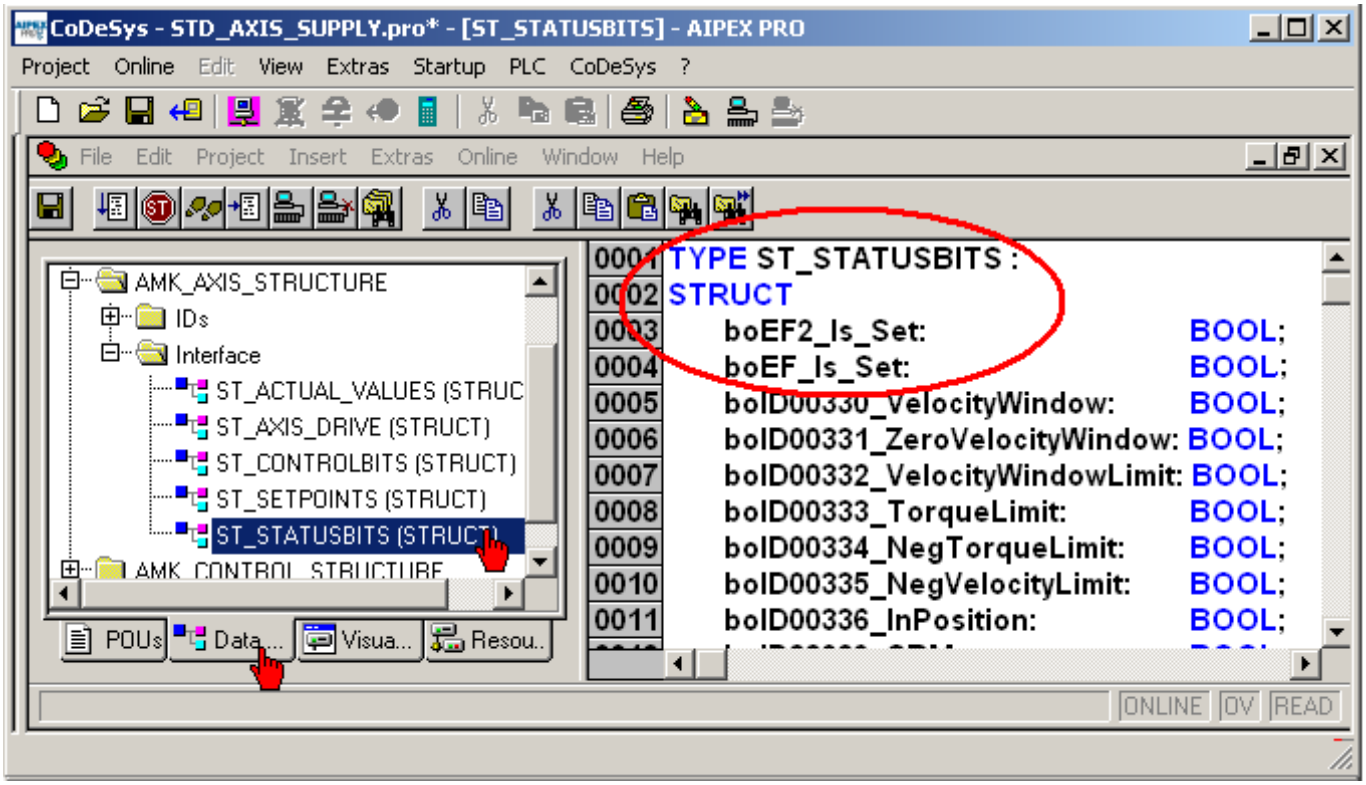
This example demonstrates how to import the bit messages Code 33135 EF2 ctr bit for power output enable and Code 33136 EF ctr bit for power output enable in addition to the available status bits.

Open AMK's function block 'STANDARD_AXIS'. Under Declare Variable, add the initial values to the 'fbWriteID26' declaration.



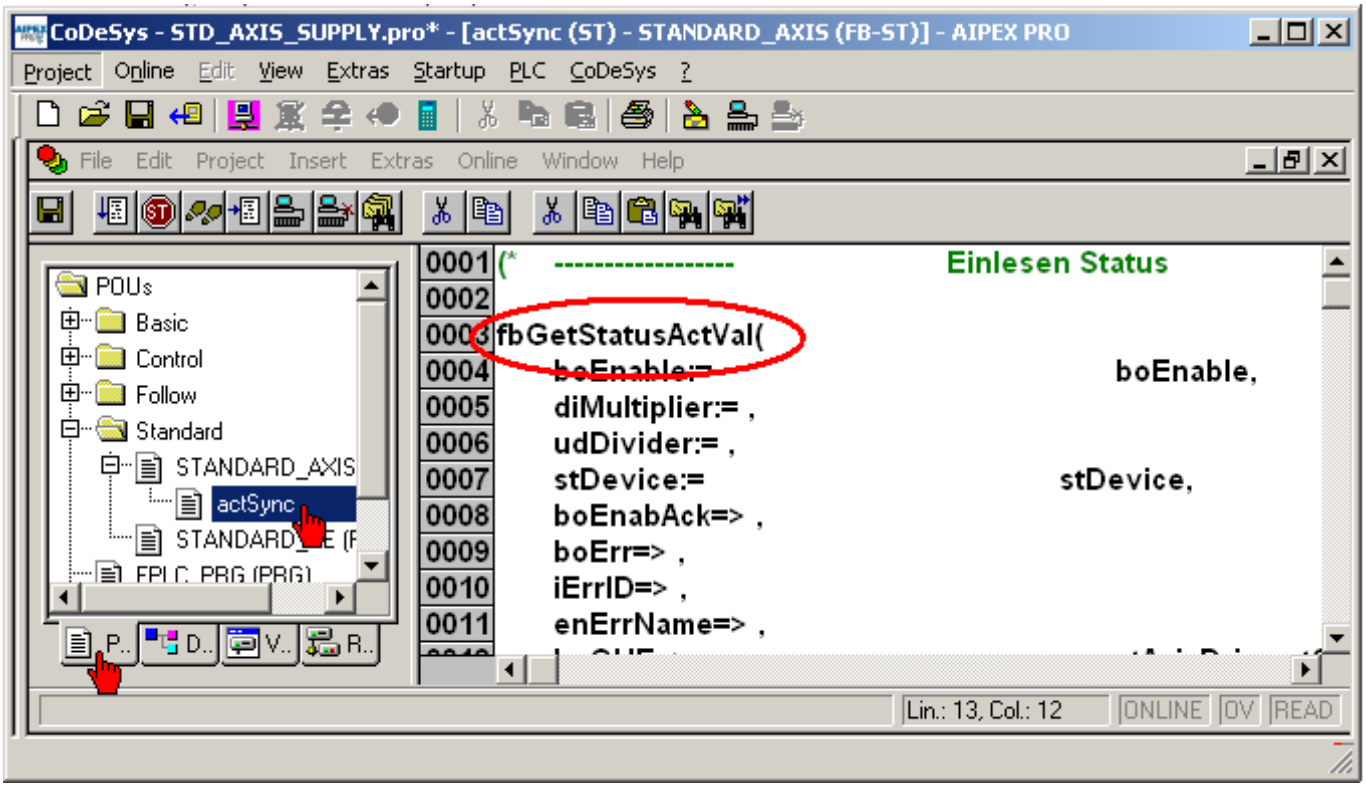
You can expand the structure 'ST_STATUSBITS' as required.
Storage location:
CODESYS V2
'Data types' → 'AMK_AXIS_STRUCTURE' → 'Interface'

CODESYS V3
'POUs' → 'Control' → 'Standard' → 'Types' → 'AMK_AXIS_STRUCTURE' → 'Interface'



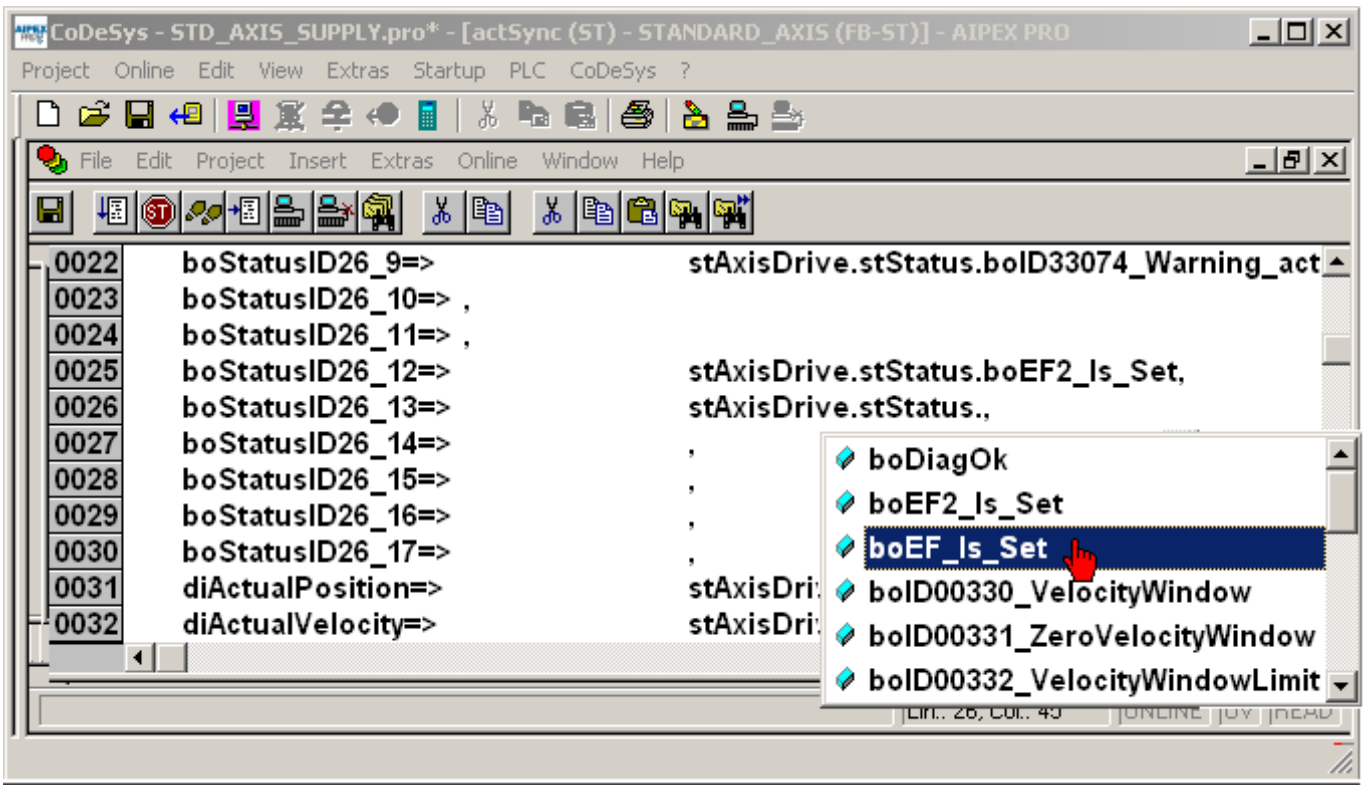
Switch to synchronous operation of **FB '**STANDARD_AXIS**'**. This can be found under: **'POUs' 'Standard' 'STANDARD_AXIS' 'actSync'**.

'fbGetStatusActVal' copies the imported status information, along with other data, into the structure 'ST_STATUSBITS'.

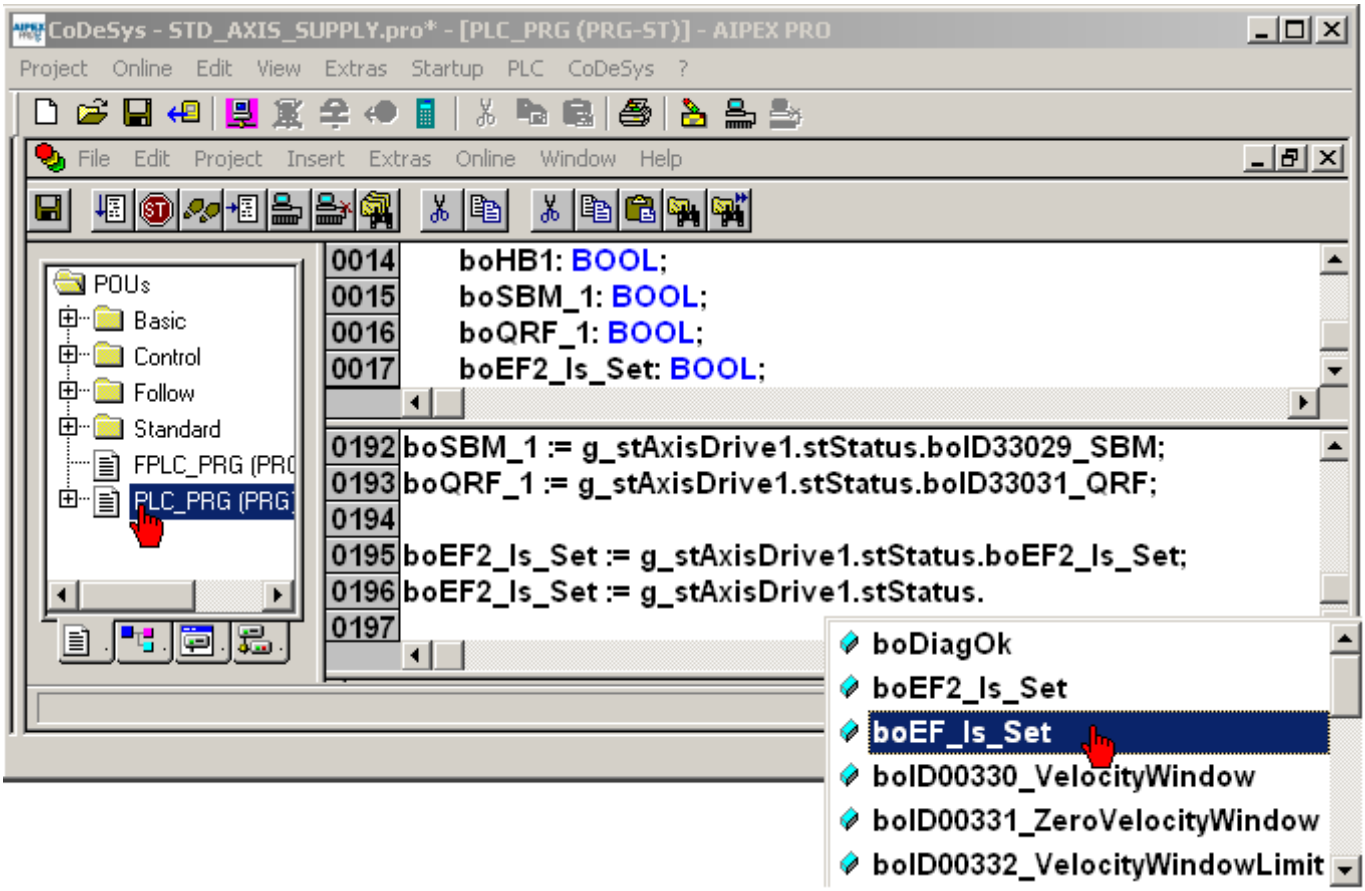


Copy list items 12 and 13 into the structure 'ST_STATUSBITS'.

You can now access 'boEF2_Is_Set' and 'boEF_Is_Set' and the pre-existing status bits.



Create the local variables in **'PLC_PRG'**. Export the status bits from the structure.

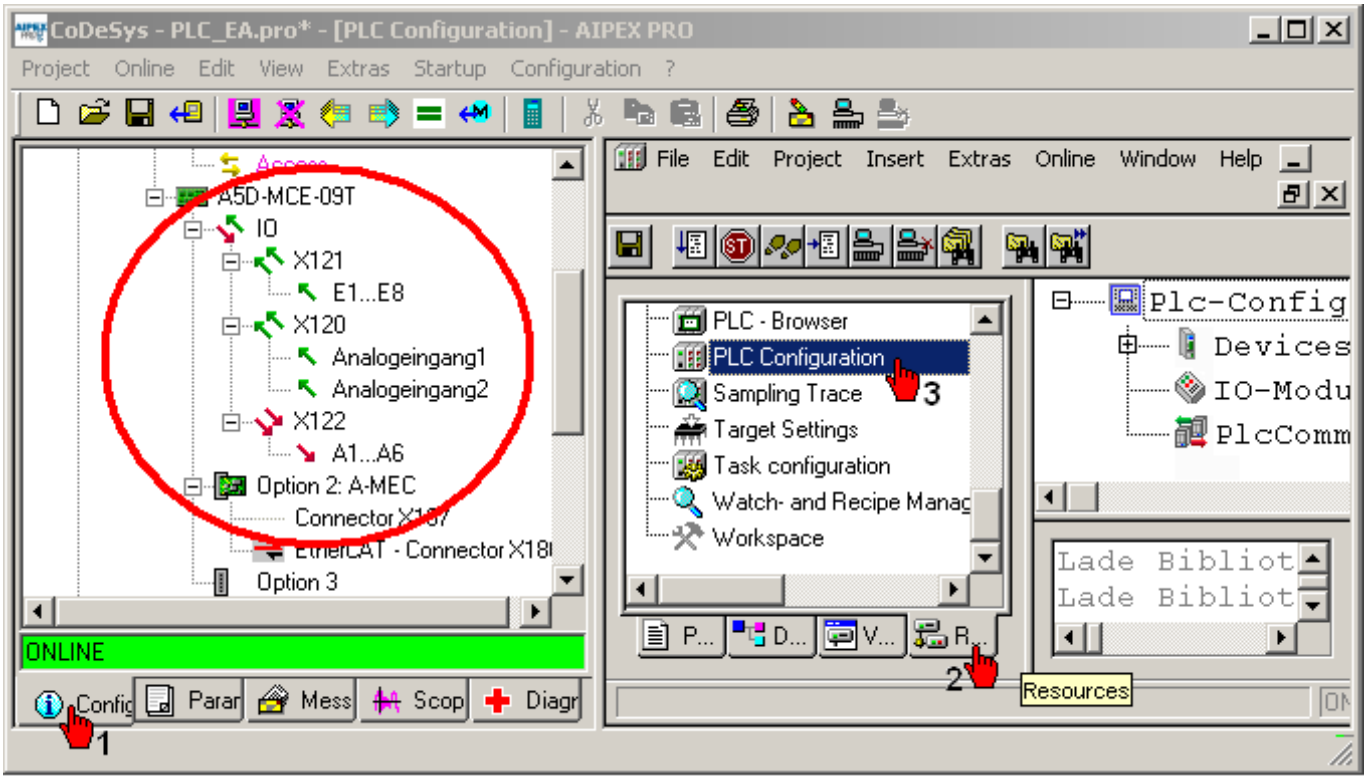


Following the initial automatic writing process (WriteID26) of the status bits, the device needs to be restarted.

4.3.2.18.8 Access to PLC IOs

For each in- and output block, one variable needs to be set in CODESYS.

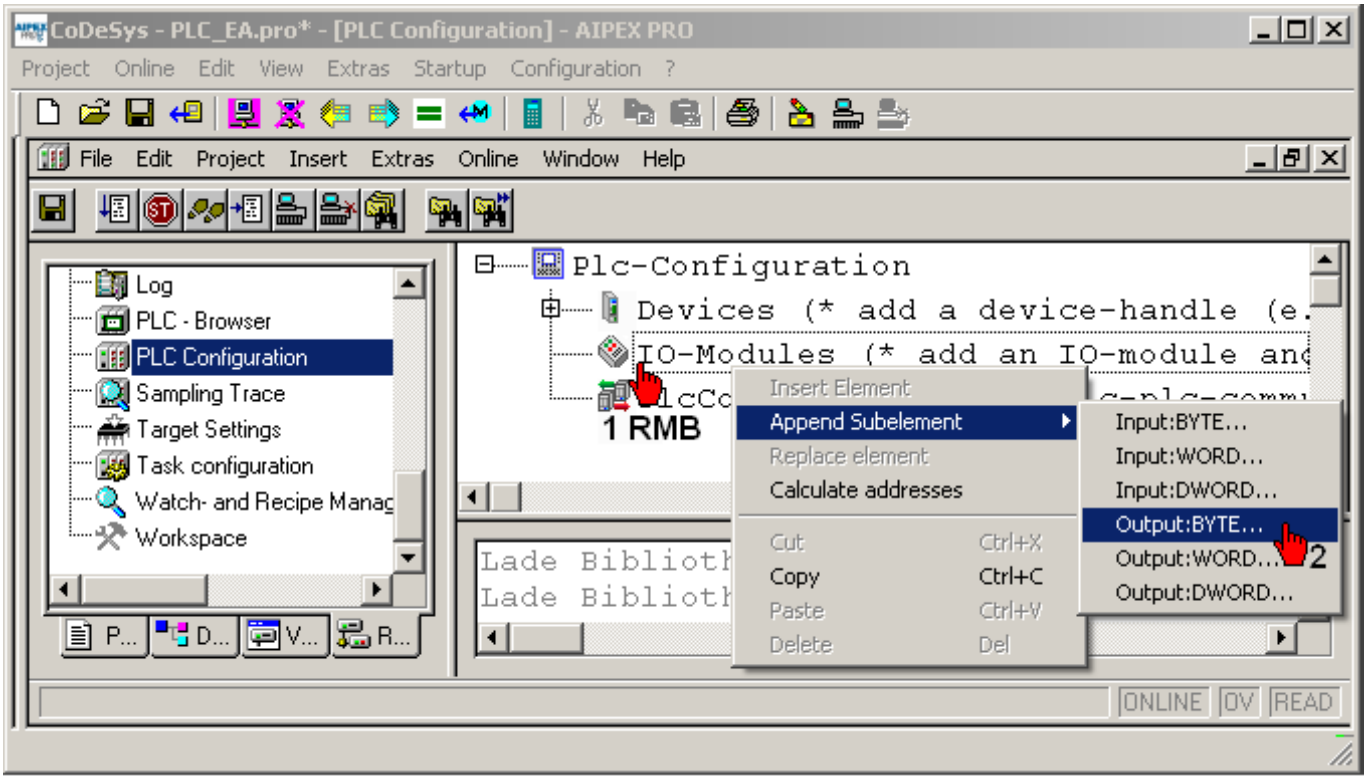
Open the 'PLC Configuration' (3) in the menu 'Ressources' (2)



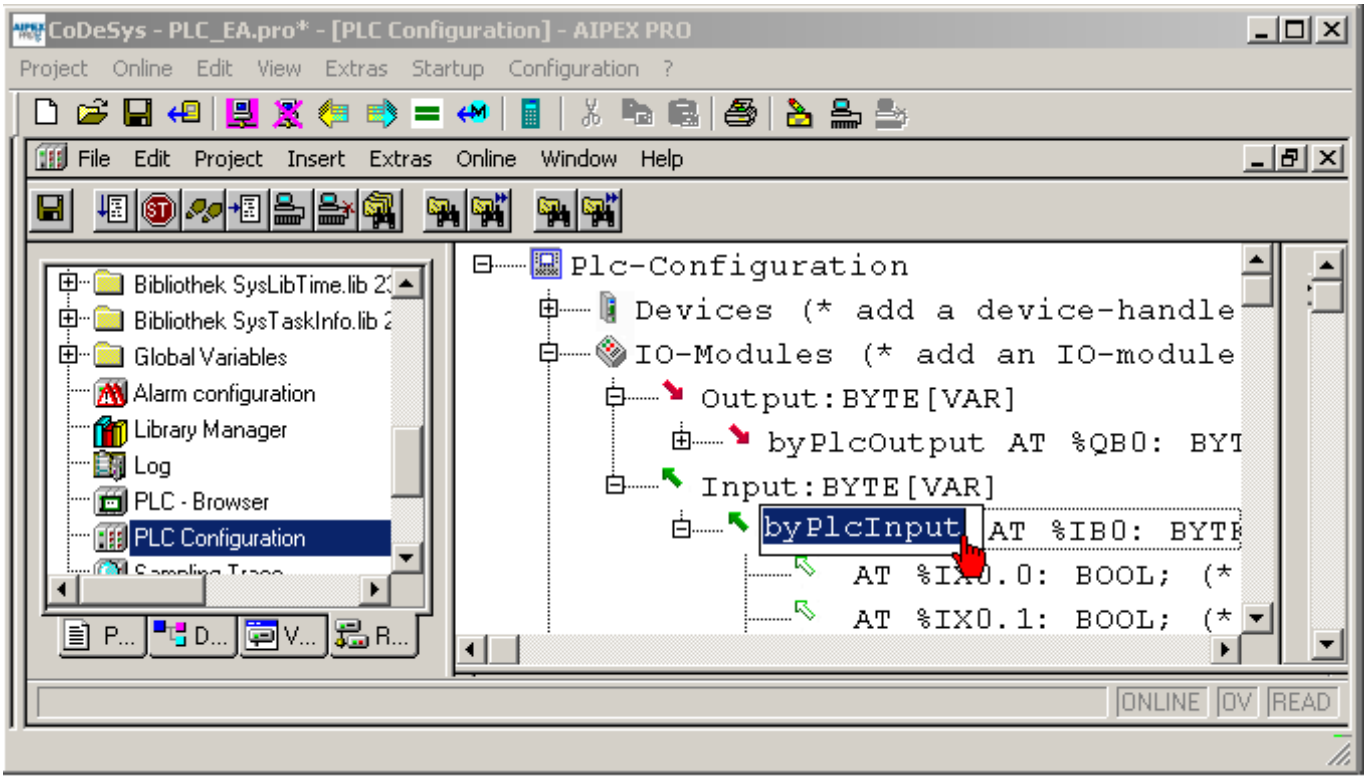
Right click on 'the I/O modules (1)' to insert new variables.

Input: Data is read in from the drives to the PLC.

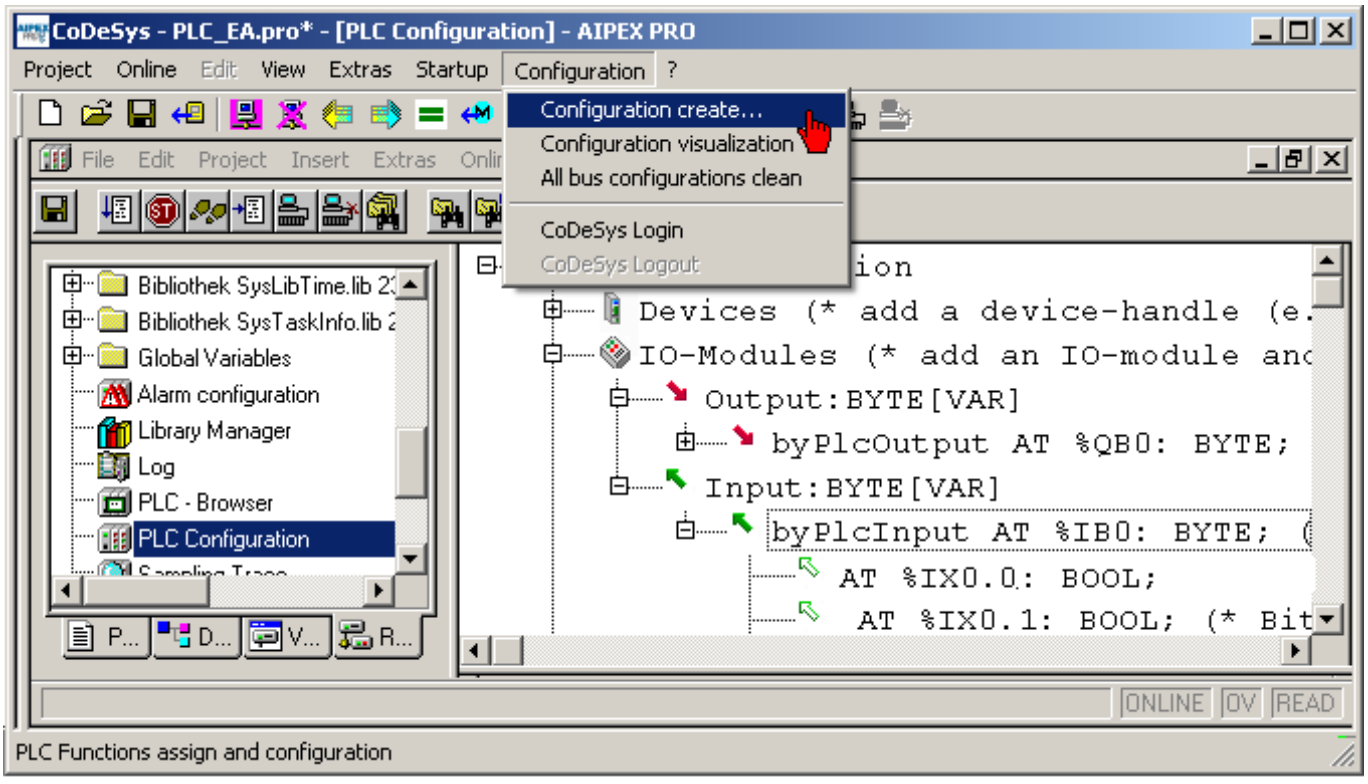
Output: Data is written from PLC to the drives.



For each physically existing module, a symbolic variable needs to be created.



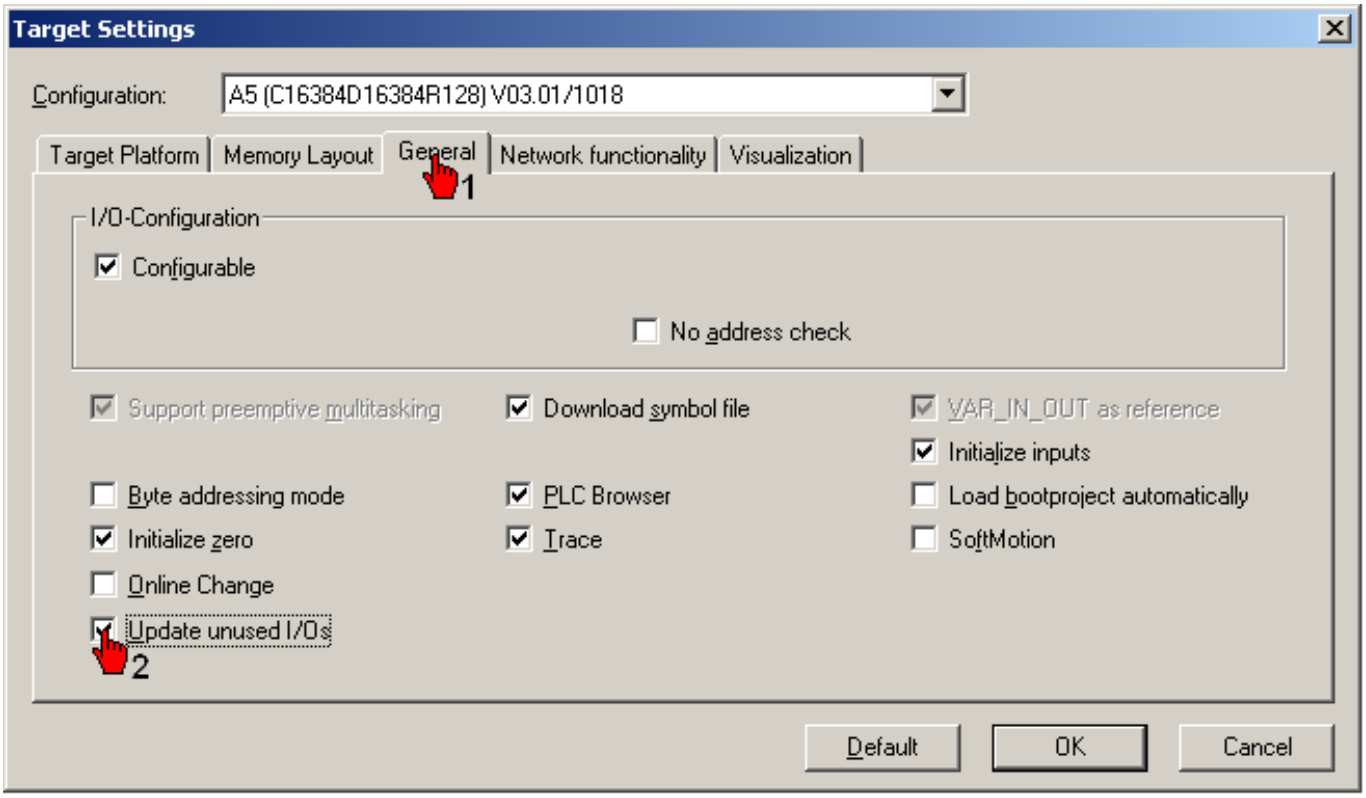
Afterwards, start the automatic message configuration creation



Assign the symbolic variables to the physically existing in- and outputs.
After the message configuration has been created, the system needs to be rebooted.

Function test

Activate the checkbox **'Update unused I/Os'** under **'Resources' 'Target setting' 'General'** or assign a symbolic name for each I/O.



4.4 Visualization of AFL blocks

For the majority of the function blocks, visualizations are prepared in CODESYS. These visualizations contain the input and output variables. Thus, a quick and easy projecting can be done to test the function blocks.

The naming of the visualizations is equivalent to those of the function blocks with a prefixed 'Vi'.

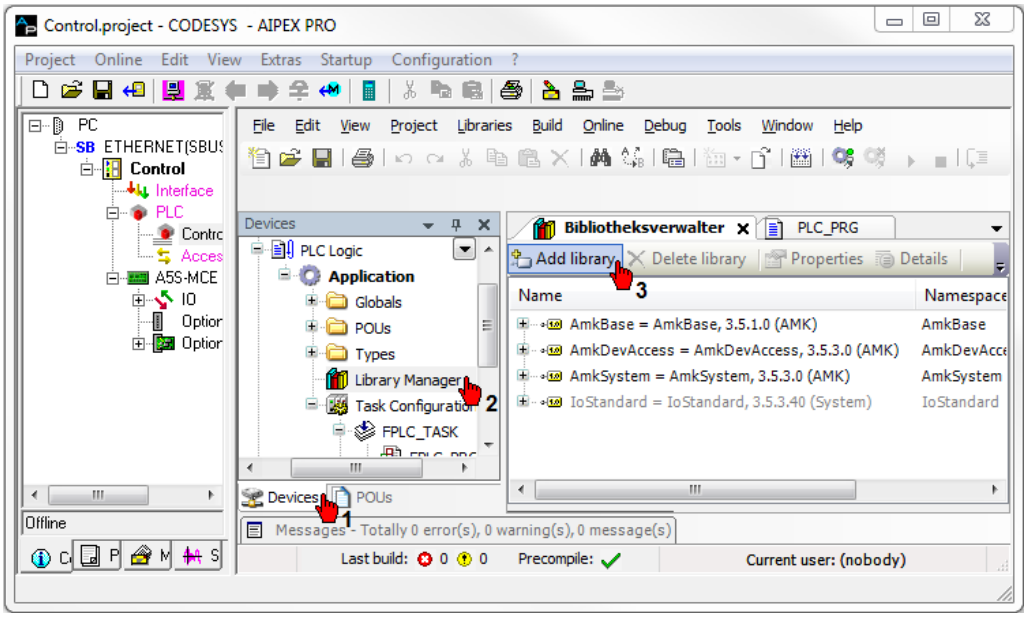
Examples:

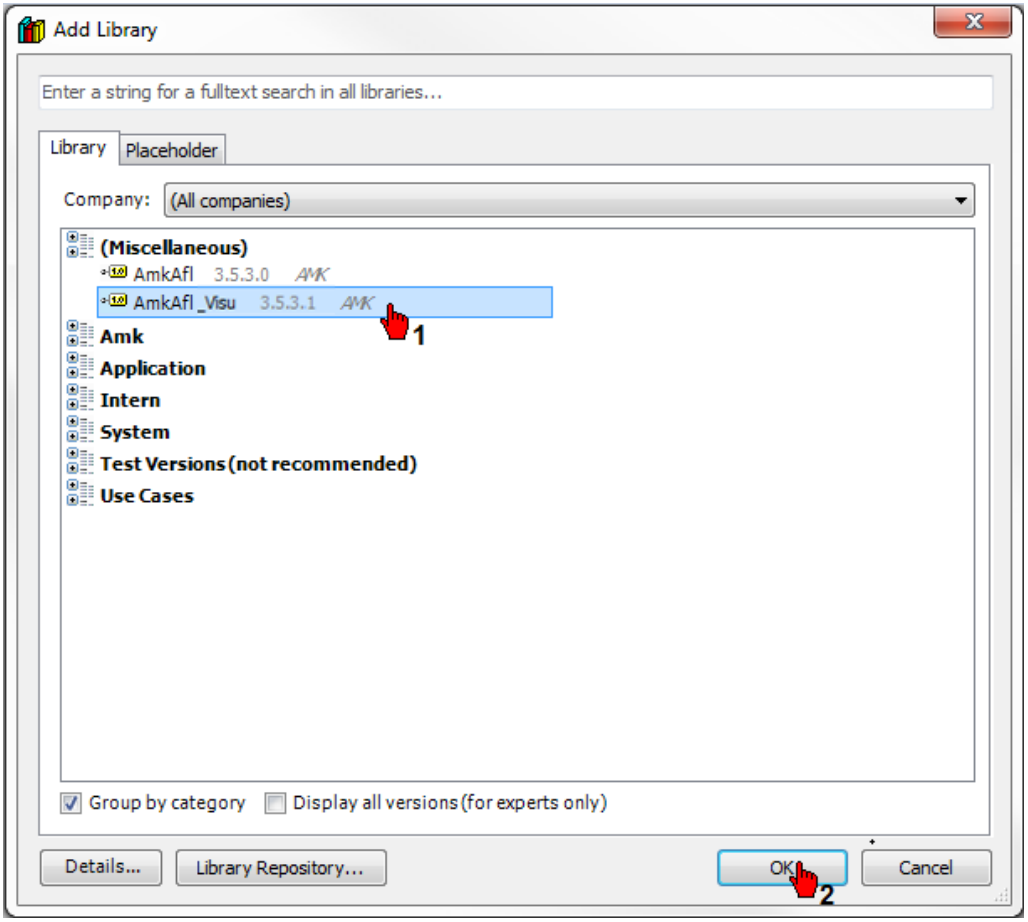
Function block	Visualisation
COMPARE_2VALUES	ViCOMPARE_2VALUES
PRINT_MARK_CONTROL	ViPRINT_MARK_CONTROL
MODULO_COUNT	ViMODULO_COUNT

4.4.1 CODESYS V3

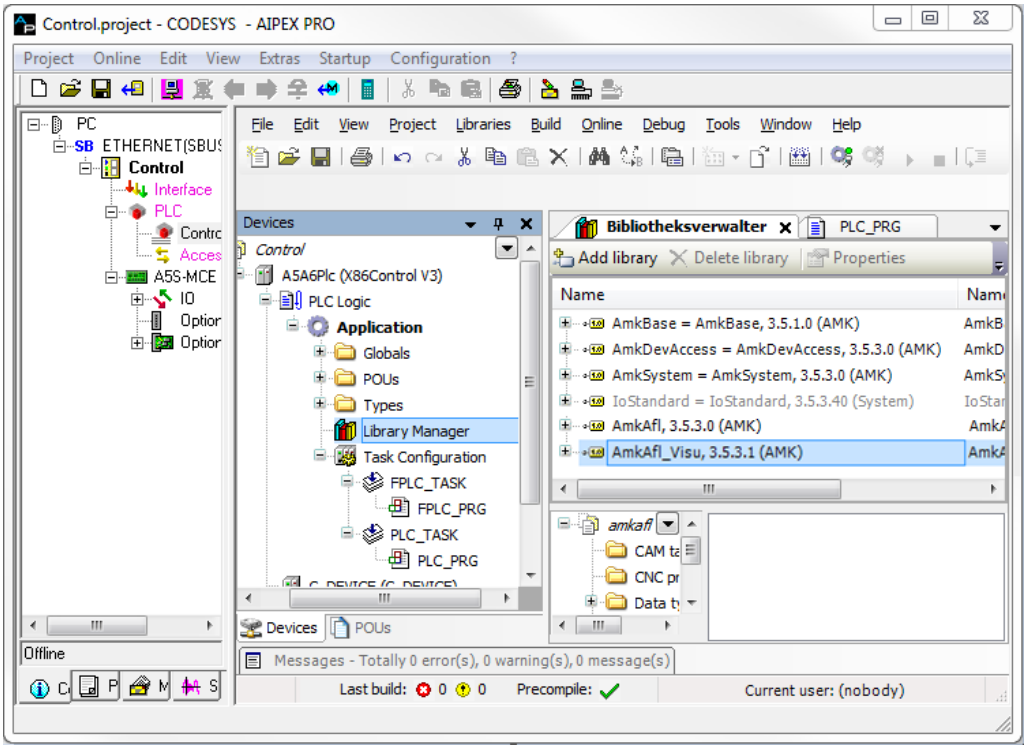
The visualizations of function blocks are include in an additional library from AFL version 3.5.3.1.

The library AmkAfl_Visu.library is not included in the standard templates and must be manually added to the CODESYS project .

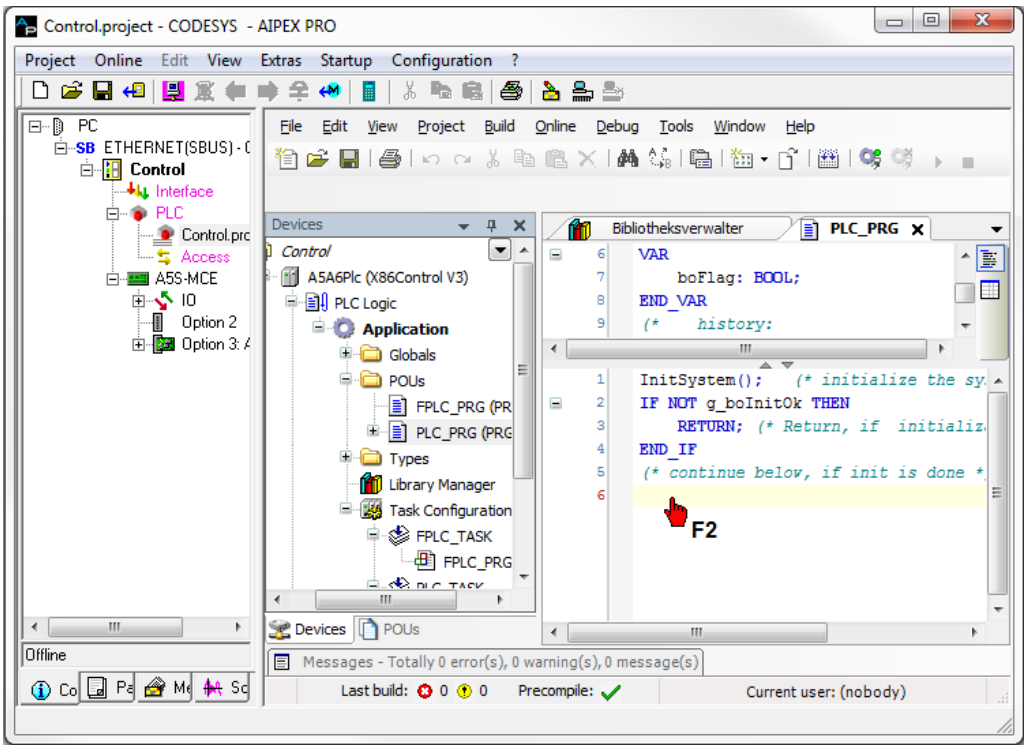

Control.project - CODESYS - AIPEX PRO
Project Online Edit View Extras Startup Configuration ?
File Edit View Project Libraries Build Online Debug Tools Window Help
PC
ETHERNET(SBUS
Control
Interface
PLC
A5S-MCE
IO
Devices
PLC Logic
Application
Globals
POUs
Types
Library Manager
Task Configuration
FPLC_TASK
Bibliotheksverwalter
PLC_PRG
Add library
Delete library
Properties
Details
Name
Namespace
AmkBase = AmkBase, 3.5.1.0 (AMK)
AmkDevAccess = AmkDevAccess, 3.5.3.0 (AMK)
AmkSystem = AmkSystem, 3.5.3.0 (AMK)
IoStandard = IoStandard, 3.5.3.40 (System)
Devices
POUs
Messages - Totally 0 error(s), 0 warning(s), 0 message(s)
Offline
Last build: 0 0
Precompile:
Current user: (nobody)
1
2
3

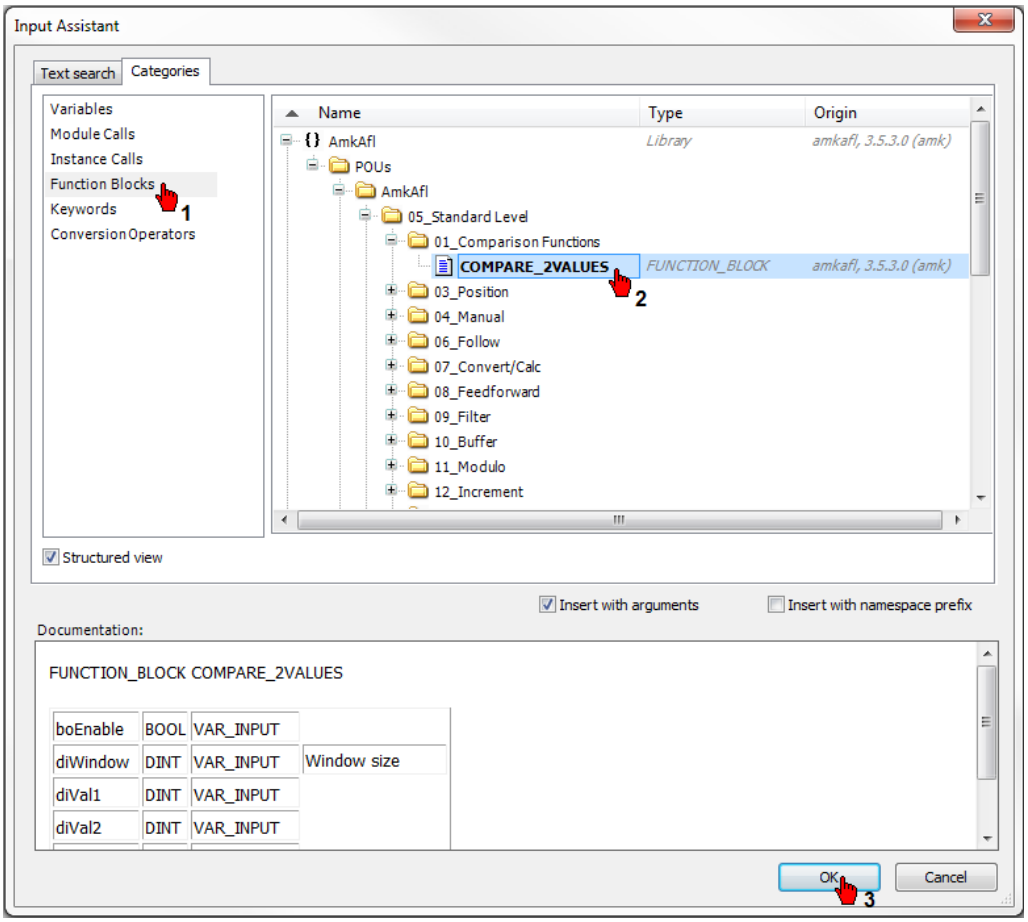

Add Library
Enter a string for a fulltext search in all libraries...
Library
Placeholder
Company:
(All companies)
(Miscellaneous)
AmkAfl 3.5.3.0 AMK
AmkAfl_Visu 3.5.3.1 AMK
Amk
Application
Intern
System
Test Versions (not recommended)
Use Cases
Group by category
Display all versions (for experts only)
Details...
Library Repository...
OK
Cancel
1
2

The library AmkAfl_Visu is added, if it is listed in the library manager.

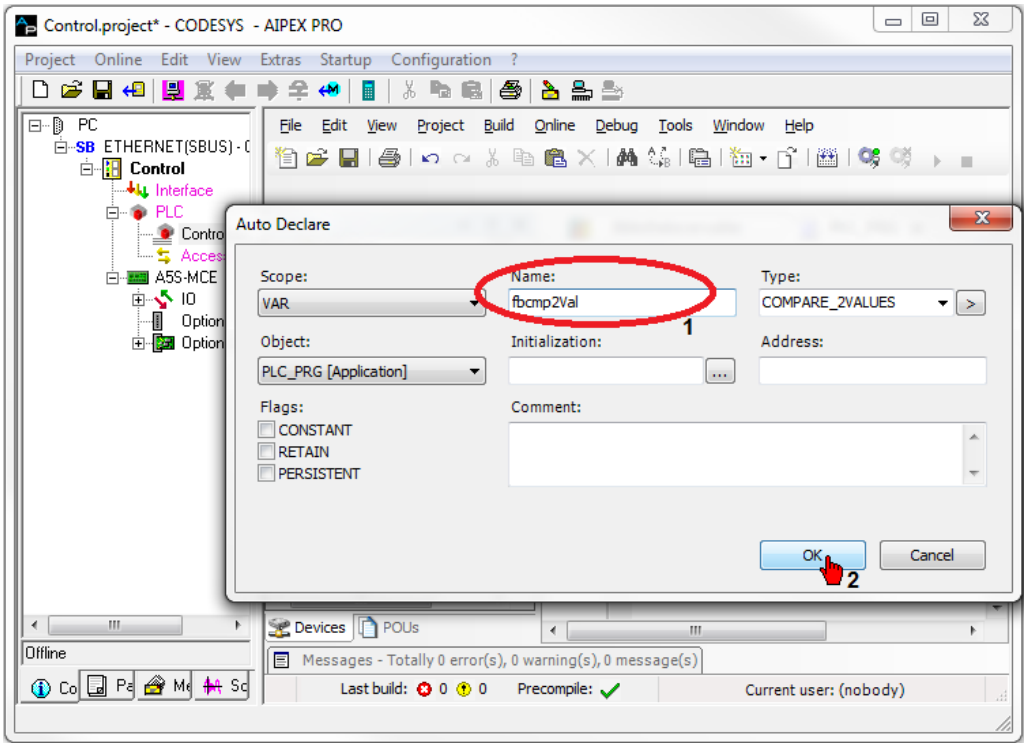


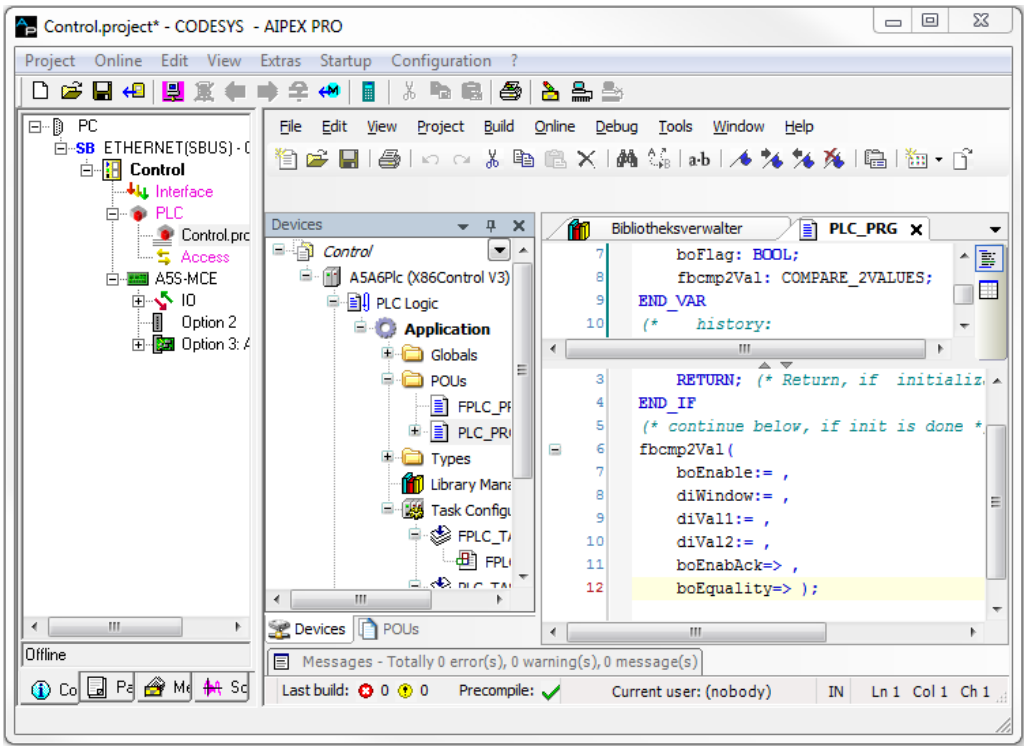
Insert function block (Example: COMPARE_2VALUES)



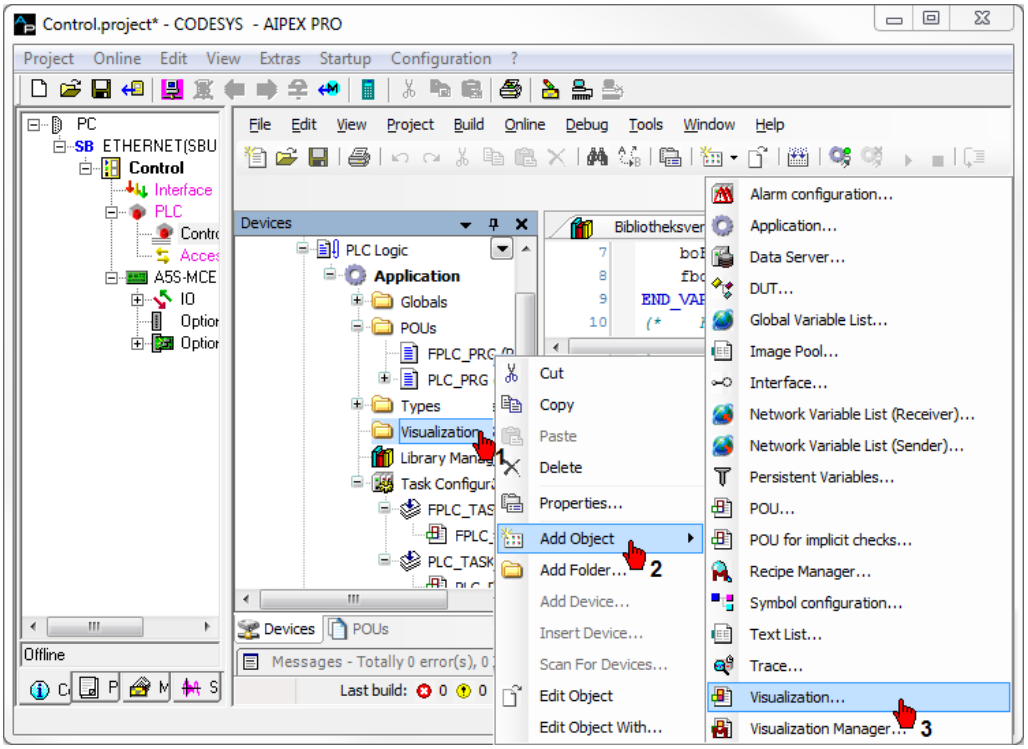


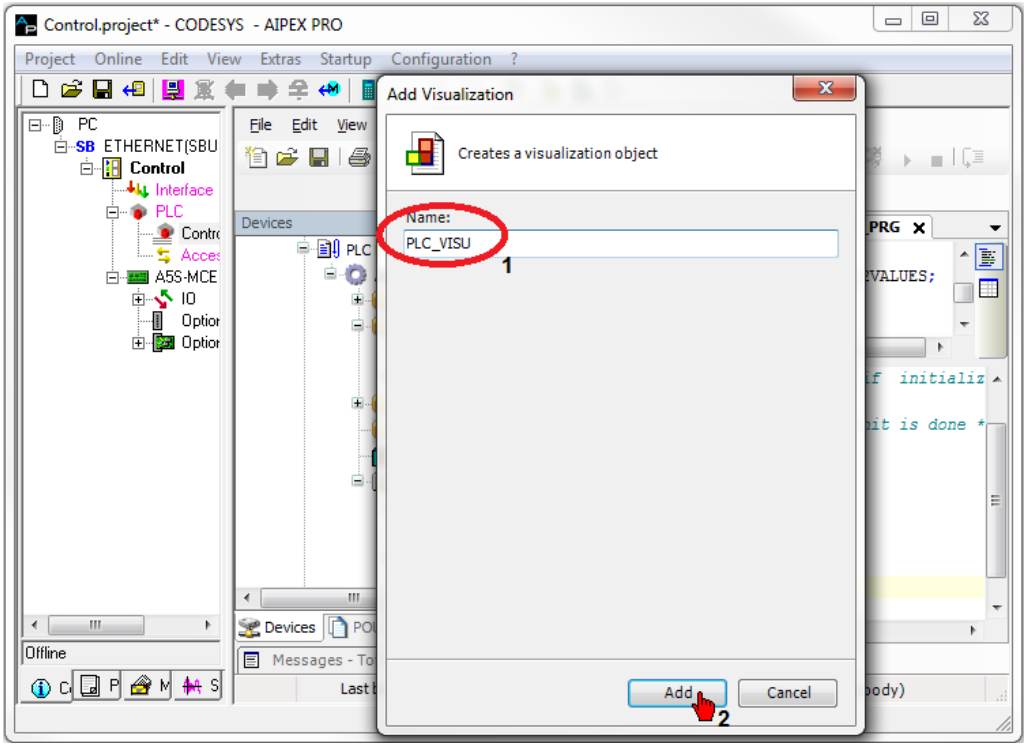
Declare function block



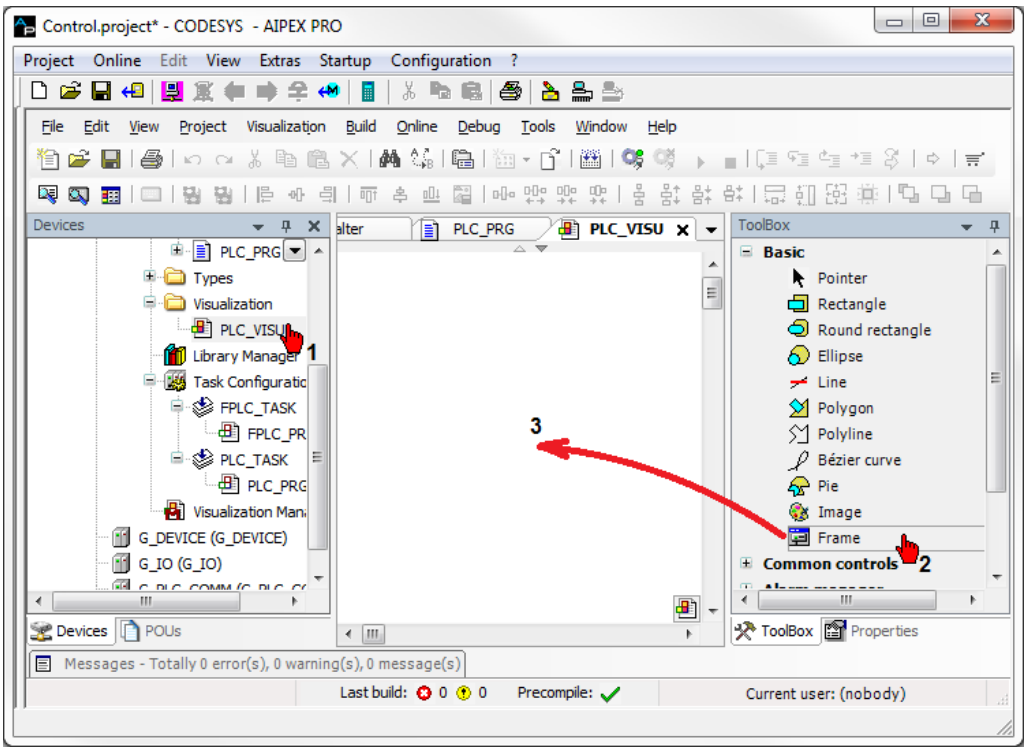


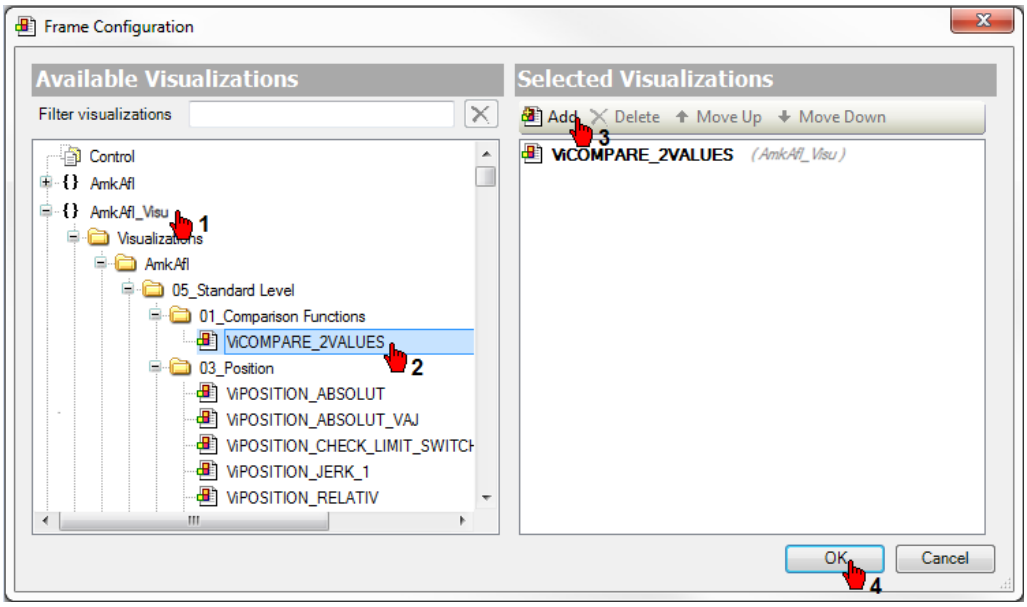
Add visualization



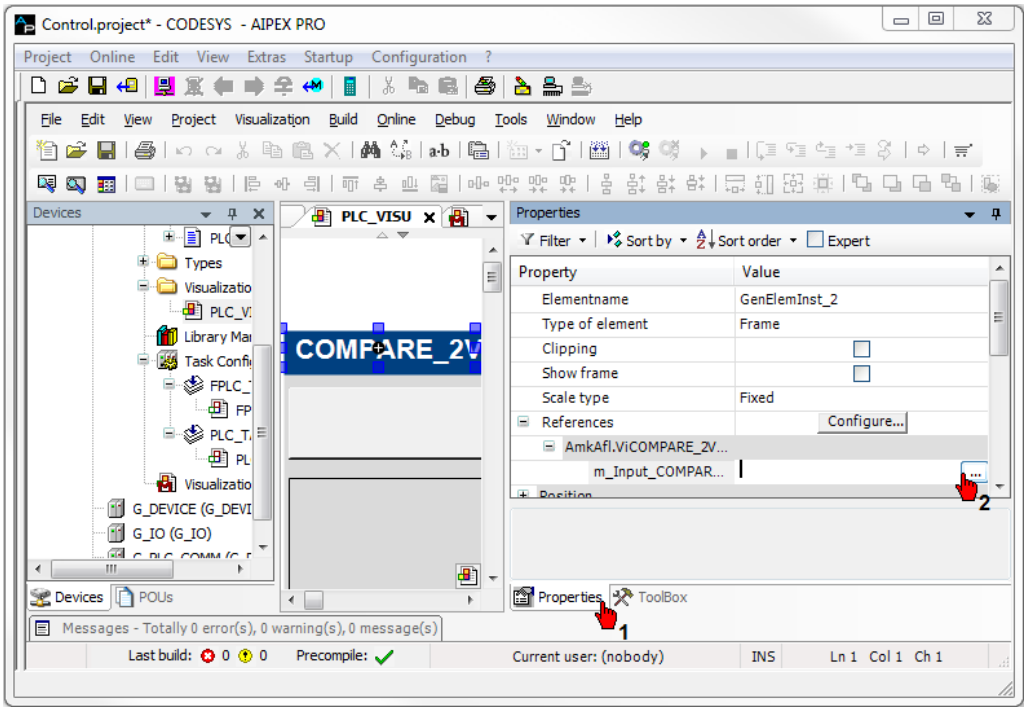


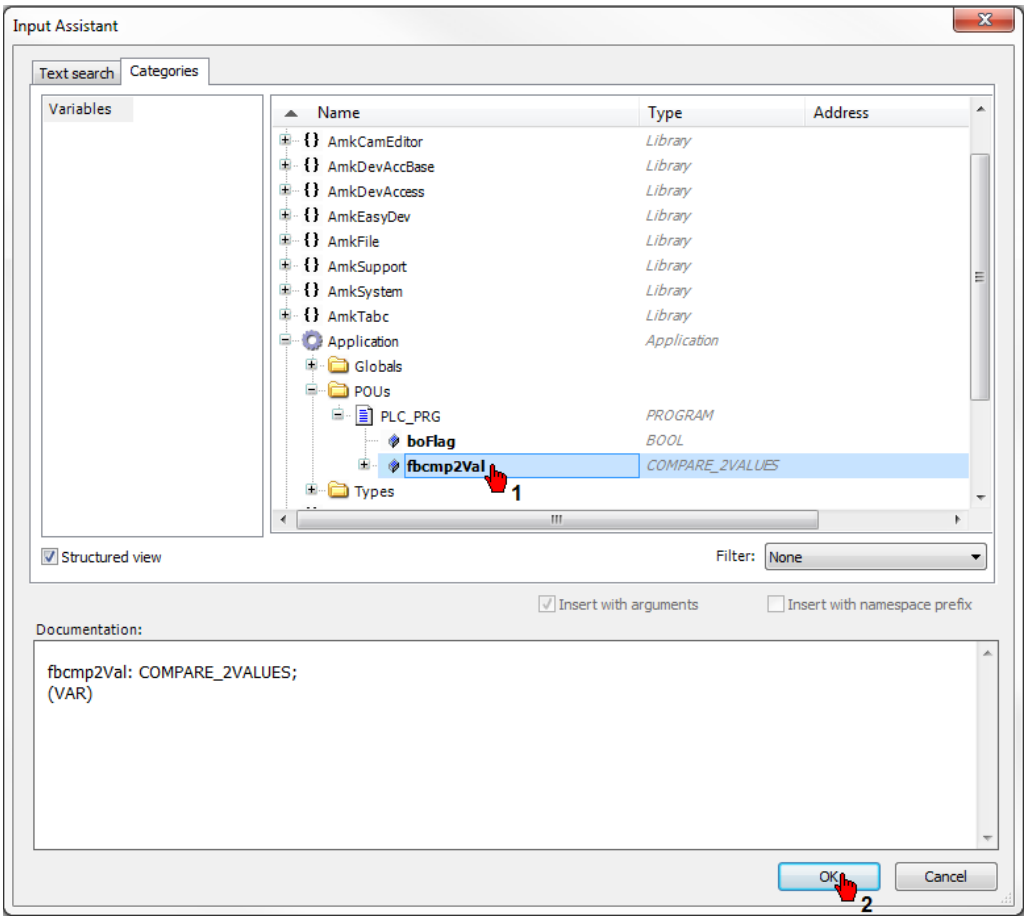
Add the frame and select the visualization (Example: COMPARE_2VALUES)



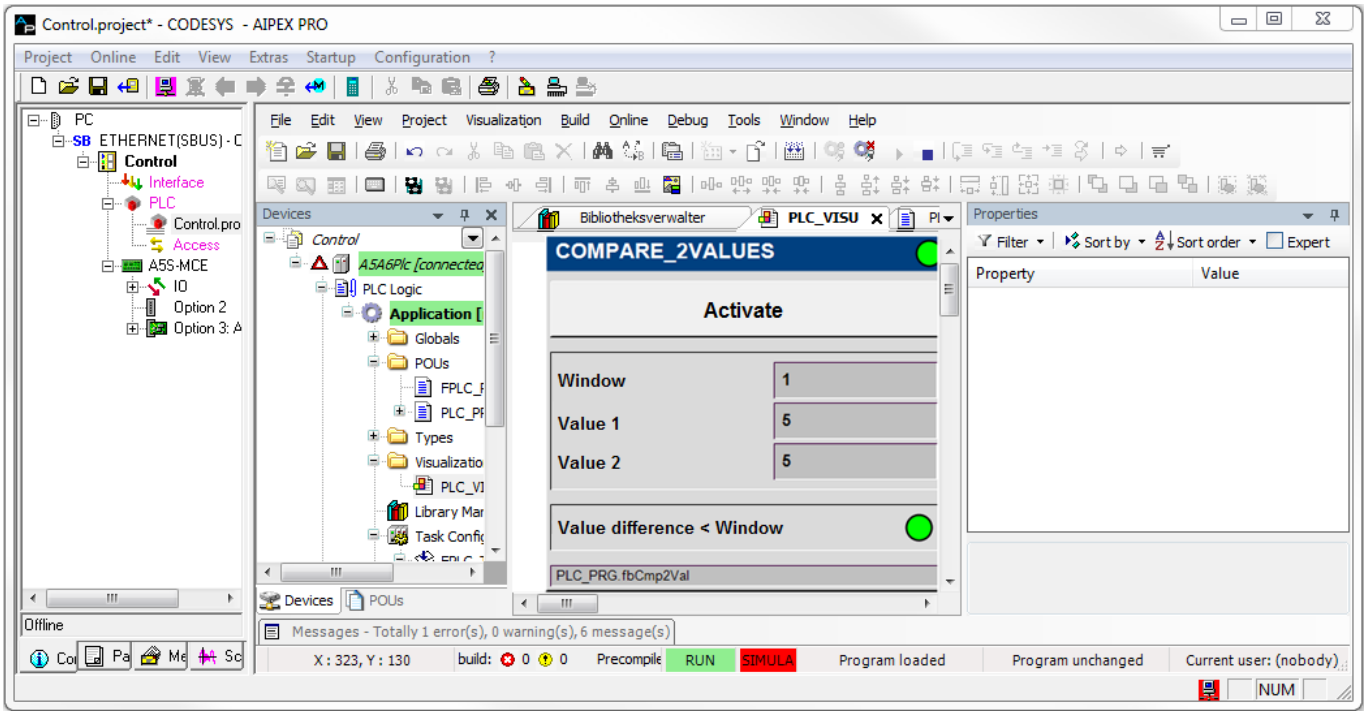


Select the function block that you want to assign the visualization.





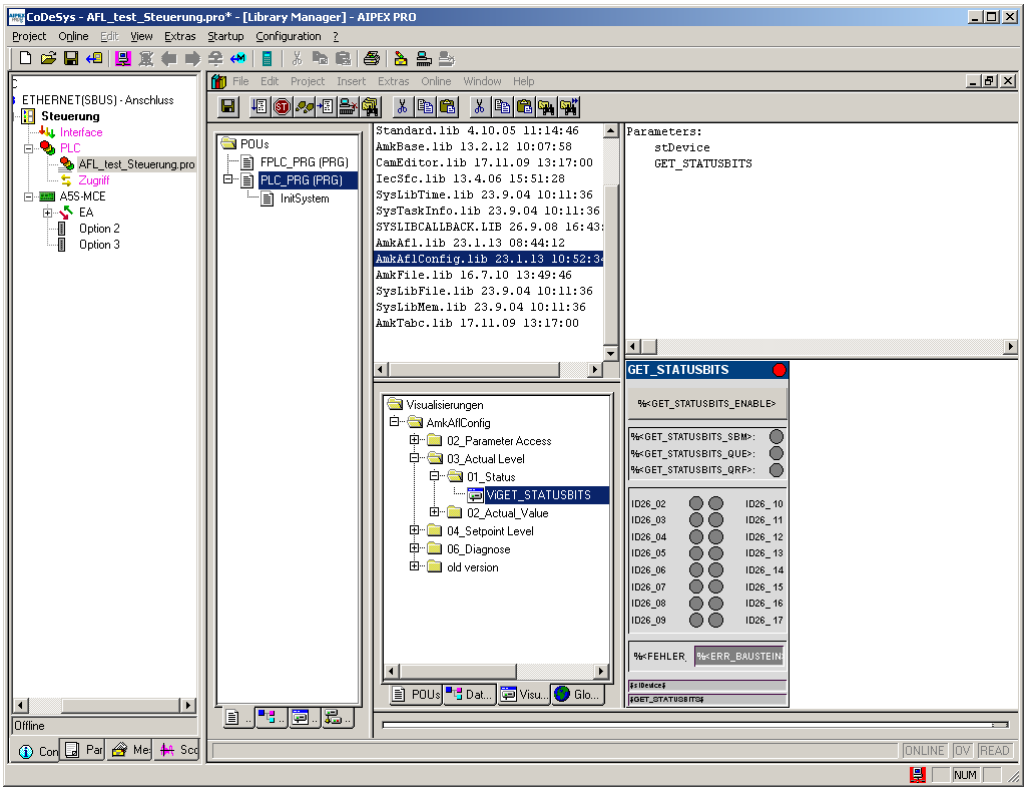
The values are only displayed if the CODESYS program was started.



You can get further information about PLC programming with CODESYS from the software description AIPEX PRO V3 (AMK part no. 204979)

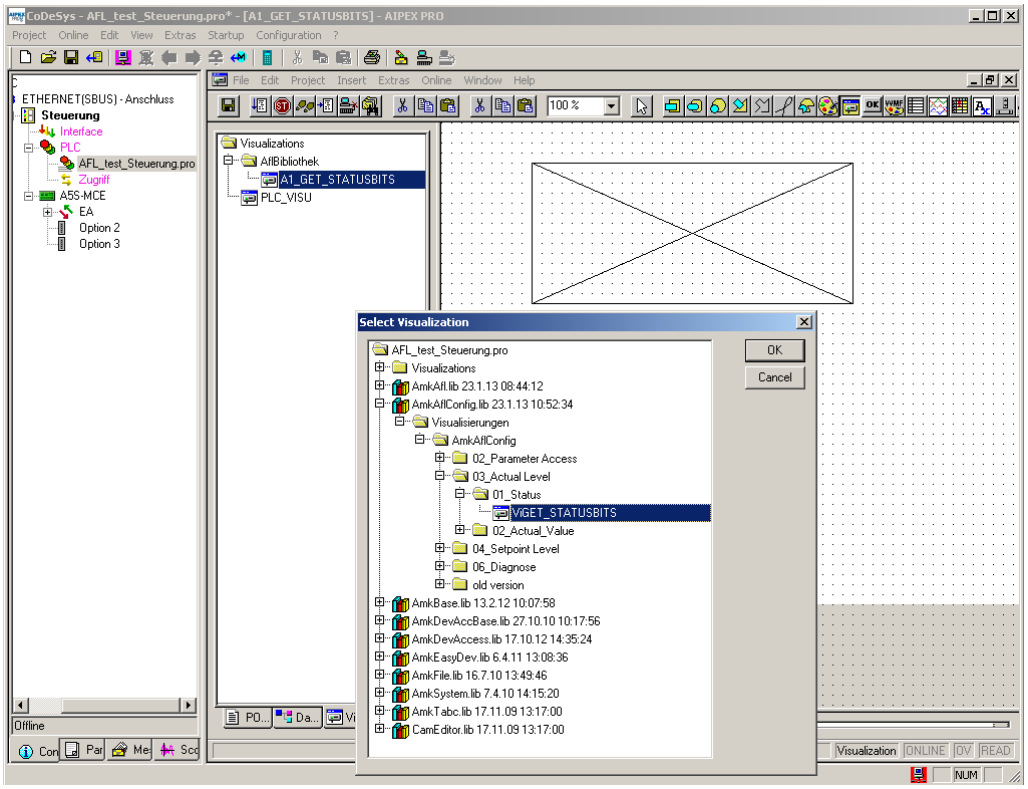
4.4.2 CODESYS V2

CODESYS user interface 'Library Manager': Visualization with its stored variables:

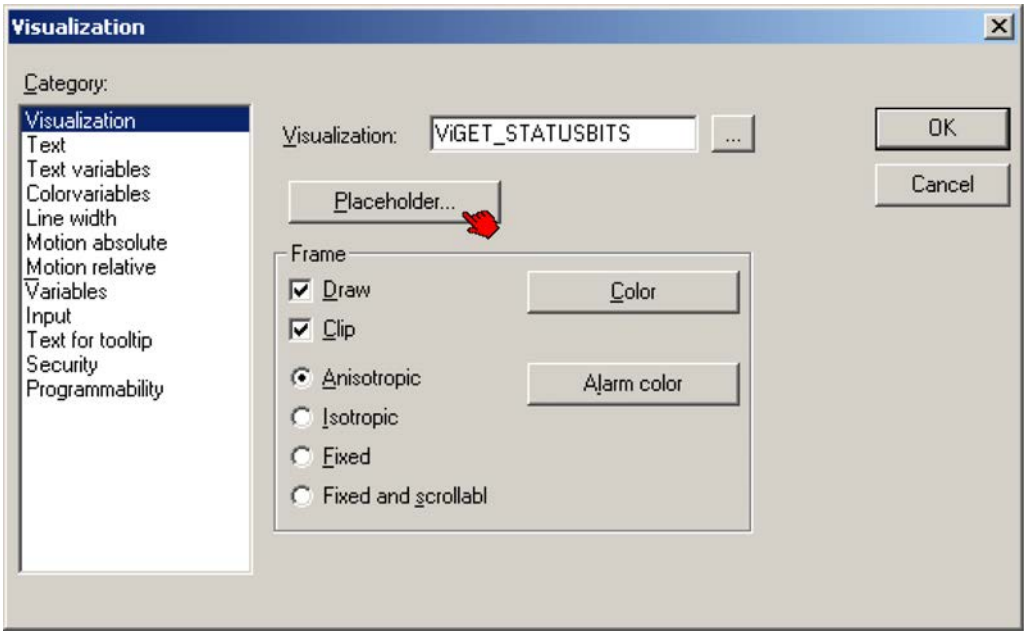


In the 'Visualizations' editor, use **'Insert'** -> **'Visualization'**.

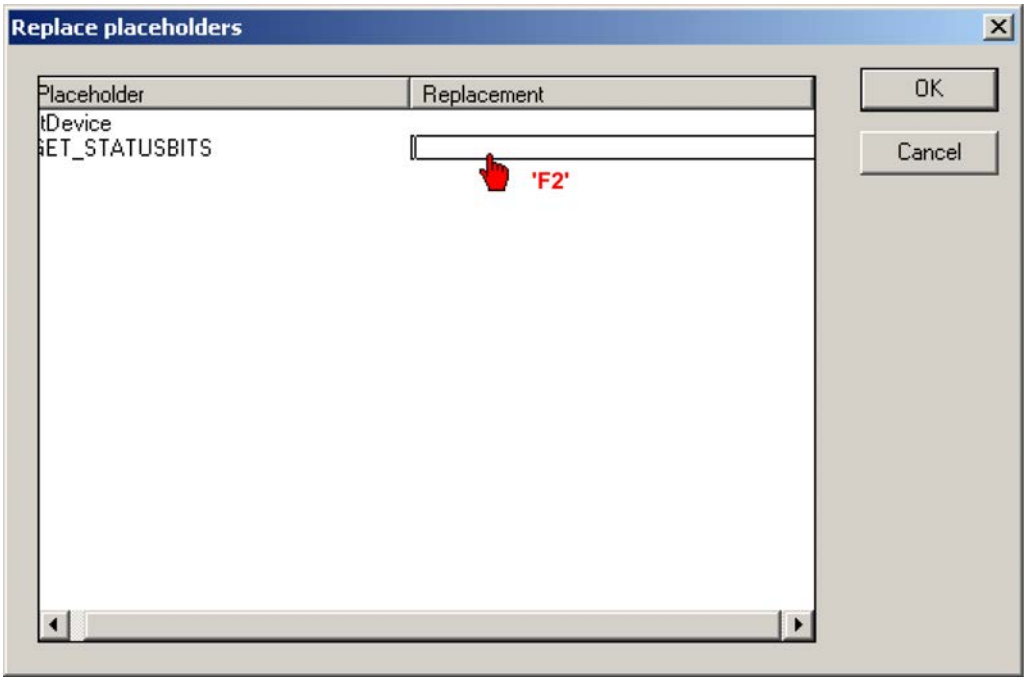
Pull up an area and select the required visualization block. You can adapt the final position and size of the visualization subsequently.



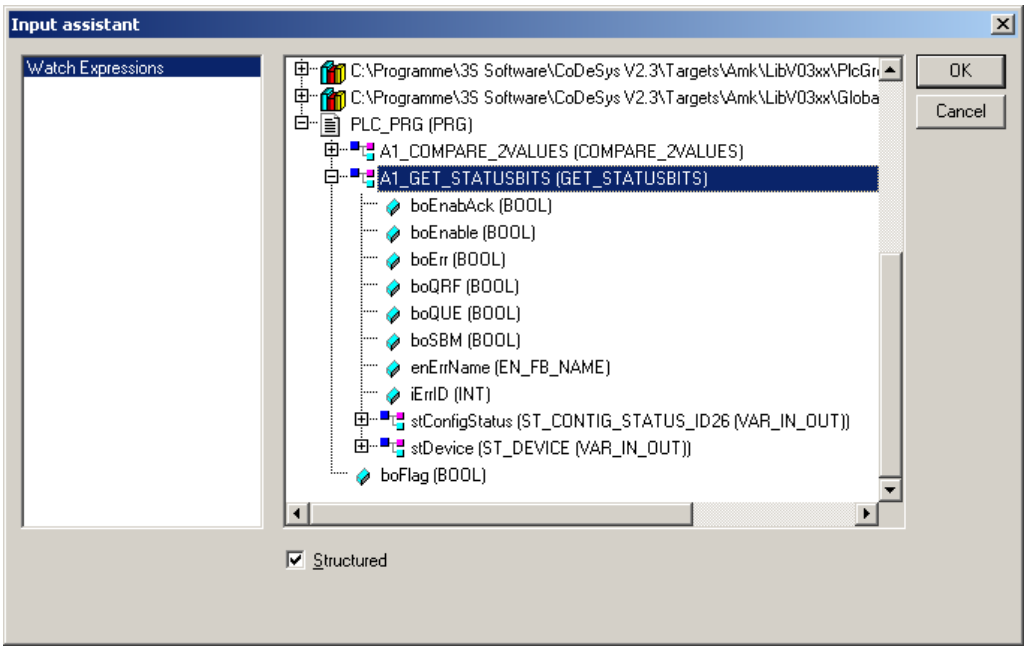
Double click on the inserted visualization block or click with the right mouse button and select **'Configure'**.



Select **'Placeholder...'**. Here, you can exchange the placeholding variables which are stored in the visualization block.
Click to the column 'Replacement' and press **'F2'**.



Select the function block you want to assign to the visualization.



Confirm all input windows with **'OK'**.

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

AIO
AMKAMAC I/O terminals

Ax-PNC
Numerical Control Motion

A-SPB
Profibus DP slave interface

ATF
AMK Tool Flasher (PC software for transferring firmware to device)

ASCII
American Standard Code for Information Interchange

AIPEX
AMK startup and parameterizing software (PC software): Programming, parameterization, configuration, diagnosis, oscilloscope, status information

ADB
AMK database - file in XML format with information about all AMK parameters

actSync
Synchronous action of an asynchronous function block

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

B

BIN
Binary (digital)

C

CRC
Cyclic redundancy check (Checksum)

CAN
Controller Area Network

D

DO
Digital output

DI
Digital input

Default
Factory setting

DEZ
Decimal

E

EnDat 2.2
Motor encoder interface protocol of the company Heidenhain

EMV
Electromagnetic compatibility

E-encoder
Absolute encoder, singleturn, EnDAT 2.1 with additional sine and cosine track

EF2
Power output stage enable

EF
Power output stage enable

EDS
Electronic data sheet

E/A
In- and outputs

EnDat 2.1
Motor encoder interface protocol of the company Heidenhain

EtherCAT
Real-time Ethernet bus

EMC
Electromagnetic compatibility

F

FSoE
Fail-Safe over EtherCAT

FPLC_PRG
Real-time PLC task, synchronized to device cycle

FL
Command (Causes a new system run-up)

Firmware
System software, loaded by AMK

FIFO
First in - first out (Memory which outputs the saved values the same sequence they were saved)

F-encoder
Absolute encoder, multiturn, EnDAT 2.1 with additional sine and cosine track

FB
Function block

G

g_yourDevice
Symbolic name of a device in a PLC project. The name is defined in CoDeSys configuration: devices

H

HEX
Hexadecimal, 0x...

I

iSA-VIS
Web visualization

iSA
AMKASMART decentralized controller with power supply

I-encoder
Incremental encoder, optical encoder with sine and cosine track and zero pulse

iSA-PNC
Numerical Control Motion

iX
AMKASMART decentralized inverter

ihXT
AMKASMART Servo motors with integrated inverter

iDT
AMKASMART Servo motors with integrated inverter

ID
Parameter identification numbers acc. to SERCOS Standard

iC
AMKASMART decentralized inverter with power supply

iSA-PCO
PLCopen

K

KW-Rxx
AMKASYN controller card for installation into compact inverter

KU
AMKASYN compact converter

KEN
AMKASYN compact power supply without recovery

KE/KW
Modular AMK drive system (contains compact power supply KE, compact inverter KW with controller card and applicable option card)

KE
AMKASYN compact power supply with recovery

KWZ
AMKASYN compact two-axes inverter to control two motors

KWD
AMKASYN compact double inverter to control two motors

L

LAN
Local Area Network

M

MDB
Motor database; it contains information about all AMK motor parameters

Modulo
Modulo processing of position setpoint and actual values

N

NK
Cam switch

P

PDO
Process Data Object

Parameter
Identification number acc. to SERCOS standard

PDK_xxxxxx_abcdefgh
Product documentation; xxxxxx - AMK part no. , abcdefgh - name

P-encoder
Absolute encoder singleturn, EnDAT 2.2 light

PGT
Periphery basic clock Fetch cycle in the basic device to which the drive controller is synchronized (The cycle time is according to ID2)

PLC_PRG
Task which is not synchronized to the device cycle

POU
Program organization unit (PLC program elements; types program, function or function block)

PRG
Program

Q

QBR
Acknowledgment motor holding brake

Q-encoder
Absolute encoder multiturn, EnDAT 2.2 light

QFL
Acknowledgment clear error; the command clear error was executed

QRF
Acknowledgment controller enable; the drive is controlled in the activated operation mode

QUE
Acknowledgment DC bus on; shows that DC bus is loaded

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.

RO
Read Only

S

STO
Safe torque off (Safety function acc. to DIN EN 61800-5-2)

SERCOS
Standardized digital interface for communication between controller and field bus participants.

SEEP
Device-internal memory, serial EEPROM

SBUS
AMK-specific protocol for serial interfaces

SafePMT
Safe parameter editor

SBM
System ready message; shows that the device is error-free In case of error. SBM will be reset

T

TwinCAT
Automation software

T-encoder
Absolute encoder, multiturn, RS485 Hiperface with sine and cosine track

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