Kollmorgen Automation Suite

KAS IDE User Guide

Document Edition: D

October 2013

Valid for Software Revision 2.7

Valid for AKD PDMM firmware version: 1.10

Part Number: 959715

Keep all manuals as a product component during the life span of the product.
Pass all manuals to future users / owners of the product.

Trademarks and Copyrights

Copyrights

Trademarks

KAS and AKD are registered trademarks of Kollmorgen™.
SERVO**STAR** is a registered trademark of Kollmorgen™.
Kollmorgen™ is part of the Danaher Motion company.
Windows® is a registered trademark of Microsoft Corporation
EnDat is a registered trademark of Dr. Johannes Heidenhain GmbH.
EtherCAT® is registered trademark of Ethercat Technology Group.
PLCopen® is an independent association providing efficiency in industrial automation.
INtime® is a registered trademark of TenAsys® Corporation.
Codemeter is a registered trademark of WIBU-Systems AG.

All product and company names are trademarks™ or registered® trademarks of their respective holders. Use of them does not imply any affiliation with or endorsement by them.

Kollmorgen Automation Suite is based on the work of:

- AdvXMLParser, an XML file parser modified to handle the ALS project (distributed under the terms of the license – see terms – see "Licenses" on page 745 for terms).
- AjaxFileUpload, software (distributed under the MPL License – see terms – see "Licenses" on page 745 for terms).
- Apache log4net library for output logging (distributed under the Apache License – see "Licenses" on page 745 for terms).
- bsdtar and libarchive2, a utility and library to create and read several different archive formats (distributed under the terms of the BSD License – see terms – see "Licenses" on page 745 for terms).
- bzip2.dll, a data compression library (distributed under the terms of the BSD License – see "Licenses" on page 745 for terms).
- Curl software library
- DockPanel Suite, a docking library for .Net Windows Forms (distributed under the MIT License – see terms – see "Licenses" on page 745 for terms).
- FileHelpers library to import/export data from fixed length or delimited files (see "Licenses" on page 745 for terms).
- GNU gzip[1] (www.gnu.org) is used by the PDMM (distributed under the terms of the GNU General Public License http://www.gnu.org/licenses/gpl-2.0.html).

[1]Copyright (C) 2007 Free Software Foundation, Inc. Copyright (C) 1993 Jean-loup Gailly. This is free software. You may redistribute copies of it under the terms of the GNU General Public License <http://www.gnu.org/licenses/gpl.html>. There is NO WARRANTY, to the extent permitted by law. Written by Jean-loup Gailly.

- GNU Tar[1] (www.gnu.org) is used by the PDMM (distributed under the terms of the GNU General Public License http://www.gnu.org/licenses/gpl-2.0.html).
- jQuery.Cookies, a Javascript library for accessing and manipulating HTTP cookies in the web browser (distributed under the MIT License – see "Licenses" on page 745 for terms).
- jQuery File Tree, a file browser plugin (distributed under the MIT License – see "Licenses" on page 745 for terms).
- jQueryRotate, a plugin which rotates images (img html objects) by a given angle on web pages (distributed under the MIT License, http://opensource.org/licenses/mit-license.php).
- JsonCpp software (distributed under the MIT License –see terms see http://json-cpp.sourceforge.net/LICENSE for terms).
- LZMA SDK (http://www.7-zip.org/sdk.html), used to compress crash dump information (available as public domain).
- Mongoose v3.7, an embedded web server library (distributed under the MIT License – see "Licenses" on page 745 for terms).
- MVVM Light Toolkit components for Model – View –ViewModel patterns with Windows Presentation Foundation (distributed under the MIT License – see "Licenses" on page 745 for terms).
- Qwt project (distributed under the terms of the GNU Lesser General Public License – see "Licenses" on page 745 for terms).
- U-Boot, a universal boot loader is used by the AKD-PDMM (distributed under the terms of the GNU General Public License, http://www.gnu.org/licenses/gpl-2.0.html). The U-Boot source files, copyright notice, and readme are available on the distribution disk that is included with the AKD-PDMM.
- ZedGraph class library, user control, and web control for .NET (distributed under the LGPL License – see "Licenses" on page 745 for terms).
- Zlib software library
- Zlib1.dll, a data compression library (distributed under the terms of the BSD License – see terms – see "Licenses" on page 745 for terms).

All other product and brand names listed in this document may be trademarks or registered trademarks of their respective owners.

Disclaimer

The information in this document (Version 2.7 published on 10/7/2013) is believed to be accurate and reliable at the time of its release. Notwithstanding the foregoing, Kollmorgen assumes no responsibility for any damage or loss resulting from the use of this help, and expressly disclaims any liability or damages for loss of data, loss of use, and property damage of any kind, direct, incidental or consequential, in regard to or arising out of the performance or form of the materials presented herein or in any software programs that accompany this document.

All timing diagrams, whether produced by Kollmorgen or included by courtesy of the PLCopen organization, are provided with accuracy on a best-effort basis with no warranty, explicit or implied, by Kollmorgen. The user releases Kollmorgen from any liability arising out of the use of these timing diagrams.

[1]Copyright (C) 2007 Free Software Foundation, Inc. License GPLv2+: GNU GPL version 2 or later <http://gnu.org/licenses/gpl.html> This is free software: you are free to change and redistribute it. There is NO WARRANTY, to the extent permitted by law. Written by John Gilmore and Jay Fenlason.

This page intentionally left blank.

Table of Contents

1 Document History

Edition	Software Version	Date	Description
0.2		September 2010	First release for KAS R1
A	2.5	October 2013	Initial release of KAS IDE and AKD PDMM
B	2.6	December 2012	EtherCAT devices backup & restore; PLCopen Registration; Superimposed Axes; Cam on the fly; Ethernet/IP; Profinet; Reset to factory settings; Save project on controller; Password protection
C	2.7	June 2013	

TIP When all source is under SVN, add the unique version number issued from SVN to served as the unique ID

This page intentionally left blank.

2 Preface

This chapter explains how to use the online help provided with Kollmorgen Automation Suite™.

2.1 System Requirements for KAS IDE and KAS Simulator

KAS IDE and KAS Simulator are compatible with any PC having the minimum following hardware:

Element	Description
Operating System	Microsoft® Windows® XP (32-bit XP Pro SP3 or above) and Windows 7 (32 and 64-bit). TIP For optimal performance, please be sure your operating system is fully updated with the latest patches.
Processor type	Intel® Pentium® M or equivalent processor at 1.5 GHz or greater.
Memory	512 MB RAM or greater (which is recommended for complex applications)
Storage	1 GB hard drive or compact flash space
Display	WXGA+ (1440 x 900) or higher-resolution monitor with 24-bit color NOTE Better results are achieved with OpenGL and 3D cards
Connectivity	1 Ethernet port, at either 100Mbits/s or 1Gbits/s. IMPORTANT A 100Mb network is required in order to allow the IDE to Runtime communication to work in all conditions. The AKD WorkBench AutoTuner and Scope both require 100Mb of bandwidth to function properly.
WebServer	A modern web browser is required to access the web server. We recommend Internet Explorer 9 , Mozilla FireFox (v.11 or later) , or Google Chrome .

Table 1-1: Minimum System Requirements for the KAS IDE

IMPORTANT KAS IDE and KAS Simulator should not be installed on a Kollmorgen Industrial PC (PAC). The IDE and Simulator are for use on PCs only.

2.1.1 System Requirements for the KAS Runtime

Requirement	Description
Operating System	The KAS Runtime is supported under XP embedded (XPe) for the industrial PC.
Recommended network environment	Only a **Local Network** connection can ensure the communicate between the KAS IDE and the KAS Runtime. IMPORTANT The KAS IDE is not able to communicate to the KAS Runtime through NAT connection.
Supported Kollmorgen Industrial PC	KAS Runtime is compatible with AKC hardware models

2.2 Using Online Help

The online help is your main reference for using KAS. However, more up-to-date information and material are available on our Web site. The online help provides extensive cross-referencing, enabling you to find more information on a given topic in other locations.

2.2.1 Alerts and Warnings

IMPORTANT Alerts you that an operation or action could have unexpected results or be

IMPORTANT irreversible. Not following warning notices could also result in minor or moderate damage (e.g. data loss) or undesirable effects.

NOTE Provides important information to ensure a thorough understanding of product use.

TIP Provides further information or advice to help you work efficiently.

2.2.2 Browse the Table of Contents

The online help can be used like any Web site with links, back and forward buttons.

On the left side of the interface, the topics listed in the Contents (TOC) provide you with assistance on every aspect of working with KAS. Navigate through the TOC books and pages to find the information you need. When you click a topic page, it displays in the workspace.

The TOC structure is based on a top-down approach with **concepts**, followed by **procedures**, and then **references**.

You can access the topics as follows:

- To learn **about some concepts**, see the Understanding chapter for conceptual explanation
- To learn **how to perform** a task, see the Using chapter for tasks description

TIP All chapters have extensive links to the other relevant sections so it does not really matter where you start.

2.2.3 Search the Online Help System

To find information, you can use:

- **Contents**
- **Glossary and Acronyms**
  To display a complete list of abbreviations and acronyms, select Glossary or Acronyms at the bottom of the Contents.
- **Index**
  Select the Index tab to open the online help index. Navigate through the index list (or use the "Search text box" at the top) to find keywords for the information you need. When you click a keyword in the index, the associated topic links are listed in the Index Results window. Click any of the links to open a specific topic.

NOTE Using the "Search text box" at the top of the index list is not working from KAS IDE. This current limitation does not happen when you open the help in your Internet browser.

- **Search**
  Select Search to open the Help Search window. This window provides a way for you to quickly search for information in the online help. Simply enter one or more keywords in the search field and click the **Search** button. Links to topics containing those keywords are listed below. Clicking on a link displays the topic in the work-space.

NOTE
- Search is **not** case sensitive but the results may be ranked differently based on case sensitivity.
- Wildcards are *not* supported.

TIP After a relevant topic is located using the **Search** command, view the Contents to understand its relationship to other related topics.

Syntax for an effective Search.

Variable	Description	Example
	Search for one or more words. When a group of words are entered into the search field, "or" is inferred.	cat dog mouse
" " (wrap a string in quotes)	Search for a phrase. **NOTE** The search engine ignores certain commonly used words. For example, a, an, the, of, to, be, you, your, when, how- ever, for, that, can (and more). If your search results are not successful, delete some of the less important words.	Successful: "Pipe Network Functions" Unsuccessful: "what is the list of Pipe Network Functions"
OR ‡ | (pipe symbol)	Search for "either of" or "any of" specific strings.	cat or dog or mouse "windy day"|"cumulus cloud"
AND ‡ + (plus symbol) & (ampersand)	Search for two or more specific strings.	cat And dog "windy day"+rain "noodle soup"&"animal crackers"
NOT ‡ ! (exclamation mark)	Search for all topics that do not contain a given word or phrase.	not fish ! flood
^ (carat symbol)	Search for all topics that contain one string but do not contain another.	cat ^ mouse
( ) parenthesis	Combinations of the above.	cat and (dog or mouse) cat or dog (! fish)

‡ The syntax is case insensitive.

About Rankings

Results returned are case insensitive. However, ranked results take into consideration case matches and assigns higher scores. Therefore, a search for “motion” followed by a search for “Motion” would return the same number of help topics, but the order in which topics display are different.

TIP

Use the Favorites Window

- If you plan to refer to a specific topic, you can click the [Add topic to favorites] button
- If you perform a search in the online help and plan to make the same query often in the future, you can click the [Add search string to favorites] button

TIP

Use the lists of Figures, Tables and Concepts

Select the Figures, and Tables available at the bottom of the Contents to display the complete list.

For more details on each icon in the header, see "Help Toolbar" on page 661.

2.2.4 Use the Context-Sensitive Help

About Context-Sensitive Help (CSH)

Context-Sensitive Help is used to link specific dialogs or windows in the KAS IDE to existing help topics. When you open a dialog or window, you can quickly open a help topic about it. This topic can be at a very specific level, or more global to a major feature.

To get specific help:

- Open the dialog box and set the focus to the item where you need help
- Click the **F1** key

2.2.5 How to Send Feedback

With KAS IDE, you can improve the content by:

- Adding comments (see call out 1)
- Rating pages 2

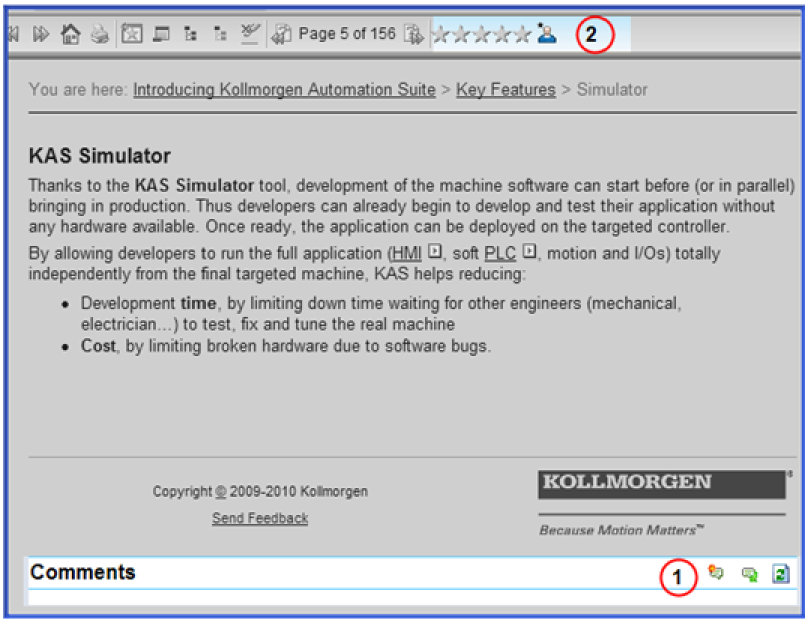


Figure 1-1: Send Feedback

This dialog lets you enter and submit a community-wide comment that can be viewed (and replied to) by all other users viewing the online help.

2.2.5.1 How to Add a Comment

1. Select the page you want to comment
2. Click the Add Comment button

NOTE The Comments window pane is displayed at the bottom

> **NOTE** of the page. As the form opens at the top of the page, you have to scroll up depending on the size on the current page.

3. If it is your first time submitting a comment, you are required to complete the registration process
4. In the Add Comment dialog, provide a subject and enter your comment
5. Click **Submit**

2.2.5.2 How to Rate a Page

1. Select the page you want to rate
2. In the toolbar, click the rating button ☆☆☆☆☆
3. In the Topic Rating popup, click on the stars to provide a rating for the page

Rating the page anywhere from one to five stars:

★	**incorrect** and needs correction (please provide a comment as well)
★★	**not helpful** at all
★★★	can be **improved**
★★★★	contains **enough** information
★★★★★	very **helpful**

2.2.5.3 How to Register

1. You must create a user profile to post comments to this online help. When requested, you have to provide information such as your username and email address.
2. Wait to receive the email, then follow the instructions in the email to complete activation.

> **NOTE** The registration process needs to be done only once.

> **TIP** You can edit your user profile with the button to modify when you want to receive email notifications.

2.3 Learning Kollmorgen Automation Suite

To learn Kollmorgen Automation Suite, you can either:

- Navigate this online help and choose chapters depending on your experience, or
- Read the printed materials

2.3.1 Access Chapters

The KAS documentation includes information for readers from a variety of backgrounds. To get the most out of the documentation, we recommend that you start by reading the chapters that are most relevant to you. Within each chapter, read through the topics in sequence.

2.3.1.1 Beginner

- Find basic information about KAS in chapter "Introducing Kollmorgen Automation Suite" on page 47
- If you are not familiar with the concepts behind KAS, read the chapter "Understanding KAS" on page 73

- An overview of the KAS IDE User Interface is in chapter "Describing KAS Graphical User Interface" on page 591
- To get information on how to run and debug the project, read paragraph "Step 3 of 6 - Launch KAS Simulator" on page 322 and paragraph "Testing and Debugging the Project" on page 327

2.3.1.2 Advanced User

- In order to design and create a project, refer to the chapter "Using the KAS IDE" on page 181
- Go to chapter "Tools" on page 393 if you need explanations about the tools used by the KAS IDE
- For in-depth information, refer to chapter "Advanced Topics" on page 459

2.3.2 Read KAS Manuals

If you prefer to read printed material, the following manuals (in PDF format) are available under the C:\Program Files\Kollmorgen\Kollmorgen Automation Suite\Help folder

KAS Title	pdf	Description
Getting Started		Covers the main steps to get your KAS system up and running **What does it contain?** • **HW Installation** (Connection and Wiring) Wiring & hardware details, connectors, system diagrams • **HW Configuration** Basic configuration and settings needed to start the HW components (HMI + Industrial PC + Fieldbus + I/O) • **SW Installation** KAS software setup

<table>
<tr><th>KAS Title</th><th>pdf</th><th>Description</th></tr>
<tr><td>30 Minutes to Motion</td><td></td><td>Covers the main topics to help you start quickly with KAS IDE.

NOTE The objective is to familiarize you with the basic principles and the way the program works by creating a simple motion application project.

What does it contain?
Key Features
Explore the Workspace
Become familiar with KAS user interface
Build a motion project
Almost every task that you perform in KAS falls under one of the following basic steps (which may not always be completed in the following order):
Start Projects - Create a project from scratch, or modify an existing project.
Add Components - Add elements to build your project, such as PLC programs, variables and Pipe Network necessary to control the motion part of your system.
Build Output - Select a device and generate the application that you will deliver to users. see "Running the Project" on page 318
Run Output - Make the output accessible to your end-users.</td></tr>
<tr><td>IDE User Manual</td><td></td><td>Contains the content to help you with KAS IDE, except the topics included in the Reference Manuals</td></tr>
<tr><td>Reference Manual - PLC Library</td><td></td><td>Contains Technical References on PLC Programming Languages and Library</td></tr>
<tr><td>Reference Manual - Motion Library</td><td></td><td>Contains Technical References on Motion Library for Pipe Network and PLCopen</td></tr>
<tr><td>PAC Web Server User Manual</td><td></td><td>Describes use of the PAC web server.</td></tr>
</table>

Table 1-2: List of KAS Guides in PDF Format

KASRelease Notes

The KAS version 2.7 Release Notes contain fixed limitations, known limitations, workarounds, and information on all hardware and software components that have been updated, changed or added in this release.

TIP The KAS IDE allows you to include references to external files such as the PDF files listed above. For more details, refer to paragraph "Use the Reference Folder" on page 343.

Additionally, you can add in the PDF your own comments, tips and tricks, provided that you have Adobe Acrobat®.

3 Introducing Kollmorgen Automation Suite

This chapter introduces Kollmorgen Automation Suite (KAS) with a product **overview** that lists the features, the components, and the different implementations.

KAS is intended for engineers who want to design and build high-performance motion control and automation systems. KAS is designed to allow you to quickly and easily compose a motion application. It can be achieved with all of the re-use and flexibility of the KAS libraries in conjunction with the IEC 61131-3 programming languages.

As can be seen, KAS can cover a wide variety of applications:

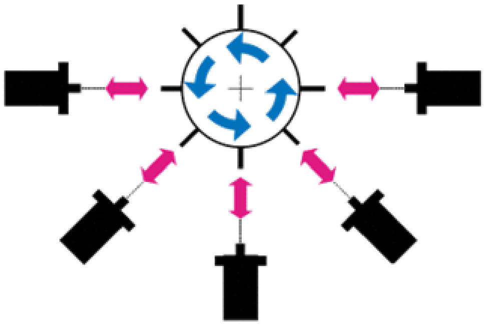


Figure 2-1: Synchronized Feeder

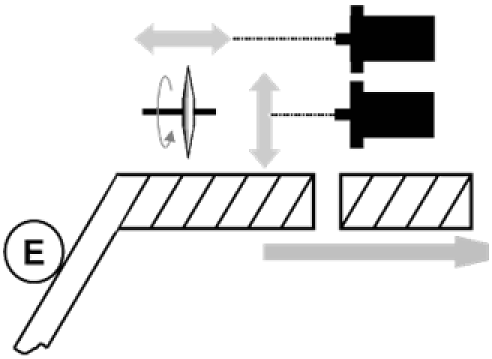


Figure 2-2: Spring Winding

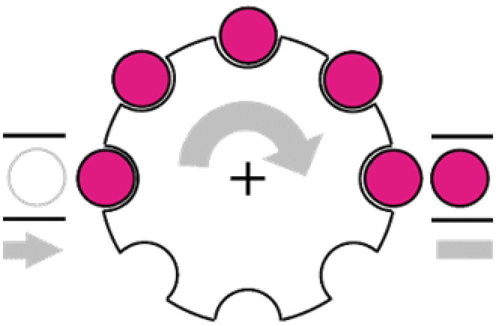


Figure 2-3: Synchronizer

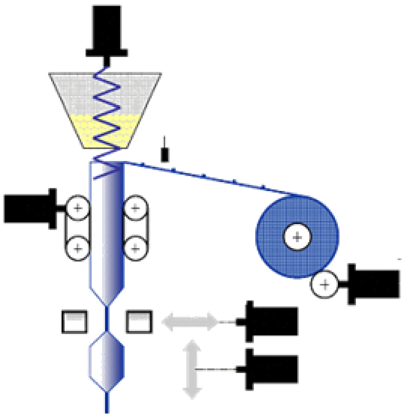


Figure 2-4: Form Fill Seal

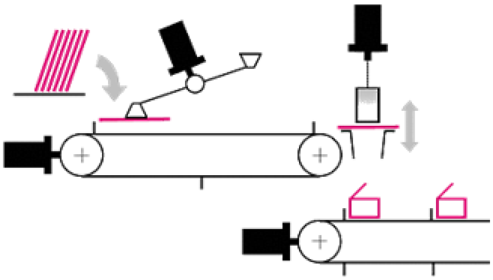
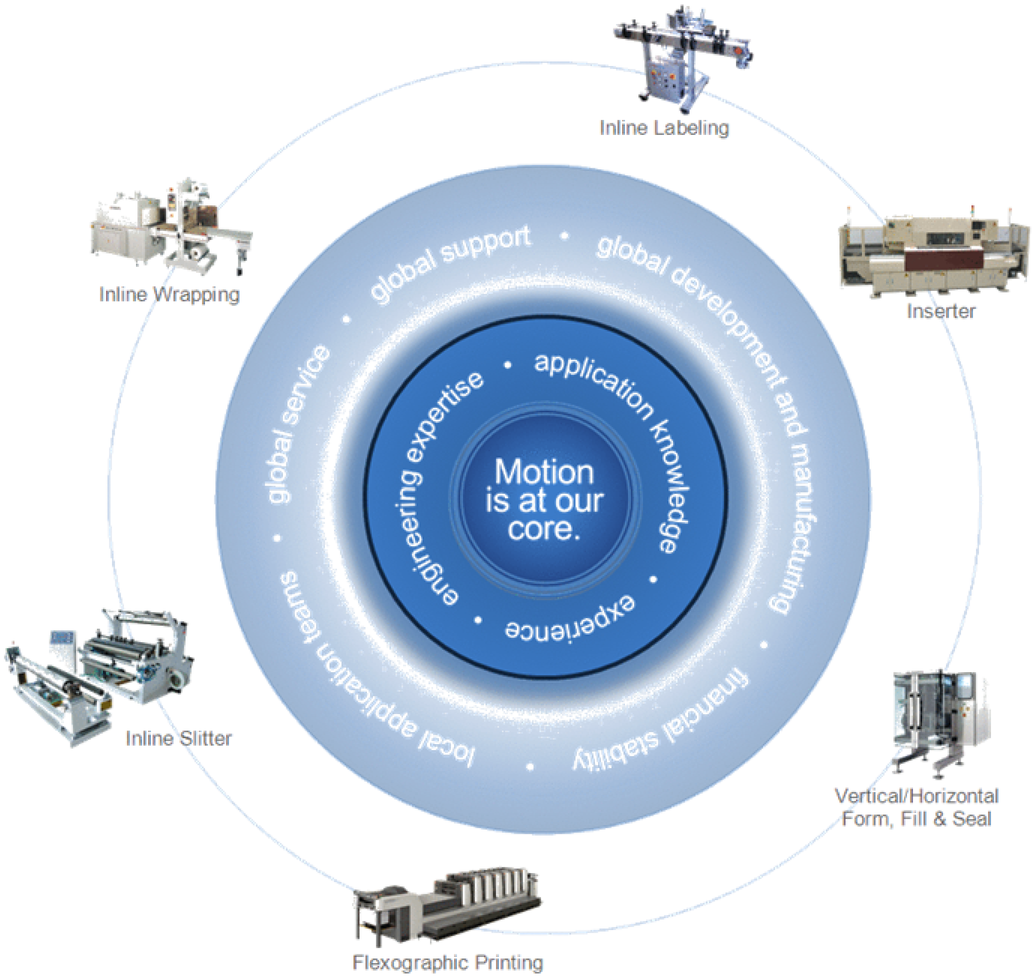


Figure 2-5: Carton Erector

3.1 Key Features

The purpose of KAS is to include in a single software package, all the tools you need (i.e. a soft PLC, configuration tools, and a motion controller) to create an automation system.

An overview of an investment in Kollmorgen for Motion Control and Automation Systems solution can be encapsulated as follows:



Kollmorgen Automation Suite (also known as KAS) is Kollmorgen's all-in-one solution for designing, developing and maintaining automation systems. As a solution offering, it brings many years of Motion Control experience to the market, and this is coupled with technical expertise and experience, global delivery capability, and strong financial performance.

KAS is a set of software packages designed to run and take advantage of Kollmorgen's extended set of integrated hardware products such as Programmable Automation Controllers, Programmable Drives, AKD drive family, award winning components like the AKM motor family, gear boxes, I/O terminals and Human Machine Interaction terminals (or Operator Interfaces.)

KAS provides machine builders with a **high-performance**, **cost-effective** and **easy to use** solution for building machines. KAS achieves this goal by integrating in a **coherent, intuitive, flexible** way the three main functionalities of a machine:

- Precise control of all moving parts (Motion control)
- Interface with machine operators (HMI)
- PLC programming of the machine (IEC 61131-3 Soft PLC)

KAS is made of two different software components:

- KAS IDE - the Integrated Development Environment allowing the development and monitoring of complex machine automation systems
- KAS Runtime - the Runtime engine offers the functionally of both **a High-Performance Motion and a PLC Engine**

3.1.1 Integrated Development Environment

KAS comes with a powerful Integrated Development Environment (IDE) (commonly named **KAS IDE**) which provides machine builders with all the necessary tools for designing, programming, configuring, debugging and maintaining machine applications. KAS uses the same interface, tools, and libraries to create applications for various types of KAS controllers (PAC, Programmable Drives)

With the KAS IDE, system engineers can:

- Create new application projects using predefined or custom application **templates**
- Define the machine hardware architecture (motion bus, fieldbus, controllers, drives and motors) as well as the machine program (HMI panels, IEC 61131-3 programs and function blocks, motion block, profiles and axes) from a centralized **Project Explorer** which is based on a tree-structure
- Develop PLC programs, functions and function blocks using the five IEC 61131-3 programming languages (ST, IL, FFLD, FBD and SFC), the IEC 61131-3 standard library and KASFunction Block libraries dedicated to motion, communication and monitoring
- Centrally manage all IEC 61131-3 variables with KAS**variable dictionary** and map logical variables to physical inputs and outputs
- Create and organize your own libraries of functions and function blocks
- Easily set up **HMI panels** by means of graphical objects that are part of the HMI control library; and map graphical objects to IEC 61131-3 variables
- Graphically design advanced multi-axis relations using **Kollmorgen's graphical motion programming** environment - also called the ***Pipe Network*** - with its tool generating code automatically
- Use ultra-fast IEC 61131-3 compiler to validate the syntactical correctness application code
- Configure hardware devices via an integrated set of configuration tools (for instance AKD drives, EtherCAT I/O terminals, Profibus, etc.)
- Access controller devices to download, start and stop the application, watch log messages and send shell commands to the target device
- Debug the application by inserting break points and stepping into the code or by monitoring internal values (IEC 61131-3 variables, motion positions, drive's internal values) directly in the editors or with KAS advanced **softscope** tool
- Access the full online documentation

3.1.2 KAS Runtime

Kollmorgen Automation Suite Runtime (commonly named **the KAS Runtime**) offers, in a single software package, the functionally of both a soft PLC and a motion controller.

The KAS Runtime (virtual machine) is a high-performance deterministic environment designed to run on different hardware platforms ranging from low-cost **programmable drives** to **high-end Programmable Automation Controllers**. This gives machine builders all the flexibility when designing their machines.

KASsupports many configurations when integrating machines:

- Ranging from **single-axis** to more than **200 tightly coordinated axes**
- With a **centralized** (Programmable Automation Controllers), **distributed** (Programmable Drives) or **mixed** (Programmable Automation Controllers + programmable drives) control architecture
- Running on a **single or multiple controllers**
- Communicating via **Ethernet, OPC, CAN** or **Profibus**
- Using the high-performance **Pipe Network** or the standard **PLCopen** function blocks
- Controlling Kollmorgen's drives (**AKD,** some of the **Servostar Sxxx** drive family), **AKM** motors, and **AKT** terminals for I/Os products

The KAS Runtime can be used in the two different contexts:

- With a controller implementation (PAC)
- With a master drive implementation (AKD PDMM)

See paragraph "Different Implementations" on page 69 for more details.

3.1.3 KAS Simulator

Thanks to the **KAS Simulator** tool, development of the machine software can start before (or parallel with) bringing in production. Thus developers can already begin to develop and test their application without any hardware available. Once ready, the application can be deployed on the targeted controller.

By allowing developers to run the full application (HMI, soft PLC, motion and I/Os) totally independently from the final targeted machine, KAS helps reducing:

- **Development time**, by limiting down time waiting for other engineers (mechanical, electrical…) to test, fix and tune the real machine
- **Cost**, by limiting broken hardware due to software bugs.

3.2 Looking at Kollmorgen Automation Suite

3.2.1 Physical View

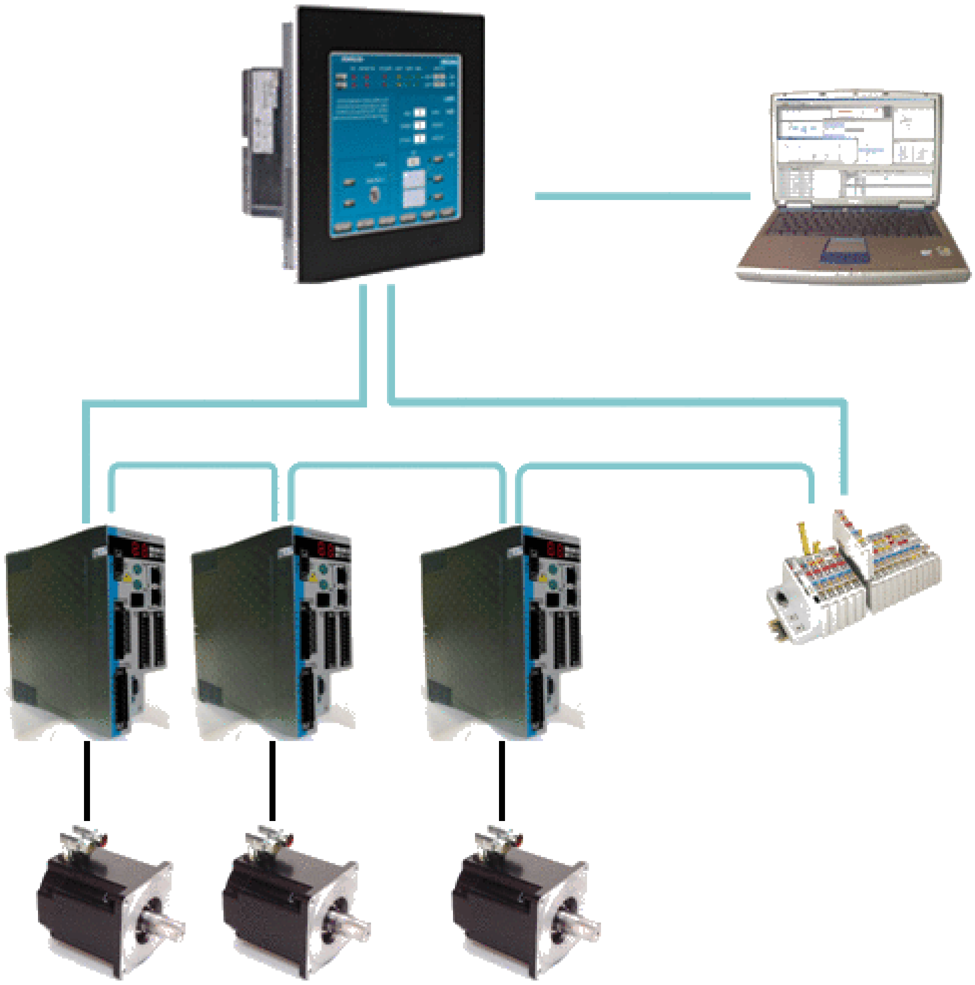


Figure 2-6: Example of Automation System

3.2.2 Logical View

An automation system usually needs an organized hierarchy of controller systems to function and usually including the following items:

Item	Call out#	Description
HMI	1	At the end-user top level, the Human Machine Interface is where the operator can monitor or operate the system. It is usually composed of a panel on a PAC.
Communication	2	HMI is linked to the middle layer via a non time critical communication system (e.g. Modbus TCP protocol on Ethernet).

Item	Call out#	Description
PLC	3	Programmable Logic Controllers is a digital computer used for automation of industrial processes, such as control of machinery on factory assembly lines. It is used to synchronize the flow of inputs from (physical) sensors and events with the flow of outputs to actuators and events.
Motion Engine	4	There are two Motion Engines available: Pipe Network and PLCopen. The Motion Engine implements different motion algorithms and functions to create, access and delete pipes, pipe blocks and axes. It also provides a set of Functions and Function Blocks that IEC 61131-3 applications can use to control the behavior of these algorithms.
Fieldbus	5	The fieldbus is the way to connect instruments in a plant design by linking the PLC to the external systems.
I/O	6	Input/Output refers to the communication between your automation system, and the outside world.
Drive	7	A Drive is an electronic device that provides power to a motor or servo.
Motor	8	At the bottom of the control chain is the motor which actually does the work.

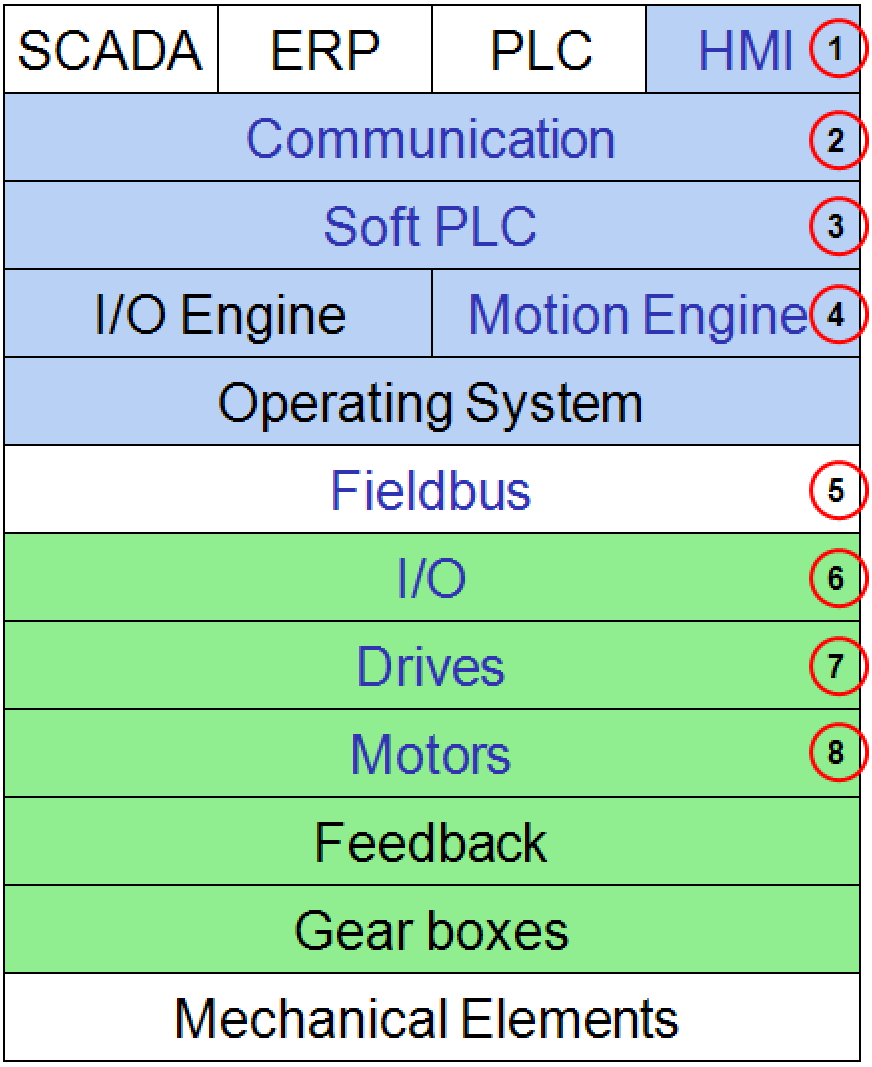


Figure 2-7: Logical Architecture

Color Legend	Description

3.2.3 Architectural View

The block diagram shows KAS architecture with a Programmable Automation Controller platform running both Windows operating system and INtime real-time kernel.

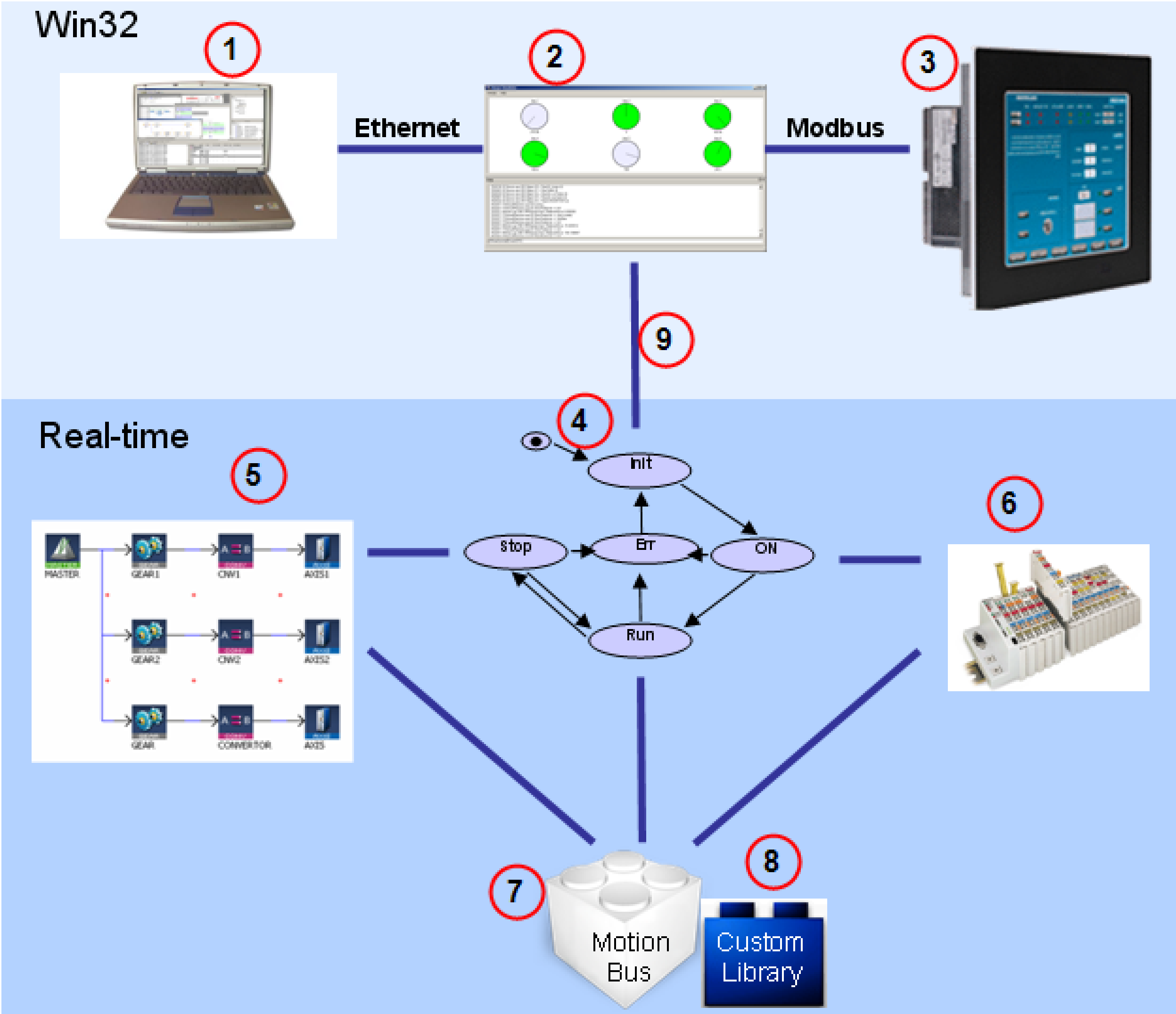


Figure 2-8: Architectural view with a Programmable Automation Controller Implementation

The **Win32** sub-system runs the non real-time part which is composed of:

Item	Call out#	Description
KAS IDE development tools	1	Allows to prepare the project (i.e. design, create and run virtually)
KAS Runtime Server	2	Also called the KAS Runtime Front-end
HMI	3	Available when integrated on a Programmable Automation Controller platform (not present when integrated a programmable drive)

Table 2-1: Architectural View - **Win32** Sub-system

The **RTOS** platform runs the KAS Runtime engine which is composed of:

Item	Call out#	Description
IEC 61131-3 virtual machine	4	Responsible for managing an IEC 61131-3 application with its programs and variables
Motion manager	5	Manages motion engines, axis objects and motion bus drivers. The KAS Runtime comes with two motion engines: Pipe Network and PLCopen. The motion engine implements different motion algorithms and functions to create, access and delete pipes, pipe blocks and axes (e.g. MLAxisCreate, MLGearInit, MLPipeAct). It also provides a set of Functions and Function Blocks that IEC 61131-3 applications can use to control the behavior of these algorithms
I/O manager	6	Manages I/Os and I/O drivers. It works closely with the VM Manager instances to map transparently all IEC 61131-3 variables declared as input or output
Motion Bus	7	A plug-in giving access to the EtherCAT network
custom function blocks	8	A plug-in implementing custom function blocks

Table 2-2: Architectural View - **RTOS** Sub-system

Interface between the **Real-time** and **Win32** sub-systems.

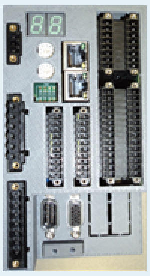
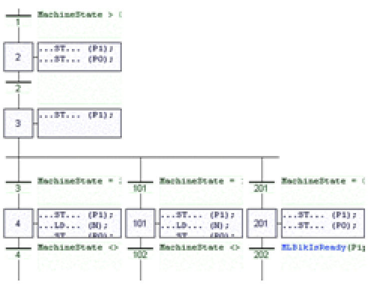
Item	Call out#	Description
interface	9	Interface between real-time and non real-time software parts is done via shared memory buffers

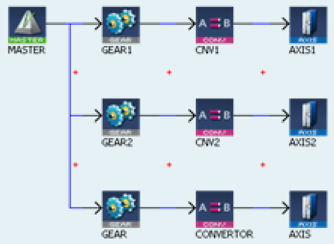
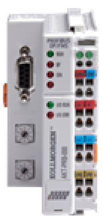
The Runtime communicates with the IDE during operation to:

- Receive further instructions from the IDE such as a direct motion command
- Provide status information to the IDE for motion and operation of the application program
- Provide information displayed on the IDE scope
- Provide Log information to the IDE

NOTE When the KAS Runtime is implemented with a programmable drive, the interface between the real-time and non real-time parts is done via Ethernet based on TCP/IP protocol.

3.3 KAS Breakdown

Domains	**Concept** (Technology)	**Task** (Tools)	**Reference**
HMI 		Kollmorgen Visualization Builder Add an HMI	HMI
Controllers PAC 	Programmable Automation Controllers	Add Controller Configure Controller	Controller
AKD PDMM 	Programmable Drive Multi-axis Master		
PLC 	IEC 61131-3	ST editor IL editor FBD editor FFLD editor SFC editor Variable dictionary Softscope	STLanguage IL Language FBD Language FFLD Language SFC Language

Domains	**Concept** (Technology)	**Task** (Tools)	**Reference**
Motion Engine 	Motion Concept	Design Pipe Network Pipe Network Editor Design CAM Cam Profile Editor Softscope	
Operating System	XP embedded INtime		
Fieldbus 	EtherCAT Profibus SynqNet	Configure EtherCAT Motion Bus	Motion bus Cables
I/O Terminal 	CANopen DeviceNET	Add I/O terminal I/O mapping to variable I/O Editor	"Remote Input/Output (I/O Terminals)" (see page 676)

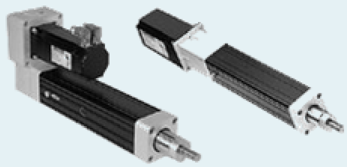
Domains	**Concept** (Technology)	**Task** (Tools)	**Reference**
Drive 	AKD S300	Add and configure drive Drive Configuration AKD Firmware Download	AKD
Motor	Kollmorgen Servomotor		AKM
Mechanical 			Linear Positioners Gearheads

Table 2-3: KAS - Technologies and Tools

3.3.1 Human-Machine Interface

Figure 2-9: Hardware to Display the Human-Machine Interface

PLCs interact with people for the purpose of configuration, alarm reporting or everyday control. A Human-Machine Interface (HMI) is employed for this purpose. A simple system uses buttons and lights to interact with the end-user. Text displays are available as well as graphics on the touch panels.

Most modern PLCs can communicate over a network to some other systems, such as a computer running a SCADA system.

The communication between the HMI and the PLC is based on Modbus over TCP/IP (Modbus TCP is the Ethernet version of Modbus) by means of a standard Ethernet cable that connects the two devices.

This communication is done in the background, asynchronously, every 200 milliseconds. Variables defined in the HMI (see "Map Variables to HMI" (see page

316)) to describe the interface are passed to the PAC or AKD PDMM this way. This means there is no data coherency in the data exchange because the variables read by the Modbus do not come from the same PLC cycle. As this data has a rather low priority and is interpreted by human feedback, it should never be noticed by the user.

3.3.2 PAC and Touch Panel PC

Designed for industrial applications, a PAC is a powerful and robust computer which can be used in close proximity to machinery.

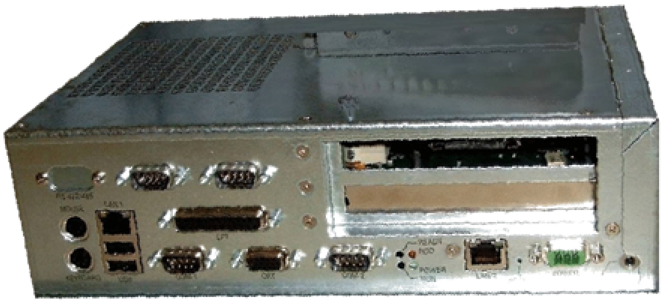


Figure 2-10: Programmable Automation Controller

To give access to the HMI when there is no dedicated HMI hardware, KAS PAC usually includes a touch-screen panel as a combined input and output device.

Figure 2-11: Touch Panel PC

3.3.3 Programmable Drive Multi-Axis Master (AKD PDMM)

Figure 2-12: High, medium and low voltage AKD PDMMs.

3.3.3.1 AKD PDMM Hardware

The AKD PDMM comprises three printed circuit boards (PCB)

- Power board
- AKD control card:
- AKD PDMM option card: QorIQ /Freescale MPC8313E RDB with P1020 processor (800MHz)

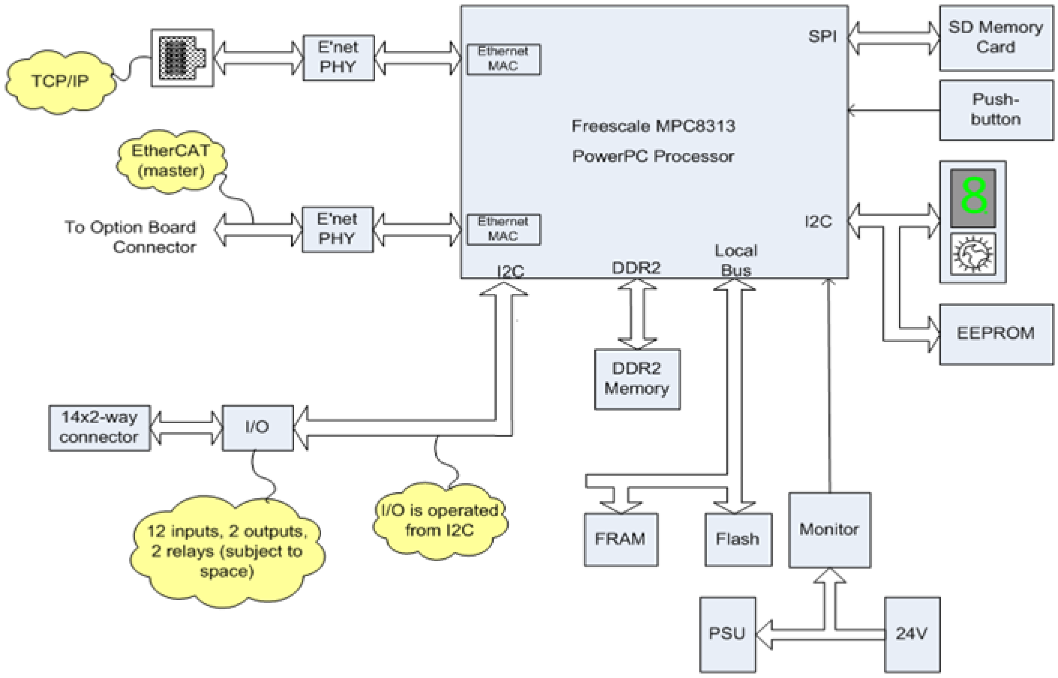


Figure 2-13: AKD PDMM card

3.3.3.2 AKD PDMM Rotary Switch

On the AKD PDMM, the rotary switch can be set on a position from 0 to 9.

Position 0

The drive tries to get an IP address from a DHCP server, but in case of no DHCP on the LAN, it uses Zero configuration networking to automatically create a usable IP address.

Position 1

The IP address of the drive can be defined manually.

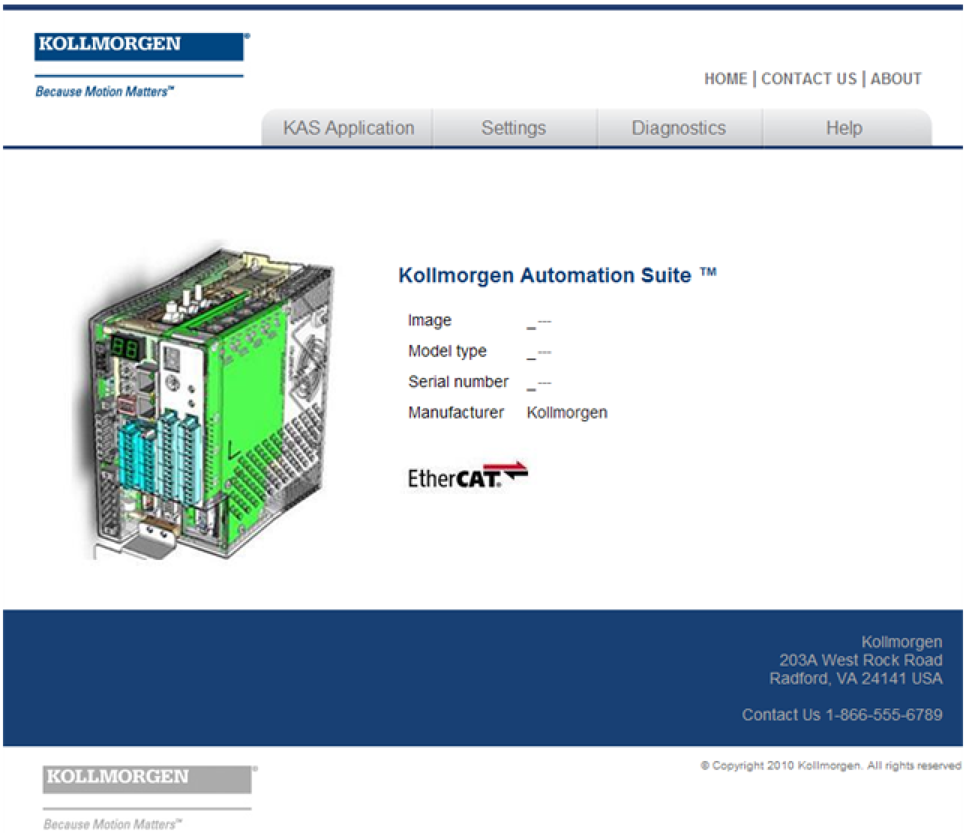
Position 2-9

The drive is pre-configured with static IP addresses ranging from 192.168.0.101 (Position 2) to 192.168.0.108 (Position 9).

3.3.3.3 AKD PDMM Web Server

The AKD PDMM contains a web server that allows you to perform the following operations:

- Read information about the AKD PDMM (model type, firmware version, version of your KAS application)
- Diagnostic your system (CPU speed and usage, total and free storage space, list the EtherCAT devices)
- Configure some parameters (change the IP address, upgrade the firmware)
- Interact with your application (Start and Stop your KAS application , see the logs)



3.3.4 Real-Time Control

Windows alone is not enough

Applications that need sub-millisecond response times, predictable execution of control processes, require extremely accurate time control based on a constant time sampling. Windows is not deterministic and has not been designed to fulfill the needs of real-time control.

Then to impose accurate, time critical processing requirements, a hard real-time operating system is required in order to enable Windows environment to control tasks. INtime is the only RTOS designed to run side-by-side to Windows.

KAS real-time computation

The real-time kernel being part of KAS contains inter-process communication and synchronization mechanisms to guarantee a real-time control of your automation system.

Real-time computations can be said to have failed if they are not completed before their deadline, where their deadline is relative to an event. A real-time deadline must be met, regardless of system load.

When it is implemented with a AKD PDMM, the real-time kernel is based on **QNX**

Whereas the kernel for the AKD drive is based on VDK.

NOTE For the KAS Simulator, KAS relies on Windows capabilities.

3.3.5 Communication and Fieldbus

3.3.5.1 Fieldbus

Fieldbus allows a machine to be connected to other machines in an automation systems network. Typically, such a connection is referred to as a “factory automation” network connection.

3.3.5.2 Motion bus

Motion requires the controller to frequently update the drive with new trajectory setpoints. The bus involved in the motion control requires to be able to handle rigid jitter and timing demands including high data throughput and low latency.

Ethernet

Ethernet is certainly the most popular communications bus today because it is used in most computer networks. Motion control devices using Ethernet allow high-speed connections to computers without requiring special hardware. This reduces the cost and time required to make high-speed connections.

EtherCAT

The EtherCAT technology overcomes the system limitations of other Ethernet solutions. The Ethernet packet is no longer received, then interpreted and copied as process data at every connection. Instead, the Ethernet frame is processed on the fly. Each slave node reads the data addressed to it, while the telegram is forwarded to the next device. Similarly, input data is inserted while the telegram passes through. The telegrams are only delayed by a few nanoseconds.

On the master side, very inexpensive, commercially available standard network interface controller (NIC) or any on board Ethernet controller can be used as hardware interface.

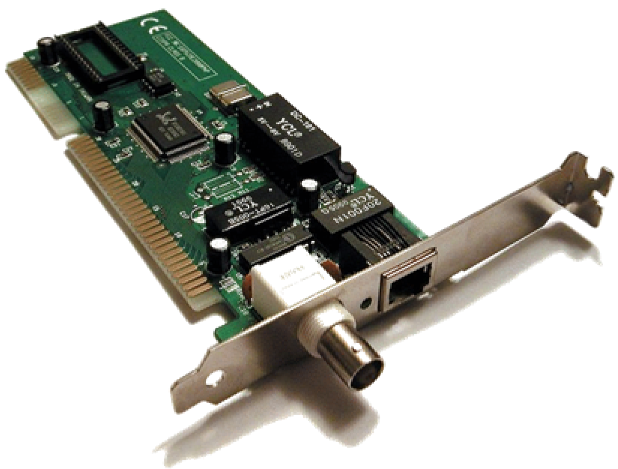


Figure 2-14: Network Interface Controller

3.3.5.3 Motion Bus Driver

A motion bus driver is a software component responsible for managing the communication link between the PAC, if any is present (see paragraph "Different

Implementations" on page 69), and the drives. On most systems this communication link is implemented via a physical wire coupled to a communication protocol.

3.3.5.4 PCI Interface Card

Plugged to a computer motherboard, this card allows attaching peripheral devices via a specific bus (for example, if your PAC does not have built-in connection for Profibus fieldbus, you can insert a specific PCI card)

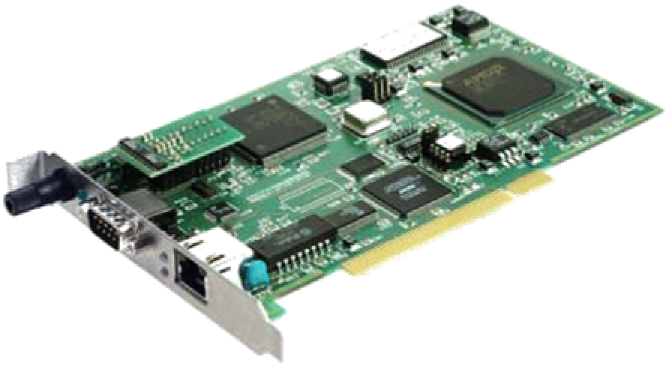


Figure 2-15: PCI Interface Card

3.3.6 Machine for Input/Output System

Input/Output refers to the communication and acquisition of data between your automation system, and the outside world (possibly a human, or another information processing system).

Inputs are the signals or data received by the automation system, and outputs are the signals or data sent from it.

Automation systems built with KAS are interrupt-driven. Typical interrupt uses include the following: system timers, disks I/O, power-off signals, and exceptions handling.

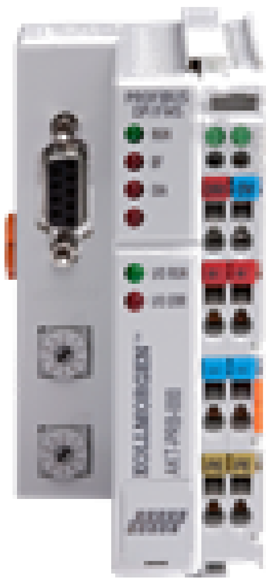


Figure 2-16: I/O Modules

I/O modules provide a convenient modular package which is simple to wire and add or change slice types.

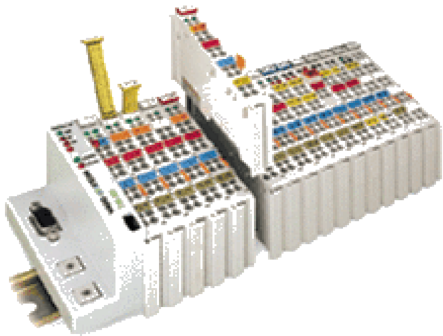


Figure 2-17: Standard I/O Couplers and Slices

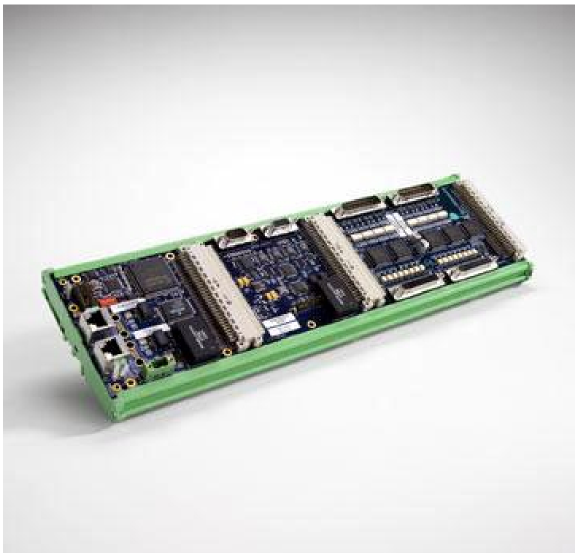


Figure 2-18: I/O Controllers

3.3.7 Drive

Figure 2-19: AKD

Figure 2-20: S300

Figure 2-21: S700

See also "Drives" on page 678 in **Hardware Devices** chapter for more details.

3.3.8 Motor

3.3.8.1 Kollmorgen Servomotors

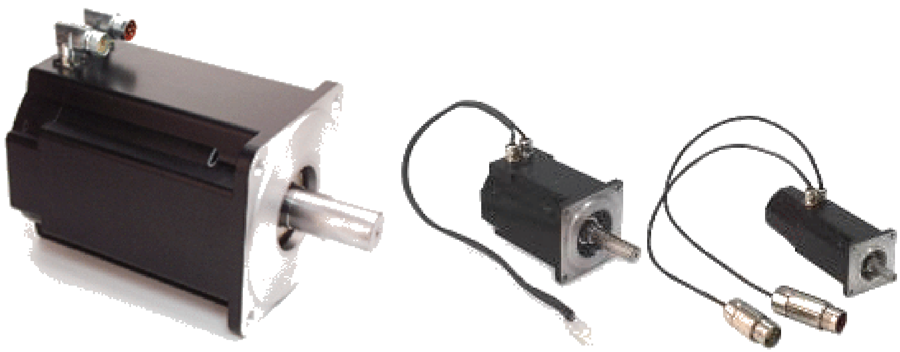


Figure 2-22: Kollmorgen AKM Servomotors

3.3.8.2 Cartridge Motor

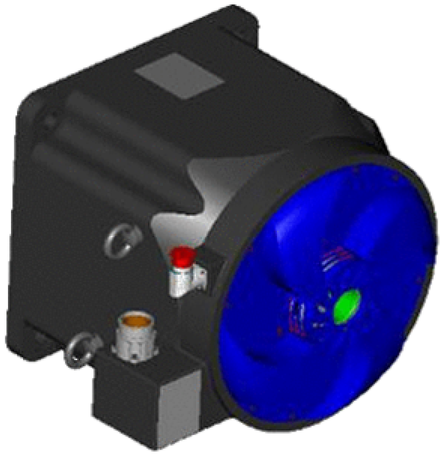


Figure 2-23: Cartridge Motor

3.3.8.3 Direct Drive Products

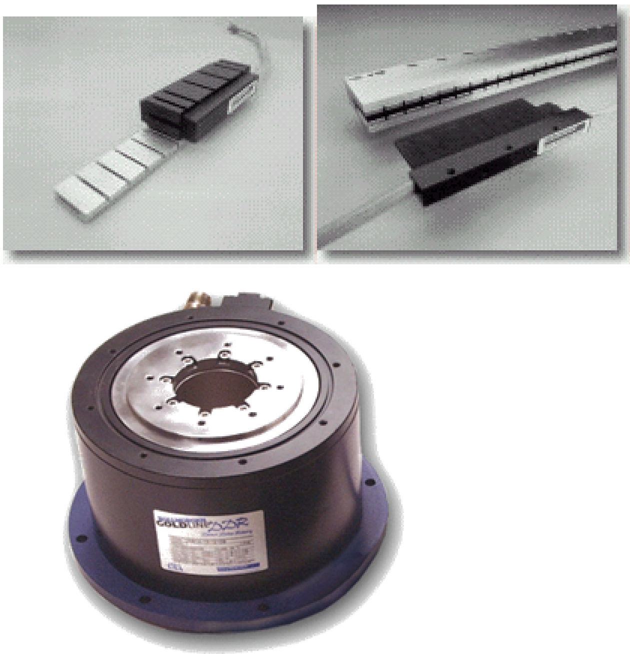


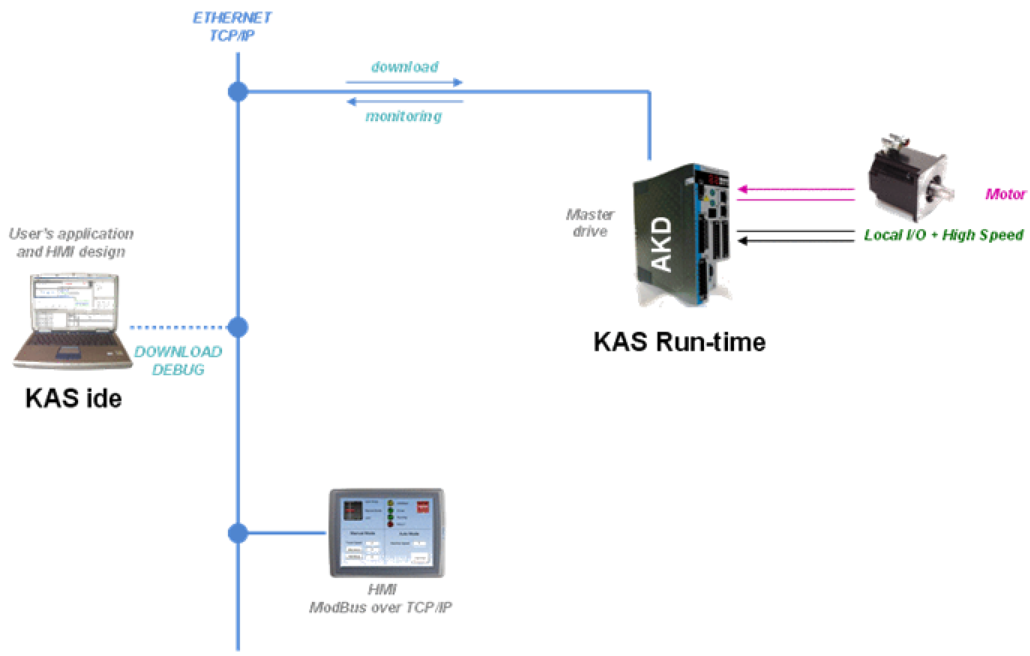
Figure 2-24: Direct Drives

3.4 Different Implementations

KAS supports the following architectures:

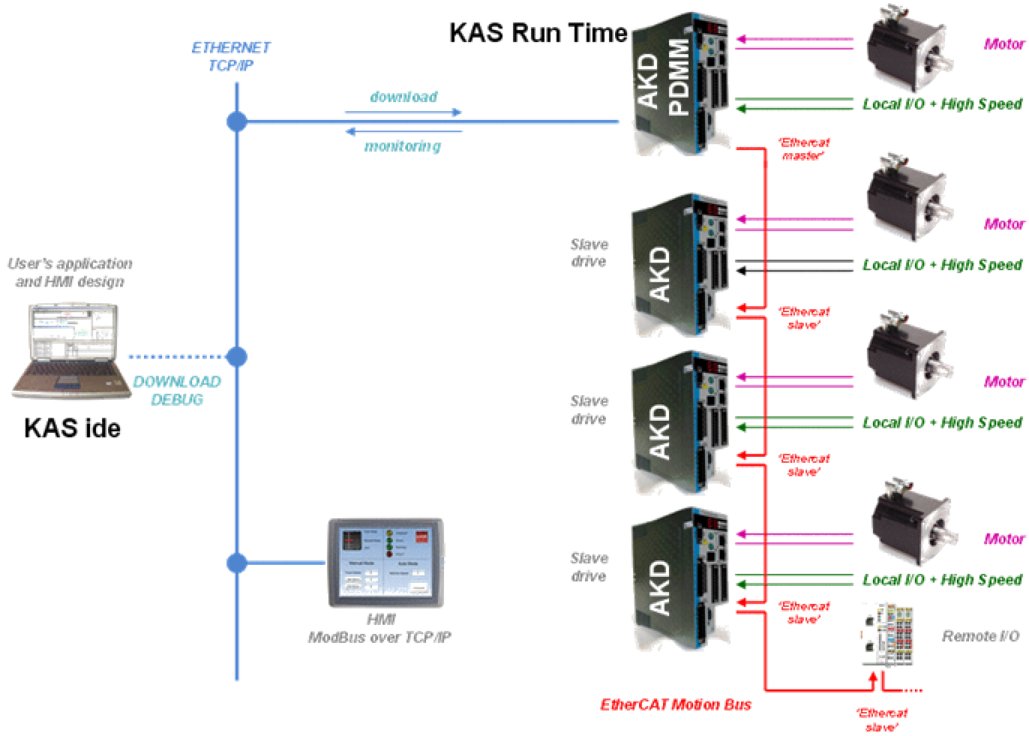
3.4.1 Single-Axis Managed by AKD Drive

The scalable system architecture begins with a base version of a 1.5 axis controlled by a programmable drive



3.4.2 Multi-Axis Managed by Drives

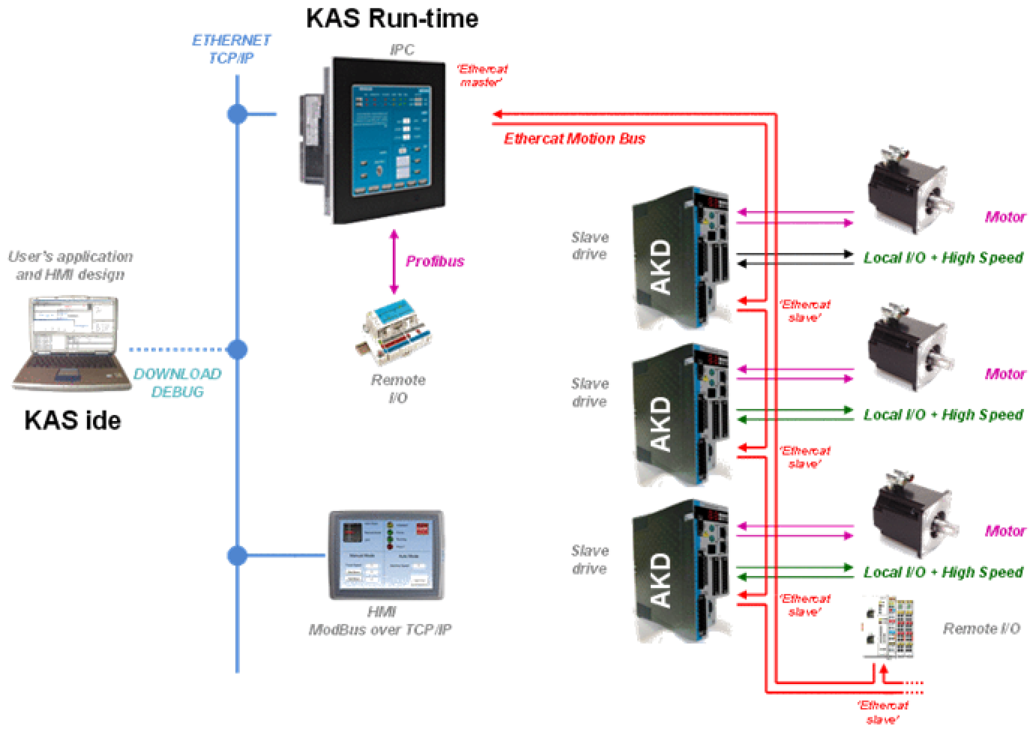
One programmable drive (AKD PDMM) acts as a master drive and sends basic commands to control all the other slave AKD drives. This configuration can manage up to 4 axes.



3.4.3 Multi-Axis Managed by PAC

A Programmable Automation Controller is controlling several drives (which can be programmable drives). This configuration can manage up to 250 axes.

NOTE Only one KAS IDE needs to be connected to the PAC.



This page intentionally left blank.

4 Understanding KAS

This chapter gives explanation about the most important **concepts** that you need to understand to use KAS.

IMPORTANT To take full advantage of KAS functions, a basic understanding of automation (programming languages and motion control) is required.

4.1 IEC 61131-3

4.1.1 Introduction

To create programs for the implementation of the PLC part of your application, the KAS IDE complies with IEC 61131-3. This standard currently defines five programming languages for programmable control systems.

The KAS IDE implements this standard to provide you with well-defined and well-known programming languages.

4.1.2 Data Types

Data types are defined within the common elements of IEC 61131-3.

Why Data typing?

Data typing is implemented to define the type of any parameter used, which helps to prevent errors early on in the programming phase. This avoids for example dividing a Date by an Integer.

When you have defined whether the data is a string, a date, an integer or a 16-bit Boolean input, there is no longer any confusion, nor any conflict between different people using the textual representation (i.e. the name of the variable).

Different kinds of Data types

Common data types are Boolean, Integer, Real, Byte, Word, Date, Time_of_Day, and String. Based on these, you can define your own personal data types, known as derived data types. In this way you can define an analog input channel as a data type, and re-use it.

List of Data types

Below are the available basic data types:

Types	Description	Values	Prefixes
BOOL	Boolean (bit)	FALSE or TRUE - stored in 1 byte	
SINT	Small signed integer in 8 bits	-128 to +127	SINT#
USINT	Small unsigned integer in 8 bits	0 to +255	USINT#
BYTE	*Same as USINT*		
INT	Signed integer in 16 bits	-32768 to +32767	INT#
UINT	Unsigned integer in 16 bits	0 to +65535	UINT#
WORD	*Same as UINT*		
DINT	Signed double precision integer in 32 bits	-2147483648 to +2147483647	
UDINT	Unsigned integer in 32 bits	0 to +4294967295	UDINT#

Types	Description	Values	Prefixes
DWORD	*Same as UDINT*		
LINT	Long signed integer in 64 bits		LINT#
ULINT	Long unsigned integer in 64 bits		ULINT#
LWORD	*Same as ULINT*		
REAL ‡	Single precision floating point stored in 32 bits	-3.4E38 to 3.4E38 and -3.4E-38 to 3.4E-38 (6 to 7 significant digits of accuracy)	
	IMPORTANT REAL is restrictive, but because it is the default, it is recommended to explicitly declare your real constants with the **LREAL#** prefix.		
LREAL ‡	Double precision floating point stored in 32 bits	-1.7E308 to 1.7E308 and -1.7E-308 to 1.7E-308 (14 to 15 significant digits of accuracy)	LREAL#
TIME	Time data type is used to specify a time variable - accuracy is 1ms. See "TIME" (see page 82) for more information.	0ms to 24h	T# or TIME#
STRING	Variable length string with declared maximum length Each character is store on 1 byte (i.e. on 8 bits)	Maximum length cannot exceed 255 characters	

‡ REAL variables are limited to 6-7 digits of accuracy and LREAL variables are limited to 14-15 digits of accuracy. Any digits after these significant digits will be lost, leading to loss of precision.

NOTE You can use **2#**,**8#** or **16#** prefixes to specify an integer in binary, octal or hexadecimal basis respectively.

4.1.2.1 Structures

A structure is a complex data type defined as a set of members. Members of a structure can have various data types. A member of a structure can have dimensions or can be an instance of another structure.

When a structure is defined, it can be used like other data types to declare variables.

Members of a structure can have an initial value. In that case, corresponding members of all declared variables having this structure type will be initialized with the initial value of the member.

To specify a member of a structured variable in PLC languages, use the following notation:

```
VariableName.MemberName
```

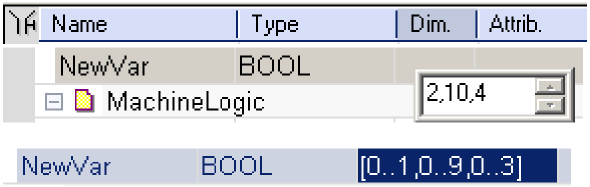
Limitation

If a member of a structure is an instance of another structure, the nested structure must be declared **BEFORE** in the list.

4.1.2.2 Arrays

You can declare arrays for internal variables by specifying dimension(s).

To declare an array, enter the number of elements in the **Dim.** column of the Dictionary (see procedure here).



For a multi-dimensional array (note that arrays have at most three dimensions), enter the number of elements for each dimension separated by commas (for example **2,10,4** is a 3 dimensional array, the first dimension has 2 elements, the second dimension has 10 elements, and the third dimension has 4 elements).

IMPORTANT All indexes are 0 based. For example, in the case of a single dimension array, the first element is always identified by *ArrayName***[0]**.

Use in ST (structured text) and IL (instruction list) languages

To specify an item of an array in ST and IL languages, enter the name of the array followed by the index(es) entered between the "**[**" and "**]**" characters. For multi-dimension arrays, enter indexes separated by comas. Indexes can be either constant or complex expressions. Below are some examples in ST language:

```
TheArray[1,7]  := value;
result  := SingleArray[i + 2];
```

Use in FBD and FFLD languages

In graphical languages, the following blocks are available for managing array elements:

[I]>>	get value of an item in a single dimension array
[I,J]>>	get value of an item in a two dimension array
[I,J,K]>>	get value of an item in a three dimension array
>>[I]	set value of an item in a single dimension array
>>[I,J]	set value of an item in a two dimension array
>>[I,J,K]	set value of an item in a three dimension array

For **get** blocks, the first input is the array and the output is the value of the item. Other inputs are indexes in the array.

For **put** blocks, the first input is the forced value and the second input is the array. Other inputs are indexes in the array.

Limitations

- Arrays have at most three dimensions.
- All indexes are 0 based.
- The total number of items in an array (merging all dimensions) cannot exceed 65535.

4.1.3 Variables

The scopes of the variables are normally limited to the organizational unit in which they are declared, e.g. local. This means that their names can be re-used in other parts without any conflict, eliminating another source of errors, e.g. the scratchpad. If the variables have global scope, they must be declared as such. Parameters can be assigned an initial value to have the right setting at start up and cold restart.

4.1.3.1 About Retain Variables

A retain variable is a PLC variable which:

- is non-volatile: stored persistently in the memory (called NVRAM) of the controller (PAC or Programmable Drive). When using KAS Simulator the retain variables are stored in a normal disk file.
- is known by all programs (when its content is changed, the change is propagated to all equations in which this variable is used)
- normally does not contain real-time critical data.

When an application is started, KAS initializes the retain variables with the value stored in the NVRAM only if the definition of the retain variables in the application and in NVRAM are the same. If the values do not match KAS will initialize the retain variables with their default values. This is known as a Cold Start.

Such a variable is used to store application specific data, like for instance to count a cutting-edge cycle in order to stop for its blade replacement after a specific number of iterations.

IMPORTANT The non-volatile memory size is hardware dependent. If the size of the retained variables is larger than the non-volatile storage space, an error will be logged and the data will not be stored in non-volatile memory. See "NVRAM" (see page 675) for more information.

For the KAS Runtime Simulator, the retained variables are saved in a file in your project repository.

4.1.3.2 Working with Variables

All variables used in programs must be first declared in the variable editor. Each variable belongs to a group and must be identified by a unique name within its group.

Groups

A group is a set of variables. A group either refers to a physical class of variables, or identifies the variables local to a program or user-defined function block. Below are the possible groups:

Groups	Description
GLOBAL	Internal variables known by all programs
RETAIN	Non volatile internal variables known by all programs
%I...	Channels of an input board - variables with same data type linked to a physical input device

Groups	Description
%Q...	Channels of an output board - variables with same data type linked to a physical output device
PROGRAMxxx	All internal variables local to a program (the name of the group is the name of the program)
UDFBxxx	All internal variables local to a User-Defined Function Block plus its IN and OUT parameters (the name of the group is the name of the program)

Data type and dimension

Each variable must have a valid data type. It can be either a basic data type or a function block. In the later case, the variable is an instance of the function block. Physical I/Os must have a basic data type. Instances of function blocks can refer either to a standard or "C" embedded block, or to a User Defined Function Block.

If the selected data type is STRING, you must specify a maximum length. This cannot exceed 255 characters.

Refer to the list of available data types for more information. Refer to the section describing function blocks for further information about how to use a function instance.

Additionally, you can specify dimension(s) for an internal variable, in order to declare an array.

Naming a variable

A variable must be identified by a unique name within its parent group. The variable name cannot be a reserved keyword of the programming languages and cannot have the same name as a standard or "C" function or function block. A variable must not have the same name as a program or a user-defined function block.

The name of a variable must begin by a letter or an underscore ("_") mark, followed by letters, digits or underscore marks. It is not allowed to put two consecutive underscores within a variable name. Naming is case-insensitive. Two names with different cases are considered as the same.

Naming Physical I/Os

Each I/O channel has a predefined symbol that reflects its physical location. This symbol begins with "**%I**" for an input and "**%Q**" for an output, followed by a letter identifying the physical size of the data. Then comes the location of the board, expressed on one or two numbers, and finally the 0-based index of the channel within the board. All numbers are separated by dots. Below are the possible prefixes for IO symbols:

%IX	1 byte input - BOOL or SINT
%QX	1 byte output - BOOL or SINT
%IW	2 bytes input - INT
%QW	2 bytes output - INT
%ID	4 bytes input - DINT or REAL
%QD	4 bytes input - DINT or REAL
%IL	8 bytes input - LINT or LEAL
%QL	8 bytes output - LINT or LEAL
%IS	STRING input
%QS	STRING output

In addition, you can give an alias (a readable name) to each I/O channel. In that case, either the "%" name or the alias can be used in programs. The alias must adhere to the same rules as a variable name.

Attributes of a variable

Physical I/Os are marked as either "**Input**" or "**Output**". Each internal variable can be configured as Read/Write or **Read Only**. Read Only variables can be mapped to Outputs, but not to Inputs. This is because Inputs can change state and a Read Only variable would not be able to change its value to match the input state.

Parameters of User-Defined Function Blocks are marked as either **IN**or **OUT**.

4.1.3.3 Retain Variables

What is a retain variable?

A retain variable is a PLC variable which:

- is non-volatile: stored persistently in the memory (called NVRAM) of the controller (PAC or Programmable Drive). When using KAS Simulator the retain variables are stored in a normal disk file.
- is known by all programs (when its content is changed, the change is propagated to all equations in which this variable is used)
- normally does not contain real-time critical data.

When an application is started, KAS initializes the retain variables with the value stored in the NVRAM only if the definition of the retain variables in the application and in NVRAM are the same. If the values do not match KAS will initialize the retain variables with their default values. This is known as a Cold Start.

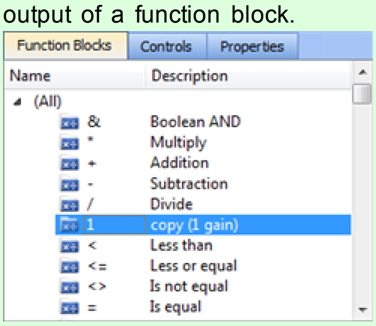
Such a variable is used to store application specific data, like for instance to count a cutting-edge cycle in order to stop for its blade replacement after a specific number of iterations.

IMPORTANT The non-volatile memory size is hardware dependent. If the size of the retained variables is larger than the non-volatile storage space, an error will be logged and the data will not be stored in non-volatile memory. See "NVRAM" (see page 675) for more information.

For the KAS Runtime Simulator, the retained variables are saved in a file in your project repository.

TIP *Retained variables should not be used as the output of Function Blocks*. Doing so may cause problems with retaining the value if the FB is executed with the Enable input off. The output structure can cause it to be conditionally updated to zero or the old value.

A work-around solution is to use a `copy (1 gain)` function to selectively enable updating the retain variable from an intermediate variable, which is the output of a function block.

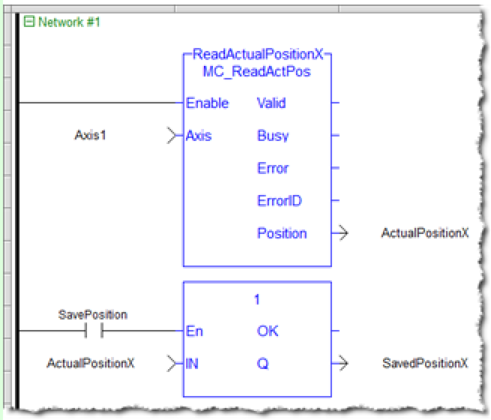


Work-around example

In the following image `ReadActualPositionX` is continually reading position

TIP

feedback from an axis, and the copy block is saving a specific value to a retain variable. `ActualPositionX` is the intermediate variable and `SavedPositionX` is the retain variable.



4.1.4 Constant Expressions

Constant expressions can be used in all languages for assigning a variable with a value. All constant expressions have a well-defined data type according to their semantics. If you program an operation between variables and constant expressions having inconsistent data types, it leads to syntactic errors when the program is compiled.

Below is the list of prefixes according to possible data types:

Type	Prefix	Description
BOOL		There are only two possible boolean constant expressions. They are reserved keywords **TRUE** and **FALSE**.
SINT	SINT#	Small integer constant expressions are valid integer values (between -128 and 127). All integer expressions having no prefix are considered as DINT integers
USINT/BYTE	USINT#	Unsigned small integer constant expressions are valid integer values (between 0 and 255). All integer expressions having no prefix are considered as DINT integers.
INT	INT#	16-bit integer constant expressions are valid integer values (between -32768 and 32767). All integer expressions having no prefix are considered as DINT integers.
UINT/WORD	UINT#	Unsigned 16-bit integer constant expressions are valid integer values (between 0 and +65535). All integer expressions having no prefix are considered as DINT integers.

Type	Prefix	Description
DINT		32-bit integer constant expressions must be valid numbers between -2147483648 to +2147483647. DINT is the default size for integers: such constant expressions do not require a prefix. **NOTE** You can use **2#**,**8#** or **16#** prefixes to specify an integer in binary, octal or hexadecimal basis respectively.
UDINT/DWORD	UDINT#	Unsigned 32-bit integer constant expressions are valid integer values (between 0 and 4294967295). All integer expressions having no prefix are considered as DINT integers.
LINT	LINT#	Long integer (64-bit) constant expressions are valid integer values. All integer expressions having no prefix are considered as DINT integers.
ULINT/LWORD	ULINT#	Unsigned 64-bit integer constant expressions are valid integer values. All integer expressions having no prefix are considered as DINT integers.
REAL		Real constant expressions must be valid numbers, and must include a dot ("**.**"). If you need to enter a real expression having an integer value, add "**.0**" at the end of the number. You can use "**F**" or "**E**" separators for specifying the exponent in case of a scientific representation. REAL is the default precision for floating points: such expressions do not require a prefix. **IMPORTANT** REAL is restrictive, but because it is the default, it is recommended to explicitly declare your real constants with the **LREAL#** prefix. **NOTE** REAL constants are limited to 6-7 digits of accuracy. Any digits after these significant digits will be lost, leading to a loss of precision.

Type	Prefix	Description
LREAL	LREAL#	Real constant expressions must be valid numbers, must include a dot ("**.**"). If you need to enter a real expression having an integer value, add "**.0**" at the end of the number. You can use "**F**" or "**E**" separators for specifying the exponent in case of a scientific representation. **NOTE** LREAL constants are limited to 14-15 digits of accuracy. Any digits after these significant digits will be lost, leading to a loss of precision.
TIME	T# or TIME#	Time-constant expressions represent durations that must be less than 24 hours. They are expressed as a number of hours followed by "**h**", a number of minutes followed by "**m**", a number of seconds followed by "**s**", and a number of milliseconds followed by "**ms**". The order of units (hour, minutes, seconds, milliseconds) must be respected. You cannot insert blank characters in the time expression. There must be at least one valid unit letter in the expression.
STRING		String expressions must be written between single quote marks. The length of the string cannot exceed 255 characters. You can use the following sequences to represent a special or not-printable character within a string: **$$** a "$" character **$'** a single quote **$T** a tab stop (ASCII code 9) **$R** a carriage return character (ASCII code 13) **$L** a line feed character (ASCII code 10) **$N** carriage return plus line feed characters (ASCII codes 13 and 10) **$P** a page break character (ASCII code 12) **$xx** any character (xx is the ASCII code expressed on two hexadecimal digits

Table 3-1: List of Prefixes for Constant expressions

4.1.4.1 Examples

Below are some examples of valid constant expressions:

TRUE	TRUE boolean expression
FALSE	FALSE boolean expression
SINT#127	small integer
INT#2000	16 bit integer
123456	DINT (32 bit) integer
16#abcd	DINT integer in hexadecimal basis
8#34712	DINT integer in octal basis
2#1000100	DINT integer in binary basis
LINT#1	long (64 bit) integer having the value "1"
0.0	0 expressed as a REAL number
1.002E3	1002 expressed as a REAL number in scientist format
LREAL#1E-200	Double precision real number
T#23h59m59s999ms	maximum TIME value
TIME#0s	null TIME value
T#1h123ms	TIME value with some units missing
'hello'	character string
'name$Tage'	character string with two words separated by a tab
'I$'m here'	character string with a quote inside (I'm here)
'x$00y'	character string with two characters separated by a null character (ASCII code 0)

Below are some examples of typical errors in constant expressions

BooVar := 1;	0 and 1 cannot be used for booleans
1a2b	basis prefix ("16#") omitted
1E-200	"LREAL#" prefix omitted for a double precision float
T#12	Time unit missing
'I'm here'	quote within a string with "$" mark omitted
hello	quotes omitted around a character string

Additionally, there are pre-defined constants. See "Step 8 of 15 - Use the Defines List" (see page 262) for information about Internal and user-defined Defines.

4.1.5 Program Organization Units

Within IEC 61131-3, the "Functions" (see page 84), "Function Blocks" (see page 84), and "Programs" (see page 86) are called Program Organization Units (POU).

In addition to the IEC standard, you can write you own code: sub-program or UDFB.

Types	IEC 61131-3	Written by end-user
Basic functions (has no memory)	"Functions" (see page 84)	"Programs" (see page 86) / "Sub-programs" (see page 86)
Instantiated functions (keep track of the past)	"Function Blocks" (see page 84) (FB)	"User-Defined Function Blocks" (see page 87) (UDFB)

4.1.5.1 Difference between Functions and Function Blocks

- Functions are expected to complete in one cycle
- Function Blocks can take several cycles to complete

Description of FB operation

Rather than halt the application, waiting for operations to complete, the FB typically gives control back to the application but does not set its **Done** output.

Examples of Operations Overrunning the Cycle Duration

- A motion command to move from one location to another can take several cycles to complete.
- Same for operations like reading/writing to files or reading and writing over TCP/IP can also take several cycles to complete.

Operation Sequence

1. When a FB is called, it starts an operation and possibly does not complete it
2. The FB is called in the next cycle, and it checks to determine if the operation is done
3. If it is done, it sets the **Done** output. If not, it continues on
4. Now the application knows that the operation is complete and can do what ever other processing it needs based on the FB being done

4.1.5.2 Functions

IEC has defined standard functions and also allows you to create your own functions (called user-defined functions). Typically, functions take several inputs and return a single output as the result of processing.

- Standard functions are for example ADD (addition), ABS (absolute), SIN (sine), COS (cosine), GT (Greater Than),….
- User-defined functions, as in the following example, can be used repeatedly once defined.

```
 FUNCTION SIMPLE_FUN : REAL
        VAR_INPUT
                A, B : REAL;
                C : REAL := 1.0;
        END_VAR
        SIMPLE_FUN := A*B/C;
END FUNCTION
```

4.1.5.3 Function Blocks

Function Blocks (FBs) take several inputs and return a group of values as the output as the result of processing.

Function Blocks are the equivalent to Integrated Circuits (IC), representing a specialized control function. They are specified at such a level that you quickly recognize the functionality of the function block and specifically what happens if it is activated or connected to other blocks in a sequence of motion commands.

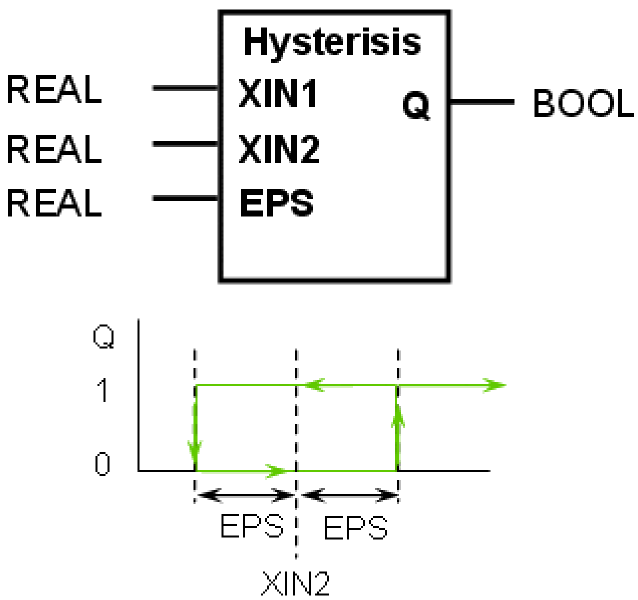
They contain data as well as an algorithm, so they can keep track of the past (which is one of the differences from Functions). They have a well-defined interface and hidden internals, like an IC or a black box. The user only sees the interface, being the inputs and outputs. The code itself is hidden.

Function Blocks can be used in any of the IEC languages. Note that in an SFC program, function blocks can be part of a step or transition created in FFLD, ST, IL and FBD.

Once defined, they can be used repeatedly, in the same program, different programs, or even different projects. This makes them highly re-usable.

There are predefined function blocks (e.g. timers, counters or triggers) and also additional function blocks that can come from libraries produced by you or other suppliers (e.g. a temperature control-loop or PID).

Example of function blocks



The function block is based on the programming language function block Diagram and has the name Hysterisis. It has three inputs (XIN1, XIN2 and EPS) of datatype REAL on the left, and one output (called Q) of type BOOL on the right-hand side.

TIP	Input names are not very usable. Please use meaningful names.

Internally, the FB contains the following body code:

```
 FUNCTION_BLOCK HYSTERISIS
VAR_INPUT
        XIN1, XIN2 : REAL;
        EPS : REAL;  (* Hysterisis ban *)
END_VAR
VAR_OUTPUT
        Q : BOOK := 0
END_VAR
```

```
IF Q THEN
        IF XIN1 < ( XIN2 - EPS ) THEN
                Q := 0 (* XIN2 decreasing *)
        END_IF;
        ELSIF XIN1 > ( XIN2 + EPS ) THEN
                Q := 1; (* XIN2 increasing *)
        END_IF;
END_FUNCTION_BLOCK
```

In this example, the body code is written in the Structured Text language:

- The first part deals with the data structure
- The second with the algorithm
- No additional data is used.

Whatever name was used for this local data inside the body, it does not conflict with matching names in other functions, function blocks, or with global expressions. This example of data encapsulation removes a major source of errors.

4.1.5.4 Programs

With the above-mentioned basic building blocks, a program can be seen as a network of functions and function blocks. Each of them being written in any of the defined programming languages.

Sub-programs

The list of programs is completed with "Sub-programs". Sub-programs are written in FBD, FFLD, ST or IL languages, and can be called by the programs of the application. Input and output parameters plus local variables of a sub-program are declared in the variable editor as local variables of the sub-program.

A sub-program can call another sub-program or a UDFB.

Unlike UDFB, local variables of a sub-program are not instantiated. This means that the sub-program always works on the same set of local variables. Local variables of a sub-program keep their value among various calls. The code of a sub-program is not duplicated when called several times by parent programs.

A sub-program cannot have more than 32 input parameters or 32 output parameters.

A good programming practice is to break up your programs into smaller modules.

See also paragraph "Application Software Structure - Definitions" on page 560.

Program Guidelines

An application is a list of programs. Programs are executed sequentially within the target cycle, according to the following model:

```
 Begin cycle
| exchange I/Os
| execute first program
| ...
| execute last program
| wait for cycle time to be elapsed
End Cycle
```

Programs are executed according to the order defined by the user. All SFC programs must be grouped (it is not possible to insert a program in FBD, FFLD, ST or IL between two SFC programs). The number of programs in an application is limited to 32767. Each program is entered using a language chosen when the program is created. Possible languages are Sequential Function Chart (SFC), Function Block

Diagram (FBD), Free Form Ladder Diagram (FFLD), Structured Text (ST) or Instruction List (IL).

Programs must have unique names. The name cannot be a reserved keyword of the programming languages and cannot have the same name as a standard or "C" Function or function block. A program must not have the same name as a declared variable. The name of a program must begin by a letter or an underscore ("_") mark, followed by letters, digits or underscore marks. It is not allowed to put two consecutive underscores within a name. Naming is case-insensitive. Two names with different cases are considered as the same.

Child SFC Programs

You can define a hierarchy of SFC programs, entered as a tree in the list of programs. A child program is controlled within action blocks of the parent SFC program.

TIP Even if you do not want to split your FFLD program, at least separate FFLD from SFC. Simply make a sub-program in FFLD called from the SFC step, and keep only the state machine in the SFC program. This makes everything simpler and more comfortable for editing and debugging.

Program Limitations

When creating your application you have to consider the following important limitations.

For **SFC** programs:

- Actions in SFC steps cannot be more than 32kB
- Condition in SFC transition cannot exceed 32kB
- Total P-code size of the program cannot exceed 64kB

For **FFLD** programs:

- Width of any network is limited to 255 columns
- Height of any network is limited to 255 rows

For any program, sub-program or UDFB written in **other** languages:

- Jump limit is 64kB
  For example, in a Free Form Ladder program, if you create a UDFB or program which is over 64kB and then decide to add a jump to label in the first network to the last network, this jump reaches the limit.
- Total P-code size of the program, sub-program or UDFB cannot exceed 64kB

4.1.5.5 User-Defined Function Blocks

The list of programs is completed with "User-Defined Function Blocks" (UDFBs). UDFBs are described using SFC, FBD, FFLD, ST or IL languages, and can be used as other function blocks in the programs of the application. Input and output parameters plus private variables of a UDFB are declared in the variable editor as local variables of the UDFB.

There is no restriction using any operation in a UDFB. A UDFB can call standard functions and function blocks.

A UDFB can call another UDFB. Note that the called UDFB must be declared before the calling one in the program list.

Each time a UDFB is instantiated, its private variables are duplicated for the declared instance. The code of the UDFB is duplicated on each call in parent programs. This leads to higher performances at run-time, but consumes code space. It is recommended to package small algorithms in UDFBs. Large parts of code must be managed in programs.

4.1.6 Programming Languages

Within the IEC 11631 standard, syntax and semantics of the programming languages have been defined, leaving no room for variance. Once you have learned them, you can use a wide variety of systems based on this standard.

The languages consist of two textual and three graphical versions:

Textual:

- Instruction List (IL)
- Structured Text (ST)

Graphical

- Sequential Function Chart (SFC)
- Free Form Ladder Diagram (FFLD)
- Function Block Diagram (FBD)

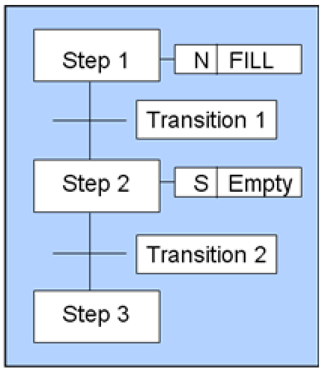
All five languages are interlinked: they provide a common suite.

The choice of programming language depends on:

- the programmer's background
- the problem at hand
- the level of describing the problem
- the structure of the control system
- the interface to other people / departments

4.1.6.1 Sequential Function Chart (SFC)

SFC describes graphically the sequential behavior of a control program. It is derived from Petri Nets.



SFC organizes the internal structure of a program, and helps to deconstruct a control problem into manageable parts, while maintaining the overview.

SFC consists of steps, linked with Action Blocks and Transitions. Each step represents a particular state of the systems being controlled. A transition is associated with a condition, which, when true, causes the step before the transition to be deactivated, and the next step to be activated. Steps are linked to action blocks, performing a specific control action. Each element can be programmed in any of the IEC languages, including SFC itself.

Alternative and Parallel Sequences

You can use alternative sequences and even parallel sequences, like those commonly required in batch applications. For example, one sequence is used for the primary process, and the second for monitoring the overall operating constraints.

As shown in the following picture, parallel sequences are also possible:

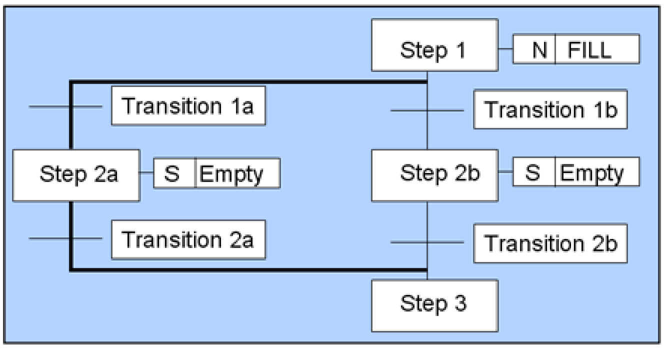


Figure 3-1: Example of a Parallel Sequence in SFC

From step 1, it either goes to step 2a or step 2b, depending on which of the transition conditions is met. Both conditions need to exclude each other.

4.1.6.2 Structured Text (ST)

ST is a very powerful high-level language with its roots in ADA, Pascal and “C”. It contains all the essential elements of a modern programming language, including selection branches (IF-THEN-ELSE and CASE OF) and iteration loops (FOR, WHILE and REPEAT). These elements can also be nested. It can be used for the definition of complex function blocks, which can be used within any of the other languages.

4.1.6.3 Function Block Diagram (FBD)

FBD is very common to the process industry. It expresses the behavior of functions, function blocks and programs as a set of interconnected graphical blocks, as in electronic circuit diagrams. It looks at a system in terms of the flow of signals between processing elements.

4.1.6.4 Free Form Ladder Diagram (FFLD)

FFLD is based on the graphical presentation of Relay Ladder Logic.

4.1.6.5 Instruction List (IL)

IL is the European counterpart of FFLD. As textual language, it looks like Assembler.

4.1.7 Alias Definitions

The compiler supports the definition of aliases (see usage in paragraph "Step 8 of 15 - Use the Defines List" on page 262).
An alias is a unique identifier that can be used in programs to replace another text. Definitions are typically used to replace a constant expression and facilitate the maintenance of programs.

There are three levels of definitions:

- Common to all the projects present on your machine
- Global to all programs within your project
- Local to one program

Common and global definitions can be edited from the "File / Open" menu of the main window. Local definitions are edited together with the corresponding program. Use the "View / Local Defines" menu command when editing a program to open its local definitions.

Definitions are entered in a text editor. Each definition must be entered on one line of text according to the following syntax:

```
#define Identifier Equivalence (* comments *)
```

Below are some examples:

```
#define OFF     FALSE              (* redefinition of FALSE con-
stant *)
#define PI      3.14               (* numerical constant *)
#define ALARM   (bLevel > 100)     (* complex expression *)
```

You can use a definition within the contents of another definition. The definition used in the other one must be declared first. Below is an example:

```
#define PI      3.14
#define TWOPI   (PI * 2.0)
```

Note that a definition can be empty, for example:

```
#define CONDITION
```

The defined word can be used for directing the conditional compiling directives.

TIP You can enter `#define` lines directly in the source code of programs in IL or ST languages.

The use of definitions can disturb the program monitoring and make error reports more complex. It is recommended to restrict the use of definitions to simple expressions that do not risk creating a misunderstanding when reading or debugging a program.

4.1.8 Handling Exceptions

The compiler enables you to write your own exception programs for handling particular system events. The following exceptions can be handled:

- Startup (before the first cycle)
- Shutdown (after the last cycle)
- Division by zero

Startup

You can write your own exception program to be executed before the first application cycle is executed:

1. Create a new main program that will handle the exception. It cannot be an SFC program.
2. In the editor of global defines, insert the following line:

```
#OnStartup ProgramName
```

NOTE **Warning:** The program is executed before all other programs within the fisrt cycle. This implies that the cycle timing can be longer during the first cycle. You cannot put breakpoints in the Startup program.

Shutdown

You can write your own exception program to be executed after the last application cycle when the runtime system is cleanly stopped:

1. Create a new main program that will handle the exception. It cannot be an SFC program.
2. In the editor of global defines, insert the following line:

```
#OnShutdown ProgramName
```

NOTE **Warning:** You cannot put breakpoints in the Shutdown program.

Division by zero

You can write your own exception program for handling the "Division by zero" exception. Below is the procedure you must follow for setting an exception handler:

1. Create a new sub-program without any parameter that will handle the exception
2. In the editor of global defines, insert the following line:

```
#OnDivZero SubProgramName
```

In the sub-program that handles the exception you can perform any safety or trace operation. You then have the selection between the following possibilities:

- Return without any special call. In that case the standard handling will be performed: a system error message is generated, the result of the division is replaced by a maximum value and the application continues.
- Call the FatalStop function. The runtime then stops immediately in Fatal Error mode.
- Call the CycleStop function. The runtime finishes the current program and then turns in "cycle setting" mode.

Handlers can also be used in DEBUG mode for tracking the bad operation. Just put a breakpoint in your handler. When stopped, the call stack will show you the location of the division in the source code of the program.

4.2 Motion Concepts

4.2.1 Introducing Motion

4.2.1.1 Motion Control Main Functions

To ensure accurate positioning and movement, motion control consists of the two following main parts:

- Setpoint generation
- Regulation

Setpoint generation

This consists of generating a trajectory defined by **position versus time**. It is purely logical and does not relate to the physical world.

Regulation

Even using the very best drives, you cannot maintain accurate positioning without a feedback loop. The regulation consists of following the generated position settings using classical feed-forward or feedback control-loops (by means of PID). Regulation is the part which takes care of the physical world of making moving motors.

These two functions can be located on the same hardware (as in a "stand-alone" servo drive) or on two separate hardware devices, linked together by a fieldbus.

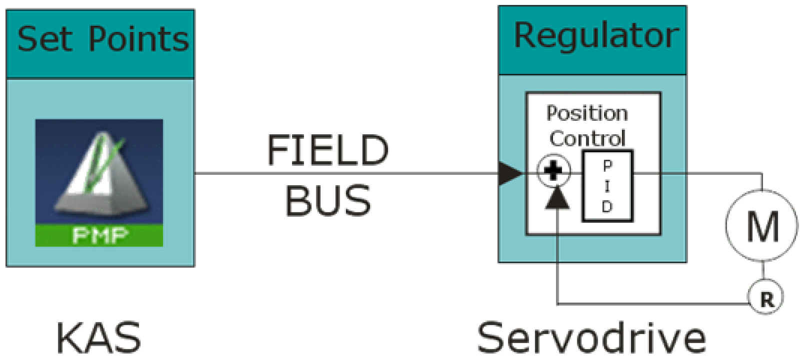


Figure 3-2: Regulation with Remote Drive

4.2.1.2

4.2.1.3 Single and Multi-Axis Motion

In **Single-Axis**, as shown in the figures above, one setpoint generator is linked to one axis.

Multi-Axis motion consists of synchronizing several axes linked to a common motion source. This source can be external, like a physical motor (called master) or an internal profile generator (called virtual master) as shown in "Figure 3-3: Multi-Axis Driven by a Virtual Master " on page 92 below.

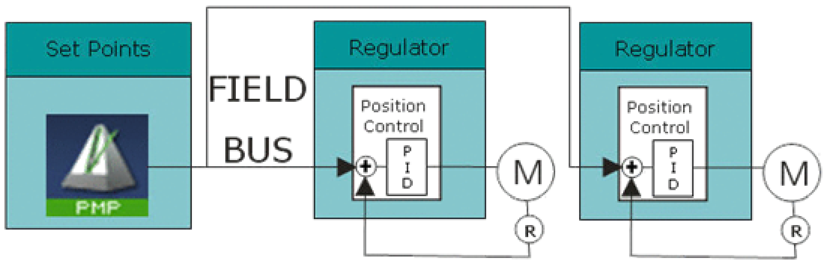


Figure 3-3: Multi-Axis Driven by a Virtual Master

4.2.1.4

4.2.1.5 Hardware Organization of Motion Functions

A complete motion control "chain" is made of two main parts that can be subdivided into several more basic functionalities. Depending on your hardware system configuration, each of these elementary functions can theoretically be embedded in different hardware modules.

One of the possible configurations is represented in the figure below.

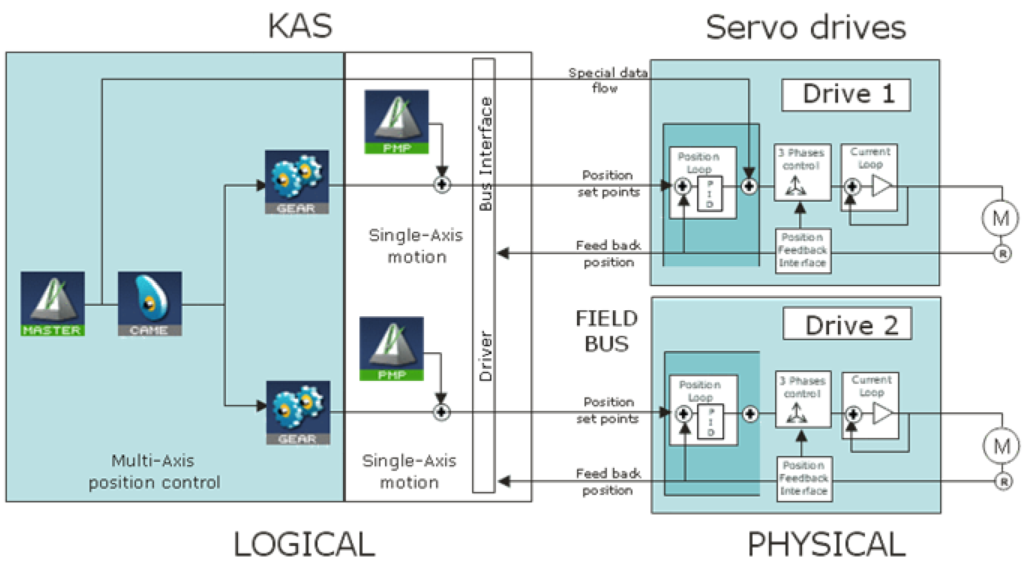


Figure 3-4: Hardware Organization of Motion Functions

The scope of Kollmorgen Automation Suite is to manage all the logical parts of the motion control and to ignore the physical aspects (which are handled by the hardware). To make the link between the logical and physical worlds, KAS includes some components that acts as interface.

Therefore, we do no longer consider regulation and the physical world in the following paragraphs. Only setpoint generation are taken into account.

4.2.1.6

4.2.1.7 Motion Profile

In motion control, a common need is to move a system from one steady position to another (point-to-point motion). Following the fastest possible motion within an allowed maximum value for speed, acceleration, and jerk, results in a third-order motion profile as illustrated below:

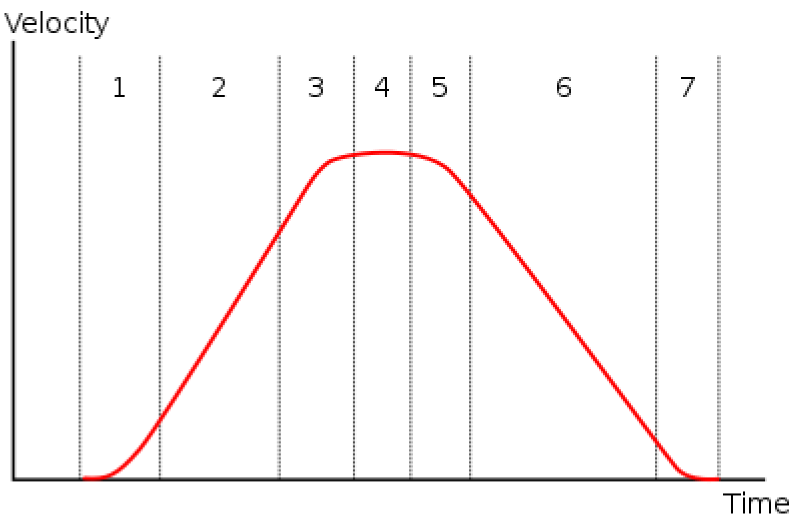


Figure 3-5: Third-order motion profile

The motion profile consists of up to seven phases defined by the following:

- acceleration increase, with maximum positive jerk
- constant maximum acceleration (zero jerk)
- acceleration decrease, approaching the desired maximum velocity, with maximum negative jerk
- constant maximum speed (zero jerk, zero acceleration)
- deceleration increase, approaching the desired deceleration, with maximum negative jerk
- constant maximum deceleration (zero jerk)
- deceleration decrease, approaching the desired position at zero velocity, with maximum positive jerk

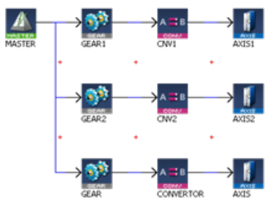
If the initial and final positions are sufficiently close together, the maximum acceleration or maximum velocity may never be reached.

4.2.2 Pipe Network or PLCopen

Using KAS there are two ways to generate motion functions and motion profiles: with Pipe Network or PLCopen.

Pipe Network

The Pipe Network enables you to create a high-performance motion algorithm which is tightly integrated to the PLC program with motion library function blocks.



For high performance, complex, or synchronized multi-axis applications, the pipe concept in KAS provides a simple conversion of mechanical applications into a graphical representation of application elements and the process flow. This format makes it easy to understand, program, and update the motion profiles and positional relationships.

The KAS application begins with the creation of a Pipe Network structure linking Master objects (source) to Axes objects (destination) and includes the definition of specific transformer motion profiles . This structure is then controlled from the PLC application using dedicated function blocks in the Motion Library.

To be able to use pipes correctly, it is necessary to first consider some definitions.

PLCopen (see PLCopen Web site)

Standard function blocks can be used and directly incorporated into the PLC application. Programming of motion is done using standard MC function blocks that can be incorporated in single-axis or multi-axis applications.

4.2.2.1 Motion Engine Differences

The following table outlines some of the main feature differences between the Pipe Network and the PLCopen motion engines. It also provides their associated function blocks.

Topic	Pipe Network	PLCopen
Function block format	Begins with **ML** ex: MLAxisRel	Begins with **MC_** ex: MC_MoveRelative
Does Function block requires instantiation?	No. Except for MLAxisStop	Most require it
Method to start execution	Most are level triggered	Most are edge triggered
Motion execution status, for function block executing motion	Use MLMotionStatus function block	Each function block includes a standard set of outputs for motion status
Function block standard input format	Requires additional function blocks to define motion parameters (speed, accel, decel, etc.)	Includes standard set of inputs to define motion (speed, accel, decel, etc.)
Axis setup method	Includes in the Pipe Network Axis block properties	Part of Axis definition screen in the Project tree
How the Axis name is setup?	Automatically done as part of Pipe Network Axis block properties	Create an instance of a Axis_ Ref variable structure in the dictionary, then assign an axis number to it in a PLC program (for procedure, see page 289)
Is there additional motion editor?	Yes (Pipe Network editor)	No
Motion buffering	Execution of multiple motion commands in a row is handled by the programmer	Function blocks have built in buffering modes
Motion jerk reduction	Primarily available by adding cams to the Pipe Network	Function blocks have jerk reduction input

Table 3-2: Differences between the Pipe Network and PLCopen

4.2.3 Pipe Network Concept

To introduce the Pipe Network concept, we can use a mechanical analogy.

In the figure below, the mechanical system is composed of three-axes and driven by one motor. All axes are connected to the motor through shafts, gears and cams. When the motor is in motion, all axes are moving synchronously. The speed relation between the Master and the Axis is achieved by using a mechanical Gear. A mechanical cam is used to get linear motion from a rotating wheel.

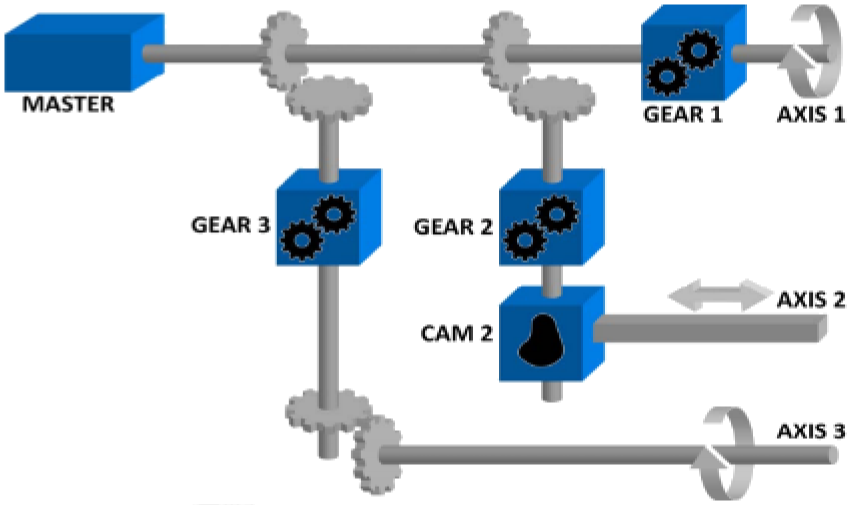


Figure 3-6: Mechanical System

The Pipe Network in the figure below corresponds to the mechanical system described above. The pipe concept is a one-to-one translation of a mechanical system into the logical world.

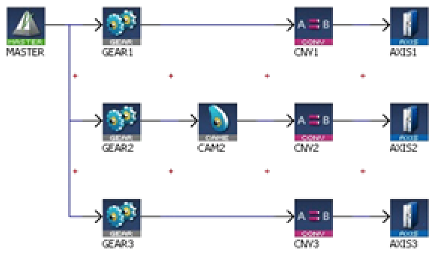


Figure 3-7: Pipe Network Structure

In our Pipe Network, the analogy is as follows:

- The main motor of the mechanical machine becomes a Virtual Master Pipe Block
- The gear boxes becomes Gear Blocks
- The mechanical cam becomes a Cam Block
- The axes becomes Axis Blocks

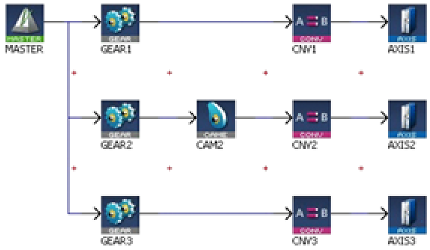
The Pipe Network concept allows motion engineers to define in a very natural way the physical relationships between the different axes of their machine.

This powerful modular approach provides a solution for almost any multi-axis requirements. It also remains open for new, additional functions that can be required in the future.

4.2.3.1 Pipe Network

To control the machine application with multi axes that are dynamically interconnected, you can design several pipes with the KAS IDE to create the global Pipe Network as shown in "Figure 3-7: Pipe Network Structure " on page 96.

Relationships between the Axes are developed and connected graphically, allowing you to visualize how the machine functions. Each horizontal flow is considered as a separate pipe. In the application below there are three pipes.



- The Pipe Network can be edited at any time.

> **NOTE** You do not have to finalize the Pipe Network before writing a PLC program, but you must compile your project to have the latest Pipe Network information available in the PLC program editor.

- Program code does not have to be written when setting up the foundation of a program, as the parameters are entered into set-up screens.

> **IMPORTANT** Pipe Network code is generated automatically by the compiler, you should not try to modify it.

- In the programs, you can define activation or deactivation statements to install or remove pipes and Pipe Blocks. This allows the dynamic adjustment of the machine behavior depending on the result.
- The Pipe Network is used for more than just coordinated motion. It contains a full library of single-axis motion commands for sections of an application where an axis operates independently.

4.2.3.2 Pipe

A pipe is a set of Pipe Blocks linked together (where position flows from one Pipe Block to the next). The general structure of a pipe is quite simple:

1. Start with an input Pipe Block (source)
2. Optionally followed by transformer Pipe Blocks
3. Followed by an output Pipe Block (convertor)
4. Finish with the destination Pipe Block

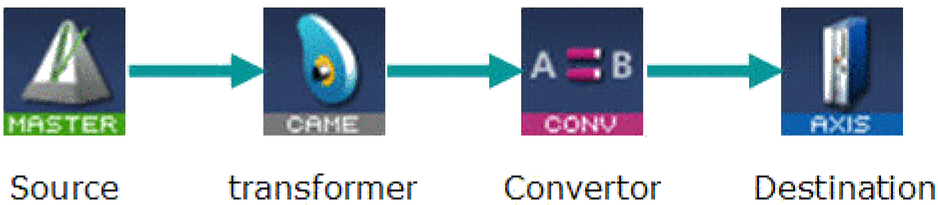


Figure 3-8: Typical Pipe Structure

> **TIP** To avoid jerk in the pipe network (which ultimately may cause a jerk in motor motion when a cam block is applied to the upstream pipe network positions) the potential position offset between the cam's first point and the input to the cam block must be taken care of in the application program by setting a cam offset or another method.

More about the different kinds of Pipe Blocks are discussed in paragraph "Step 12 of 15 - Adding Motion" on page 279.

4.2.3.3 Pipe Block

Pipes are built using logical entities called Pipe Blocks.

A Pipe Block is an object whose purpose is to modify a flow of values with strict time constraints. Pipe Blocks normally have both input and output flows of values.

Based on their functions, there are four kinds of Pipe Blocks:

Function	Description
Input (source)	Works as generator of values: • sample external source objects or create a discrete flow of values as an input to the pipe
Transformer	• apply a specific algorithm to the input value to produce their output (transformations can be linear or complex: e.g. cam) • can create events depending on the incoming values
Output (convertor)	Block that can end a pipe: • convert the incoming values from user units to correct system units for the destination objects
Destination	Simply models a physical axis of the machine

The following table provides a short description of each Pipe Block:

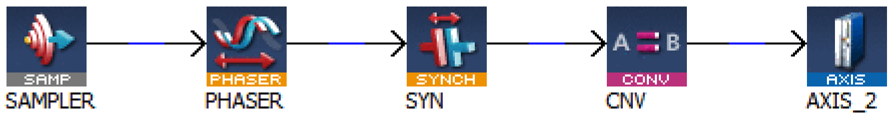
Function	Pipe Block	Description
Input	**Master**	Virtual master generating values (position) at each cycle
Input	**Sampler**	Samples external value (encoder, resolver, PLC variable etc.)
Transformation		
Mathematical	**Derivator**	Applies a derivation on the input data flow
Mathematical	**Integrator**	Integrates the input data flow
Mathematical	**Adder**	Adds two data flows
Event-driven	**Synchronizer**	Starts and stops a sub-pipe in a controlled way
Event-driven	**Delay**	Delay the data flow during some cycles
Event-driven	**Comparator**	Monitor the input data flow and detects the crossing of a particular value
Event-driven	**Trigger**	Computes the local pipe value from the timestamp of a Fast Input event
Modification	**Cam**	Applies a cam table (also called Cam Profile) to the input data flow
Modification	**Gear**	Applies a gearing ration on the input data flow
Modification	**Phaser**	Applies a phase offset to the input
Output	**Convertor**	Converts input data flow to a position and forwards it to an axis
Destination	**Axis**	Models a physical axis

Table 3-3: Pipe Network - List of Pipe Blocks

Master

Use a Master Pipe Block to create a virtual master to link two or more axes. The Profile generator in the Master block is trapezoidal. If a parabolic type profile is required, use a PMP Pipe Block. If the master is an external encoder or another axis, use the Sampler Pipe Block.

Sampler



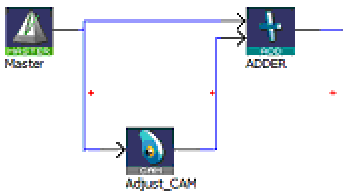
Use a Sampler Pipe Block to read an external encoder as an input signal into the Pipe Network or to directly read the actual position of another axis.

Gear

Use a Gearing Pipe Block to perform electronic gearing. The Gear Pipe Block allows gear ratios and the slope of the gear change to be initially set, then changed from within the application program.

Cam

Use a Cam Pipe Block to optimize the motion profile. Use an Adder block with a Cam block to dynamically change the distance moved during each period (or modulo) of motion.

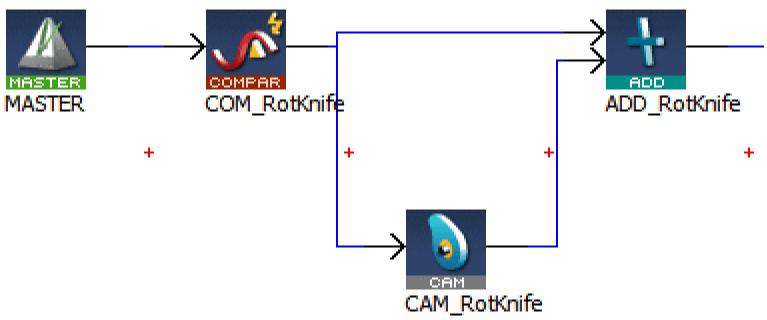


Cam Profiles are created using the cam creation tool.

Comparator

By tracking the position at one point of the Pipe Network, you can use a Comparator Pipe Block to synchronize when code is executed in a PLC application program.

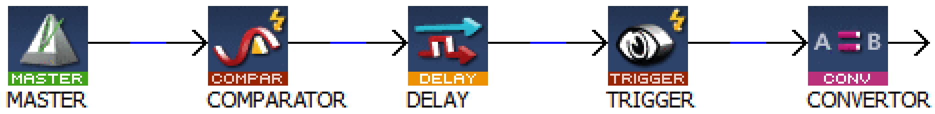
The following example shows the changing of the offset move by changing the amplitude (or offset) of the Cam Pipe Block.



In a PLC application program, the MLCompWriteRef function block is used to arm the comparator block and MLCompCheck function block is used to check the position. By using condition statements in a user program, specific actions (such as changing the move distance of the offset) can then be taken.

Another example shows the use of a Comparator Pipe Block to determine if a high-speed input is within the acceptable position range.

Trigger



Use a Trigger Pipe Block to read the position when a high-speed input is triggered on the machine. The trigger block allows you to “catch" the position at a particular location in the Pipe Network, as required by the application.

Delay

Use a Delay Pipe Block to delay the flow of position through a Pipe Network. One potential use is to place it before a Trigger block in a pipe which is not connected to a drive. There is a delay of five servo update cycles between the dynamic position in the Pipe Network and the triggering of a high-speed digital input.

Phaser

Use a Phaser Pipe Block to perform a dynamic phase adjustment inside the Pipe Network. This block can be used to phase-advance or phase-retard a position as required to synchronize different motion elements on a machine.

Synchronizer

Use a Synchronizer Pipe Block to synchronize two axes. This Pipe Block is useful in applications where it is necessary to start the motion of a second axis and sync to the first.

Axis

Models the link from the Pipe Network to a physical axis.

Changing Information Flow from Position to Velocity

You can change the Pipe Network flow of information from position to velocity by using the **Convertor** Pipe Block. This Pipe Block is normally set up to receive position, so it must be changed to receive the expected input signal type as shown below:

Change the mode of Converter block to SPEED (and not POSITION mode).

4.2.3.4 Axis Pipe Block

Making the link between the logical and physical worlds, the Axis pipe block manages the data on positions.

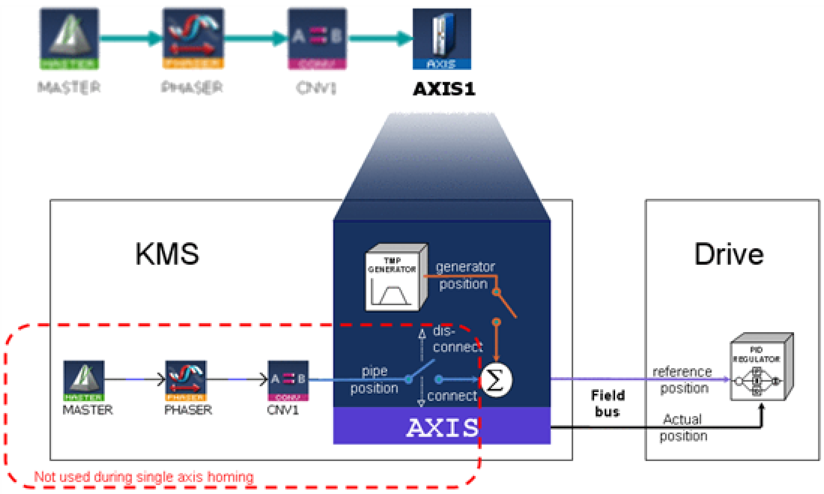


Figure 3-9: Axis Pipe Block Positions

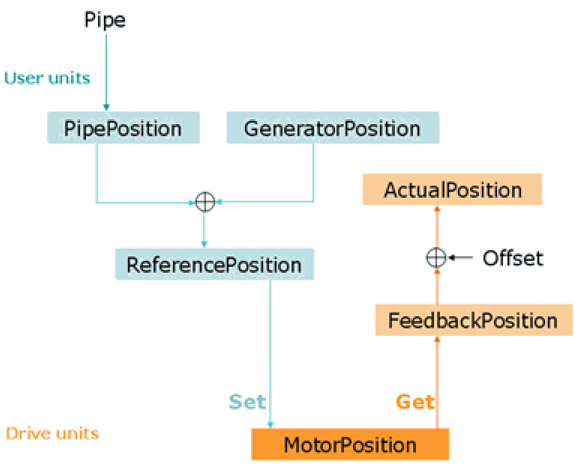
About Associated Data on Positions

The following data are illustrated in the figure below

- **PipePosition:** input position of the Axis issued by the upstream **pipe** and sent to the motion bus (equivalent to the output position of the convertor block)
- **GeneratorPosition:** position profile **generated by the axis block** and sent to the motion bus (it is the summation of all motion commanded to the axis, except for the changes in PipePosition)
- **ReferencePosition:** output position sent to the motion bus
- **ActualPosition:** real position (taking Offset into account) provided by the drive through the motion bus.
  The ActualPos is calculated by adding offsets to the Feedback position:

  ```
  ActualPos = FeedbackPos + ZeroOffset + Actu-
  alOffset + PipeOffset
  ```

- **FeedbackPosition:** absolute position provided by the drive through the motion bus



Reference Position

The motion command to a servo drive is called the Reference Position. The Reference Position is the sum of a position command from the axis generator and the Pipe Network.

```
Reference Position := Pipe Position + Generator Position
```

Actual Position

The Actual Position of the axis is returned from the drive, and it takes into account any offsets due to:

- Position offset established after homing using the MLAxisWritePos function block
- Pipe position offset established after MLCNVConnect is used to connect the Pipe Position to an Axis.

Axis Block Initialization

A call to the MLAxisInit function block is required to implement motion for the axis.

- All positions and offsets are set to zero
- The Axis Block motion generator is initialized with the proper ranges
- The values are "aligned": ReferencePosition = Pipe Position + Generator Position

Axis Connection to a Pipe

A call to the MLPN_CONNECT Function or the MLCNVConnect function block is required to get motion generated in the pipe to the Axis

- Pipe Offset is calculated as follows: Pipe Offset = Pipe Position – Reference Position
- The values are “aligned”: Reference Position = Pipe Position + Generator Position

Realigning Positions

A call to the MLAxisReAlign function block is used to realign the axis after an error occurs

- Motion must come to a stop first
- The MLAxisReAlign is executed
  You must set the movement of this block to MLAxisReadActPos - MLAxisCmdPos
- The target position must be reached before any additional motion can occur.
  It can be checked by using the MLAxisReAlgnRdy function block

Set Zero Axis

A call to the MLAxisWritePos function block is used to set a position offset at the Axis when the Pipe Network is not yet connected

- Pipe Position and Pipe Offset are set to zero
- Generator Position is set to equal to Zero Position
  (Zero Position is defined in MLAxisWritePos function block)
- Then Reference Position equals Pipe Position + Generator Position

Homing

Homing is the process of moving the motor to a known physical reference point on the machine.

Drive Homing

The AKD contains various pre-configured homing modes that avoid creating code. These home modes are drive-controlled and selected using the AKDHome function block .

Controller Homing

This homing type requires code in the application or UDFBs to perform the homing move.

Each axis is homed using MLAxis function blocks only (the Pipe Network is not used). Typically homing is done with MLAxisRel and MLAxisAbs to make motion and MLAxisWritePos to set a position offset.

Single-Axis Operation

This includes motion done on an individual axis: jogging, absolute move, or incremental moves. If these are single-axis based, then motion is executed with the MLAxisMoveVel, MLAxisAbs, and MLAxisRel FBs. These motions are typically done during machine setup or adjustment and are often referred to as manual mode. For these operations, the Pipe Network does not need to be connected to the axis.

Multi-Axis Operation

For multi-axis applications, automatic operation requires motion synchronization between two or more axes and the Pipe Network is required to achieve the synchronization. To start up the Pipe Network the following two functions must be executed in an application program:

```
PipeNetwork(MLPN_ACTIVATE):
PipeNetwork(MLPN_CONNECT);
```

Multi-axis synchronized motion is then accomplished using a motion block associated with one of the three input Pipe Blocks:

- Master: MLMasterRun, MLMasterRel, and MLMasterAbs
- PMP: MLPmpAbs, MLPmpRel
- Sampler: MLSmpConnect, MLSmpConnectEx

Monitoring an axis

There are function blocks to monitor the performance and status of an axis. The key function blocks are as follows:

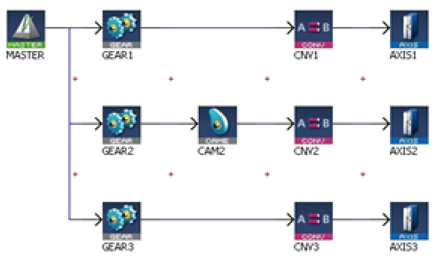
- MLAxisCmdPos - The commanded position to the servo drive
- MLAxisReadActPos - The actual position of the axis
- MLAxisStatus - The status of the axis: enabled/disabled , bus connection, Pipe Network connection, drive executing an axis stop function, drive finished a stop
- MLAxisReadGenStatus - The status of the Axis generator: acceleration, run , deceleration, change designation point, single step
- MLAxisGenIsRdy - Is Axis generator ready

4.2.3.5 Executing Motion

Two types of Pipe Blocks are used to command motion in a Pipe Network: Axis block and Input block.

- Axis block starts motion directly on one axis.
- Input blocks start motion that affect all axes that are connected downstream in a Pipe Network. Input blocks can be one of three types:
  - Master - Trapezoidal motion
  - PMP - Parabolic Motion
  - Sampler - Externally generated motion from another axis or external encoder

In the following example, executing MLAxisMoveVel, MLAxisAbs and MLAxisRel Functions can be used to cause motion on a particular axis. Whereas MLMstRun, MLMstAbs and MLMstRel functions cause motion on Axis1, Axis2 and Axis3.



For information on **error management**, see page 512.

For explanations on **restarting the motion**, refer to paragraph "Restarting Motion" on page 512

4.2.3.6 Pipe Block Lifetime

Activation

The pipe is activated when the output of the Convertor Pipe Block is connected to its related Axis (all characteristics are reset to the declaration values and the history of the block begins).

Usage

As long as the pipe remains active, its values are cyclically calculated. Functions can be performed and events can be created.

Deactivation

The pipe is deactivated when the deactivate function is applied to the pipe (all internal current values are lost and the block no longer exists).

4.2.3.7 Motion State Machine

The Motion State Machine is driven by the IEC 61131-3 application with the help of dedicated function blocks.

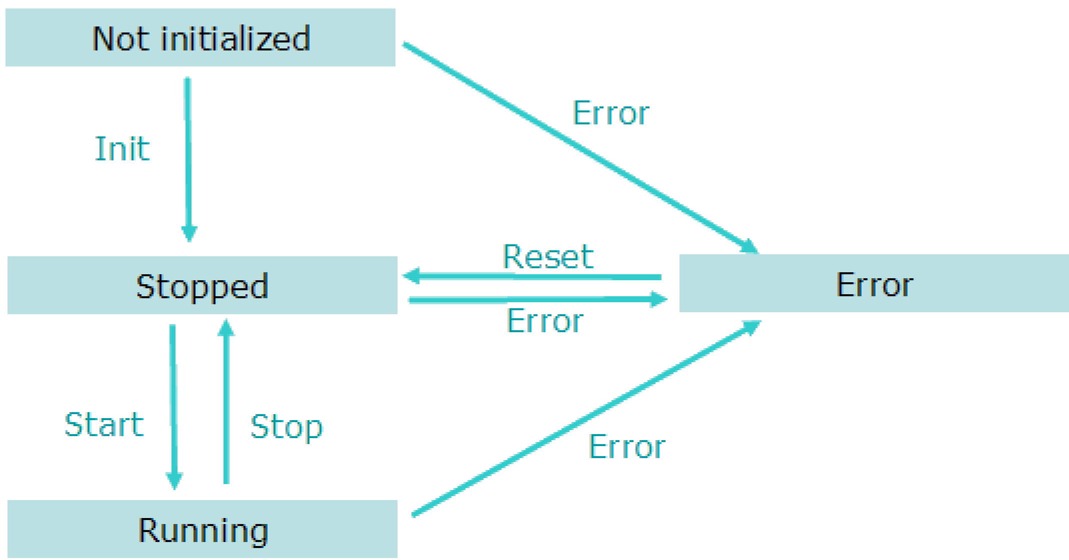


Figure 3-10: Motion State Machine

Each arrow represents a transition from one State to another.

4.2.3.8 Phase of Execution in the Pipe Network

Absolute phase of execution

The absolute phase of execution of a pipe is the elapsed time between any fixed reference and the next computation for the specified pipe.

Relative phase of execution

The relative phase of execution between two pipes is the elapsed time between the computation of the first pipe and the second one.

The relative phase of execution between two pipes of the same Pipe Network is zero. The phase of execution between two Pipe Networks cannot be specified by the user and depend on the pipe activation time of the application execution.

4.2.3.9 Use Motion Function Block for Pipe Network

Use motion library function blocks in your PLC application program to interface to the Pipe Network (see procedure here).

ML function blocks are used to:

1. Create and initialize the Pipe Network
2. Perform motion at a single-axis or multi-axis level
3. Read information from points in the Pipe Network

   For example:

   - Read a high-speed input position from a Trigger Pipe Block
   - Read Command or Reference position from an Axis Pipe Block
   - Determine when a position has been reached in a Comparator Pipe Block
4. Modify how the blocks work in the Pipe Network

   For example:

   - Change the phase offset of Phaser Pipe Block
   - Change the amplitude or offset of a CAM profile
   - Change the speed of a Master Pipe Block

Buffer Mode

With the Pipe Network engine, when a motion function block is executed while another one is presently executing, there is an immediate change. That means the previous function block is aborted and the new one immediately becomes the active move and begins executing.

Motion Init

During initialization, the IEC 61131-3 application can create (by means of the MLMotionInit function) the different motion objects it needs (pipes, blocks, axes):

- Pipe Create
- Profile Create

NOTE When the state machine leaves the Init state, the creation of new motion objects is no longer allowed, in order to avoid memory allocation problems while running the application.

Motion Start

The Start method (MLMotionStart function) initializes the motion engine, motion bus driver, and initializes EtherCAT network to operational mode. MLMotionStopbus driver deactivates the execution of the motion engine.

The function blocks MLMotionStart, MLMotionStop and MLMotionRstErr can be used by the IEC 61131-3 applications to navigate between states: i.e. Not initialized, Running, Stopped and Error.

Using the Q output of ML Function Blocks for the Pipe Network

There is a Q output on most ML function blocks. The operation of the Q output is different for different ML function blocks. The Q output can be useful in PLC application programs.

Examples:

- MLAxisMoveVel.Q is set when the motion has reached jog speed
- MLAxisRel.Q is set when the motion profile is complete
- MLAxisStop.Q is set when motion is stopped (zero speed)
- MLPrfWriteIOffset.Q is set if cam offset has been changed to the new value

For more details on Q output, refer to paragraph "What is the difference between Q and OK?" on page 107

4.2.3.10 Function - General rules

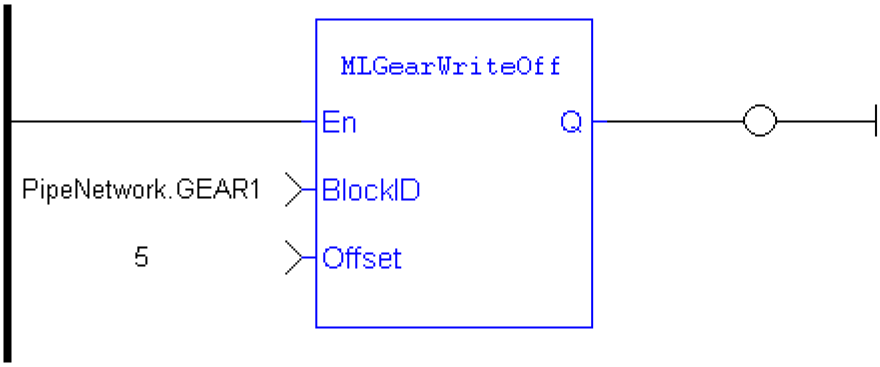
This section outlines rules for using ML function in the Pipe Network context.

Languages

Function that interact with the Pipe Network start with ML (for example MLAxisRel, MLPrfWriteOScale, or MLMstRel). These function can be used in all four of the 61131-3 PLC languages.

BlockID Inputs

The BlockID input is a DINT ID. It is the second input to a Pipe Network function when using FFLD:



The BlockID input is the first one if programming in Structured Text:

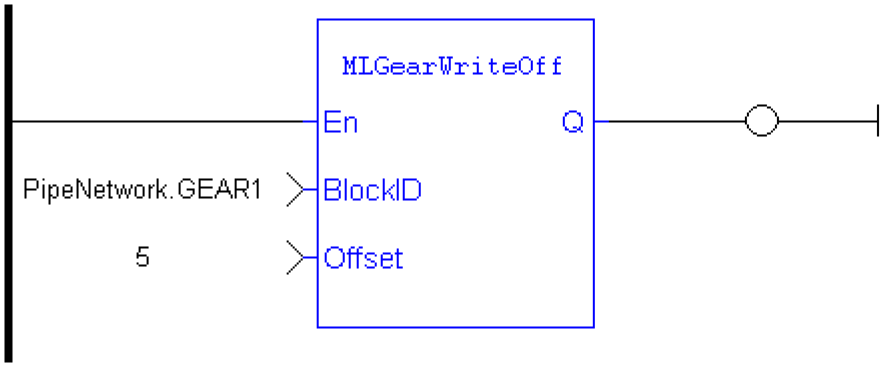
```
MLGearRatSlp( BlockID (*DINT*) );
```

This input identifies the block in the Pipe Network that the function interacts with, and if using the graphical Pipe Network Editor the used variable starts with **PipeNetwork.xxx** (except if it is acting on a CAM profile, in which case the input is named ProfileID and the variable starts with **Profiles.xxx**).

> **TIP**
> As a general rule, when selecting a Pipe Block as the BlockID for a ML function, choose a Pipe Block with the same type which is in the name of the ML function.
> For example, MLMstxxx functions expect a Master block to be chosen for the BlockID; MLAxisxxx functions need an Axis block to be chosen for the AxisID input; and MLPrfxxx functions need a Profile entered for ProfileID, etc.

Output status

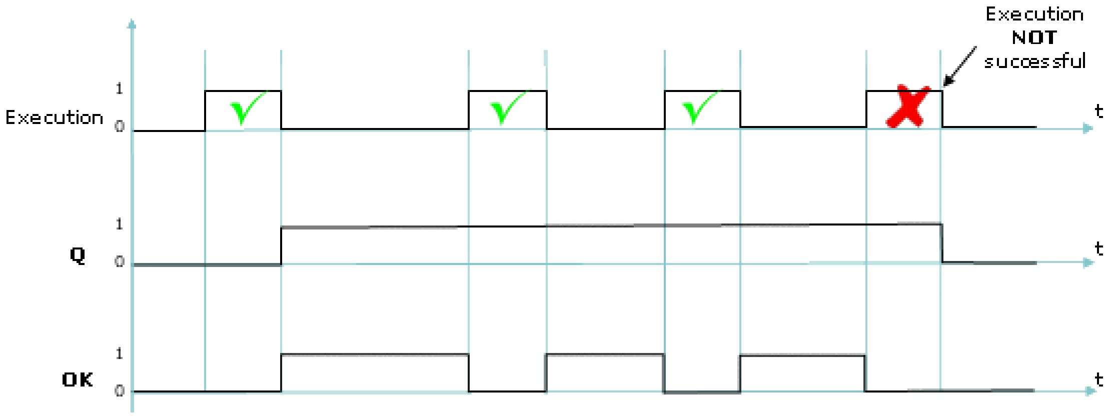
Most Pipe Network function have a default Boolean output labeled either **Q** or **OK**:



What is the difference between Q and OK?

OK returns true when function successfully executes.

Q output is initially set to 0 until the first time the block is successfully executed in a running program. After this execution, the Q output is set to 1. It remains to 1 until the function does NOT execute successfully. Alternately, after an unsuccessful execution the Q output is set to 0. It remains to 0 until a successful execution resumes



When Q is set to True?

Some function change the Q output from low to high immediately after it starts executing, but others (including most functions that command motion) wait to change the output until the function has completely finished executing.
You need to check the description for each individual function block to be sure how its Q output is behaving.

Input parameters

The En input parameter, which is used to execute the function, is not edge-triggered. If a function is seen in the PLC code and its En input is positive, the function executes. For example, a MLAxisRel command continuously executes relative moves in a program if it is called each program cycle; thus it acts as a Run/Jog command if continually commanded.

Missing input parameters

All inputs to a function must be entered in order for code to compile.

Position versus distance

Position is a value defined within a coordinate system.

DeltaPosition is a relative measure related to technical units. It is the difference between two positions.

Default Block Parameters

The parameters set when initializing a Pipe Network block are used as defaults when calling functions. These parameters can be modified in a program by using specific functions to set these values. But if a value is never set in a program the parameter entered during initialization is used.

For example:

When making a Master relative move (MLMstRel) you input the DeltaPosition, but not the velocity or acceleration. You can set the velocity for the move by using the MLMstWriteSpeed function before calling the relative move. If the speed is not set in the program, the default parameter entered during the initialization (i.e. in the properties dialog box of the Pipe Blocks) is used.

4.2.4 Pipe Blocks Description

Master

PURPOSE

In contrast to the independent axes approach, synchronized axes must have something to put them in synchronization. This is the main goal of the Master pipe block which contains a TMP (Trapezoidal Motion Profile) generator, which gives the cadence to the machine. It starts, stops and runs the machine at the desired speed.

The TMP Generator provides linear acceleration and deceleration, and also constant speed operation. These values are pure logical values, with generally no direct physical representation. It is a source block which frequently serves as a virtual master for a system comprised of several pipes.

A TMP Generator may be commanded to produce a movement of specified length (distance), or to accelerate to setpoint rate and operate at that rate until commanded to operate at a different rate. Acceleration and deceleration rates are also specified by the application.

PARAMETERS

Parameter	Description
SAMPLING_PERIOD	Sampling period of the generator expressed according to the cycle (e.g. 2.0 means the sampling is done once every 2 cycles)
MODE	The available modes are Modulo and "No Modulo"
MODULO_POSITION	Modulo Position for cyclic motion systems expressed in user logical units
TRAVEL SPEED	Travel speed value expressed in user position units per second. The travel speed value is used to set the constant speed part of the trapezoidal motion profile
ACCELERATION	Acceleration value expressed in user position units per second squared. The acceleration value is always used to generate the first part of the trapezoidal motion profile
DECELERATION	Deceleration value expressed in user position units per second squared. The deceleration value is always used to generate the last part of the trapezoidal motion profile
INITIAL_POSITION	Initial position value expressed in user position units. Used only at the pipe activation to initialize the position starting point

See details for INITIAL_POSITION and TRAVEL_SPEED parameters

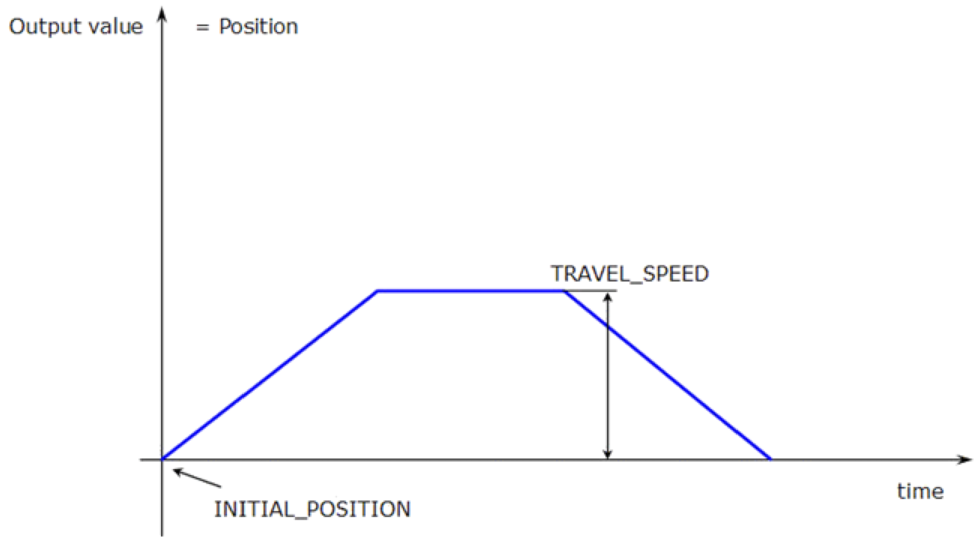


Figure 3-11: TMP Parameters: INITIAL_POSITION and TRAVEL_SPEED

See details for ACCELERATION and DECELERATION parameters

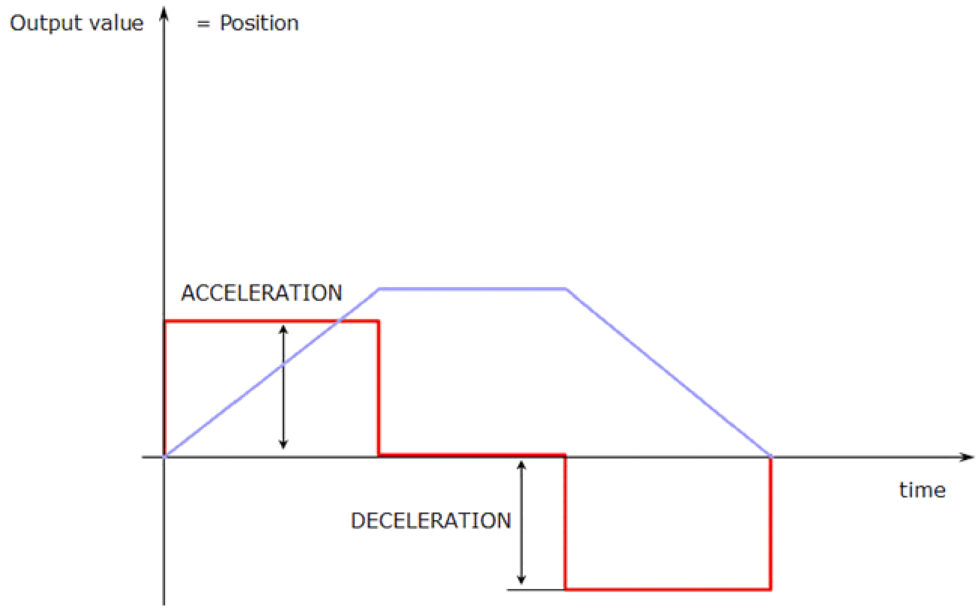


Figure 3-12: TMP Parameters: ACCELERATION and DECELERATION

See details for MODE "No Modulo" parameters

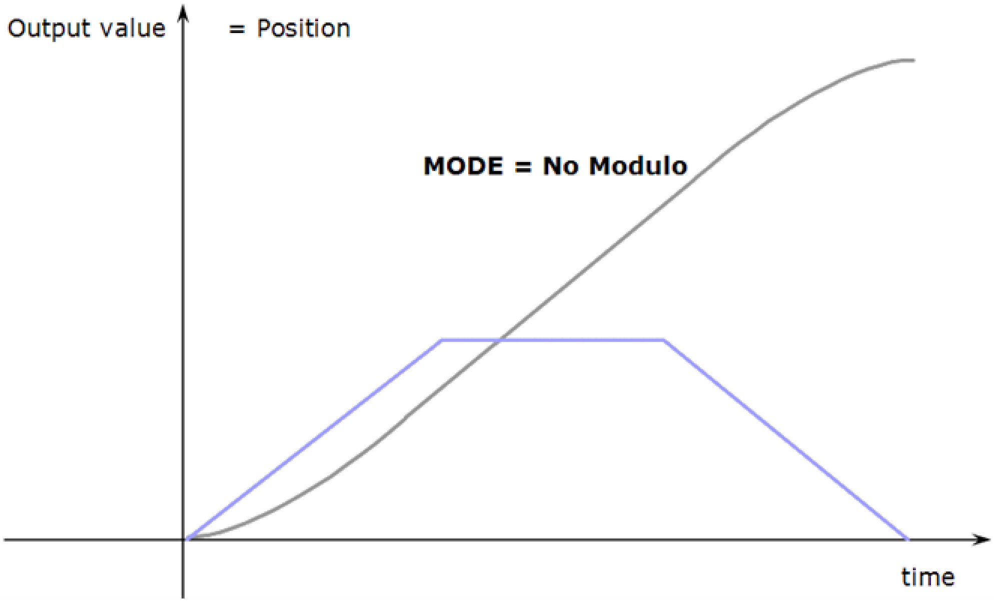


Figure 3-13: TMP Parameters: MODE "No Modulo"

See details for MODE Modulo and MODULO_POSITION parameters

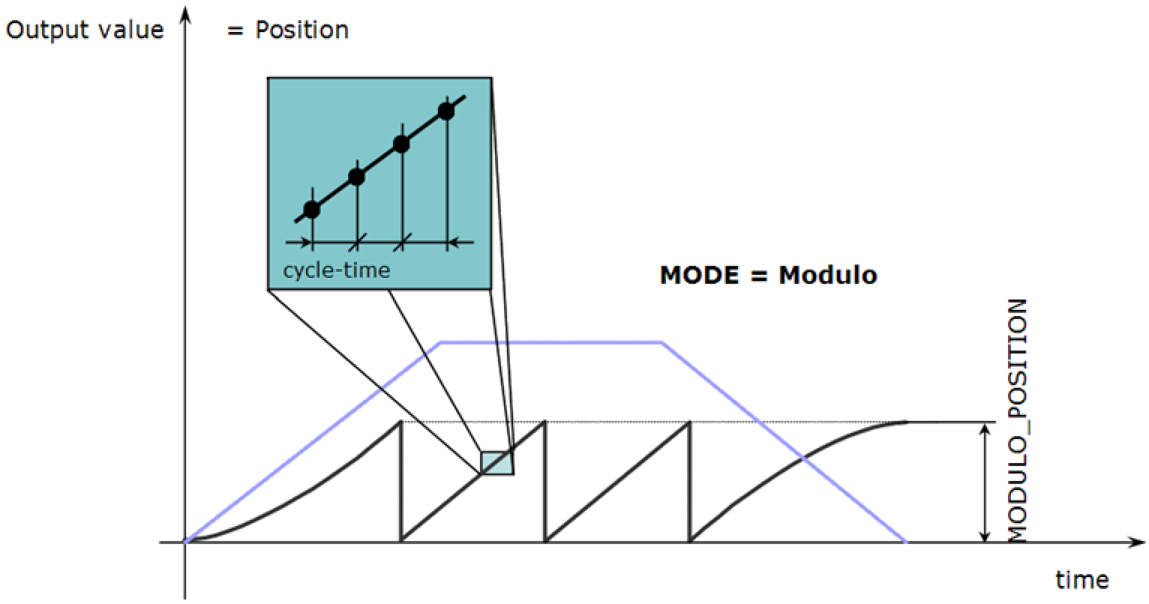


Figure 3-14: TMP Parameters: MODE Modulo and MODULO_POSITION

ASSOCIATED DATA

- **OutputValue:** output value of the data flows
- **IsReady:** Boolean set to TRUE when the pipe block is ready

PMP

PURPOSE

PMP (Parabolic Motion Profile) pipe block generates a flow of values with a second derivative (acceleration) which produces a trapezoidal trajectory. The PMP Generator is similar to the TMP Generator. However, it is useful in applications where jerk (third derivative of the motion) limiting is necessary. Although you can specify the maximum instantaneous rate of change of acceleration.

USES

The PMP Generator is utilized as a virtual master to generate a simple point-to-point profile in machinery where large masses are being rotated or delicate webs (used in industry) are being processed. In fact, it is used in any application where jerk must be limited.

The PMP Generator is also capable of producing forward-backward motions with a non-stop, jerk-free transition through zero speed (see the figure below). This feature is frequently used for linear axes which must make a quick back-and-forth motion without any pause at one end.

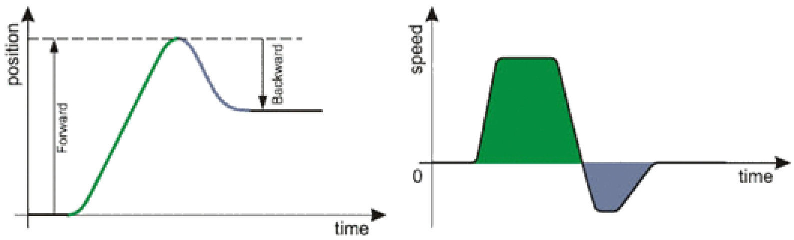


Figure 3-15: PMP Generator forward & backward motion profile

PARAMETERS

Parameter	Description
SAMPLING_PERIOD	Sampling period of the generator expressed in seconds
MODULO_POSITION	Modulo Position for cyclic motion systems expressed in user logical units
FIRST_TRAVEL_SPEED and LAST_TRAVEL_SPEED	Travel speed values expressed in user position units per second. The travel speed values are always used to set the constant speed part of the motion profile
ACCELERATION	Acceleration value expressed in user position units per second squared. The acceleration value (subject to constraints imposed by the JERK parameter) is always used to generate the portions of the motion profile where velocity is changing
JERK	Jerk value expressed in user position units per second cubed. The jerk value is used to generate rounded part of the speed motion profile. Jerk is the derivative of the acceleration, so it specifies the acceleration ramp
INITIAL_POSITION	Initial position value expressed in user position units, used only at the pipe activation to initialize the position starting point

See details for FIRST_TRAVEL_SPEED, LAST_TRAVEL_SPEED and ACCELERATION parameters

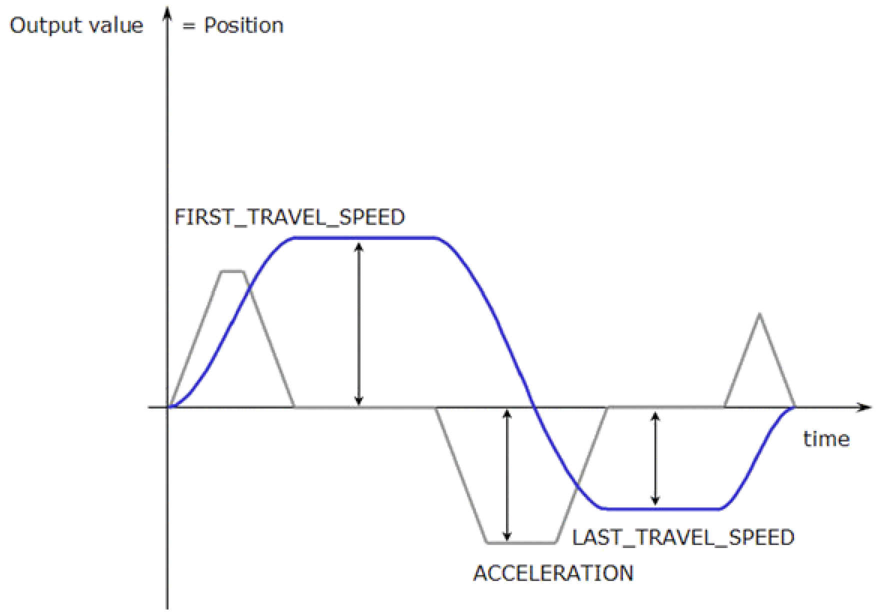


Figure 3-16: PMP Parameters: FIRST_TRAVEL_SPEED, LAST_TRAVEL_SPEED and ACCELERATION

See details for INITIAL_POSITION, "No Modulo" and MODULO_POSITION parameters

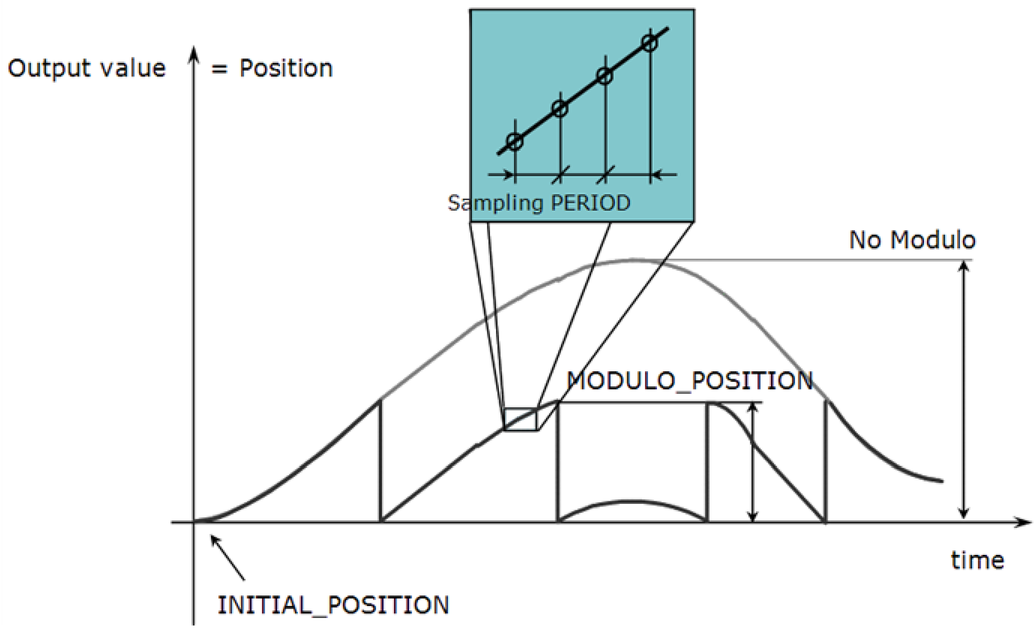


Figure 3-17: PMP Parameters: INITIAL_POSITION, "No Modulo" and MODULO_POSITION

Example of PMP motion profiles: Relative move

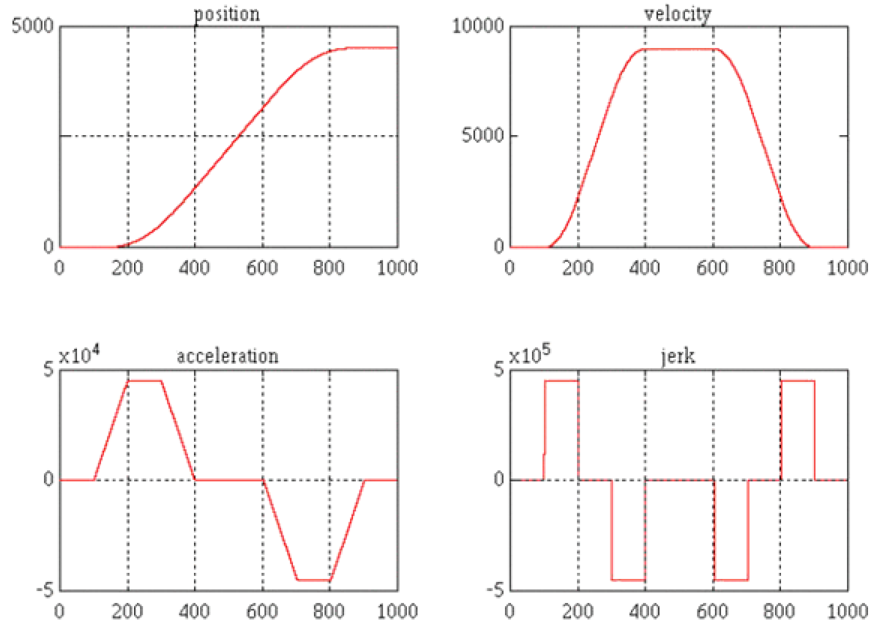


Figure 3-18: PMP Motion Profiles for a Relative Move

Example of PMP motion profiles: Forward-Backward motion

The figure below shows the position, speed, acceleration and jerk profiles generated by a move of 4500 position units forward followed immediately by a backward move of 2000 position units.

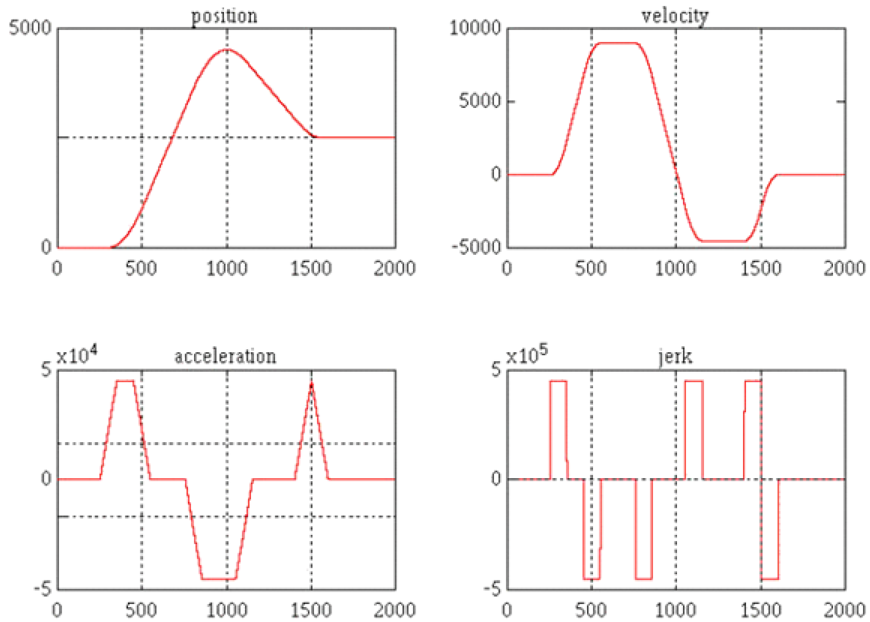


Figure 3-19: PMP Motion Profiles for a Forward-Backward Motion

ASSOCIATED DATA

- **OutputValue:** output value of the data flows
- **IsReady:** Boolean set to TRUE when the pipe block is ready

Sampler

PURPOSE

The purpose of the sampler block is to periodically sample and place into a pipe some output of a source object. The sampled output might typically be the POSITION or SPEED of the source object measured by a resolver, an encoder or some other types of sensor.

The sampler implements a logical connection between an external master (source object outside the KAS system) and one or more pipes for the purpose of slaving the motion of the KAS system to the external master by placing the sampled values into the pipes.

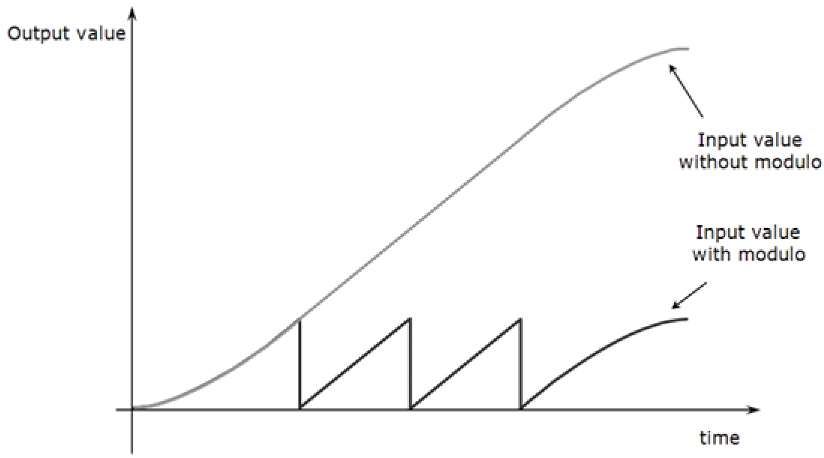


Figure 3-20: Sampler

PARAMETERS

<table>
<tr><th>Parameter</th><th>Description</th></tr>
<tr><td>MODE</td><td>The available modes are POSITION and SPEED

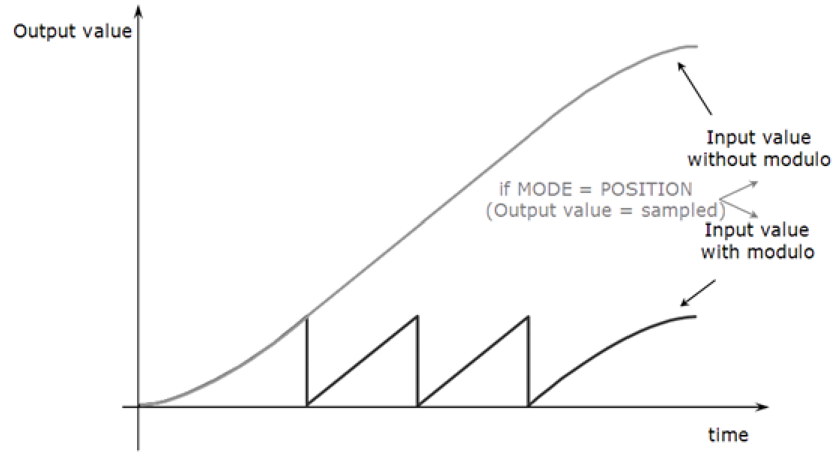


Figure 3-21: Sampler Mode Position

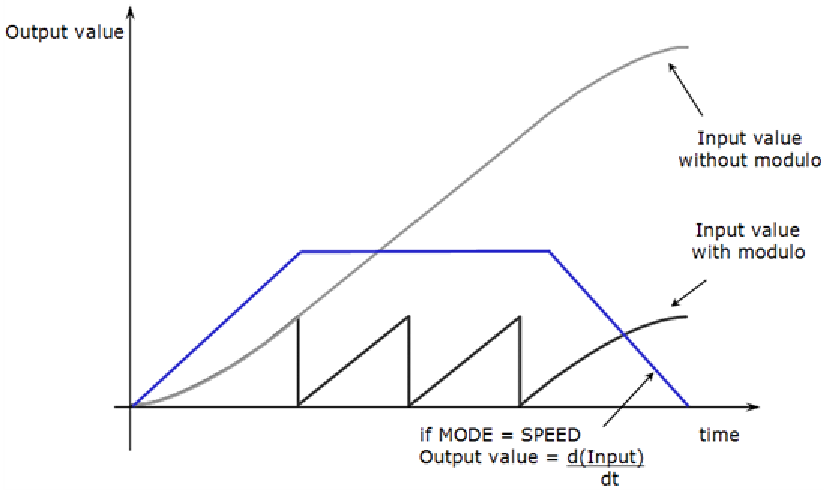


Figure 3-22: Sampler Mode Speed</td></tr>
<tr><td>SAMPLING_
PERIOD</td><td>Period of the sampler expressed in seconds

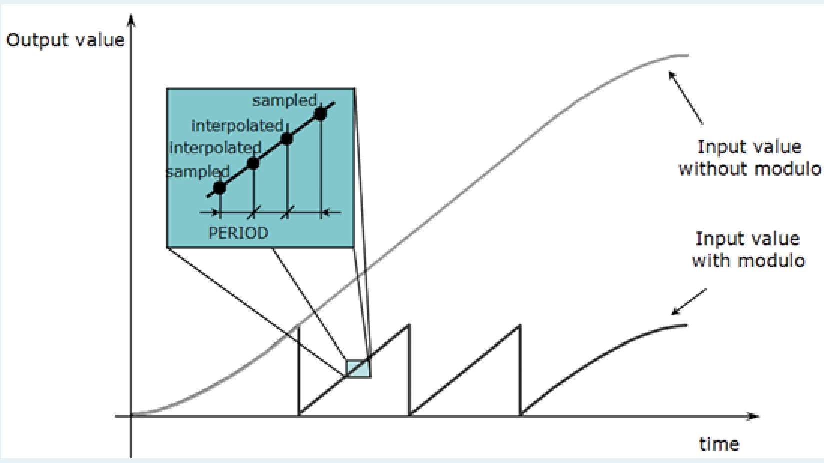


Figure 3-23: Sampler Period</td></tr>
</table>

Example of Sampler Pipe Block

The figure below illustrates the concept. The Sampler feeds motion trajectory data derived from an encoder (or resolver) coupled to the remote machine into the Pipe Network.

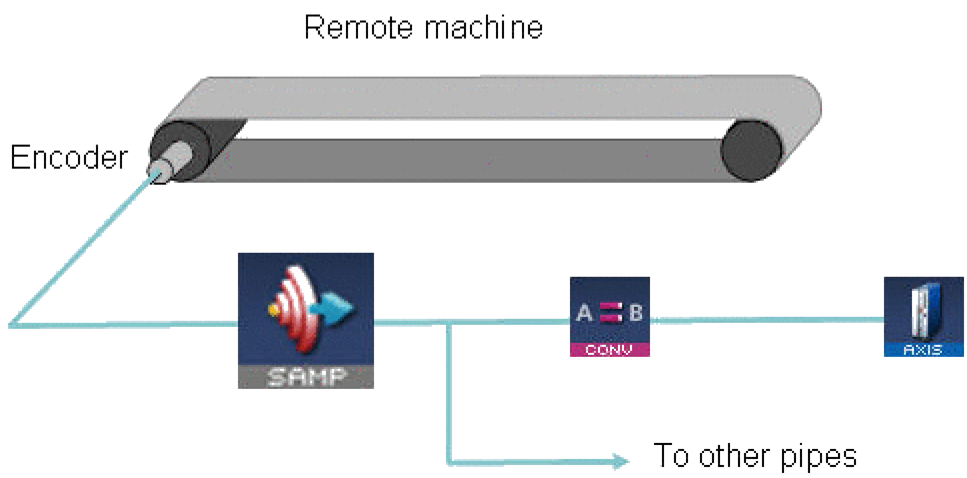


Figure 3-24: Sampler Pipe Block Used to Track an External Master

ASSOCIATED DATA

- **OutputValue:** output value of the data flows
- **IsReady:** Boolean set to TRUE when the pipe block is ready

Synchronizer

PURPOSE

The Synchronizer provides the capability to de-synchronize and re-synchronize an axis to an internal or external master like a mechanical clutch / brake. It is used where a slave axis must be stopped and, when restarted, achieve perfect, jerk-free re-synchronization with the master. The ramping distance (increment of slave axis motion within which ramp up or ramp down occurs) and the slave axis resting position are adjustable.

PARAMETERS

Parameter	Description
MODULO_POSITION	Value of the period of a cyclic system expressed in user units. The parameter is defined to correctly manage the periodicity (modulo) of the input values
CURVE TYPE	When synchronizing, specifies which type of curve (parabolic or polynomial) has to be implemented for merging with the master
OUTPUT PHASING	Set the output phasing value (position reached once the axis is stopped) of the synchronizer block

Example of Synchronizer Pipe Block

Such a pipe block can be used, for instance, when an item is missing on a conveyor.

Figure below illustrates the application of a Synchronizer which enables a slave axis to be stopped, started and re-synchronized to an external master.

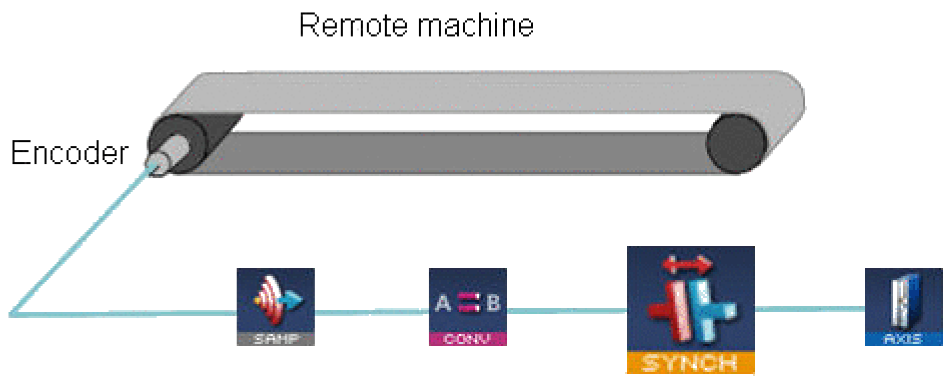


Figure 3-25: Synchronizer Pipe Block to Start, Stop and Re-synchronize a Slave Axis

ASSOCIATED DATA

- **OutputValue:** output value of the data flows
- **IsReady:** Boolean set to TRUE when the pipe block is ready

Phaser

PURPOSE

A Phaser produces a flow of output values which are offset (phase shifted) a specified amount from its input. A typical application of a Phaser is to provide independent phase adjustment capability on an axis.

The Phaser has some similarities with the gear pipe block, however its intended use is quite different. The typical application for a Phaser pipe block is to drive a periodic system: that is to say, a machine where the axes are globally increasing (or decreasing) their position. On the other hand, the gear pipe block, with OFFSET and RATIO parameters, is intended for bounded applications (applications where the integral of speed on a complete cycle is zero). Using the wrong one at the wrong place will cause unnecessary complications.
In addition, you must always consider the position as the input value (and not the speed).

PARAMETERS

Parameter	Description
OUTPUT_MODULO_ POSITION	Defined to correctly manage the periodicity (modulo) of the output values. Expressed in user units
PHASE	Magnitude of the number added to the input value. Phase value may also be negative. A negative phase value is subtracted from the input value. Phase is expressed in user logical units
PHASE_SLOPE_TYPE	You can choose among two modes to define the slope: • Phase_Slope_Max: means that a phase change is fully implemented in a single step. • Phase_Slope_User: You can select this mode to specify the phase slope.

Parameter	Description
PHASE_SLOPE	Rate at which phase changes are implemented, expressed in user logical units per second. A slow rate parameter is provided to limit the implementation of step changes of phase
STANDBY_VALUE	Value assumed by the phaser output when the phaser is in "stopped" condition, expressed in user logical units

ASSOCIATED DATA

- **OutputValue:** output value of the data flows
- **IsReady:** Boolean set to TRUE when the pipe block is ready

Delay

PURPOSE

Delay the data flow a number of cycles.

PARAMETERS

Parameter	Description
CYCLE DELAY	Number of cycles for postponement

ASSOCIATED DATA

- **OutputValue:** output value of the data flows
- **IsReady:** Boolean set to TRUE when the pipe block is ready

Adder

PURPOSE

Adds two data flows (the output is the algebraic sum of the two inputs).

Before being added, input values may be amplified and shifted (multiplication factor and offset are individually defined for each input).

PARAMETERS

Parameter	Description
RATIO	Multipliers for the input data flows
OFFSET	Offset values for the input data flows

```
Output = (Ratio_1 * Input_1 + Offset_1) + (Ratio_
2 * Input_2 + Offset_2)
```

RULES

NOTE **Important!** The two following rules apply to the Adder pipe block

Rule 1: The pipe blocks connected to the Adder inputs (e.g. a Cam and a Gear) must have the same output modulo positions.

Rule 2: The modulo position of the pipe blocks connected to the Adder inputs must have the same value (or a multiple) as the modulo position of the pipe block connected to the output of the Adder.

ASSOCIATED DATA

- **OutputValue:** output value of the data flows
- **Entry1:** input value 1

- **Entry2:** input value 2

Derivator

PURPOSE

The Derivator is a general pipe block whose purpose is to calculate the first derivative of its input values with respect to time.

It is usually used to change incoming position into velocity. It often works together with the GEAR block as gearing in velocity to avoid jumps when suddenly changing the position.

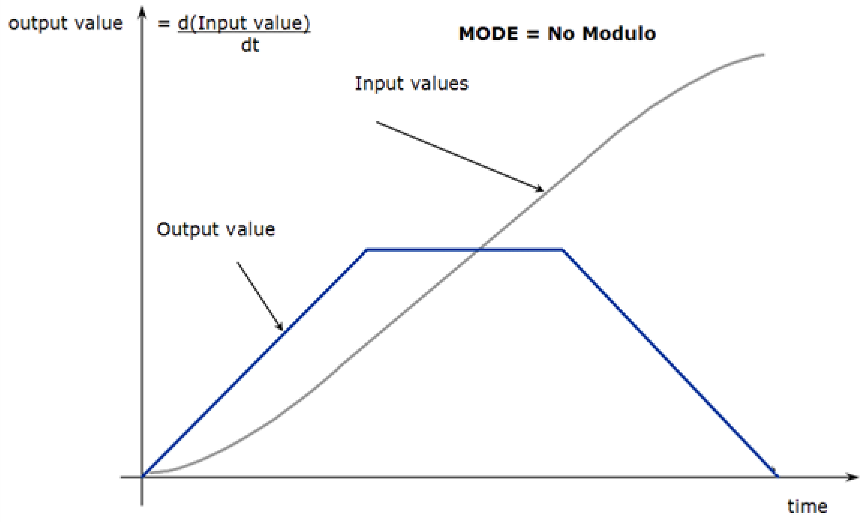


Figure 3-26: Derivator - "No Modulo" Mode

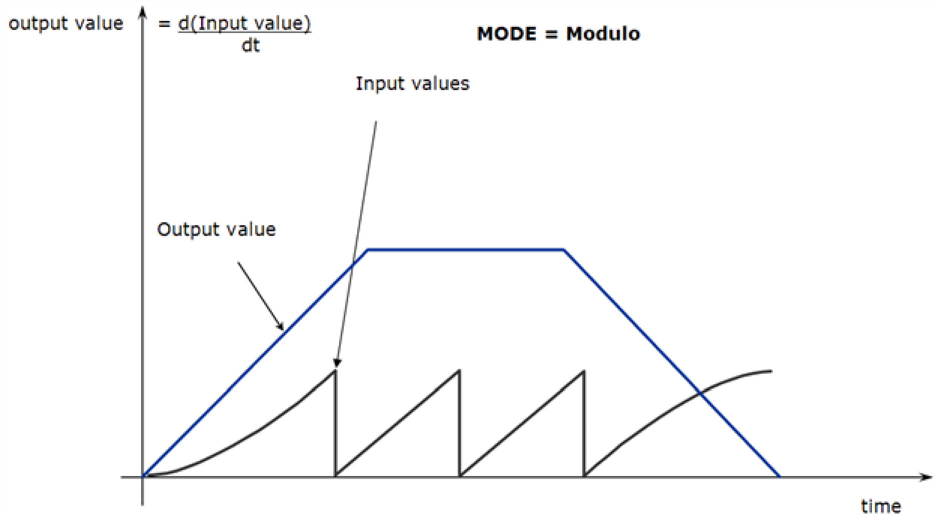


Figure 3-27: Derivator - Modulo Mode

PARAMETERS

Parameter	Description
INPUT_MODULO_POSITION	Value of the period of a cyclic system expressed in user units. The parameter "INPUT_MODULO_POSITION" is defined to correctly manage the periodicity (modulo) of the input values. For example, if the input value increases each millisecond by one (degree) then the output value will be a thousand (degrees per second). Now lets imagine that the input value skips suddenly from 359 to 0 • If VALUE PERIOD = 360, the output will continue to indicate 1000 (degrees per second), indicating that roll-over into the next period has been properly handled. • If VALUE PERIOD = 1000, the output will then indicate -359,000 (degrees per second), indicating that the input has incorrectly interpreted roll-over as a 359 degree change in input in one millisecond.

INITIAL BEHAVIOR

The first calculation of a Derivator pipe block just after the pipe installation indicates zero regardless of the initial input value.

ASSOCIATED DATA

- **OutputValue:** output value of the data flows
- **IsReady:** Boolean set to TRUE when the pipe block is ready

Integrator

PURPOSE

Integrates the input data flow.

Usually used to change velocity to position, and the output is the starting point from where the integration starts.

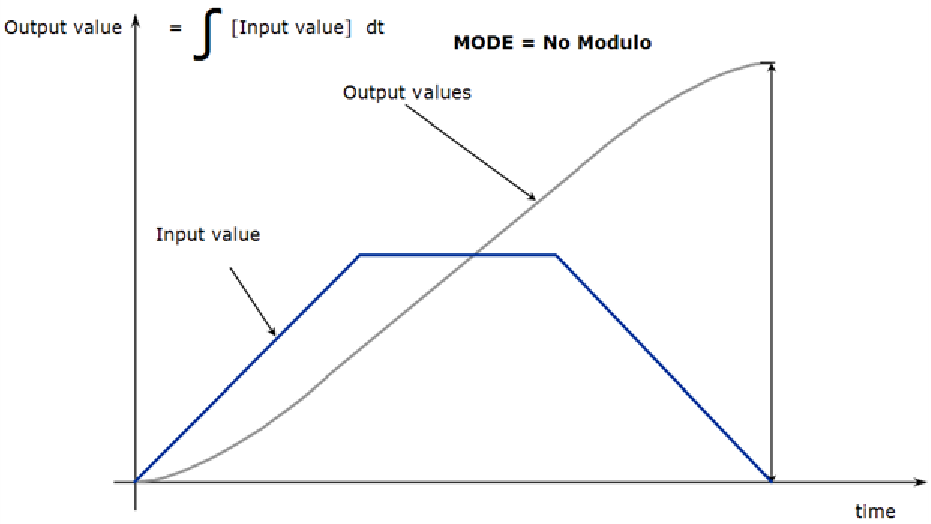


Figure 3-28: Integrator - "No Modulo" Mode

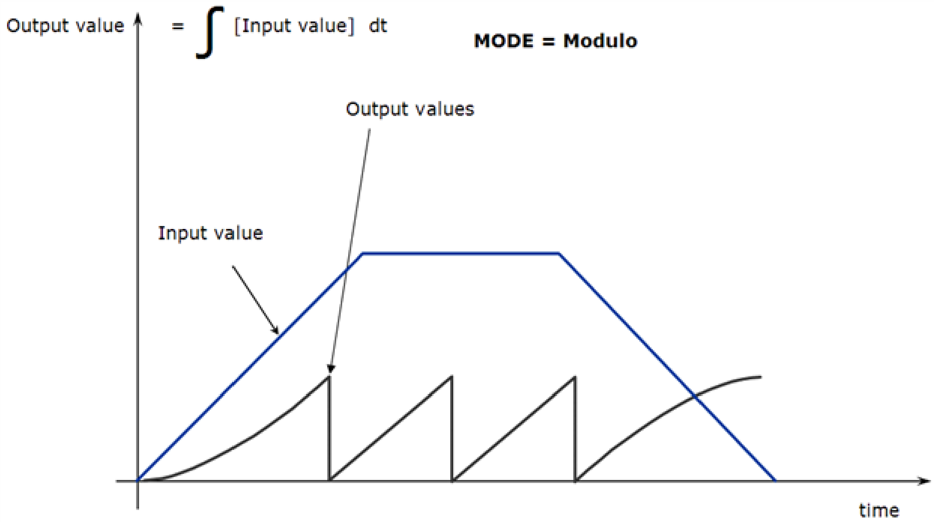


Figure 3-29: Integrator - Modulo Mode

PARAMETERS

Parameter	Description
MODE	The available modes are Modulo and "No Modulo"
OUTPUT_MODULO_ POSITION	When mode is set to Modulo, integrate the input values with respect to time. "OUTPUT_MODULO_POSITION" is defined to correctly manage the periodicity (modulo) of the output values

ASSOCIATED DATA

- **OutputValue:** output value of the data flows
- **IsReady:** Boolean set to TRUE when the pipe block is ready

Trigger

PURPOSE

Computes the local pipe value from the timestamp of a Fast Input time event (with no influence on the incoming flow of values).

Typical application is for registration.

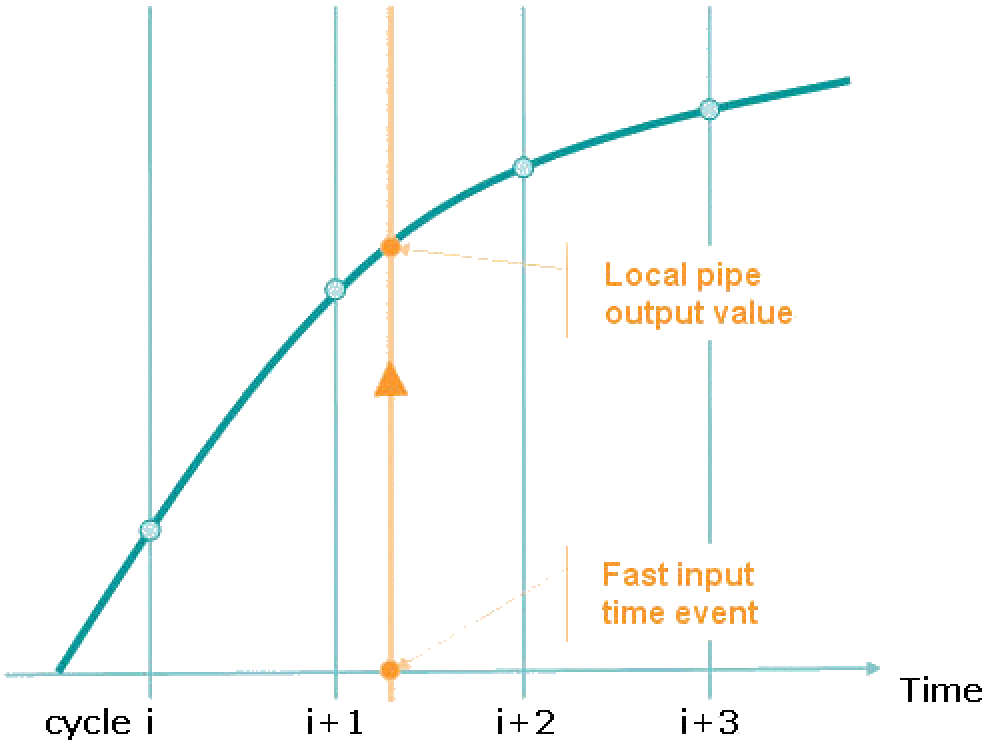


Figure 3-30: Trigger Extrapolates Output Value Based on Fast Input Timestamp

PARAMETERS

Parameter	Description
INPUT AXIS	Name of the axis where the drive has a Fast Input connection
INPUT ID	Identifier of the input object
TRIGGER MODE	Mode can be either RISING or FALLING EDGE

ASSOCIATED DATA

- **OutputValue:** output value of the data flows
- **TRIG_POS:** interpolated position calculated when the time event was triggered (reserved for debugging purposes)
- **TRIG_TIME:** time when the event was triggered (reserved for debugging purposes)
- **DELTA_TRIG_TIME:** reserved for debugging purposes

See also "Fast inputs" for more details.

Cam

PURPOSE

The Cam block is used to generate motion profiles of any shape. The profile generally represents the position transformation.

> **TIP** To avoid jerk in the pipe network (which ultimately may cause a jerk in motor motion when a cam block is applied to the upstream pipe network positions) the potential position offset between the cam's first point and the input to the cam block must be taken care of in the application program by setting a cam offset or another method.

DECLARATIONS

Separating the declaration of the Cam and profile parameters for the Cam pipe block provides the capability to declare and prepare several different cam profiles, and then apply one of these dynamically to the Cam pipe block. Profile switching may be done on the fly, without losing the synchronization and with no dead time.

In addition, the periodicity of the cam output values can be specified when used with a periodic system.

PARAMETERS

Parameter	Description
PROFILE NAME	Name of the current profile assigned to the cam. It must be a declared profile object
OUTPUT_MODULO_ POSITION	Value of the period of the cam output values expressed in user units, for a cyclic system

See details for cam parameters

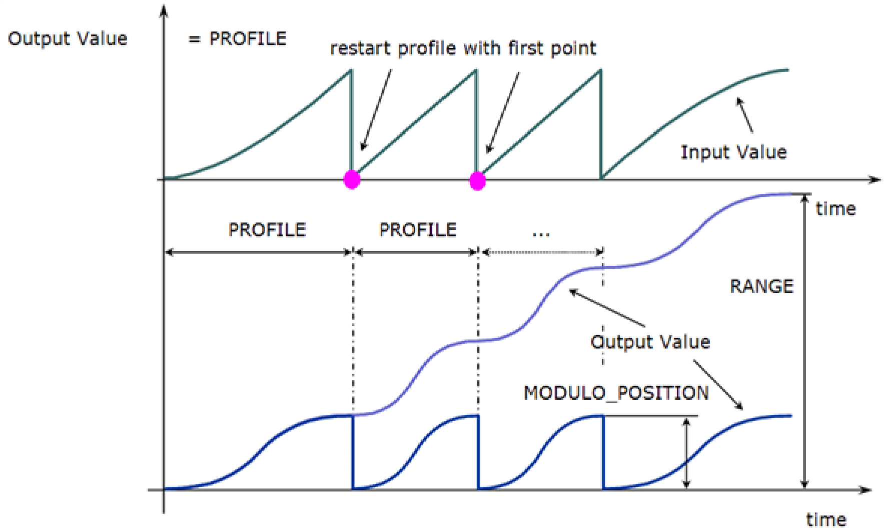


Figure 3-31: Cam Parameters

When a MODULO_POSITION is defined, the output value is reset each time it reaches the MODULO_POSITION.

SHAPE SPECIFICATION

The shape of the cam profile must be processed by the Cam Editor utility before it is usable by the Pipe Network Editor.

The shape of the profile is represented by a table of numerical values. These values can be generated using software tools such as spreadsheets or specialized cam software.

The KAS Cam Editor software tool provides the capability to visualize, analyze, edit and smooth profiles.

Cam blocks have gain as well as offset adjustment capabilities. Axis position is usually the profile variable; however, velocity or torque profiles may also be generated.

CAM'S INPUT-OUTPUT TRANSFER FUNCTION

The mathematical relationship of the cam output as a function of the input and the cam parameters is as follows:

$$\textbf{If } O_{in} \leq X_i \leq O_{in} + A_{in} \textbf{ then}$$

$$Y_i = O_{out} + (fct((X_i - O_{in})/A_{in}) * A_{out})$$

Within the stated limits, the following functions apply:

If $X_i < O_{in}$ then $Y_i = O_{out} + (fct(0.0) * A_{out})$

If $X_i > O_{in} + A_{in}$ then $Y_i = O_{out} + (fct(1.0) * A_{out})$

With:

X_i	Input value	Y_i	Output value
O_i	Input offset	O_{out}	Output offset
A_{in}	Input amplitude	A_{out}	Output amplitude
fct	the function defining the shape		

Example of Cam Pipe Block

The figure below illustrates the use of the Cam blocks in a three-axes container filler mechanism. The cam profile for axis 1 controls the volume of liquid dispensed and the fill rate; Axis 2 raises and lowers the container; and Axis 3 indexes containers under the filling mechanism. All three axes track the main machine motion profile produced by a TMP Generator.

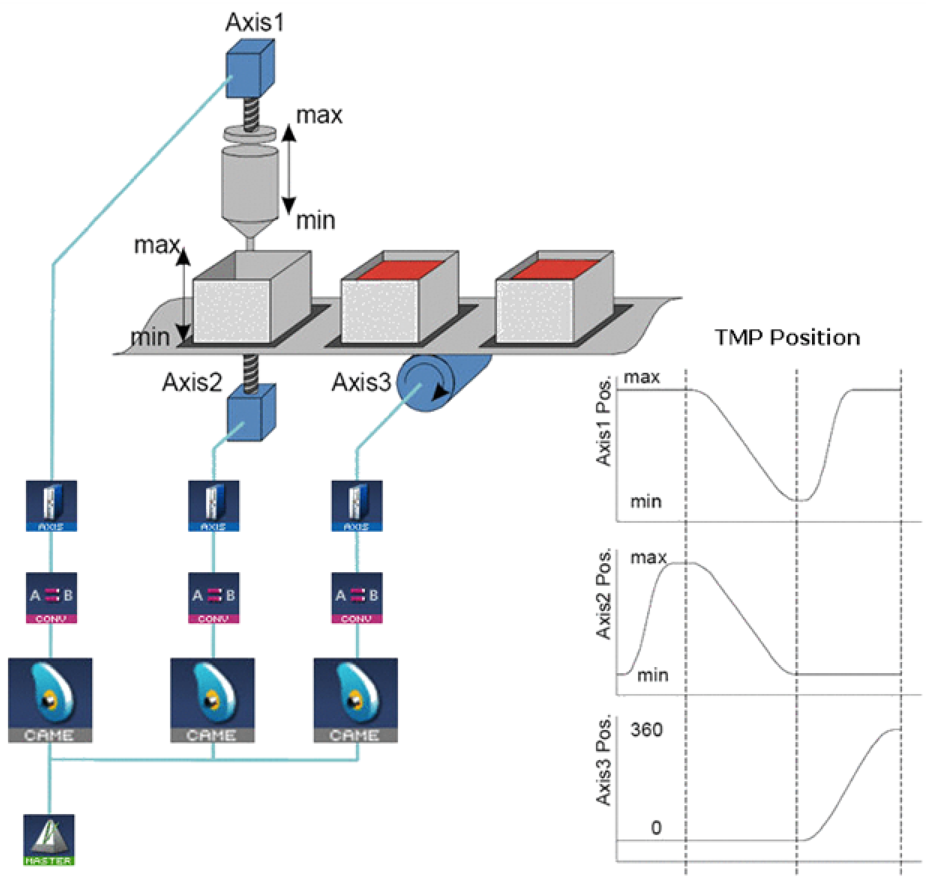


Figure 3-32: Cam Blocks Control Operation of a Three Axis Filling Mechanism

ASSOCIATED DATA

- **OutputValue:** output value of the data flows
- **IsReady:** Boolean set to TRUE when the pipe block is ready

Gear

PURPOSE

The purpose of the Gear block is to amplify/attenuate (with a ratio) and shift (with an offset) the flow of values. A Gear may have a ratio and offset less or greater than one, or even zero. Ratio and offset may be changed dynamically during application execution. A slope may be specified to limit the rate at which step changes in ratio and offset are implemented.

PARAMETERS

Parameter	Description
RATIO	Ratio coefficient
OFFSET	The input offset value
RATIO and OFFSET SLOPE	Sets the maximum rate of change at the pipe block output resulting from changes in RATIO or OFFSET parameters. When set to the MAX (which is the default setting), the slope is infinite. Units are user units per second for OFFSET SLOPE and 1/second for RATIO SLOPE
Modulo	When set to TRUE, adapts the output values according to the periodicity (modulo)

```
Output = Ratio * Input + Offset
```

ASSOCIATED DATA

- **OutputValue:** output value of the data flows
- **IsReady:** Boolean set to TRUE when the pipe block is ready
- **INPOS:** reserved for debugging purposes

Comparator

PURPOSE

A Comparator monitors the flow of pipe data and causes a specified action when the flow of values at its input crosses a specified reference value. A Comparator is often used for synchronizing the operation of an actuator to the position of a product or axis in a machine cycle. The Comparator block does not modify flow values and has no effect on the axis and its periodicity.

PARAMETERS

Parameter	Description
MODULO_POSITION	Value of the period of a cyclic system expressed in user units. The parameter "MODULO_POSITION" is defined to correctly manage the periodicity (modulo) of the input values.
REFERENCE	The Comparator checks if the input value of the Comparator is greater or equal to this reference value
THROUGH_ZERO	Through zero reference mode can be set or not: • YES: used to properly detect a periodic threshold crossing of motions on periodic axis where the flow values are always greater than or equal to zero but lower than the Modulo Position. In this mode, the flow values must first cross one period limit and then, as soon as a value is greater than or equal to the reference, the ready flag becomes true • NO: applies mainly to bounded motions, and the Comparator's ready flag is false as long as the flow value is less than the reference and becomes true as soon as the flow value is greater than or equal to the reference.

USING THROUGH ZERO REFERENCE MODE

The necessity to use the through zero reference mode is illustrated with the following example. Assume that the system is a periodic system with a Modulo Position of 500. The system is running in the positive direction (pipe flow values increase). Imagine that the position of the system is now 400 and you want to wait for the system to reach 326 again. If you ask for the Comparator to detect the 326 reference in normal mode, it will immediately set the ready flag at true (400 > 326) but this is not what you want. If you ask for the Comparator to detect the 326 value in through zero reference mode, it will wait for the system to cross one zero reference (cross the position value = 0) and then will trigger the application when the correct condition is fulfilled.

COMPARATOR RESPONSE TIME CONSIDERATIONS

There is a big difference in response time when using a Boolean equation to compare a value with a reference, versus using a Comparator pipe block do to the same processing. With the Boolean equation, KAS periodically performs the comparison, ignoring any dynamics taking place between successive comparisons, which could result in delays in triggering sequences, and possible loss of information when the pipe-flow value crosses the reference momentarily between comparisons. With a Comparator, the value of the ready flag is intrinsically updated each time a new pipe-flow value is computed. Therefore, it is impossible to lose any transitions.

Example of Comparator Pipe Block

The figure below illustrates an application of the Comparator. In this example, an output valve controlled by a Comparator is added to the filling mechanism from the example in the Cam pipe block. When cam position crosses the value "Trigger 1", the Comparator initiates the "Open Routine" which, in turn, opens the output valve. Next, the Comparator is set to the value "Trigger 2". When the cam position crosses the "Trigger 2" value, the Comparator initiates the "Close Routine" and the valve is closed. The Comparator is again set to the value of "Trigger 1" and the cycle restarts. A user output resident in the Drive operates the valve.

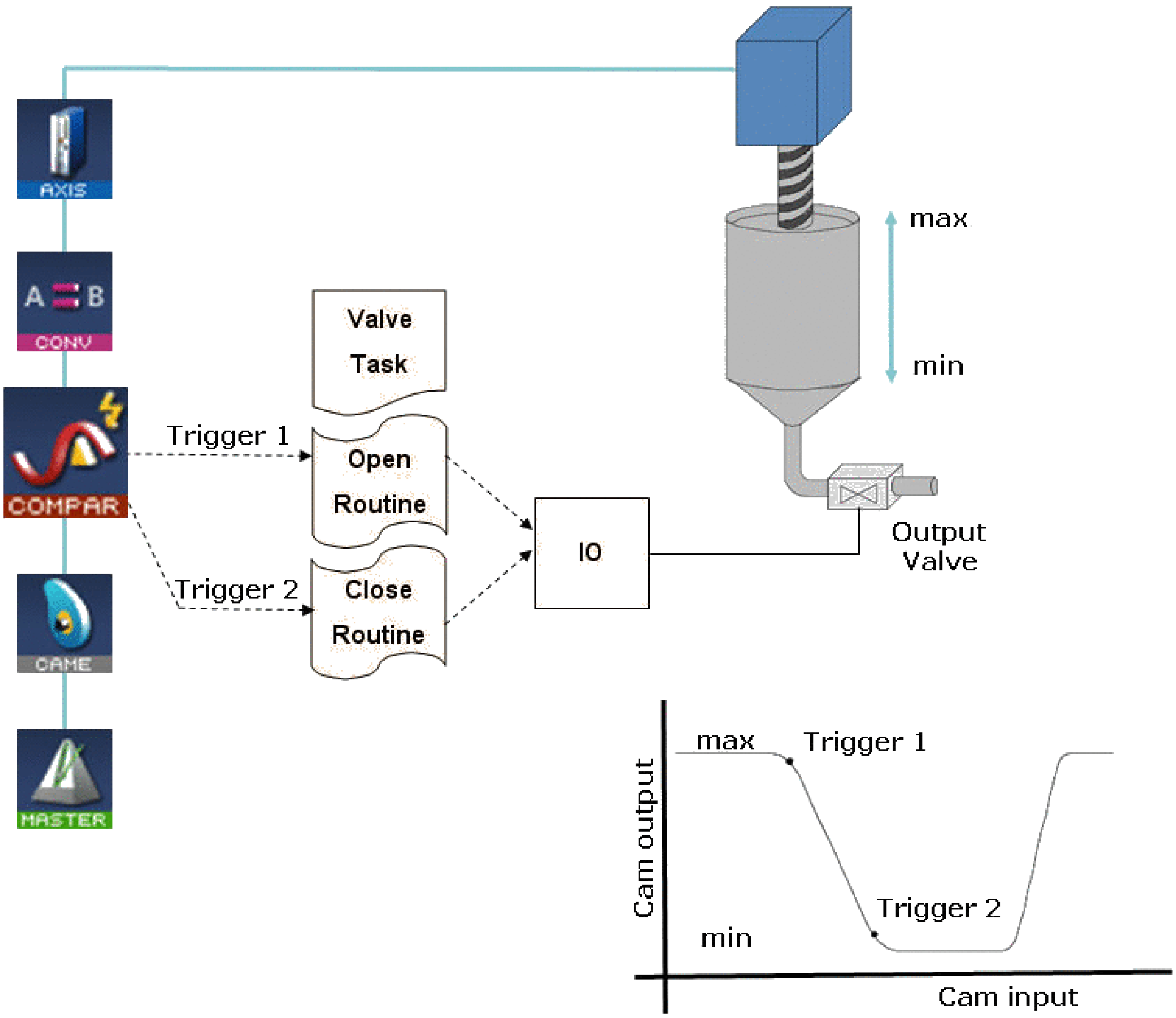


Figure 3-33: Comparator Used to Control a Valve on a Filler Mechanism

ASSOCIATED DATA

- **OutputValue:** output value of the data flows
- **IsReady:** Boolean set to TRUE when the pipe block is ready

Convertor

PURPOSE

The convertor block is necessary to define the connection between a pipe and a destination object. Depending on convertor mode, the incoming numerical values are converted to POSITION or SPEED setpoints with no periodicity.

This conversion has no effect on the axis units and their periodicity.

This block must be present at the end of a pipe, typically right before an axis block.

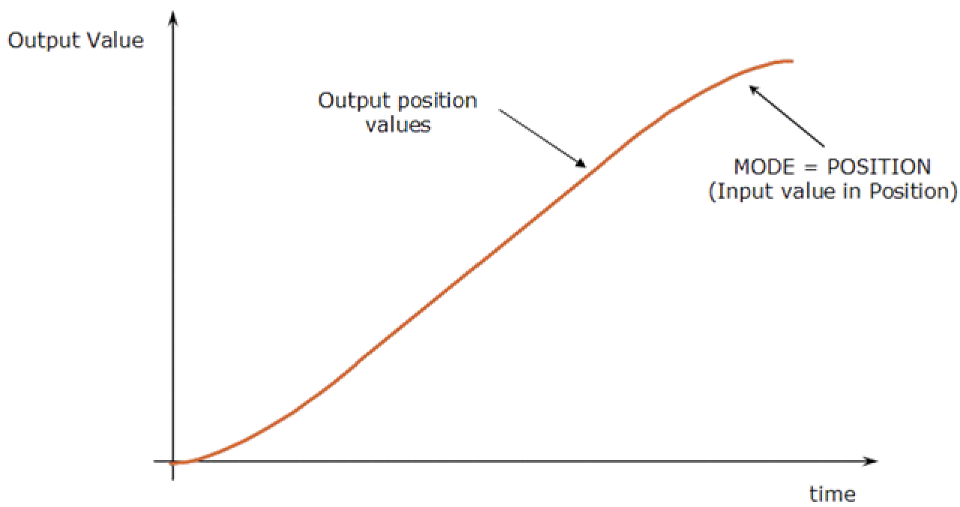


Figure 3-34: Convertor - Position Mode "No Modulo"

Note that Output position values are identical to input values when inputs in position mode (by range)

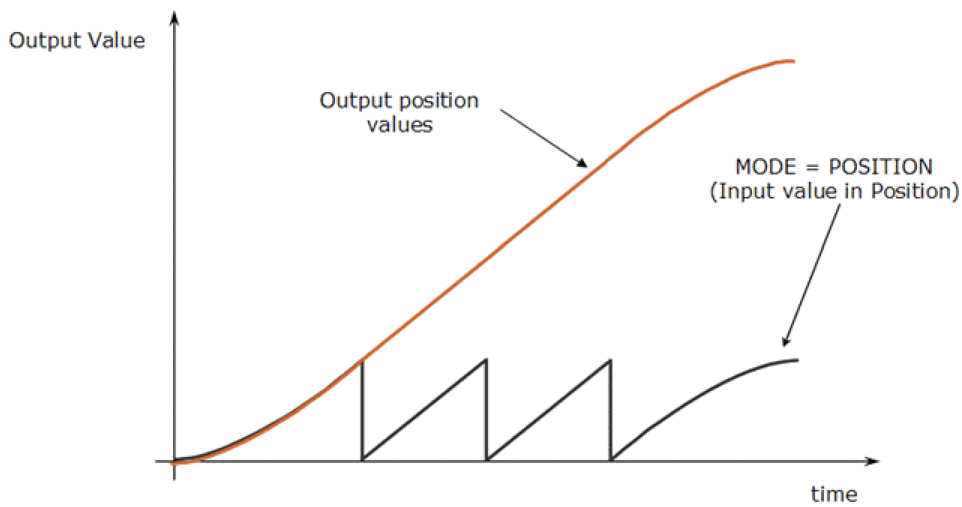


Figure 3-35: Convertor - Position Mode (Modulo)

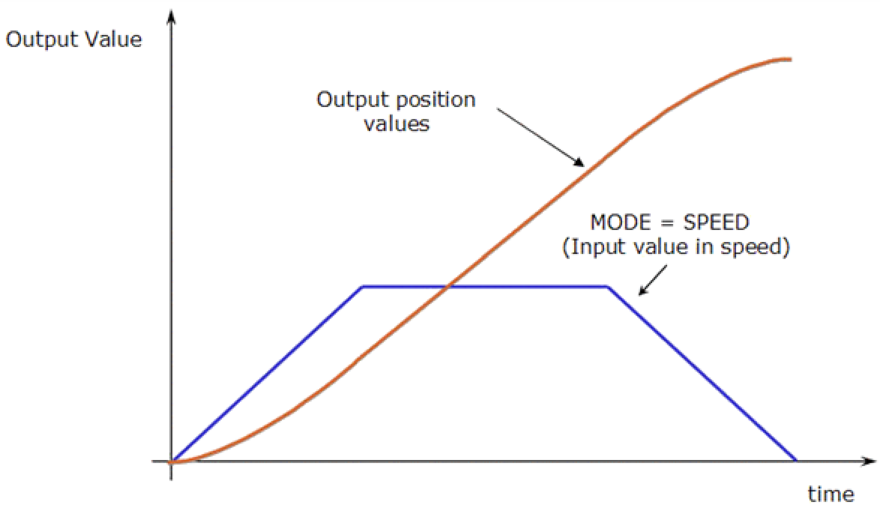


Figure 3-36: Convertor - Speed Mode

PARAMETERS

Parameter	Description
MODE	The available modes are: • POSITION: The values drive the position of the motor. At pipe activation, the current (axis) position is set to the first value given by the pipe by moving the motor. Speed and acceleration are derivatives of position. The torque is set according to the regulator needs. Units are the axis physical units. • SPEED: The values drive the speed of the motor. At pipe activation, the current position is not affected. Position is the integral of speed, and acceleration is the derivative of speed. The torque is set according to the regulator needs. Units are the axis physical units per second.

ASSOCIATED DATA

- **OutputValue:** output value of the data flows
- **IslinkedToAxis:** Boolean set to TRUE when the Convertor pipe block is linked to an axis block

Axis

PURPOSE

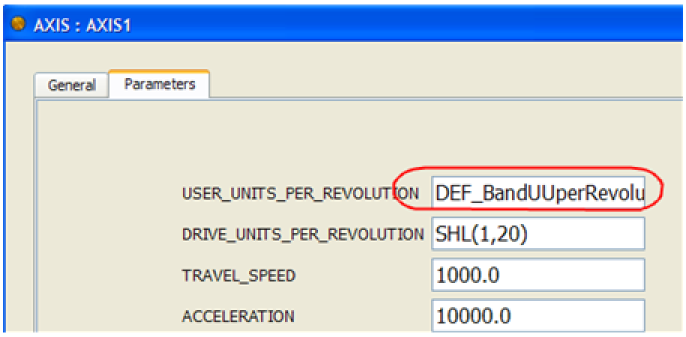
Models the link from the Pipe Network to a physical axis.

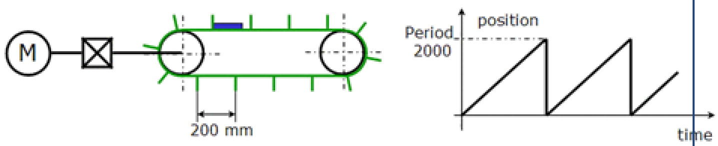
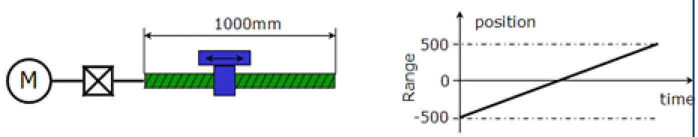
Gives access (through the fieldbus) to remote drive's functions and parameters.

Automatically updates the image of the remote drive's status and error information.

PARAMETERS

Parameter	Description
MOTION BUS	Select in the drop-down menu the type of motion bus associated to the axis
ADDRESS	Specify the address number depending on the motion bus

Parameter	Description
USER UNITS PER REVOLUTION	To divide the current axis into graduations adapted to your project, you must define the unit that is equivalent to one revolution of the physical motor. (e.g. 3600 means that you define the user unit to be tenth of a degree) **TIP** You can rely on expression to define values **See example with expressions** Gear factor 1:3 and 1000.0 User Units per one gear shaft revolution // user units per revolution calculation example #define DEF_BandGear 3.0 // gearbox ratio #define DEF_BandUnit 1000.0 // user units for 1 mechanical turn #set DEF_BandUUperRev DEF_BandUnit/DEF_BandGear  **Figure 3-37:** Define Value with Expressions For more details on Definitions, refer to § "**Use the "Defines" list**"
DRIVE UNITS PER REVOLUTION	Number of units associated to the Drive for one revolution of the physical motor.
TRAVEL SPEED	Travel speed value expressed in user length units per second. The travel speed value is used to set the constant speed part of the trapezoidal motion profile
ACCELERATION	Acceleration value expressed in user length units per second squared. The acceleration value is always used to generate the first part of the trapezoidal motion profile
DECELERATION	Deceleration value expressed in user length units per second squared. The deceleration value is always used to generate the last part of the trapezoidal motion profile
INITIAL_POSITION	Initial position value expressed in user logical units. Used only at the pipe activation to initialize the position starting point

Parameter	Description
MODE	The available modes are Modulo and "No Modulo" Depending on the type of the moving object the axis acts on, you can define the MODULO_POSITION parameter or not. **Modulo** Moving objects, performing a never ending cyclical motion are called periodic (e.g. printing cylinder, cutting wheel). In the following example, if a user unit = 0.1 mm has been chosen, a Modulo Position = 2000 Units could be selected for this transportation system.  **Figure 3-38:** Mode Modulo **No Modulo** Objects always moving within a certain position range (forward/backwards) can be called linear or range axes (e.g. lift axis, moving tables). In the following example, if a user unit = 0.1 mm has been chosen, a position range = 0 to 10'000 Units could be selected for this moving table.  **Figure 3-39:** Mode "No Modulo"
MODULO_POSITION	Modulo Position for cyclic motion systems expressed in user logical units

See details for INITIAL_POSITION and TRAVEL_SPEED parameters

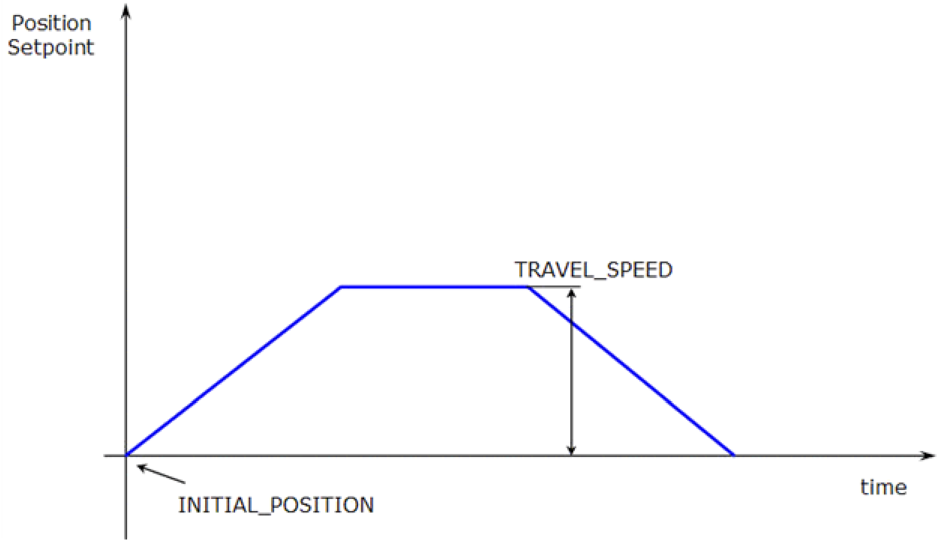


Figure 3-40: Axis Parameters: INITIAL_POSITION and TRAVEL_SPEED

See details for ACCELERATION and DECELERATION parameters

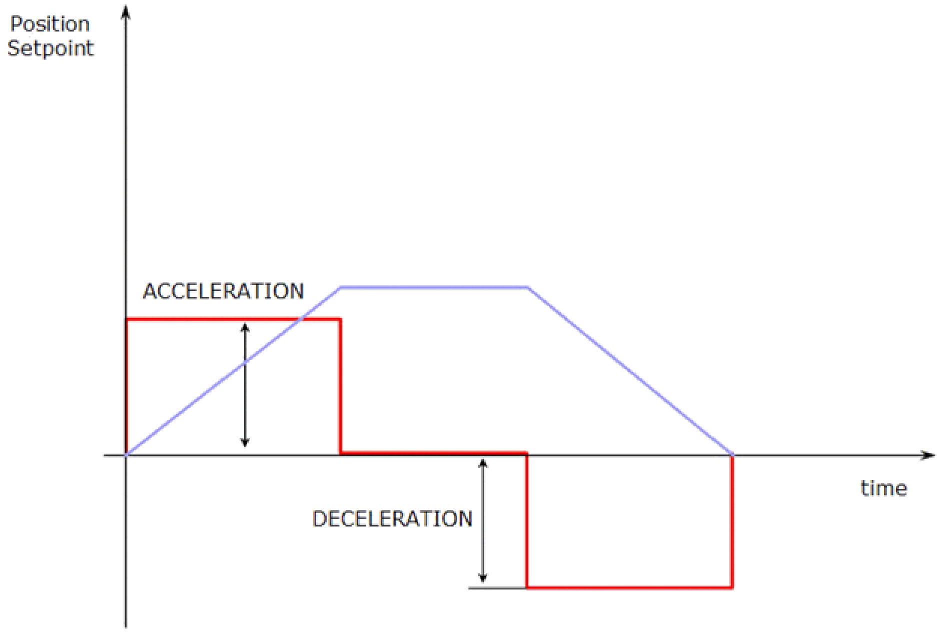


Figure 3-41: Axis Parameters: ACCELERATION and DECELERATION

See details for MODE "No Modulo" parameters

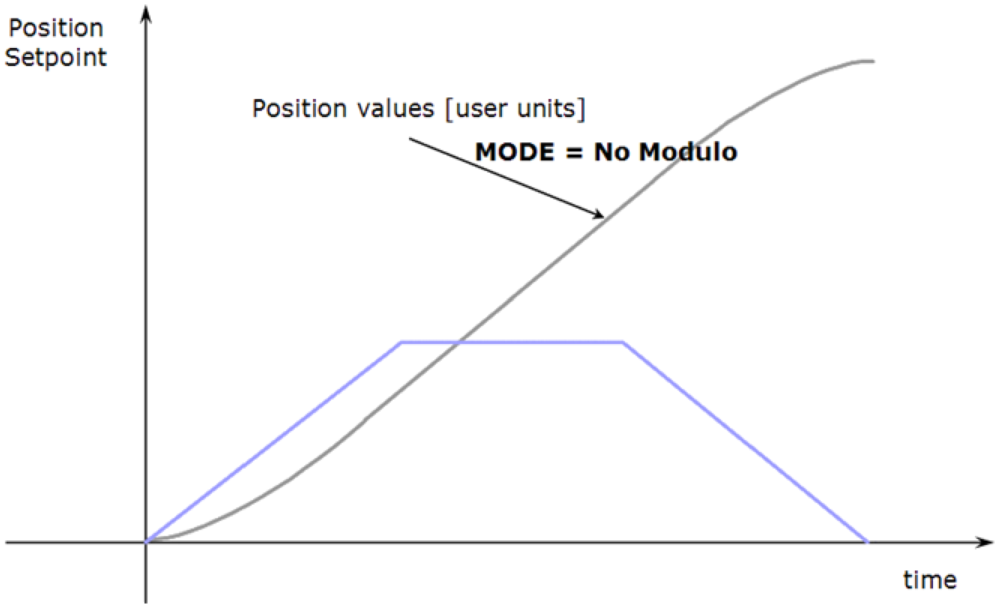


Figure 3-42: Axis Parameters: MODE "No Modulo"

See details for MODE Modulo and MODULO_POSITION parameters

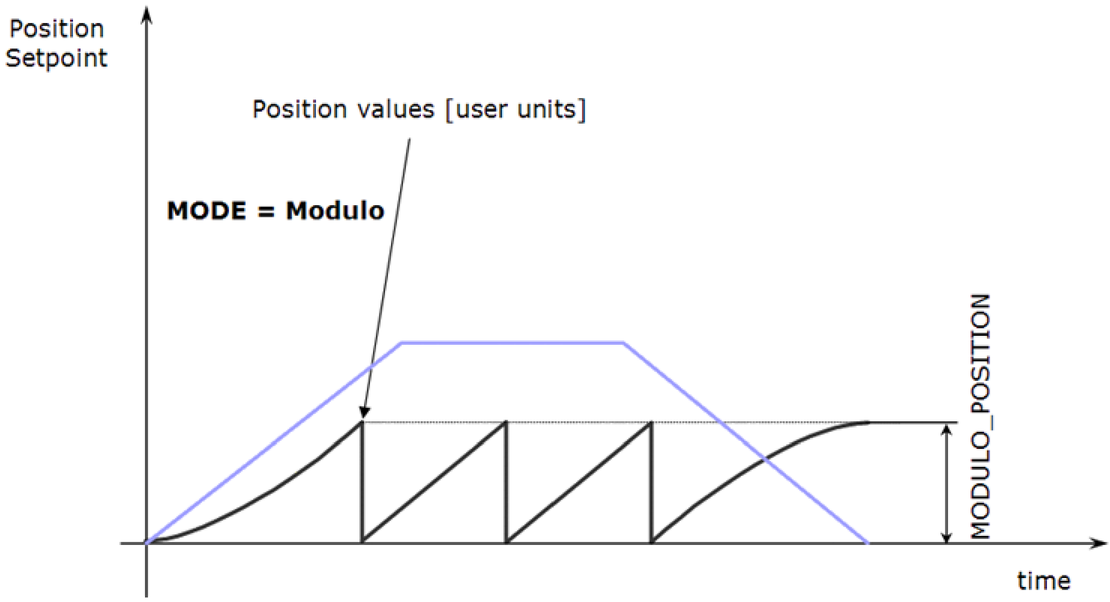


Figure 3-43: Axis Parameters: MODE Modulo and MODULO_POSITION

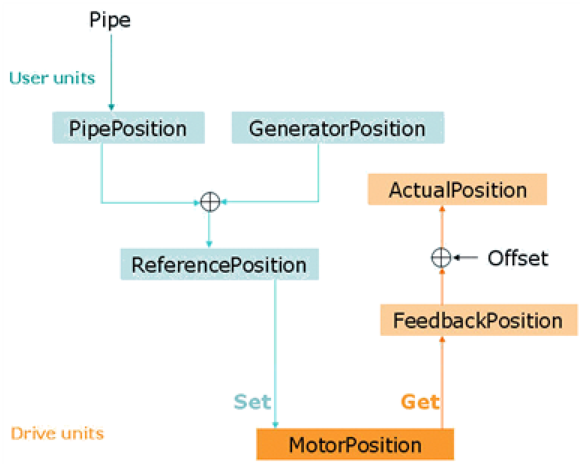
Associated data on Positions

The following data are illustrated in the figure below

- **PipePosition:** input position of the Axis issued by the upstream **pipe** and sent to the motion bus (equivalent to the output position of the convertor block)

- **GeneratorPosition:** position profile **generated by the axis block** and sent to the motion bus (it is the summation of all motion commanded to the axis, except for the changes in PipePosition)
- **ReferencePosition:** output position sent to the motion bus
- **ActualPosition:** real position (taking Offset into account) provided by the drive through the motion bus.
  The ActualPos is calculated by adding offsets to the Feedback position:

  ```
  ActualPos = FeedbackPos + ZeroOffset + Actu-
  alOffset + PipeOffset
  ```
- **FeedbackPosition:** absolute position provided by the drive through the motion bus



4.2.5 PLCopen®

The Kollmorgen Automation Suite supports the International PLCopen® motion standard.

The PLCopen international standard was created to obtain uniformity of motion function blocks and motion startup between machine control products. The PLCopen function blocks for Motion Control, is based on IEC 61131-3 "Function Blocks" (see page 84) concept with the following factors in consideration:

- Simplicity - ease of use for the application program builder and installation & maintenance
- Efficiency - in the number of function blocks, for efficiency in design (and understanding)
- Consistency - conforming to the IEC 61131-3 standard
- Universality - hardware independent
- Flexibility - future extensions / range of applications
- Completeness

KAS supports PLCopen motion in the following program formats: FFLD, SFC, ST,IL and FBD. PLCopen blocks in KAS start with "MC" (example: MC_MoveAbsolute). MC blocks are an alternative to using the ML Motion function blocks (example: MLAxisAbs) and associated Pipe Networks in many applications. Using MC Motion function blocks does not require a separate motion editor. Users who are familiar with PLCopen are automatically familiar with PLCopen inside the KAS IDE.

4.2.5.1 PLCopen Function Blocks

The following function block (FB) library is designed for the purpose of controlling one or more servo axes using the IEC 61131-3 PLCopen standard (for more details on FB, refer to "Function Blocks" (see page 84)).

To offer flexibility, ease of use and reusability, the library consists of command-oriented function blocks that have a reference to the axis, e.g. the abstract data type **Axis**.

The PLCopen Library contains function blocks for:

- **Control**: function blocks to define and initialize motion, control power, and reset errors
- **I/O**: function blocks to control interaction with Digital I/O and Touch Probe and trigger registration functionality
- **Info**: function blocks to provide information on motion, position, status, and the ability to read and write other drive parameters
- **PLCopen Motion**: function blocks to execute different types of motion
- **Profile**: function blocks for master/slave motion
- **Reference**: function blocks to reset position
- **Registration:** function blocks to perform registration

MC_MoveAbsolute performs a single-axis move to a specified endpoint position.

MC_MoveRelative performs a single-axis move of a specified distance relative to the actual position at the time of the start of execution.

MC_MoveAdditive commands a controlled motion of a specified relative distance. Can also be used to interrupt a motion currently being performed. In this case the MotionAdditive FB causes the speed, acceleration, and deceleration of the motion already running to be changed to the parameters specified in the MC_MoveAdditive FB. If the MC_MoveAdditive FB is activated in Continuous Mode, the specified relative distance is added to the actual position (at the time of execution).

MC_MoveSuperimposed commands a controlled motion of a specified relative distance additional to an existing motion. The existing Motion is not interrupted, but is superimposed by the additional motion.

MC_MoveVelocity commands a never-ending controlled motion (jog) at a specified velocity.

MC_Stop commands a controlled motion stop and transfers the axis to the "Stopping" state. It aborts any ongoing function block execution. When the Done output is set, the state transfers to StandStill. While the axis is in Stopping state, no other FB can perform any motion on the same axis.

MC_Power controls the power stage: enable(on) and disable (off).

MC_ReadStatus returns Axis status details with respect to the motion currently in progress.

MC_ReadStatus indicates Drive-related errors.

MC_ResetError makes the transition from the state ErrorStop to StandStill by resetting all internal axis-related errors and clearing pending commands – it does not affect the output of the FB instances.

MC_ReadParameter & MC_ReadBoolParameter return the value of a Drive parameter. The returned value has to be converted to Real if required. If not possible, the vendor has to provide a supplier-dependent FB for it.

MC_WriteParameter & MC_WriteBoolParameter modify the value of a Drive parameter.

MC_ReadActualPosition returns the value of the actual position.

MC_MachRegist performs Mark-to-Machine registration

MC_MarkRegist performs Mark-to-Mark registration

MC_StopRegist turns off registration

MC_CamTblSelect selects the CAM tables by setting the pointers to the relevant tables.

MC_CamIn engages the CAM.

MC_CamOut disengages the slave from the master axis immediately in a cam block.

MC_GearIn commands a ratio between the VELOCITY of the slave and master axis.

MC_GearOut disengages the slave from the master axis.

MC_AddSuperAxis adds a superimposed axis to a specified axis.

MC_RemoveSuperAxis removes a superimposed axis from an axis.

4.2.5.2 PLCopen Function Blocks - Overview

Queuing

A queuing mechanism is provided for all PLCopen motion function blocks including single-axis and master/slave moves. This mechanism allows the application to queue a next move while the active move is executing. The buffer modes, described below, define the transition from the active move to the next move.

Buffer Modes

Some of the FBs have an input called BufferMode. With this input, the FB can either work in a Non-buffered mode (default behavior) or in a Buffered mode. The difference between those modes is when they start their action:

- A command in a non-buffered mode acts immediately, even if this interrupts another motion
- A command in a buffered mode waits untill the current FB sets its **Done** output (or **InPosition**, or **InVelocity**,..).

There are six buffer modes that can be specified at the BufferMode input of the function blocks.

Buffer mode	Value	Short name	Description
MC_ BUFFER_ MODE_ ABORT	**0**	Abort	A move that specifies Abort aborts the active move, removes the next move from the queue, and immediately becomes the active move and begins executing

Buffer mode	Value	Short name	Description
MC_ BUFFER_ MODE_ BUFFERED	**1**	Buffer	One of three events can happen with a move that specifies Buffer: • Case 1. If there is no active move, this move immediately becomes the active move and begin executing. • Case 2. If there is an active move but no next move queued, this move is queued as the next move, and begins executing when the active move has completed and decelerated to zero velocity. • Case 3. If there is an active move and a queued next move, this move does not execute but returns the error "queue full" at the ErrorID output.
MC_ BUFFER_ MODE_ BLENDING_ PREVIOUS	**2**	Blend to Previous	A move specifying Blend-to-Active behaves the same as Buffer in cases 1 and 3. In case 2, this move is queued as the next move. The active move stays at its programmed velocity to its endpoint. When the active move reaches its endpoint, this move becomes active and begins to accelerate or decelerate to its programmed velocity
MC_ BUFFER_ MODE_ BLENDING_ NEXT	**3**	Blend to Next	A move specifying Blend-to-Next behaves the same as Buffer in cases 1 and 3. In case 2, this move is queued as the next move. When the expected time is reached, the active move begins to accelerate or decelerate so that it reaches this move's programmed velocity at the time the active move reaches its endpoint
MC_ BUFFER_ MODE_ BLENDING_ LOW	**4**	Blend to Low	A move specifying Blend-to-Low behaves like Blend-to-Active if the active move's velocity is lower than this move's velocity. It behaves like Blend-to-Next if this move's velocity is lower than the active move's velocity
MC_ BUFFER_ MODE_ BLENDING_ HIGH	**5**	Blend to High	A move specifying Blend-to-High behaves like Blend-to-Active if the active move's velocity is higher than this move's velocity. It behaves like Blend-to-Next if this move's velocity is higher than the active move's velocity.

S-curve and Trapezoidal Acceleration/Deceleration

S-curve

If the Jerk input of a motion function block is non-zero, S-curve acceleration/deceleration is used. The Acceleration input specifies the maximum acceleration/deceleration reached during changes in velocity. The Deceleration input is unused. The Jerk input specifies the constant rate of change of acceleration and deceleration used to cause a smooth transition to and from maximum acceleration/deceleration.

The "Figure 3-44: Small Jerk Acceleration " on page 138 below is a velocity plot of the acceleration of a move when Jerk is a small value. The smaller the Jerk value, the more gradual the rate of change of acceleration/deceleration when transitioning from one velocity to another.

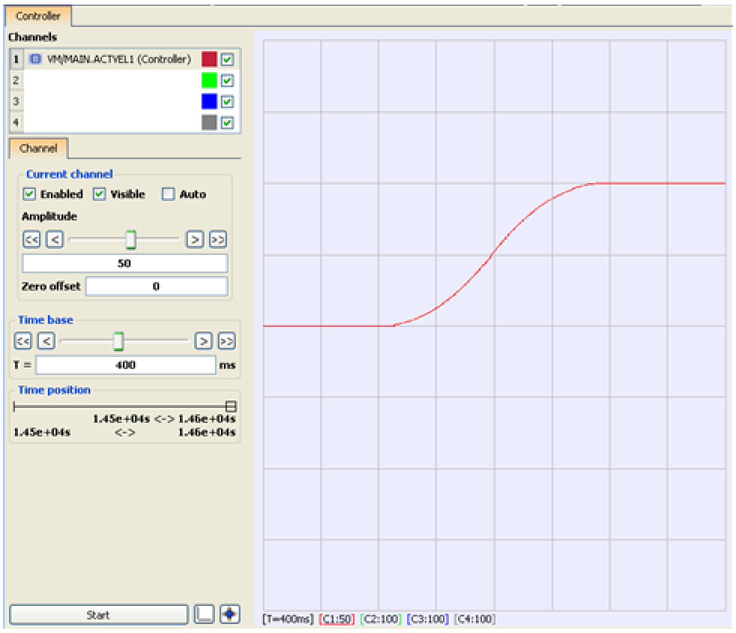


Figure 3-44: Small Jerk Acceleration

The "Figure 3-45: Large Jerk Acceleration " on page 138 below is a velocity plot of the acceleration of a move when Jerk is a large value. The larger the Jerk value, the more abrupt the rate of change of acceleration/deceleration when transitioning from one velocity to another.

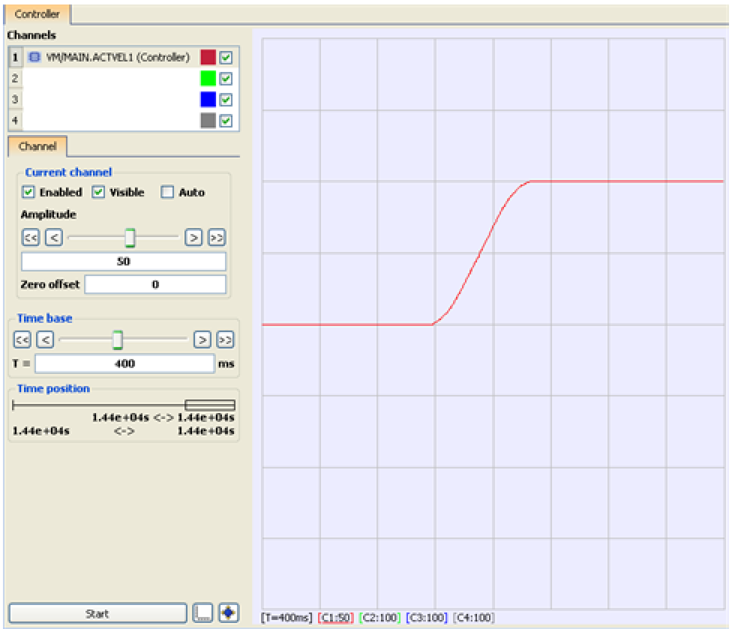


Figure 3-45: Large Jerk Acceleration

Trapezoidal

If the Jerk input of a motion function block is zero, trapezoidal acceleration/deceleration is used. The Acceleration input specifies the linear acceleration rate. The Deceleration input specifies the linear deceleration rate.

The "Figure 3-46: Trapezoidal Acceleration " on page 139 below is a velocity plot of the acceleration of a move when trapezoidal acceleration is used (Jerk = 0).

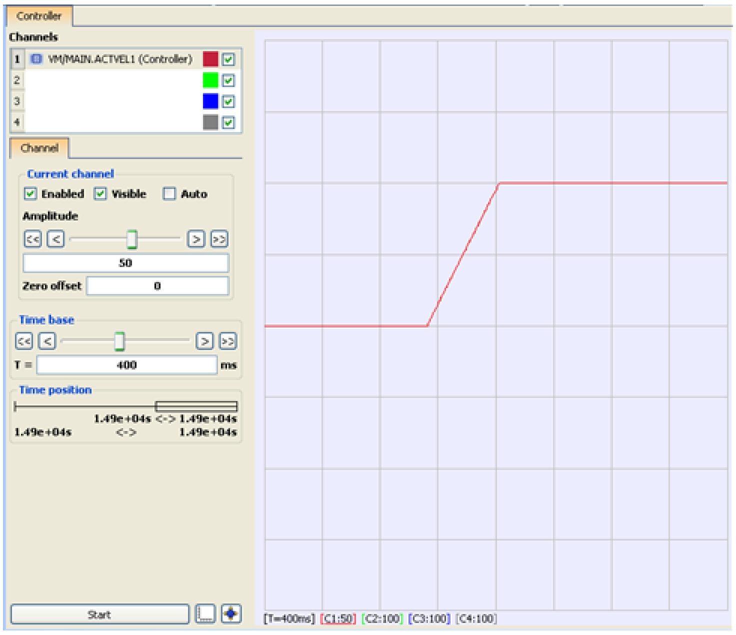


Figure 3-46: Trapezoidal Acceleration

Selection of Acceleration and Jerk Parameters for Function Blocks

Definition

Acceleration is the first derivative of velocity, or the rate of change of velocity. The Acceleration rate therefore specifies how quickly an axis may change its velocity.

Jerk is the second derivative of velocity, or the rate change of acceleration. The Jerk rate therefore specifies how quickly an axis may change its acceleration. Jerk therefore controls how abrupt the axis begins and ends the acceleration and deceleration curves.

See also "Motion Profile" (see page 93).

Rules

The amount of time an axis takes to change its velocity from one velocity to another is a function of both Acceleration and Jerk. The larger the values of acceleration and jerk, the more quickly an axis will attain its programmed velocity. The following are generalizations that can be made about acceleration, jerk and their relationships to each other.

- The higher the acceleration rate, the faster the axis will obtain programmed velocity
- The higher the jerk rate, the more responsive the axis will be to changes in command
- Excessive jerk typically, more noticeably contributes to harsh acceleration than excessive acceleration
- Too low of a jerk value contributes to slow axis responsiveness to changing commands
- Lower jerks tend to soften the beginning and end of acceleration, while higher jerks sharpen the beginning and end of acceleration
- Typically, Jerk > Acceleration, Acceleration > Velocity

Methods

There are several methodologies to determine proper acceleration and jerk values. These methodologies allow you to calculate parameters given different desired profiles. Once parameters are calculated, you can then modify them as desired to obtain the results you want. Acceleration and Jerk values are subject to the limits of ratios as explained below.

1/3,1/3,1/3 time, given velocity and time. This allows you to calculate an appropriate acceleration and jerk, if you would like an axis acceleration/deceleration profile to "jerk" or ramp acceleration up for 1/3 of the time, accelerate 1/3 of the time and ramp acceleration down 1/3 of the time. Time is the desired amount of time to reach desired velocity. Note, this is the time to change velocity, not the time to complete the move.

```
Acceleration = (3 * Velocity)/(2 * time)

Jerk = 3* Acceleration / time
```

1/3,1/3,1/3 velocity, given velocity and time. This allows you to calculate an appropriate acceleration and jerk, if you would like an axis acceleration/deceleration profile to "jerk" or ramp acceleration up for 1/3 of the velocity change, accelerate 1/3 of the velocity change and ramp acceleration down 1/3 of the velocity change. Where velocity is the desired velocity change, and time is the desired amount of time to reach the desired velocity change. Note, this is the time to change velocity, not the time to complete the move.

```
Acceleration = (5 * Velocity) / (3 * time)

Jerk = (3 * Acceleration ^2) / (2 * velocity)
```

Calculate Jerk, given Velocity, acceleration and time. If you already know the maximum acceleration of the axis, and want to simply calculate a Jerk given the velocity and time, you can use the following equation. Note, this is the time to change velocity, not the time to complete the move.

```
Jerk = (2 * Acceleration) / ( time – ( velocity
/ (2 * acceleration)))
```

Limitations on Acceleration and Jerk

The ratios of Acceleration to Jerk and Velocity to Jerk are limited on most function blocks.

- The ratio of Velocity to Acceleration must be less than **20**. A value of 20 suggests a time to accelerate to velocity of approximately 20 seconds, assuming infinite jerk. As jerk is decreased, this acceleration time would be increased.
- The ratio of Acceleration to Jerk must be less than **2**. A value of 2 suggests the time to jerk to the acceleration rate is approximately 2 seconds.

Profile Generator

Each servo axis has three Profile Generators which has its own queue. The three Profile Generators are: Normal, Superimposed, and Phasing.

- Normal handles all single-axis and master-slave moves
- Superimposed handles MC_MoveSuperimp moves exclusively
- Phasing handles MC_Phasing phase shifts exclusively

The three Profile Generators allow these types of moves to execute simultaneously.

AXIS_REF Structure

The PLCopen specification indicates a data structure to be used for identifying the axis at a function block input. AXIS_REF contains two members:

Member	Type	Description
AXIS_NUM	UINT	The axis number
AXIS_ENGINE	DINT	Reserved. Do not modify this variable or rely on its value.

For more details on Axis Number, see page 292

You have to create and initialize this data structure in your application.

Axis Parameters

The table below is a list of parameters currently supported. These parameters are read and written by the function blocks MC_ReadParam, MC_ReadBoolPar, MC_WriteParam, and MC_WriteBoolPar.

Parameter	Name	R/WA	Description
1	Command Position	Read Only	Axis command position – includes any command deltas from superimposed axes, user units
10	Actual Velocity	Read Only	Axis actual velocity, User unit/sec
11	Command Velocity	Read Only	Axis command velocity – includes any command deltas from superimposed axes, User unit/sec
1000	Phase Shift	Read Only	The amount of phase shift applied by MC_Phasing, user units
1001	Superimposed Distance	Read Only	The cumulative distance traveled via MC_MoveSuperimp moves, user units
1002	Master Offset	Read / Write	Write: the amount to increment the master offset for an active master/slave move, user units. Read: the amount of master offset applied, user units.

Parameter	Name	R/WA	Description
1003	Slave Offset	Read / Write	Write: the amount to increment the slave offset for an active master/slave move, user units. Read: the amount of slave offset applied, user units.
1004	Active Move Type	Read Only	The active move type (see table below)
1005	Next Move Type	Read Only	The queued (next) move type (see table below)
1006	Position Error	Read Only	Position error in user units
1007	Raw Feedback	Read Only	Raw Feedback position in user units
1008	Rollover	Read / Write	The axis rollover position in user units
1009	Velocity Compensation Factor	Read / Write	The factor used to multiply the velocity compensation value to account for the number of updates of delay in transmission of the feedback value from the drive to the control
1010	Velocity Compensation Filter	Read / Write	The number of updates in which to apply a change in velocity compensation
1011	Axis In-Position	Read Only	True if the axis has no active or next move queued, the command delta is 0, and the actual position is within the in-position bandwidth of the command position. False otherwise, Boolean.

Parameter	Name	R/WA	Description
1012	Axis In-Position Bandwidth	Read / Write	The bandwidth about the command position to determine the state of the in-position flag. User units
1013	Drive Warning	Read Only	(Boolean) Drive Warning Status
1014	Drive Status	Read Only	Drive Status Word (Similar to MLAxisStatus)
1015	User Units Per Rev	Read Only	User units per motor revolution (UU/FBU Ratio). See "About the User Units to Feedback Units Ratio" (see page 145) below.
1016	Actual Torque	Read Only	The actual torque being delivered by the drive, expressed in thousandths of max torque
1017	Drive Address	Read Only	Drive address value to be used in EtherCAT fieldbus functions as drive address. Before using in fieldbus functions, this value needs to be converted to integer by using a convert any to DINT function
1018	Sensor Delay	Read / Write	Compensation for Physical sensor delay for MC_TouchProbe (FunctionBlock),, in microseconds.
1019	Interpolated Command Position	Read Only	Command position solely from this axis's interpolator (in user units). This value does not include any command deltas from other axes that are currently superimposed upon it.

Parameter	Name	R/WA	Description
1020	Interpolated Command Velocity	Read Only	Command velocity solely from this axis's interpolator (in user units). This value does not include any command deltas from other axes that are currently superimposed upon it.
1021	Registration Compensation	Read Only	The latest calculated registration compensation value. This value is updated each time a good registration mark is encountered. This value is in User Units.
1022	Distance Between the Last Two Good Registration Marks	Read Only	Distance between the last two good registration marks. This value is in User Units.
1023	Number of Consecutive Good Registration Marks	Read / Write	Number of consecutive good registration marks. This value is incremented each time a good registration mark is encountered and automatically zeroed when a bad registration mark is encountered. The ability to write this parameter is provided to allow the application to zero this value.
1024	Number of Consecutive Bad Registration Marks	Read / Write	Number of consecutive bad registration marks. This value is incremented each time a bad registration mark is encountered and automatically zeroed when a good registration mark is encountered. The ability to write this parameter is provided to allow the application to zero this value.
1025	Good Registration Mark Occurred	Read Only	(Boolean) True indicates that a good registration mark was encountered. This Boolean will be automatically reset after it has been read.

Parameter	Name	R/WA	Description
1026	Bad Registration Mark Occurred	Read Only	(Boolean) True indicates that a bad registration mark was encountered. This Boolean will be automatically reset after it has been read.
1027	Fast Input Occurred	Read Only	(Boolean) True indicates that a fast input occurred. This Boolean will be automatically reset after it has been read.
1028	Feedback Units Per Rev	Read Only	Feedback units per motor revolution (UU/FBU Ratio). See "About the User Units to Feedback Units Ratio" (see page 145) below.
1029	Coordinated Motion Applied Command Position	Read Only	Amount of motion actually applied to the PLCopen axis drive by Coordinated Motion commands.
1030	Coordinated Motion Command Position	Read Only	Amount of motion requested of a PLCopen axis by the Coordinated Motion commands.

TIP

About the User Units to Feedback Units Ratio

Parameters 1015 and 1028 are set during the MC_CreateAxis (Function) function block execution. These two parameters work together to form the User Units to Feedback Units Ratio (UU/FBU Ratio). The drive interface units are fixed by the drive and define the drive units per revolution, which is used to command the drive per the ratio.

Example

For an AKD drive where the drive interface units are set to 1048576 units per revolution:

- A ratio of 360 UU / 1048576 FBU will generate 360 UU per revolution of the drive motor.
- A ratio of 720 UU / 1048576 FBU will generate 720 UU per revolution of the drive motor.
- A ratio of 720 UU / 2097152 FBU will generate 360 UU per revolution of the drive motor.
- A ratio of 360 UU / 2097152 FBU will generate 180 UU per revolution of the drive motor.

As noted in MC_CreateAxis (Function), the Feedback Units per Revolution term must be a power of 2.

Axis Positions Data

The following position data are related to PLCopen Axis

- **ActualPosition:** is the position of the axis read from the drive interface which is read from the feedback device

feedback device	→ *drive interface*	→ **ActualPosition**

- **CommandPosition:** is the command position that is sent to the drive interface to command the axis.
  This position is tied to the Status output of the MC_Power function block:
  - When the **Status = 1** the command position is a combination of the Normal, Superimposed and Phasing commands
  - When the **Status = 0** the command position tracks the Actual Position

NormalCmdPos + SuperimposedCmdPos + PhaseCmdPos+ the accumulated command deltas of all superimposed axes	→ **CommandPosition**

- **NormalCmdPos:** is the command position generated by the Normal Interpolator when interpolating a single axis move or a slave move

MC_MoveAbsolute MC_MoveRelative MC_MoveAdditive MC_MoveVelocity MC_Halt MC_CamOut MC_GearOut MC_Stop	→ *Normal Interpolator*	→ **NormalCmdPos**

When interpolating a slave move, the PhaseCmdPos is incorporated in the generation of the NormalCmdPos.

MC_GearIn MC_GearInPos MC_CamIn	→ (MasterActualPos + PhaseCmdPos) * *(gear ratio or cam profile)*	→ **NormalCmdPos**

- **SuperimposedCmdPos:** is the command position generated by the Superimposed Interpolator when interpolating a MC_Superimp move

MC_Superimp	→ *Superimposed Interpolator*	→ **SuperimposedCmdPos**

- **PhaseCmdPos:** is the command position generated by the Phasing Interpolator when interpolating a MC_Phasing master phase shift

MC_Phasing	→ *Phasing Interpolator*	→ **PhaseCmdPos**

Possible Move Types

MoveID	Description	Related FB
0	No move	
1	Distance move	MC_MoveRelative, MC_MoveSuperimp and MC_Phasing
2	Position move	MC_MoveAbsolute
3	Velocity move	MC_MoveVelocity
4	Halt move	MC_Halt

MoveID	Description	Related FB
5	Gear-in move	MC_GearIn
6	Gear-out move	MC_GearOut
7	Reference move	MC_Reference
8	Stop move	MC_Stop
9	Gear-in pos. move	MC_GearInPos
10	Cam profile move	MC_CamIn
11	Cam-out move	MC_CamOut

Rollover

The Rollover Position is specified in user units in the PLCopen Axis Data dialog. When this value is non-zero, the axis' position is reset to zero when it reaches the rollover position.

For example, if the rollover position is 360 and the axis is traveling in the positive direction, the axis position counts up until it reaches 360 where it resets to 0 and then continues counting up from there.
If the axis is traveling in the negative direction, the axis position counts down until it reaches 0, where it resets to 360 and then continues counting down from there.

Refer to MC_MoveAbsolute's's description for an explanation of its operation when Rollover Position is nonzero.

When the Rollover Position is zero, rollover is not in effect and the axis position continues to count up when traveling in the positive direction and count down when traveling in the negative direction.

PLCopen Function Block ErrorID Output

These are the possible errors that could be returned at the ErrorID output of the function blocks.

ErrorID	Description
0	no error
1	queue full
2	abort mode required
3	invalid axis
4	One of four possible scenarios: • invalid master axis • master axis and slave axis are the same • master axis is currently slaved to the specified slave axis • master axis and slave axis do not have the same update rate.
5	invalid parameter number
6	invalid move
7	invalid override
8	buffer mode required
9	invalid parameter data
10	move cannot be executed because an axis error exists, the axis is in the stopping state or the axis is disabled.
11	invalid buffer mode
12	move was aborted due to an E-stop

ErrorID	Description
13	move was aborted due to a controlled stop
14	invalid start mode
15	invalid cam profile
16	invalid slave count
17	input value is out of range
18	cannot access time stamp or latched position data
19	data not available For example, if a MC_ReadParm of FollowingError (1006) is programmed on a simulator axis for which no following error is available, an error 19 (data not available) is returned.
20	Motion engine is not running
21	Invalid ratio of velocity to acceleration, or acceleration to jerk. See more details here
22	Too many profiles – the number of selected profiles is limited to 256.
23	Internal Error
24	Object already exists
25	Block needs to be called between MLMotionInit and MLMotionStart
26	The axis is owned by another group
27	The axis is already present in the group
28	No axes are mapped to the group
29	Invalid name. A name cannot be an empty string
30	Name is already used
31	The number of axes is too small
32	The number of axes is too large
33	Invalid update rate
34	Invalid axis group
35	Invalid kinematic parameter (position, speed, velocity, acceleration, deceleration, or jerk)
36	Invalid floating point number
37	Invalid object index
38	Invalid number of positions in array
39	Position is not a valid number
40	Invalid coordinate system
41	Axis is not initialized
42	Array is not the correct size
43	A move cannot be made while the group is in a stopping state.
44	The axes group is not in the Disabled or Standby state
45	The axes group is in the Disabled state
46	The axis is not in this axes group
47	Duplicate points specified
48	Invalid radius specified
49	Colinear points were specified for the border points of a circle
50	Cannot construct a circle with specified parameters
51	Invalid circle mode

ErrorID	Description
52	Invalid path choice for circular move
53	Invalid transition mode
54	The axis group does not have exactly two axes
55	Invalid engine type for axis
56	An axis or its associated drive is in an error state.
57	Cannot execute function because both queues are not empty.
58	Lines are parallel
59	Not enough room for transitions
60	Abort mode not allowed for Circular Absolute moves
61	Invalid transition parameters
62	Transition not allowed with Abort Move.
63	The axes group is not in the Standby state.
64	The maximum number of axes has been created.
65	The axis is not powered on.
66	Error in data reported from drive.
67	The axes group is not in the ErrorStop state.
68	The axes group is still in Error State.
69	The axes group is not in Standby or ErrorStop state.

4.2.5.3 PLCopen Function Blocks - General Rules

The general rules for PLCopen are:

- "Input parameters" (see page 149)
- "Missing input parameters" (see page 150)
- "Output Exclusivity" (see page 150)
- "Output Status" (see page 150)
- "Sign Rules" (see page 150)
- "Error Handling Behavior" (see page 150)
- "Behavior of Done Output" (see page 150)
- "Behavior of CommandAborted Output" (see page 151)
- "Behavior of Busy Output" (see page 151)
- "Inputs Exceed Application Limits" (see page 151)
- "Output 'Active'" (see page 151)

Input parameters

Unless specified otherwise in the function block's description, the input parameters are read with the rising edge of the Execute input.

The input parameters can be as follows:

- Function Blocks with **Execute**

  These FBs will be executed on the rising edge. They will continue to execute until completed, but is based on the rising edge of this input only. So once activated, this FB executes even if the input is off or on.

- Function Blocks with **Enable**

  These FBs will continuously be executed every PLC cycle, as long as the Enable remains high.

- Function with **En**

This is very similar to ENABLE on Function blocks. But, as already explained in paragraph "Difference between Functions and Function Blocks" on page 84, functions are expected to complete in one cycle.

Missing input parameters

If any input parameter of a function block is missing (**open**), the compiler generates an error.

Output Exclusivity

The outputs **Busy**, **Done**, **Error**, and **CommandAborted** are mutually exclusive: only one of them can be TRUE on one function block. If **Execute** is TRUE, one of these outputs has to be TRUE .
Only one of the outputs **Active**, **Error**, **Done** and **CommandAborted** is set at the same time.

Output Status

The **Done**, **Error**, **ErrorID** and **CommandAborted** outputs are reset with the next rising edge of **Execute**.

If an instance of a function block receives a new **Execute** before it finishes (as a series of commands on the same instance), the function block does not return any feedback, like **Done** or **CommandAborted**, for the previous action.

Sign Rules

Velocity, **Acceleration**, **Deceleration** and **Jerk** are always positive values. **Position** and **Distance** can be positive or negative.

Error Handling Behavior

Two outputs deal with errors that can occur while executing a function block. These outputs are defined as follows:

- **Error**: the rising edge of **Error** informs you that an error occurred during the execution of the function block
- **ErrorID**: Error number.

Done, **InVelocity**, **InGear**, and **InSync** mean successful completion so these signals are logically exclusive to **Error**.
Instance errors do not always result in an axis error. Some bring the axis to **StandStill**(.

Behavior of Done Output

The **Done** output (as well as **InGear**, **InSync**) is set when the commanded action has been completed successfully.
With multiple function blocks working on the same axis in a sequence, the following applies: when one movement on an axis is interrupted with another movement on the same axis without having reached the final goal, **Done** of the first function block is not set.

When a motion command is executed, there are three possible outcomes:

1. It completes successfully. At that time, the **Done** output goes high.
2. It is aborted prior to completing by a subsequent motion command. At that time, the **CommandAborted** output goes high.
3. It encounters an error prior to completing or an invalid input is specified. At that time, the **Error** output goes high.

These outputs stays in this state until that motion function block is executed again. At that time, the **Done**, **CommandAborted** and **Error** outputs goes low; and the **Busy** output goes high, provided all the inputs are valid.

Behavior of CommandAborted Output

CommandAborted is set when a commanded motion is interrupted by another motion command.
The reset-behavior of **CommandAborted** is like that of **Done**. When **CommandAborted** occurs, the other output signals such as **InVelocity** are reset.

Behavior of Busy Output

The **Busy** output indicates that the function block is still working, with new output values to be expected.
Busy is SET at the rising edge of **Execute** and RESET when one of the outputs **Done**, **Aborted** or **Error** is set. It is recommended that this function block is kept in the active loop of the application program for at least as long as **Busy** is True, because the outputs can still change.
For one axis, several function blocks can be busy, but only one can be active at a time. Exceptions are **MC_SuperImposed** and **MC_Phasing**, where more than one function block related to one axis can be active.

Inputs Exceed Application Limits

If a function block is commanded with parameters which result in a violation of application limits, the instance of the function block generates an error.

Output 'Active'

The **Active** output is set at the moment the function block takes control of the motion of the respective axis.

Coordinated Motion

Use the following table to determine the state of the Busy and Active outputs when transitioning from one move to the next.

Function Block Output	First Move	Second Move
Busy	False	True
Active	False	False
Done	True[1]	False

List of Input Parameters

The input parameters are listed as follows:

- Function Blocks with **Execute**
- Function Blocks with **Enable**
- Function with **En**

List of PLCopen function blocks with **Execute**

These FBs will be executed on the rising edge. They will continue to execute until completed, but is based on the rising edge of this input only. So once activated, this FB executes even if the input is off or on.

[1]This output will be false if the move is interrupted by a MC_GrpHalt, MC_GrpStop, or if the second move's buffer mode is "Aborting".

Function Block	Description
MC_MoveAbsolute MC_MoveRelative MC_MoveAdditive MC_MoveSuperimp MC_MoveVelocity MC_Halt MC_CamIn MC_CamOut MC_GearIn MC_GearOut MC_GearInPos	A positive transition of this input requests to queue the move
MC_Phasing	A positive transition of this input requests to queue the phase shift move
MC_SyncSlaves MC_TouchProbe MC_AbortTrigger MC_SetPosition	A positive transition of this input causes this function block to execute
MC_WriteBoolPar MC_WriteParam	A positive transition of this input writes the specified parameter
MC_Reference	A positive transition of this input requests to queue the reference move and arm the reference trigger event(s)
MC_CamTblSelect	A positive transition of this input reads and initializes the specified profile
MC_Stop	A positive transition of this input initiates a stop move. While this input is held high, no other move can be queued for this axis

List of PLCopen function blocks with **Enable**

These FBs will continuously be executed every PLC cycle, as long as the Enable remains high.

Function Block	Description
MC_ReadBoolPar MC_ReadParam	When this input is high, the specified parameter is read
MC_SetOverride	When this input is high, the override factors is written
MC_ReadActPos	When this input is high, the axis's actual position is returned
MC_ReadActVel	When this input is high, the axis's actual velocity is returned
MC_ReadAxisErr	When this input is high, the axis's error status is returned
MC_ReadStatus	When this input is high, the function block outputs is updated
MC_Power	If this input is high and the drive is currently disabled, this function block requests to close the servo loop and enable the drive. If this input is low and the drive is currently enabled, this function block requests to open the servo loop and disable the drive

List of PLCopen functions with input parameter **En**

This is very similar to ENABLE on Function blocks. But, as already explained in paragraph "Difference between Functions and Function Blocks" on page 84, functions are expected to complete in one cycle.

Function	Description
MC_CreateAxis	When this input is high, a PLCopen axis is created
MC_InitAxis	When this input is high, the specified axis is initialized
MC_EStop	When this input is high, an E-stop is generated for the specified axis
MC_ResetError	When this input is high, the specified axis's errors is reset
MC_AddSuperAxis	When this input is high, the specified axis is added to the superimposed axis list of the receiving axis.
MC_RemSuperAxis	When this input is high, the specified axis is removed from the superimposed axis list for the specified receiving axis.

4.2.5.4 State machine

The following diagram normatively defines according to PLCopen the behavior of the axis at a high-level when multiple motion control function blocks are "simultaneously" activated. This combination of motion profiles is useful in building a more complicated profile or in handling exceptions within a program. In real implementations there can be additional states defined at a lower level.

The basic rule is that motion commands are always taken sequentially, even if the PLC has the capability of real parallel processing. These commands act on the state diagram of the axis.

The axis is always in one of the defined states (see diagram below). A change of state is reflected immediately when issuing the corresponding motion command (please note that the response time of 'immediately' is system dependent).

There are seven states defined:

1. Stand Still
2. Homing
3. Discrete Motion
4. Continuous Motion
5. Synchronized Motion
6. Stopping
7. Error Stop

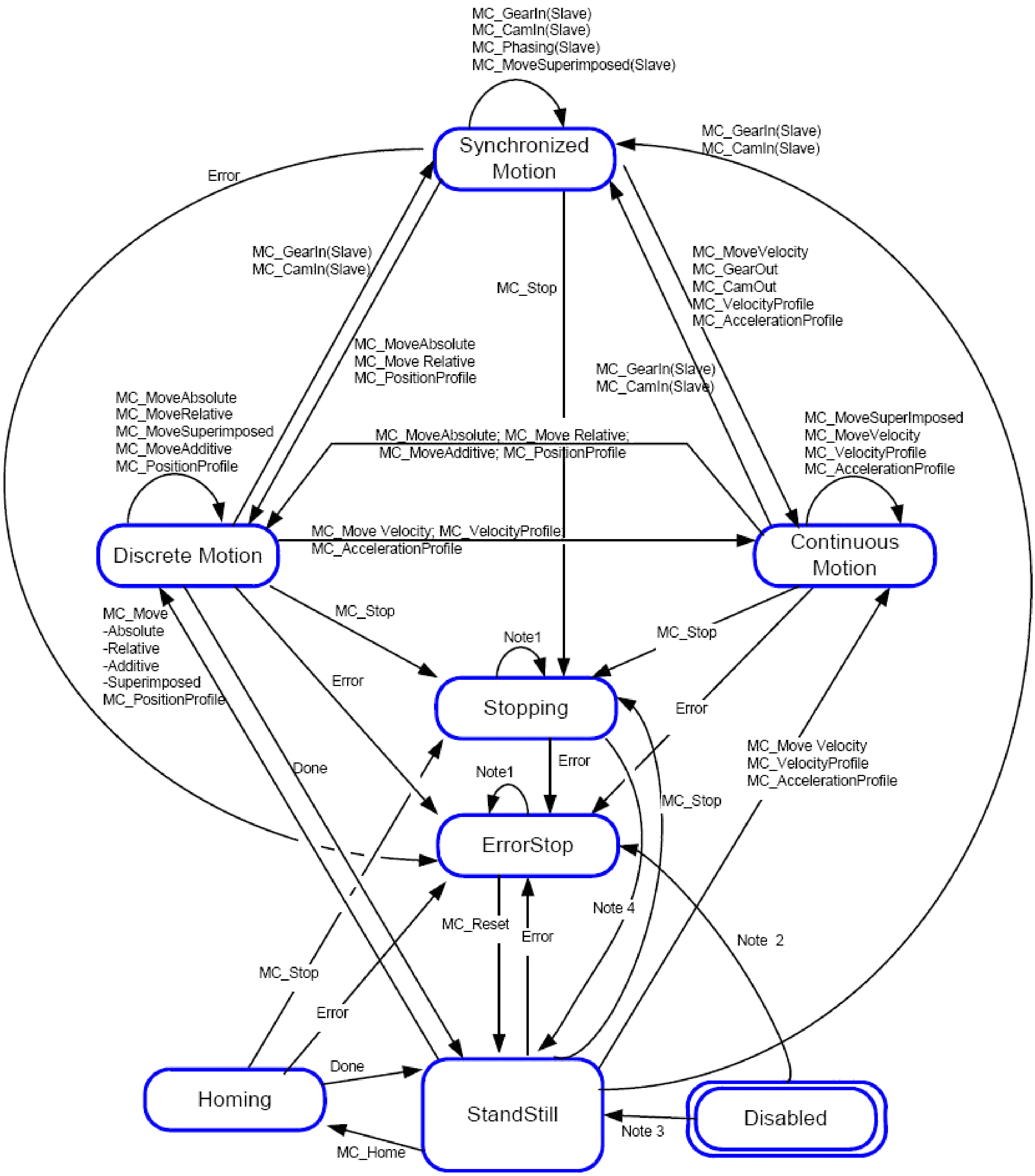


Figure 3-47: Motion State Machine (PLCopen)

NOTE

Note 1: In this state **ErrorStop** or **Stopping**, all function blocks can be called, although they are not executed, except MC_Reset and Error which generate the transition to **StandStill** or **ErrorStop** respectively
Note 2:MC_Power FB is called with Enable=TRUE and there is an error in the

> **NOTE**
> Axis
> **Note 3**: MC_Power FB is called with Enable=TRUE and there is **no** error in the Axis
> **Note 4**: MC_Stop.Done **and not** MC_Stop.Execute

A normal procedure would start in **StandStill**. In this state the power can be switched on per axis (via the command MC_Power). Also, you can access the **Homing** state (via the issue of the command Home per axis), which after normal completion returns to **StandStill**. From here you can transfer an axis to either **Discrete Motion** or **Continuous Motion**. Via the **Stopping** state you can return to **StandStill**. **ErrorStop** is a state to which the axis transfers in case of error. Via a (manual) Reset command, you can return to **StandStill**, from which the machine can be moved to an operational state again.

Please note that the States define the functionality of the function blocks.
Function Blocks which are not listed in the State Diagram do not affect the state of the axis, meaning that, whenever they are called, the state does not change. They are:

- MC_ReadStatus
- MC_ReadAxisErr
- MC_ReadParameter
- MC_ReadBoolParameter
- MC_WriteParameter
- MC_WriteBoolParameter
- MC_ReadActualPosition
- MC_CamTableSelect

State **Disabled**

The **Disabled** state describes the initial state of the axis. In this state, the movement of the axis is not influenced by the FBs. The axis feedback is operational.
If the MC_PowerFB is called with Enable=TRUE while being in **Disabled**, this either leads to **Standstill** if there is no error inside the axis, or to **ErrorStop** if an error exists.
Calling MC_Power with Enable=FALSE in any state, the axis goes to the state **Disabled**, either directly or via any other state. If a motion generating function block controls an axis while the MC_Power FB with Enable=FALSE is called, the motion generating function block is aborted (CommandAborted).

Disable means power off without error.

State **ErrorStop**

The intention of the **ErrorStop** state is that the axis goes to a stop, if possible. No further FBs are accepted until a reset has been done from the **ErrorStop** state. The transition Error refers to errors from the axis and axis control, and not from the function block instances. These axis' errors can also be reflected in the output of the function blocks "FB instances errors".
Issuing MC_Home in any other state than **StandStill** goes to **ErrorStop**, even if MC_Home is issued from the **Homing** state itself.

ErrorStop is valid as highest priority and applicable in case of an error. The axis can have either power enabled or disabled, and can be changed via MC_Power. However, as long as the error is pending the state remains **ErrorStop**.

From **StandStill** to **Stopping**

Calling the FB MC_Stop in state **StandStill** changes the state to **Stopping** and back to **Standstill** when “Execute = FALSE”. The state **Stopping** is kept as long as the input “Execute” is true. The “Done” output is set when the stop ramp is finished.

StandStill is power on without an error.

State machine for multi-axes motion control

The diagram is focused on a single-axis. The multi-axes function blocks (e.g. MC_CamIn, MC_GearIn or MC_Phasing) can be looked at, from a state diagram point of view, as multiple single-axes all in specific states. For instance, the CAM-master can be in the state **Continuous Motion**. The corresponding slave is in the state **Synchronized Motion**. Connecting a slave axis to a master axis has no influence on the master axis.

4.3 EtherCAT Motion Bus Concepts

To exchange data between the controller (master) and the devices (slaves), the KAS Runtime relies on the EtherCAT motion bus. This communication can be done in two modes: cyclic and non-cyclic (mailbox).

In **cyclic mode**, a single frame containing the data of all slaves (input and output) travels along all slaves and goes back to the master. Data is read and/or written "on the fly" by each slave.

Slave device input and output data definitions:

- Outputs are written by the master and read by the slave device
- Inputs written by the slave device and read by the master

EtherCAT Process Image

This cyclic frame is called the EtherCAT Process Image. It contains the Process Data, which is defined during network initialization.

The cyclic data is grouped in predefined blocks called Process Data Objects or "PDO" (see page 725).

NOTE PDOs contain real-time cyclic data which is deterministic. Non-cyclic data is not deterministic and is defined by Service Data Objects ("SDO" (see page 718)).

References

- EtherCAT Specification V1.0 - refer to http://www.ethercat.org (in Member Area - Downloads)
- Büttner, H.; Janssen, D.; Rostan, M. (2003), EtherCAT - the Ethernet fieldbus, (PDF), PC Control Magazine 3: 14-19

4.3.1 Functional Principle

Typical automation networks are characterized by short data-length per node, typically less than the minimum payload of an Ethernet frame. Using one frame per node per cycle leads to low bandwidth utilization and thus to poor overall network performance. EtherCAT therefore takes a different approach, called "processing on the fly" (for more details, refer to paragraph "EtherCAT Implementation" on page 163).

With EtherCAT, the Ethernet packet or frame is no longer received, and then interpreted and copied as process data at every node. Instead, the EtherCAT slave devices read the data addressed to them while the telegram passes through the device. Similarly, input data is inserted while the telegram passes through. The frames are only delayed by a fraction of a microsecond in each node, and many nodes - typically the entire network - can be addressed with just one frame.

4.3.2 EtherCAT Features

Summary

EtherCAT is characterized by outstanding performance, very simple wiring, and openness to other protocols. EtherCAT sets new standards where conventional fieldbus systems reach their limits: 1000 I/Os in 30 µs, optionally twisted-pair cable or optical fiber and, thanks to Ethernet and Internet technologies, optimum vertical integration. With EtherCAT, the costly Ethernet star topology can be replaced with a simple line structure - no expensive infrastructure components are required. Optionally, EtherCAT can also be wired in the classic way using switches, to integrate other Ethernet devices. Where other real-time Ethernet approaches require special connections in the controller, for EtherCAT, very cost-effective standard Ethernet cards suffice.

EtherCAT is versatile: Master to Slave, Slave to Slave and Master to Master Communication is supported (see figure below). Safety over EtherCAT is available. EtherCAT makes Ethernet down to the I/O level technically feasible and economically sensible. Outstanding features of this network include full Ethernet compatibility, Internet technologies (even in simple devices), maximum utilization of the large bandwidth offered by Ethernet, and outstanding real-time characteristics at low costs.

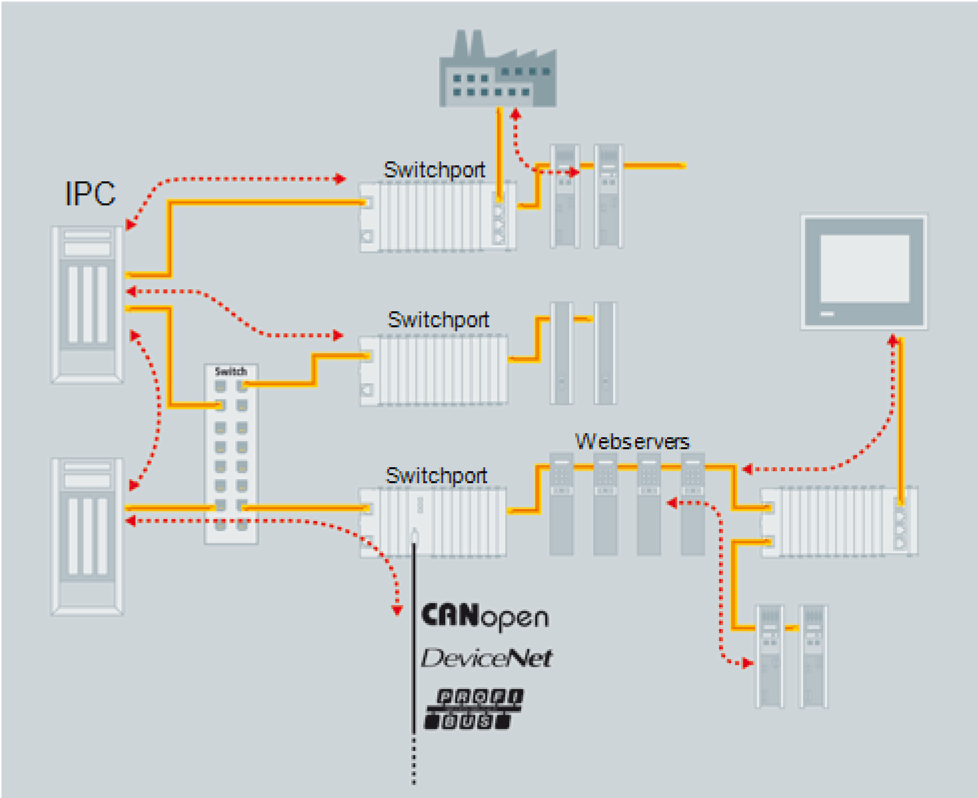


Figure 3-48: Versatile Network Architecture

4.3.2.1 Protocol

The EtherCAT protocol is optimized for process data and is transported directly within the standard IEEE 802.3 Ethernet frame using Ethertype 0x88a4. It can consist of several sub-datagrams, each serving a particular memory area of the logical process images, that can be up to 4 gigabytes in size. The data sequence is independent of the physical order of the nodes in the network, and addressing can be in any order. Broadcast, multicast and communication between slaves is possible and must be done by the master device. If IP routing is required, the EtherCAT protocol can be inserted into UDP/IP datagrams. This also enables any control with Ethernet protocol stack to address EtherCAT systems.

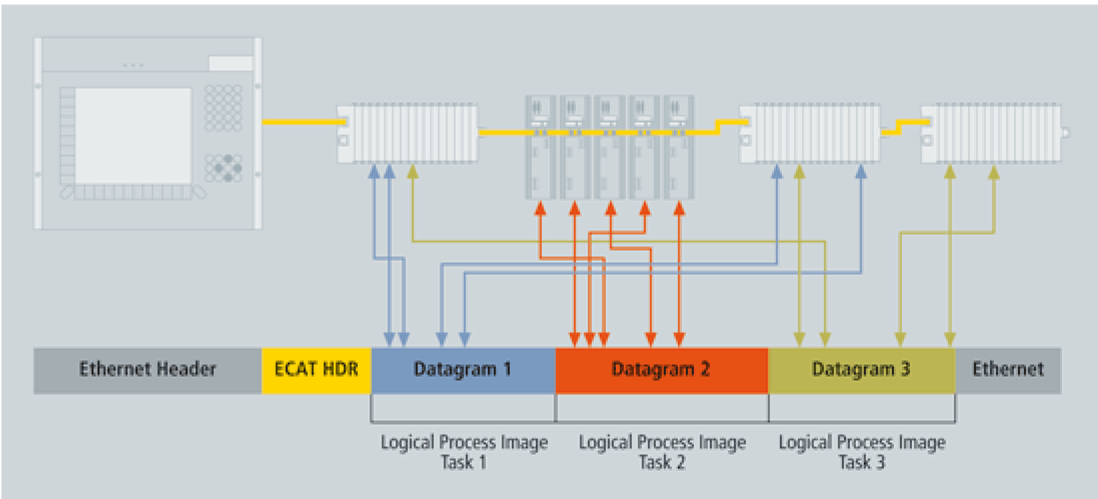


Figure 3-49: Process Data is Inserted in Telegrams

KAS supports **CANopen over EtherCAT** (CoE). It enables the advantages of EtherCAT in terms of transfer characteristics to be combined with proven, profile-specific drive functions.

KAS also uses **File Access over EtherCAT** (FoE) protocols to provide options for efficiently exchanging firmware via the bus (see "Figure 3-54: Several Device Profiles and Protocols can coexist " on page 163).

4.3.2.2 Topology

Using full-duplex Ethernet physical layers, the EtherCAT slave controllers close an open port automatically and return the Ethernet frame if no downstream device is detected. Slave devices can have several ports. Using these features, EtherCAT can support almost any physical topology, such as line, tree or star. The bus or line structure known from the fieldbuses thus also becomes available for Ethernet. The combination of line and branches or stubs is also possible: any EtherCAT device with three or more ports can act as a junction, and no additional switches are required. The classic switch-based Ethernet star topology can be used either with switches configured to forward traffic directly between ports, or with special slave devices: the switches are then located between the network master and the slave devices. The special slave device assembly (remember standard slave devices don't have a MAC address) attached to one switch port together forms an EtherCAT segment, which is either addressed via its MAC address or via port-based VLANs. Since the 100BASE-TX Ethernet physical layer is used, the distance between any two nodes can be up to 100 m (300 ft). Up to 65535 devices can be connected per segment. If an EtherCAT network is wired in ring configuration (requiring two ports on the master device), it can provide cable redundancy.

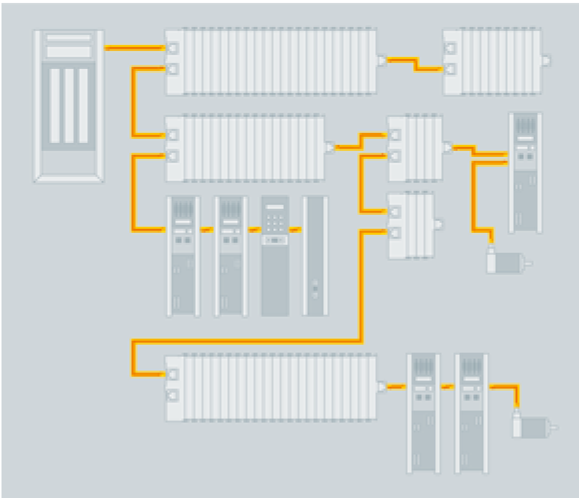


Figure 3-50: Flexible Topology: Line, Tree or Star

The topology implemented in KAS is wired in line. Consequently, as soon as the EtherCAT communication is broken, the controller is not able to communicate with any of the other network devices.

4.3.2.3 Distributed Clock (Synchronization)

A distributed clock is an EtherCAT feature that allows synchronization, with a reference clock, of all EtherCAT slaves and the master. This solves problems related to clock-shifting between the master and the devices.

This mechanism also leads to very low jitter of significantly less than 1 µs. Even if the communication cycle jitters, it is still compliant with the IEEE 1588 Precision Time Protocol standard.

Therefore, EtherCAT does not require special hardware in the master device and can be implemented in software on any standard Ethernet MAC, even without a dedicated communication coprocessor.

The typical process of establishing a distributed clock is initiated by the master by sending a broadcast to all slaves at a specific address. On reception of this message, all slaves latch the value of their internal clock twice, once when the message is received and once when it returns (remember EtherCAT has a ring topology). The master can then read all latched values and calculate the delay for each slave. This process can be repeated as many times as required to reduce jitter and to average out values. Total delays are calculated for each slave depending on their position in the slave-ring and are uploaded to an offset register. Finally the master issues a broadcast read-write on the system clock, which makes the first slave the reference clock and forcing all other slaves to set their internal clock appropriately with the now known offset.

To keep the clocks synchronized after initialization, the master or slave must regularly send out the broadcast again to counter any effects of speed difference between the internal clocks of each slave. Each slave has to adjust the speed of their internal clock or implement an internal correction mechanism whenever they have to adjust.

The system clock is specified as a 32-bit counter with a base unit of 1 ns starting at January 1st 2000, 0:00.

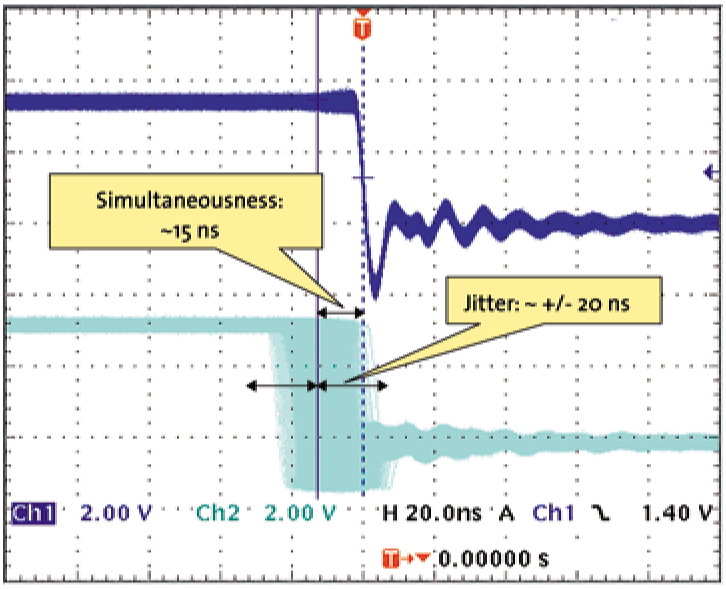


Figure 3-51: Synchronicity and Simultaneousness

Scope view of two distributed devices with 300 nodes and 120 m of cable between them.

4.3.2.4 Performance

Short cycle times can be achieved because the host microprocessors in the slave devices are not involved in the processing of the Ethernet packets to transfer the process images. All process data communication is handled by the slave controller hardware. Combined with these features, this makes EtherCAT a high-performance distributed I/O system: Process data exchange with 1000 distributed digital I/O takes about 30 µs, which is typical for a transfer of 125 byte over 100Mb/s Ethernet. Data for and from 100 servo axes can be updated with up to 10 kHz. Typical network update rates are 1-30 kHz, but EtherCAT can be used with slower cycle times, too, if the DMA load is too high on your PC.

Process Data	Update Time
256 distributed digital I/O	11 µs = 0,01 ms
1000 distributed digital I/O	30 µs
200 analog I/O (16 bit)	50µs ↔ 20 kHz
100 Servo Axis, with 8 Bytes input and output data each	100 µs
1 Fieldbus Master-Gateway (1486 Bytes Input and 1486 Bytes Output Data)	150 µs

Table 3-4: EtherCAT Performance Overview

The communication with 100 servo axes is also extremely fast: every 100µs, all axes are provided with command values and control data and report their actual position and status. The Distributed Clocks technique enables the axes to be synchronized with a deviation of significantly less than 1 microsecond. And even at this pace, there is more than sufficient bandwidth for asynchronous communications such as TCP/IP, parameter download or diagnostic data upload.

4.3.2.5 Safety over EtherCAT

The protocol enhancement called Safety over EtherCAT (FSoE) enables safety-related communication and control communication on the same network. The safety protocol is based on the application layer of EtherCAT, with no influence on the lower layers. It is certified according to IEC 61508 and meets the requirements of Safety Integrity Level (SIL) 3.

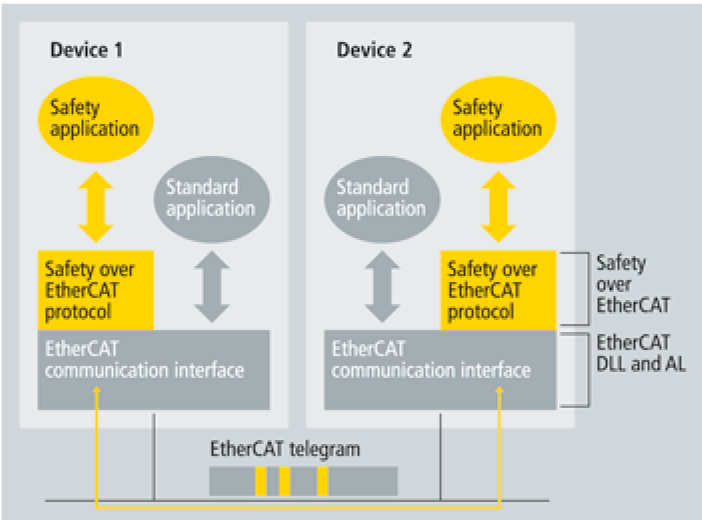


Figure 3-52: Safety over EtherCAT Software Architecture

4.3.2.6 Gateways

Gateway devices are available for the integration of existing fieldbus components (e.g., CANopen, DeviceNet, Profibus) into EtherCAT networks. Also, other Ethernet protocols can be used in conjunction with EtherCAT: the Ethernet frames are tunneled via the EtherCAT protocol, which is the standard approach for Internet applications. The EtherCAT network is fully transparent for the Ethernet device, and the real-time characteristics are not impaired, since the master dictates exactly when the tunneled transfers are to occur and how much of the 100Mb/s media the tunneled protocols can use. Therefore, all Internet technologies can also be used in the EtherCAT environment.

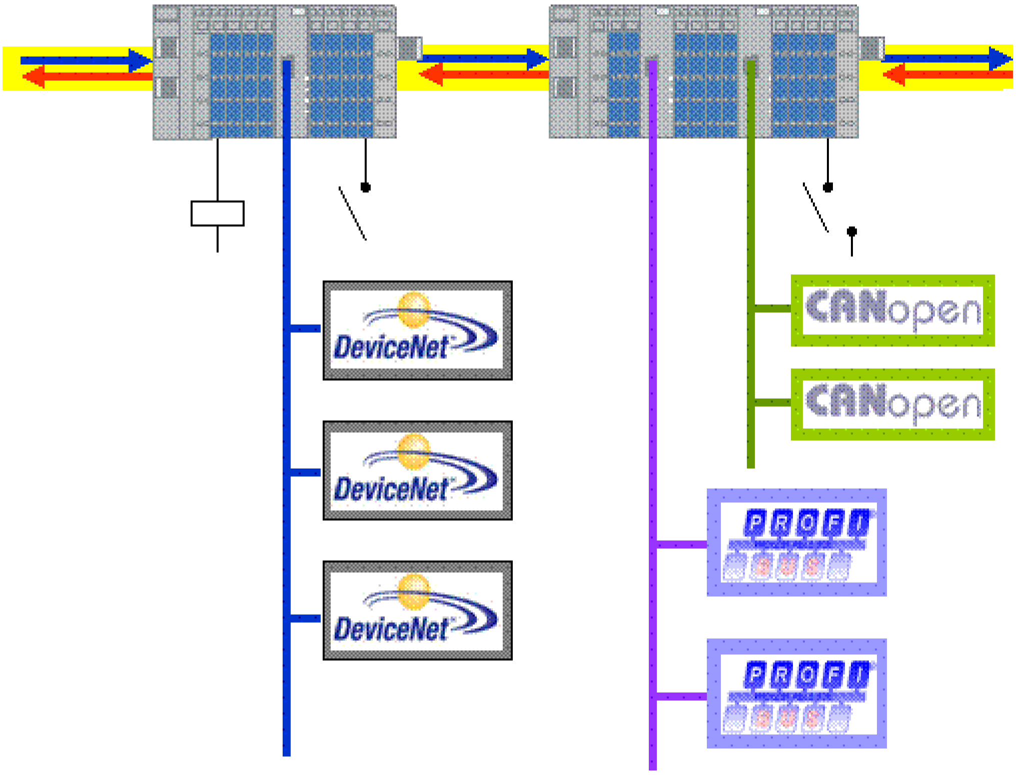


Figure 3-53: Fieldbus Gateway

4.3.2.7 Device profiles

The device profiles describe the application parameters and the functional behavior of the devices, including the device class-specific state machines. For many device classes, fieldbus technology already offers reliable device profiles, such as for I/O devices or drives. Users are familiar with these profiles and the associated parameters and tools. Therefore, no EtherCAT-specific device profiles have been developed for these device classes. Instead, simple interfaces for existing device profiles are offered. This greatly assists users and device manufacturers alike during the change from existing fieldbuses to EtherCAT.

CANopen over EtherCAT (CoE)

CANopen device and application profiles are available for a wide range of device classes and applications, ranging from I/O components, drives, encoders, proportional valves and hydraulic controllers to application profiles for plastic or textile machinery. EtherCAT can provide the same communication mechanisms as the familiar CANopen mechanisms: object dictionary, PDO (process data objects) and SDO (service data objects), and even the network management is comparable. EtherCAT can thus be implemented with minimum effort on devices equipped with CANopen. Large parts of the CANopen firmware can be re-used. Objects can optionally be expanded in order to account for the larger bandwidth offered by EtherCAT.

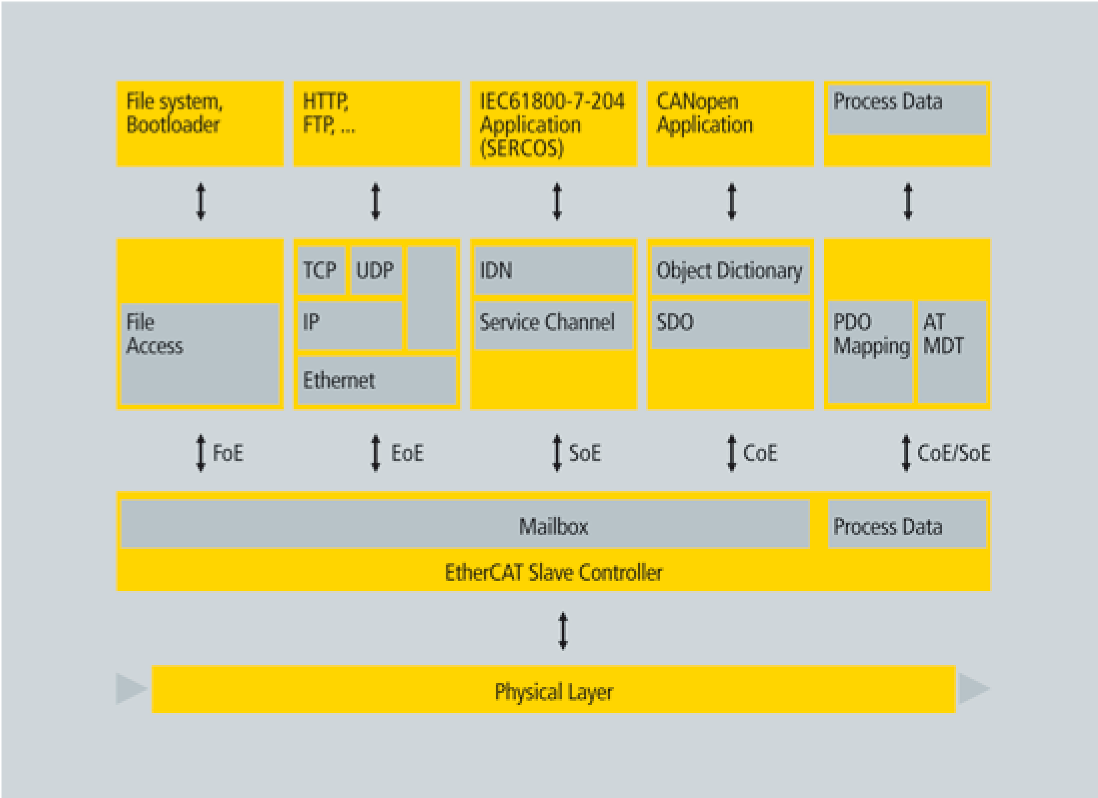


Figure 3-54: Several Device Profiles and Protocols can coexist

4.3.2.8 File Access over EtherCAT (FoE)

This very simple protocol, similar to TFTP, enables access to any data structure in the device. Therefore, standardized firmware upload to devices is possible, regardless of whether or not they support TCP/IP.

4.3.3 EtherCAT Implementation

The EtherCAT Technology was developed with very low cost devices in mind, like I/O terminals, sensors, and embedded controllers. EtherCAT only uses standard Ethernet frames according to IEEE 802.3. These frames are sent by the master device, and the slave devices extract and/or insert data on the fly. Thus EtherCAT uses standard Ethernet MACs, where they really make sense: in the master device. EtherCAT slave controllers are also used where such dedicated chips really make sense: in the slave device, where they handle the process data protocol in hardware and provide maximum real-time performance regardless of the local processing power or software quality.

4.3.3.1 Master Configuration

EtherCAT communicates a maximum of 1486 bytes of distributed process data with just one Ethernet frame. Therefore, unlike other solutions where the master device in each network cycle has to process, send and receive frames for each node, EtherCAT systems typically only need one or two frames per cycle for the entire communication with all nodes, so EtherCAT masters do not require a dedicated communication processor. The master functionality puts hardly any load on the host CPU, which can handle this task easily while processing the application program: so EtherCAT can be implemented without special or expensive active plug-in cards, just by using a passive NIC card or the on-board Ethernet MAC. Implementation of an

EtherCAT master is very easy, particularly for small and medium-sized control systems and for clearly defined applications.

For example, a PLC with a single process image: if it does not exceed the 1486 bytes, cyclic sending of a single Ethernet frame with the cycle time of the PLC is sufficient (as shown in "Figure 3-55: Master-Implementation with one Process Image " on page 164). Because the header does not change at run-time, the only thing required is that a constant header be added to the process image and that the result be transferred to the Ethernet controller.

The process image is already sorted, since with EtherCAT mapping does not occur in the master, but in the slaves - the peripheral devices insert their data at the respective points in the passing frame. This further unburdens the host CPU. It was found that an EtherCAT master entirely implemented in software on the host CPU uses less of its processing power than much slower fieldbus systems implemented with active plug-in cards; servicing the DPRAM of the active card alone puts more load on the host.

System configuration tools provide the network and device parameters (including the corresponding boot-up sequence) in a standardized XML format.

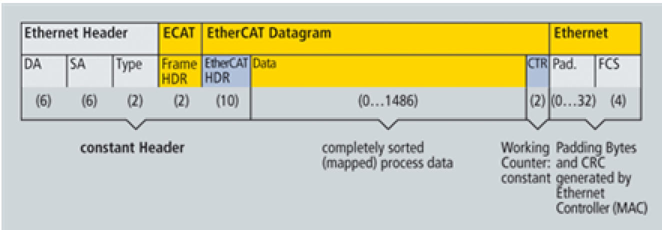


Figure 3-55: Master-Implementation with one Process Image

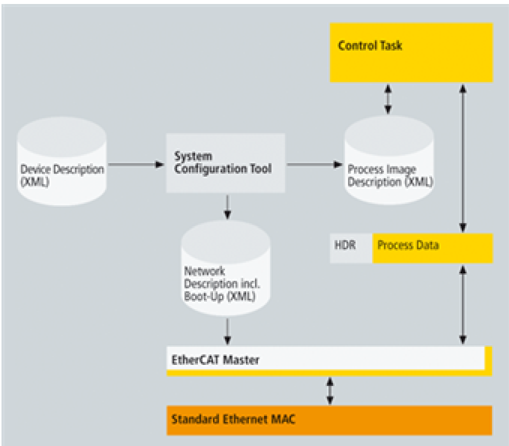


Figure 3-56: Structure of EtherCAT Master Implementation

ESI and ENI Files

The EtherCAT master uses the information from the ENI file to initialize and configure the EtherCAT network. The ESI files are provided by the vendor for each device. They contain information about the device functionality and its settings. The ESI files are used by the KAS IDE to generate the ENI file. The KAS controller's EtherCAT master uses the ENI file for network initialization and configuration.

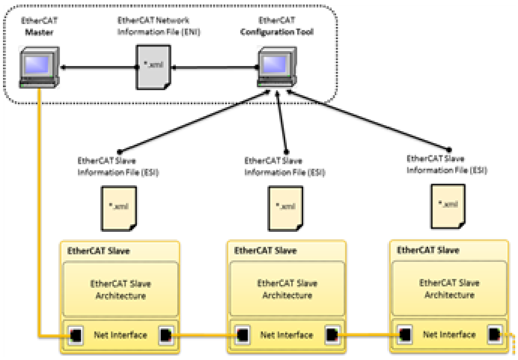


Figure 3-57: EtherCAT Network Architecture[1]

ENI File	A network configuration file in XML format, the ENI file describes the network topology, the initialization commands for each device, and commands which have to be sent cyclically. The ENI file is provided to the master, which sends commands according to this file. The KAS IDE creates the ENI file after a network discovery, which can be exported or imported. A scan and compile should be redone, if the network changes, in order to regenerate the ENI file.
ESI File	A device description in XML format. This is a fixed file provided by the supplier of a given EtherCAT device. The ESI file contains information about the device's functionality and settings. EtherCAT device vendors must provide an ESI file, which is used by the KAS IDE to compile the network information (e.g. process data structures, initialization commands) and create the ENI file.

4.3.3.2 Slave Configuration

A cost-effective EtherCAT slave controller (ESC) is used in the slave devices. With EtherCAT the slave does not need a microcontroller at all. Simple devices that get by with an I/O interface can be implemented only with the ESC and the RJ45 connector. The process data interface (PDI) to the slave application is a 32-bit I/O interface. This slave without configurable parameters needs no software or mailbox protocol. The EtherCAT State Machine is handled in the ESC. The boot-up information for the ESC comes out of the EEPROM that also supports the identity information of the slave. More complex slaves that are configurable have a host CPU on board. This CPU is connected to the ESC with an 8-bit or 16-bit parallel interface or via a serial connection.

EtherCAT Slave Controller

The slave controllers typically feature an internal DPRAM and offer a range of interfaces for accessing this application memory:

- The SPI (serial peripheral interface bus) is intended particularly for devices with small process data quantity, such as analog I/O modules, sensors, encoders or simple drives.
- The parallel 8/16-bit microcontroller interface corresponds to conventional interfaces for fieldbus controllers with DPRAM interface. It is particularly suitable for more complex devices with larger data volume.
- The 32-bit parallel I/O interface is suitable for the connection of up to 32 digital inputs/outputs, but also for simple sensors or actuators operating with 32 data bits. Such

[1]Image courtesy of EtherCAT.org, http://www.ethercat.org/pdf/english/etg2200_v2i0i1_slaveimplementationguide.pdf

devices do not need a host CPU at all (as shown in "Figure 3-59: Slave Hardware: FPGA with direct I/O " on page 166).

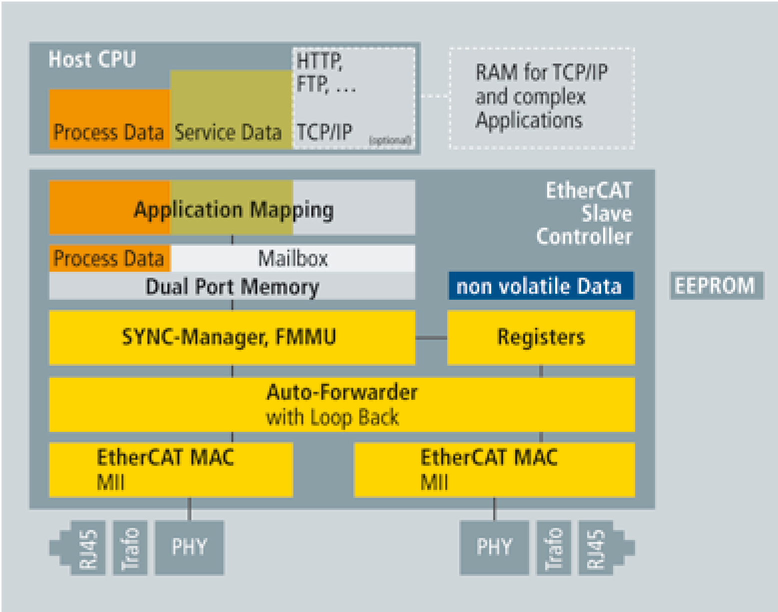


Figure 3-58: Slave Hardware: FPGA with Host CPU

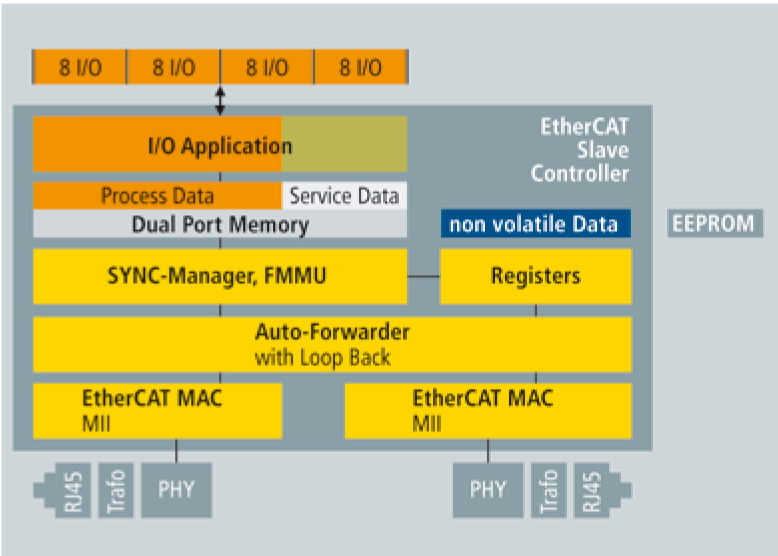


Figure 3-59: Slave Hardware: FPGA with direct I/O

4.3.3.3 State Machine

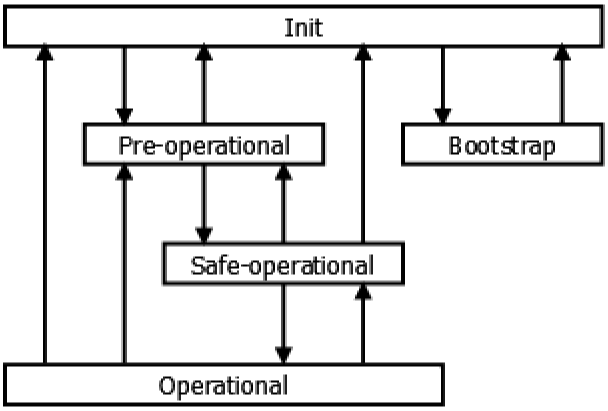


Figure 3-60: EtherCAT State Machine

Where the transitions are:

- from **Init** to **Pre-Operational (Pre-Op)**: Master configures the Sync Manager channels for Mailbox communication
- from **Pre-Op** to **Safe-Operational (Safe-Op)**: Master configures parameter using the Mailbox
- from **Safe-Op** to **Operational (Op)**: Master sends valid Outputs

The different states are:

- **Init**
  No communication on the Application Layer
  Master has access to the DL-Information registers
- **Pre-Operational (Pre-Op)**
  Mailbox communication on the Application Layer
  No Process Data communication
- **Safe-Operational (Safe-Op)**
  Mailbox communication on the Application Layer
  Process Data communication. Only Inputs are evaluated (Outputs in **Safe** state)
- **Operational (Op)**
  Inputs and Outputs are valid
- **Bootstrap**
  Recommended if firmware updates are necessary
  No Process Data communication
  Communication via Mailbox on Application Layer
  Only FoE protocol available

4.3.3.4 PDOs for AKD and S300/S700 (default)

The KAS Motion Engine interacts with the AKD and S300/S700 drives through CANopen objects in the selected PDOs. Some of the ML and MC function blocks require specific CANopen objects in the PDO(s).

The default AKD PDO selection includes all of the CANopen objects needed by the motion engine and function blocks. The default S300/S700 PDOs include only the minimal CANopen objects required by the KAS motion engine.

The following tables identify which CANopen objects are required by the motion engine and function blocks, and whether they are available in the default PDO for AKD or S300/S700, and their corresponding AKD drive parameter There are two types of PDOs:

- **RxPDO** — from Controller to Drive
- **TxPDO** — from Drive to Controller

IMPORTANT Check these tables to make sure the objects needed by the features your application uses are included in the AKD and S300/S700 PDO selection.

From Controller to Drive (RxPDO)

Index - subindex	Object Name	Require-d	AKD	S300 S700	Associated ML FB	Associated MC FB	Associated AKD para-meter
0x6040 - 0	Control word	Yes	Yes	Yes		MC_ ClearFaults, MC_Power	
0x60C1 - 1 or 0x6062 - 0	Position demand value	Yes	Yes	Yes	Related to Axis pipe block pos-itions (for more details, see page 101)	MC_ ReadParam (1)	PL.CMD
0x20A4 - 0 or 0x2802 - 0	Latch control word	No	Yes	Yes	MLAxisCfgFastIn MLAx-isTimeStamp, , all Trigger MLTrig FB	MC_ TouchProbe , MC_ AbortTrig-ger	CAP0.EN, CAP1.EN, CAP0.MODE, CAP1.MODE
0x60B2 - 0 or 0x60F6 - 1	Additive torque value (Torque Feed Forward)	No	Yes	Yes	MLAxisAddTq	n/a [1]	IL.BUSFF
0x60FE - 1	Digital out-puts (used by Onboard I/O mappings)	No	Yes	No	n/a	n/a	DOUTx.STATE
0x3470 - 3	AOUT.VALU-E (used by Onboard I/O mappings)	No	Yes	No	n/a	n/a	AOUT.VALUE-U

From Drive to Controller (TxPDO)

Index - subindex	Object Name	Require-d	AK-D	S30-0 S70-0	Associated ML FB	Associated MC FB	Associated AKD parameter
0x6041 - 0	Status word	Yes	Yes	Yes	n/a [2]	n/a	n/a
0x6063 - 0 or 0x6064 - 0	Position actual value	No	Yes	Yes	MLAxisFBackPos, MLAx-isReadActPos	MC_ ReadActPo-s	PL.FB

[1] means Not Applicable
[2] means Not Applicable

Index - subindex	Object Name	Require-d	AK-D	S30-0 S70-0	Associated ML FB	Associated MC FB	Associated AKD parameter
0x2050 - 0 or 0x35C9 - 0	Position actual value 2	No	Yes	Yes	MLAx-isRead2ndFB	For a **Digit-izing** axis: Secondary feedback can be read by reading the actual position of the axis which is assigned to the sec-ondary feed-back. Digitizing axes always use the second feedback for the Drive. KAS does not allow a digit-izing axis on a drive which has not a servo axis already assigned	PL.FB (if DRV.CMDSOURC-E = 1)
0x606C - 0	Velocity actual value	No	Yes	No	MLAxisReadVel	MC_ ReadActVel	VL.FB
0x6077 - 0	Torque actual value	No	Yes	Yes	MLAxisReadTq	MC_ ReadPara-m (1016)	IL.FB
0x20A5 - 0 or 0x2901 - 0	Latch status word	No	Yes	Yes	MLAxisCfgFastIn, MLAxisTimeStamp , all Trigger MLTrig FB	MC_ TouchProb-e, MC_ AbortTrig-ger	CAPx.STATE
0x20A6 - 0 or 0x2902 - 0	Latch pos-ition	No	Yes	Yes	MLAxisCfgFastIn, MLAxisTimeStamp , all Trigger MLTrig FB	MC_ TouchProb-e, MC_ AbortTrig-ger	CAPx.T (for time) CAPx.PLFB (for position)
0x60FD - 0	Digital inputs (used by Onboard I/O map-pings)	No	Yes	No	n/a	n/a	DIN.STATES

Index - subindex	Object Name	Required	AK-D	S300 S700	Associated ML FB	Associated MC FB	Associated AKD parameter
0x3470-4	AIN.VALUE (used by Onboard I/O mappings)	No	Yes	No	n/a	n/a	AIN.VALUE
0x60F4	Following error	No	Yes	Yes	MLAxisReadFEUU	MC_ReadParam (1006)	PL.ERR

Examples

Below are three examples where the PDO object is passed as an argument in the function block.

```
MLSmpConnectEx('1001:Position actual value 2') ;
```

The argument is a concatenation of the EC address with the PDO object name.

```
MLCNVConnectEx(PipeNetwork.CNV1, PipeNetwork.AXIS1, EC_ADDITIVE_TORQUE_
VALUE, 0 ) ;
```

The argument is a constant based on the object index.

```
ECATGetObjVal(1001, 'Position actual value') ;
```

The argument is the PDO object name.

4.3.4 CANopen

4.3.4.1 CANopen Status machine

The states of the status machine can be revealed by using the status word.

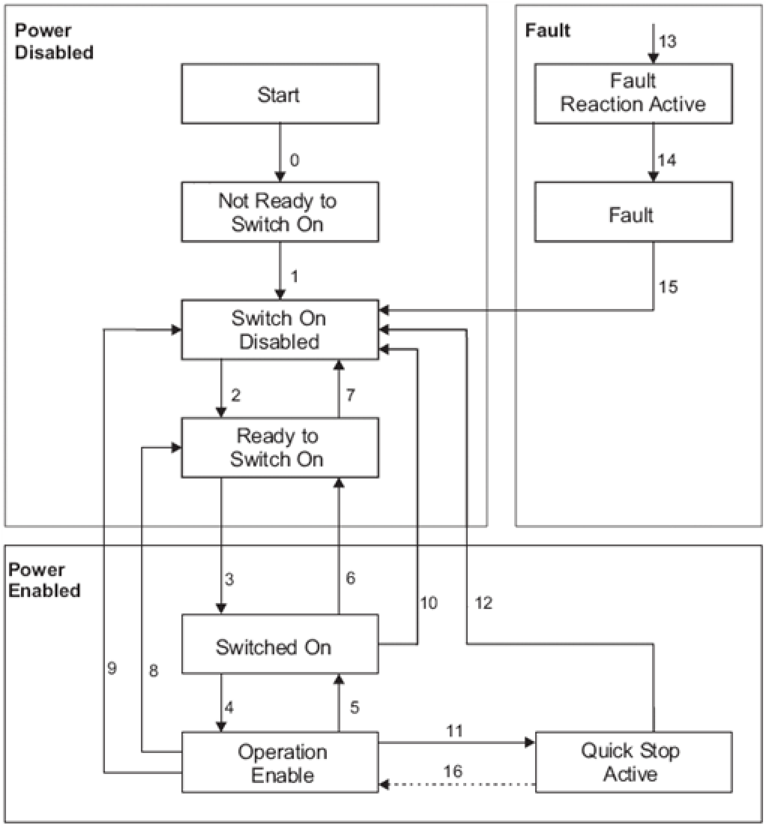


Figure 3-61: CANopen Status Machine

The start state is a pseudo-state indicating the start when the state machine is activated during the start-up sequence of the device drive's application software.

Status description

Status	Description
Not ready to switch on	The drive is not ready to switch on. The controller has not indicated readiness for service. The drive is still in the boot phase or in the fault status
Switch on disabled	The drive cannot be enabled via the EtherCAT interface; because for example there is no connection to a power source
Ready to switch on	The drive can be enabled via the control word. DC-link voltage can be switched on, parameters can be transferred, motion functions cannot be performed yet.
Switched on	The drive is enabled but the setpoints are not yet transferred from the EtherCAT interface. The drive is idle. DC-link voltage must be switched on, parameters can be transferred, but motion functions cannot be performed yet. Output stage is switched on (enabled). Operation Enable No fault present; output stage is enabled; motion functions are enabled.
Operation enabled	The drive is enabled and the setpoints are transferred from the EtherCAT interface. No fault present; output stage is enabled; motion functions are enabled.
Quick stop active	The drive has been stopped with the quick stop ramp; output stage is enabled; motion functions are not enabled.

Status	Description
Fault reaction active	A fault has occurred and the drive is stopped with the emergency stop ramp
Fault	A fault is active, and the drive has been stopped and disabled

Table 3-5: Status Description

Transitions of the status machine

The drive device supports the transitions and actions as listed in the table below. The event initiates the transition. The transition is terminated after the action has been performed.

Transition	Event	Action
0	Automatic transition after power-on or reset application	Drive device self-test and/or self initialization has to be performed.
1	Automatic transition	Communication has to be activated.
2	Shutdown command from control device or local signal	None
3	Switch on command received from control device or local signal	The high-level power has to be switched on, if possible.
4	Enable operation command received from control device or local signal	The drive function has to be enabled and all internal setpoints cleared.
5	Disable operation command received from control device or local signal	The drive function has to be disabled.
6	Shutdown command received from control device or local signal	The high-level power has to be switched off, if possible.
7	Quick stop or disable voltage command from control device or local signal	None
8	Shutdown command from control device or local signal	The drive function has to be disabled, and the high-level power has to be switched off, if possible.
9	Disable voltage command from control device or local signal	The drive function has to be disabled, and the high-level power has to be switched off, if possible.
10	Disable voltage or quick stop command from control device or local signal	The high-level power has to be switched off, if possible.
11	Quick stop command from control device or local signal	The quick stop function has to be started.
12	Automatic transition when the quick stop function is completed and quick stop option code is 1, 2, 3 or 4, or disable voltage command received from control device (depends on the quick stop option code)	The drive function has to be disabled, and the high-level power has to be switched off, if possible.
13	Fault signal	The configured fault reaction function has to be executed.

Transition	Event	Action
14	Automatic transition	The drive function has to be disabled; the high-level power has to be switched off, if possible.
15	Fault reset command from control device or local signal	A reset of the fault condition is performed, if no fault exists currently on the drive device; after leaving the Fault state, the Fault reset bit in the control word has to be cleared by the control device.
16	Enable operation command from control device, if the quick stop option code is 5, 6, 7, or 8	The drive function has to be enabled.

Table 3-6: Transition Events and Actions

4.3.4.2 Control word

The status machine for the control word corresponds to the CANopen status machine.

The control word indicates the received command controlling the state machine. It is only read during **Operational** status. The control commands allow the manipulation of the state of a drive by setting its control word . Such commands are built up from the logical combination of the bits in the control word and external signals (e.g. enable output stage).

Bits definition of the control word

Bit	Name
0	Switch on
1	Disable Voltage
2	Quick Stop
3	Enable Operation
4	Operation mode specific
5	Operation mode specific
6	Operation mode specific
7	Reset Fault (only effective for faults)
8	Pause/halt
9	reserved
10	reserved
11	reserved
12	reserved
13	Manufacturer-specific
14	Manufacturer-specific
15	Manufacturer-specific

Table 3-7: Bit Assignment in Control Word

The commands are coded as given in the table below.

Command	Bits of the control word					Transitions
	Bit 7	Bit 3	Bit 2	Bit 1	Bit 0	
Shutdown	0	X	1	1	0	2,6,8
Switch on	0	0	1	1	1	3
Switch on + enable operation	0	1	1	1	1	3 + 4 (Note)
Disable voltage	0	X	X	0	X	7,9,10,12
Quick stop	0	X	0	1	X	7,10,11
Disable operation	0	0	1	1	1	5
Enable operation	0	1	1	1	1	4,16
Fault reset	up	X	X	X	X	15

Table 3-8: Command Coding

Note: automatic transition to Enable operation state after executing SWITCHED ON state functionality.

Bits marked by an X are irrelevant.

4.3.4.3 Status word

The status machine for the control word corresponds to the CANopen status machine.

The current state of the status machine can be read out with the aid of the status word .

The status word is only updated and written by the drive in **Safe-Op** and **Operational** states.

Bits definition of the status word

Bit	Name
0	Ready to switch on
1	Switched on
2	Operation enable
3	Fault
4	Voltage enabled
5	Quick stop
6	Switch on disabled
7	Warning
8	Manufacturer-specific (reserved)
9	Remote (always 1)
10	Target reached
11	Internal limit active
12	Operation mode specific (reserved)
13	Operation mode specific (reserved)
14	Manufacturer-specific (reserved)
15	Manufacturer-specific (reserved)

Table 3-9: Bit Assignment in Status Word

The bit combinations coding the following states are listed in the table below.

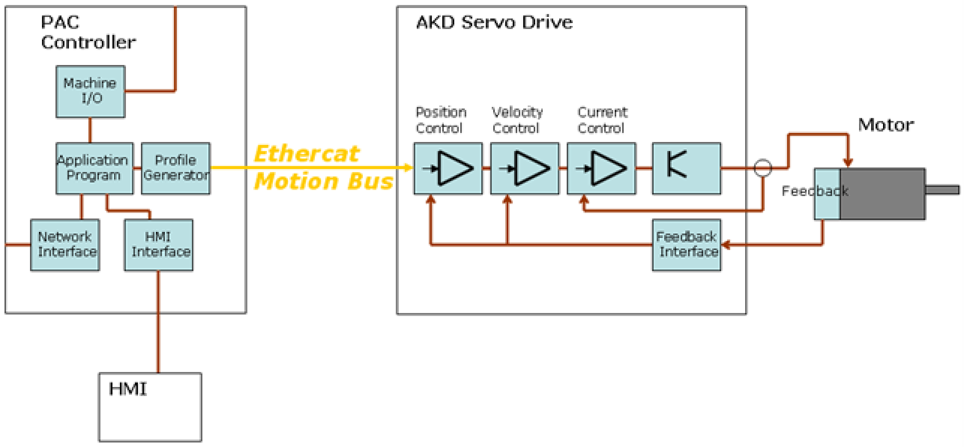
Status word MSB (15..12) (11 .. 8) (7 .. 4) (3 .. 0) LSB	State
xxxx xxxx x0xx 0000	Not ready to switch on
xxxx xxxx x1xx 0000	Switch on disabled
xxxx xxxx x01x 0001	Ready to switch on
xxxx xxxx x01x 0011	Switched on
xxxx xxxx x01x 0111	Operation enabled
xxxx xxxx x00x 0111	Quick stop active
xxxx xxxx x0xx 1111	Fault reaction active
xxxx xxxx x0xx 1000	Fault

Table 3-10: State Coding

Bits marked by an X are irrelevant

4.4 AKD Drive

The **servo loops** in a KAS system are located within the AKD Drive. The **profile generator** used for all the motion in your application is located in the PAC.



4.4.1 AKD Drive

4.4.1.1 Connection Modes

When AKD drive has to be configured, it is important to understand the distinction between the two functional modes:

- Unconnected (Offline)
- Connected (Online)

Offline mode

When the KAS IDE is not connected to the IPC, all the AKD are in offline mode.

In this mode, AKD drives are emulated: when you modify the value of a parameter, a command is sent to a logical drive that interprets the command and updates the in-memory parameter. An offline drive allows you to use KAS IDE without having any drive hardware. The parameters of a drive are simulated. An offline drive allows you to create a drive configuration as well as exploring the different AKD views. Because it is a simulation there are a number of operations that are not possible.

Online mode

An online drive is working with a specific physical drive on your network.

Online mode updates the parameters directly in the AKD. When you modify the value of a parameter, a command is sent to the drive and the corresponding parameter is updated.

NOTE An AKD drive cannot be deleted while it is connected.

NOTE The AKD drive will come out of Online Configuration mode and start an the application if you start an application while the drive is in that mode.

NOTE **Important!** Before coming out of Online Configuration mode (either by clicking the Online Configuration button or clicking the Start button when the device is in Online Configuration mode) you should ensure the following conditions exist.

1. Drive is not enabled
2. Service motion is not enabled
3. Auto-tuning is not enabled

Recommendation: Clicking on the Disable button will force the above scenario.

4.4.1.2 AKD Configuration According to EtherCAT State

The drive configuration can only take place when the AKD is Online. As shown below, it can happen when the EtherCAT fieldbus is in the following state: Pre-Op or Op.

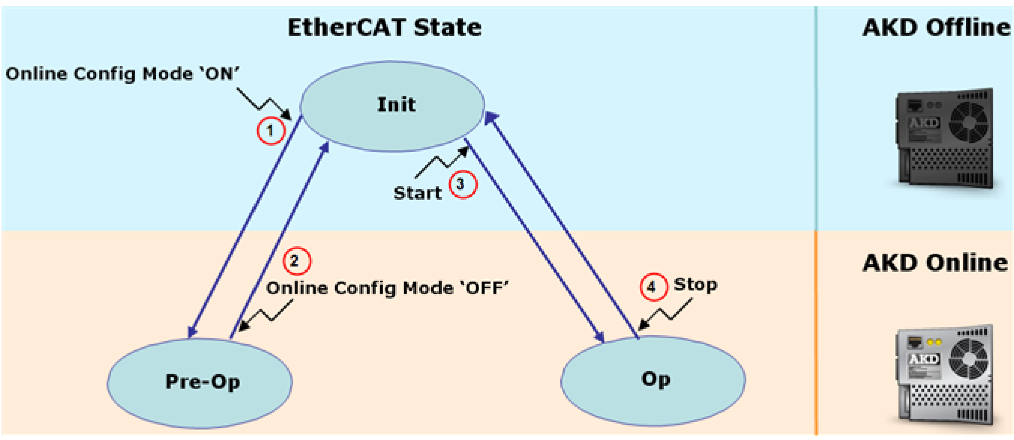


Figure 3-62: AKD Configuration According to EtherCAT State

Action	Name	Description
1	Online Config Mode "ON"	Set the EtherCAT fieldbus in Pre-Op state to enable the AKD Online Configuration (to see where you can access this button in the KAS IDE, see page 196) This step goes online to connect all AKD and update their parameters (for procedure, see page 190)

Action	Name	Description
2	Online Config Mode "OFF"	This step disconnect all AKD drives
3	Start the Drive	Set the EtherCAT fieldbus in the Operational state [1] (to see where you can access this button, see page 651) This step goes online to connect and start all AKD drives AKD configuration is possible from the different AKD views (with some restrictions for the views: Service Motion and Performance Servo Tuner)
4	Stop the Drive	This step disconnect and stop all AKD drives

Table 3-11: AKD Drive - List of Actions

4.4.1.3 About AKD Parameters

When the KAS IDE is establishing a connection to the IPC, each AKD within the project which is mapped to a physical drive stores its parameters in a file and performs the connection to the mapped drive. When the connection is done, the parameters of the logical drive (AKD offline) is uploaded to the physical drive. The reverse operation is done during the disconnection from the IPC.

4.5 Tasking Model / Scheduling

In the KAS Runtime, both the Motion and Programmable Logic Controller (PLC) Programs are run every cycle. The cycle update time is set when configuring the EtherCAT motion bus (see "EtherCAT Master Settings" (see page 214)).

The cycle time becomes effective only when the Motion is started (i.e. when the PLC code initializes the Motion by calling the MLMotionIni function block), and the application runs on a PAC.

The time base remains much longer than the cycle time as long as the Motion is **not** yet started, or if the application runs on the KAS Simulator (for more details, see page 683.). In these cases, the PLC execution rate is approx. 10 milliseconds.

4.5.1 Priority Between Motion and PLC

The Motion computation is always executed each cycle, and occurs before executing the PLC programs application. The figure below shows the execution in the following order:

1. I/O related to the PLC program are serviced (for more details, see page 178)
2. Motion command, position feedback from each axis and other elements in the EtherCAT PDO object are sent and received on the EtherCAT motion bus (this includes servo drives and Remote I/O)
3. PLC programs are executed
4. NVRAM variables are saved (for more details, see page 178)

[1] Depending on the number of AKD drives physically present in the EtherCAT network, the KAS IDE can slow down when getting data.
The KAS Runtime is **not concerned** with this limitation.

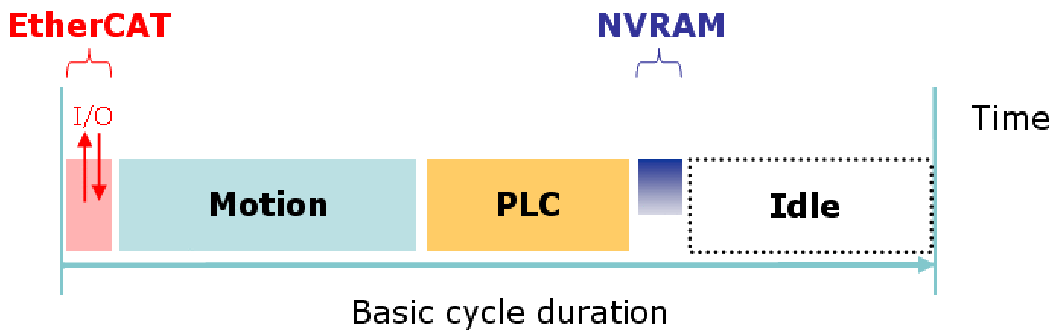


Figure 3-63: Priority Between Motion and PLC

> **NOTE** The Motion time (see figure above) must be shorter than the basic cycle duration at each cycle. This condition is checked at each cycle and if the cycle is overran, Kollmorgen Automation Suite generates a fatal error and the application execution is stopped.

4.5.1.1 EtherCAT Processing Time

The EtherCAT frame is executed at the beginning of the cycle. During this period, all the values related to EtherCAT (PDO) are exchanged, including:

- **Inputs** are read
- **Outputs** are set

Based on the I/O mapping to PLC variables, the I/Os are updated before they are effectively used during the PLC period.

As a consequence, when the PLC variables set an Output, it is updated during the EtherCAT frame of the next cycle.

About Variation during the EtherCAT Processing

The EtherCAT period is subject to time variation along the cycles due to the following reasons:

- Some EtherCAT function blocks are using the asynchronous SDO communication, which is not deterministic.
- Some EtherCAT slave devices support mailbox protocols.
  The master cyclically reads the mailbox of the EtherCAT slaves (polling of mailbox is performed every 50 cycles and is spread on several cycles depending on the number of EtherCAT slaves)

See also the FAQ about SDO communication.

4.5.1.2 NVRAM Processing Time

Due to a slow processing when saving the Retain Variables to the NVRAM, this action is not performed each cycle. The save operation is performed in the background every 20 seconds (frequency increases to each 2 seconds when the application is running).

When executed during a cycle, it occurs after the PLC period.

4.5.1.3 What happens when a PLC Program is overrunning the Cycle Duration

Large application can require more than one cycle to completely execute all the PLC programs.

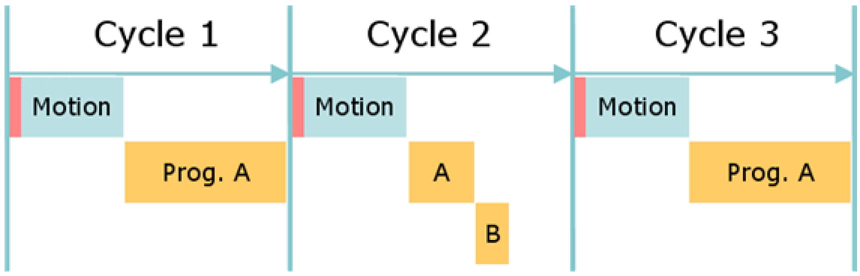


Figure 3-64: Application Overrunning the Basic Cycle

The figure above shows an example of an application with two PLC programs (A and B). It takes two cycles to execute all the code in the two programs.

- Cycle 1 executes most of Prog. A
- Cycle 2 finishes Prog. A and executes Prog. B

NOTE Even if there is time left over in the cycle, execution of Prog. A does not start until the next cycle

- Cycle 3 starts executing Prog. A again

An application overrun has no effect other than a short delay in the application execution. Execution of the real-time application is recovered as soon as the overload disappears.

IMPORTANT If Outputs are set when a program runs over several basic cycles, unexpected and potentially dangerous effects can happen.

NOTE When running with the KAS Simulator, there is no overrunning because the cycle is extended to include all the PLC programs, right after the Motion computation.

See Also: "Priority Between PLC Programs " (see page 179)

4.5.2 Priority Between PLC Programs

In turn, PLC programs are assigned a priority. At times of heavy demand for processing time, the operating system serves programs with higher priority first.

For more details, see how to:

- Set the PLC cycling

See Also: "Priority Between Motion and PLC" (see page 177)

This page intentionally left blank.

5 Using the KAS IDE

This chapter provides explanations and procedures to accomplish common **tasks** with the KAS IDE.

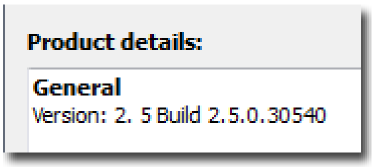
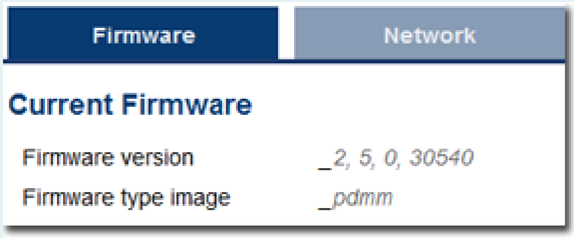
5.1 KAS IDE to Runtime Compatibility

The KAS software includes two main software components, the IDE and the Runtime. The IDE resides on your PC and the Runtime resides on your programmable automation controller or programmable drive. For optimum operation, the IDE you install and run on your PC must be compatible with the Runtime you install on your controller or programmable drive.

The KAS installer and PDMM Runtime contain the version information in the filename:

```
KAS-Setup-2.5.0.30540.exe
KAS-PDMM-2.5.0.30540.img
```

The version can be determined by the following:

Version Source	Found From	Example
IDE, Controller or Simulator	Help menu > About	
PDMM	Web server Settings Firmware tab	

The compatibility between the IDE and the Runtime is defined by matching the digits in the versions. The IDE and Runtime use the same version scheme:

```
major . minor . micro . revision
```

- If the IDE to Runtime major.minor is NOT equal, they are NOT compatible.
- If the IDE to Runtime major.minor is equal, but the micro is NOT equal, they are not 100% compatible.
- If the IDE to Runtime major.minor.micro is equal, they are compatible.

Examples

IDE	Runtime	Compatible?	IDE to Runtime Connection
2.5.xx.xxxx	1.2.xx.xxxx	NO, No connection.	The IDE displays an error message.
2.5.0.xxxx	2.5.1.xxxx	YES*	Connection possible. The IDE displays warning and requires user to press “ok” to continue.
2.5.0.xxxx	2.5.0.xxxx	YES	Normal connection.
2.5.0.30540	2.5.0.30540	YES	Normal connection.

* Not a recommended configuration. The IDE will operate, but there can be different features available between the IDE and the Runtime. It is recommended to upgrade either the IDE or the Runtime to matching versions.

5.2 KAS P-Code to Runtime Compatibility

It is possible that after a PAC or PDMM runtime re-installation that the old P-code and runtime `major.minor.micro` versions may be different. Runtime will not start if this occurs.

PDMM A generic E24 error will be displayed and an error will be logged if P-code version is different than the runtime version.

PAC An error will be logged if the P-code version is different than the runtime version.

5.3 Starting the KAS IDE

Open **All Programs** and start the **KAS IDE** application located under the **Kollmorgen** folder and **Kollmorgen Automation Suite** subfolder.

5.3.1 View Version Information

You can access the version information using the **About** command in the **Help** menu.

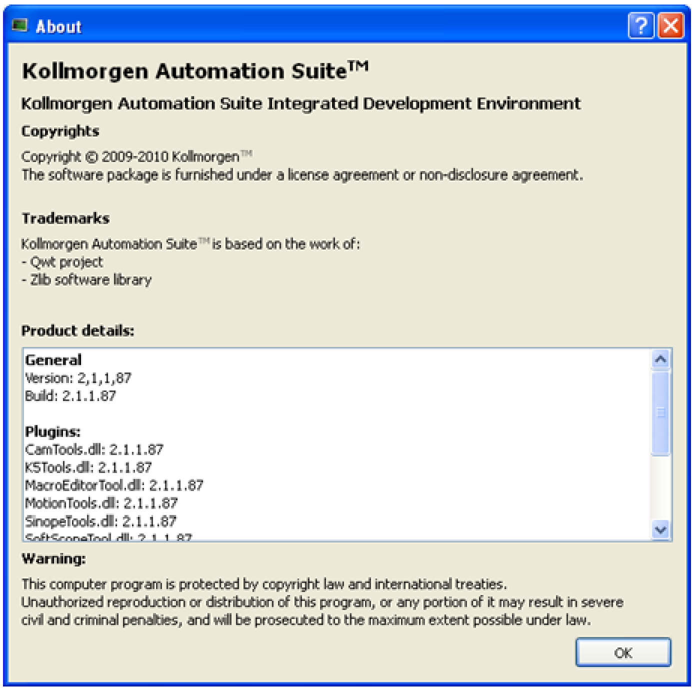


Figure 4-1: About Window

This window displays the application versions as well as all the plug-in versions included in the KAS IDE and loaded during start up.

5.3.2 Access Help System

You can access the online help using the **Documentation** command in the **Help** menu.

See also "Use the Context-Sensitive Help" on page 43

5.3.3 KAS Log Window

5.3.3.1 Log Information

The KAS log window (see "Figure 4-2: Log Messages " on page 184) provides a running display of activity related to the execution of the application. Items displayed include application startup and initialization information.

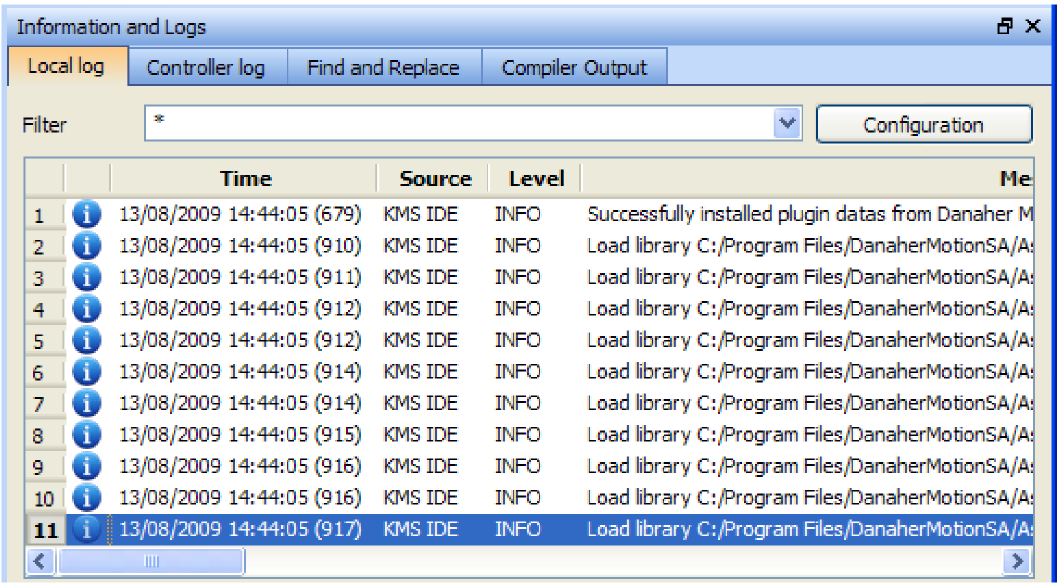


Figure 4-2: Log Messages

See also "Information and Logs" on page 626

5.3.4 KAS GUI

For a better understanding of **KAS** menus, toolbar and workspace items (description and manipulation), refer to paragraph "Describing KAS Graphical User Interface" on page 591

5.4 Creating a Project

5.4.1 Step 1 of 15 - Add a Controller

5.4.1.1 Add the Controller

To add a controller to your project:

- Click the **New** command in the **File** menu to start the Controller Creation Wizard
- Select the controller name within the list and click the **Next** button

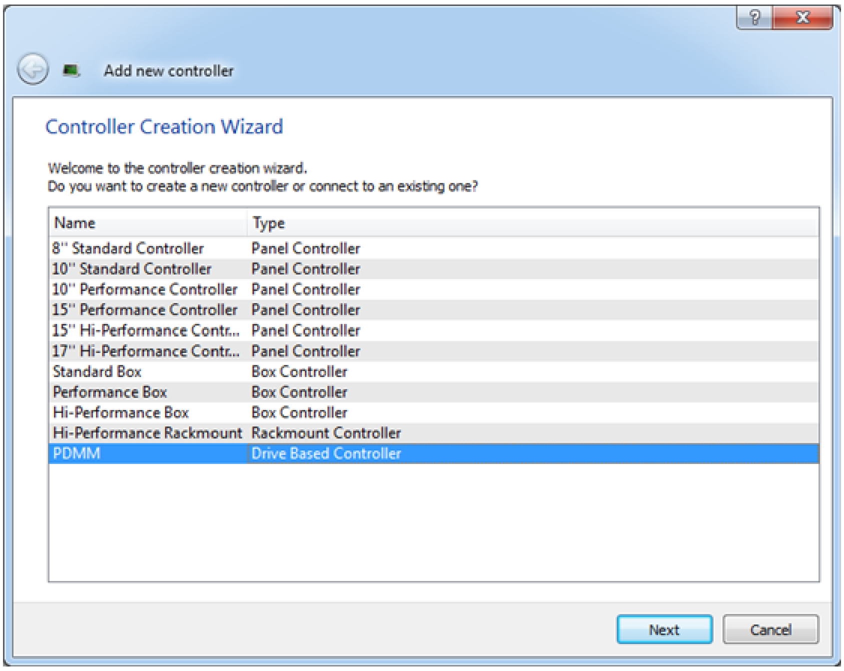


Figure 4-3: Select a Controller

- Choose the motion engine option (Pipe Network or PLCopen) and select the application template (see list below)

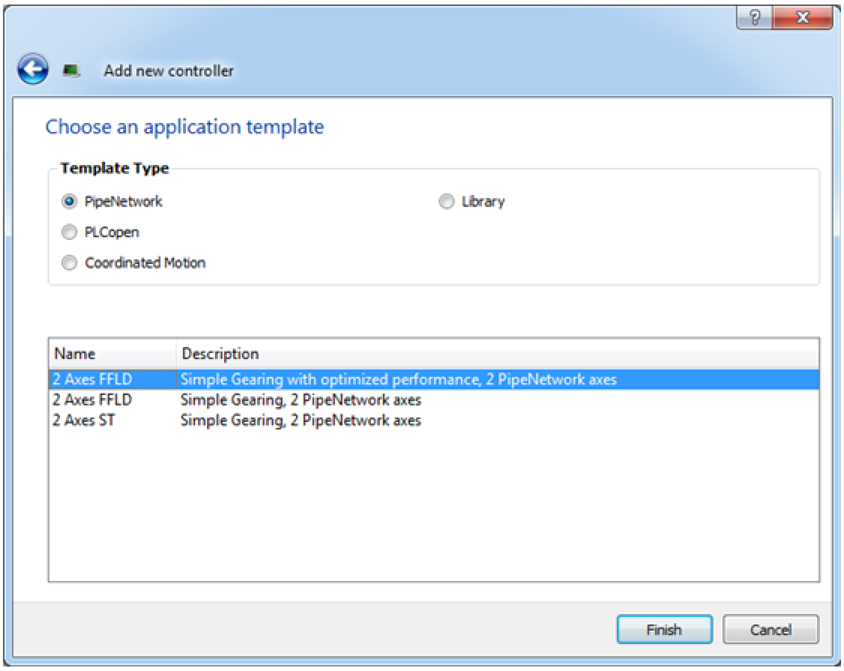


Figure 4-4: Select an Application Template

- Click the **Finish** button
- Click the **Save As** command in the **File** menu
- Define the Project Name and its Location
- Click **OK**

List of available application templates

Template Type	Template name	Description
Pipe Network	2 Axes FFLD	Simple Gearing, 2 PipeNetwork axes (FFLD only)
	2 Axes ST	Simple Gearing, 2 PipeNetwork axes (ST only)
	2 Axes FFLD	Simple Gearing with optimized performance, 2 PipeNetwork axes (SFC, ST, FFLD, and FBD)
PLCopen	2 Axes FFLD	Simple Gearing, 2 PLCopen axes (FFLD only)
	2 Axes ST	Simple Gearing, 2 PLCopen axes (ST only)
	2 Axes FFLD	Simple Gearing with optimized performance, 2 PLCopen axes (SFC and FFLD)
Coordinated Motion	2 Axes-Linear / Circular	Raster Scan Motion Path, 2 PLCopen axes
	3 Axes - Linear / Circular	Raster Scan Motion Path, 2 PLCopen axes and 1 PipeNetwork axis.
KAS Runtime	Library	Allows you to create a custom library (See also "Step 10 of 15 - Create and Use Custom Libraries" on page 265)

5.4.1.2 Step 1 of 15 - Add a Controller

Add the Controller

To add a controller to your project:

- Click the **New** command in the **File** menu to start the Controller Creation Wizard
- Select the controller name within the list and click the **Next** button

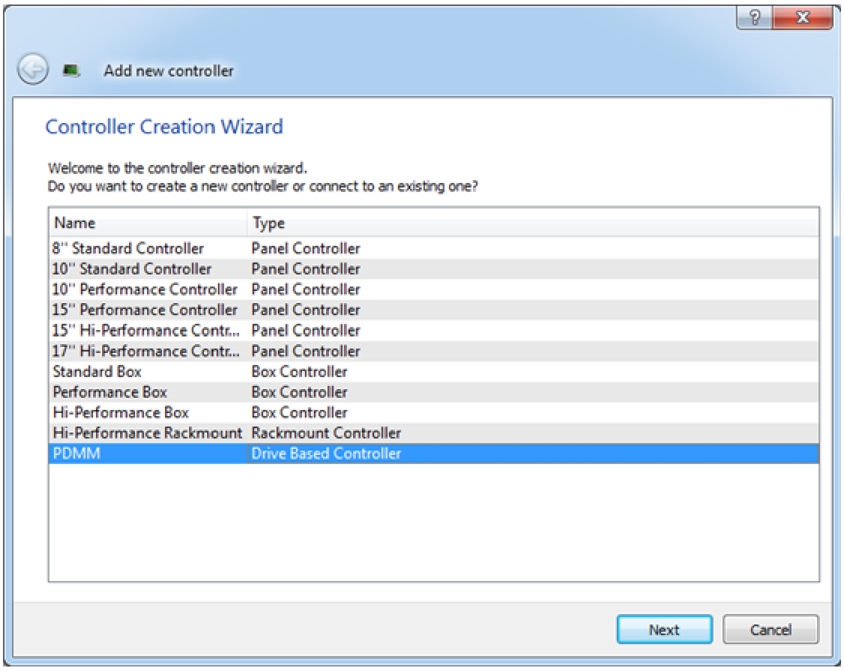


Figure 4-5: Select a Controller

- Choose the motion engine option (Pipe Network or PLCopen) and select the application template (see list below)

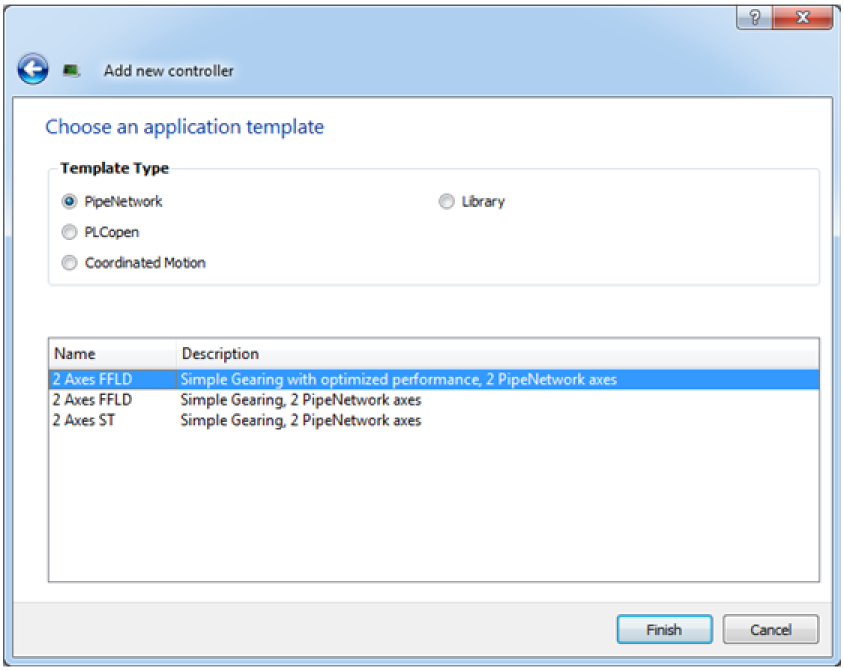


Figure 4-6: Select an Application Template

- Click the **Finish** button
- Click the **Save As** command in the **File** menu
- Define the Project Name and its Location
- Click **OK**

List of available application templates

Template Type	Template name	Description
Pipe Network	2 Axes FFLD	Simple Gearing, 2 PipeNetwork axes (FFLD only)
	2 Axes ST	Simple Gearing, 2 PipeNetwork axes (ST only)
	2 Axes FFLD	Simple Gearing with optimized performance, 2 PipeNetwork axes (SFC, ST, FFLD, and FBD)
PLCopen	2 Axes FFLD	Simple Gearing, 2 PLCopen axes (FFLD only)
	2 Axes ST	Simple Gearing, 2 PLCopen axes (ST only)
	2 Axes FFLD	Simple Gearing with optimized performance, 2 PLCopen axes (SFC and FFLD)
Coordinated Motion	2 Axes-Linear / Circular	Raster Scan Motion Path, 2 PLCopen axes
	3 Axes - Linear / Circular	Raster Scan Motion Path, 2 PLCopen axes and 1 PipeNetwork axis.
KAS Runtime	Library	Allows you to create a custom library (See also "Step 10 of 15 - Create and Use Custom Libraries" on page 265)

5.4.1.3 Configure the Controller

The controller is configured using the Controller Properties dialog box.

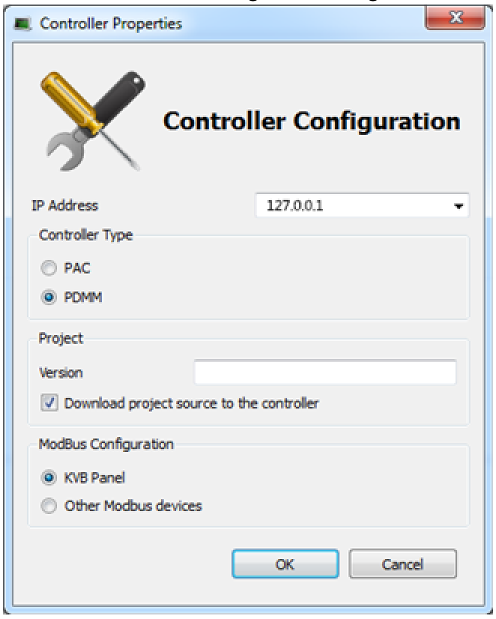


Figure 4-7: Configure the Controller Properties

To set-up the controller:

1. In the Project Explorer, right-click on the new controller to open the contextual menu
2. Select the **Properties** command
3. Define the IP Address

   A note about addressing

   - For the KAS Runtime Simulator, enter the localhost IP address: 127.0.0.1
   - For runtime system on PAC or AKD PDMM, enter the IP address of the controller (e.g. 10.155.100.150)

     NOTE You must ensure that controller is accessible by the KAS IDE machine (see FAQ section for IT issues)

4. Choose the controller type

   NOTE You must select the correct Controller type before compiling your application (the PLC code generated for PAC and AKD PDMM have different endianness). A warning is displayed if you try to start your application with an incompatible Controller type.

5. (Optional) Specify a version number (the string can be composed of any character)

   TIP Versioning can be useful when you make improvements to your application and need a version control system (See also "Use a Version Control System" on page 341). The version is saved in your project file. When you make a build for a PAC, it is also saved in the **versinfo.xml** file saved under the Application folder.

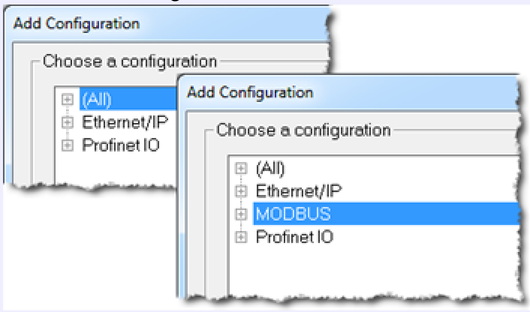
6. (Optional) Choose whether the project's source code should be downloaded to the controller. This is enabled by default and your preference is saved with the project.

Disabling this option means a comparison of source on the controller and in the IDE will not be available.

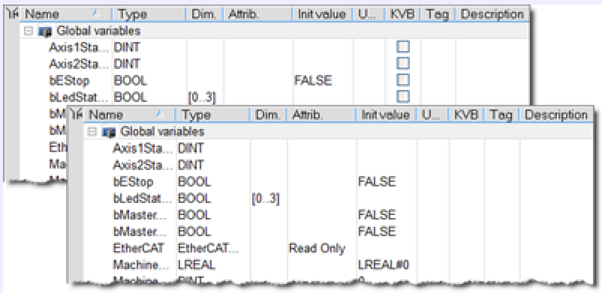
7. Select whether Modbus will go to a KVB Panel or be handled by another Modbus device.

NOTE

- If KVB Panel is selected there will be no node to add a Modbus configuration in the Fieldbus Editor.

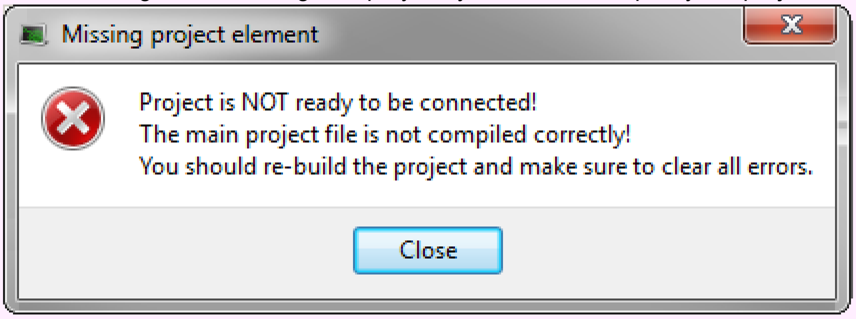


- If Other Modbus devices is selected then the KVB checkbox for Dictionary items will be hidden.. Motion configuration will need to be added from the Fieldbus Editor.



8. Click **OK**

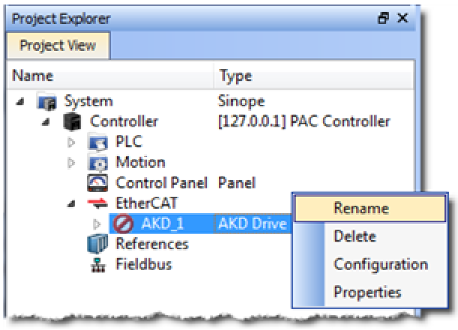
IMPORTANT You must compile your project before trying to connect to a Controller! The following error message displays if you do not compile your project!



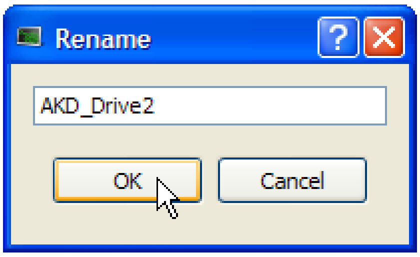
5.4.2 Step 2 of 15 - Add and Configure Drive

5.4.2.1 Add the Drive

1. In the Project Explorer, right-click the **EtherCAT** node to open the menu
2. Select the **Add AKD Drive** command
(this option is only enabled when you are **not** connected to the controller)
3. A "Generic" AKD drive is added.
4. Click **Finish** when you are done (for more details about the AKD drive GUI, click here)
5. In the Project Explorer, right-click the AKD Drive node and select the **Rename** command



6. Define the name for the new drive
   Note that the name is limited to 10 characters and can only include letters, numbers, and underscores.



7. Click **OK**

TIP An alternative method to add a drive is to rely on the auto scan feature.

When an AKD drive is added to the project tree, it must be mapped to a physical drive. This step is explained in paragraph "EtherCAT Mapping Device" on page 200

5.4.2.2 Configure the AKD Drive

1. In the Project Explorer, double-click the new AKD Drive or select Configuration from the context menu to open all the parameters linked to it

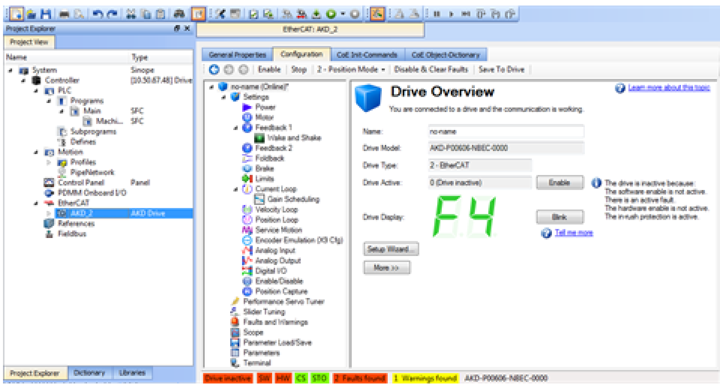
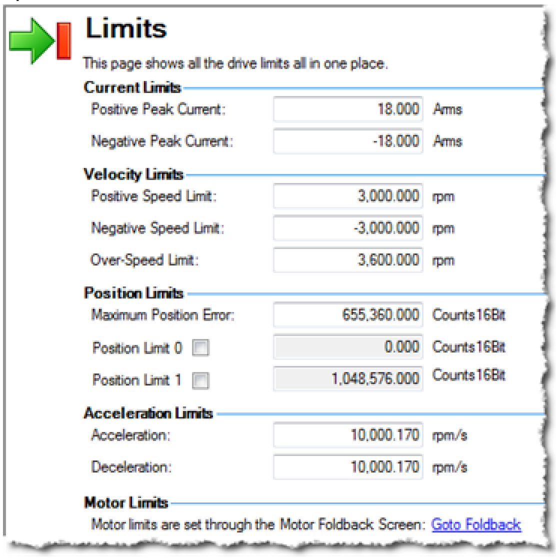
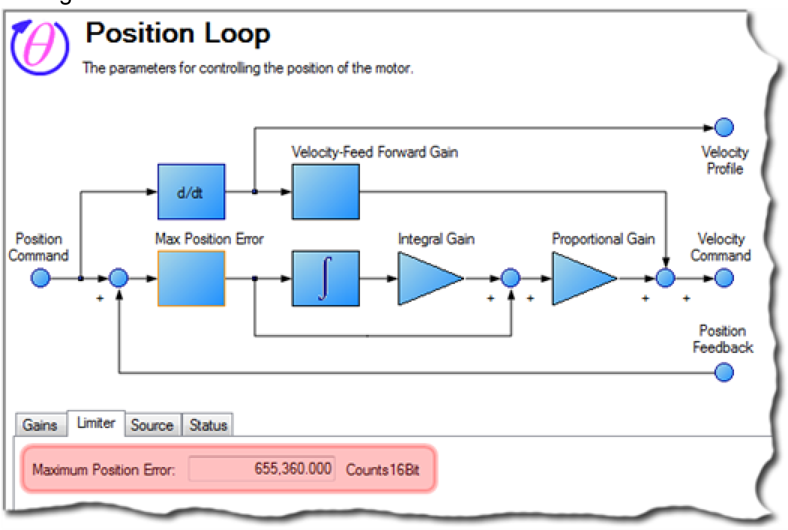


Figure 4-8: AKD Configuration

2. Define the motion parameters in the **Limits** tab to configure the limited motor torque, speed, and acceleration



3. Define the motion parameters in the **Position Loop** tab to configure the limiting following error



4. Define the resolution of the feedback position in the **Feedback** tab
   Note that for all feedback types, the motor position feedback sent from the AKD drive to the PAC through EtherCAT is normalized to 20 bits/rev or 1048576 counts/rev
5. Then, you must define the units to be used for the motion [1] :
   - For Pipe Network, refer to paragraph "Step 12 of 15 - Adding Motion" on page 279
   - For PLCopen, refer to paragraph "Axis Data" on page 294

NOTE User units in the PLC language editors are:

- Position : User unit
- Velocity User unit/sec
- Acceleration: User unit/sec^2

Several AKD tabs contain units that follow the standard AKD format:

[1]The normal units screen in the AKD Work bench GUI is not included in the IDE

NOTE

- Position: 16 bits/rev
- Velocity: RPM
- Acceleration: RPM/ Sec

6. To ensure high performance, define the load for your servo system.. KAS IDE provides several options for performing the drive tuning:
   - Slider Tuning - Allows adjustment to the desired bandwidth using the slider (pre-calculated tuning)
   - Performance Servo Tuner - Takes the drive through an automatic tuning sequence
   - Manual Tuning - Allows you to set gains individually for Current Loop, Velocity Loop, and Position Loop

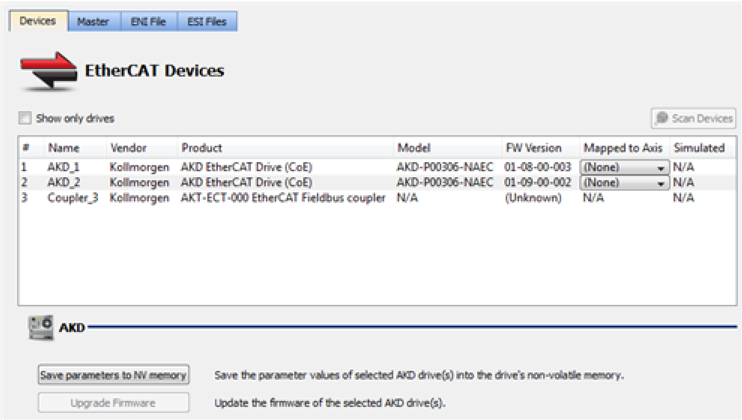
For more details on AKD configuration, see page 175

For more details on AKD Firmware Download, refer to the FAQ section.

NOTE After your application is downloaded to the controller you can activate the **Online Configuration** Mode to configure your drives with the **Setup Wizard...** For more details, see page 192

5.4.2.3 Save and Retrieve Parameter Files

The AKD parameters can be saved to non volatile memory in the drive. For more details, refer to paragraph "Save parameters to NV memory" on page 198



#	Name	Vendor	Product	Model	FW Version	Mapped to Axis	Simulated
1	AKD_1	Kollmorgen	AKD EtherCAT Drive (CoE)	AKD-P00306-NAEC	01-08-00-003	(None)	N/A
2	AKD_2	Kollmorgen	AKD EtherCAT Drive (CoE)	AKD-P00306-NAEC	01-09-00-002	(None)	N/A
3	Coupler_3	Kollmorgen	AKT-ECT-000 EtherCAT Fieldbus coupler	N/A	(Unknown)	N/A	N/A

5.4.2.4 AKD Setup Wizard...

The wizard allows you to configure drives once the following conditions have been met:

- The scan has been performed
- Your project is compliant with the physical devices on the EtherCAT network
- You have activated the Online Configuration mode

You then have access to the AKD parameters that are used when the drive is running.

Figure 4-9: AKD Setup Wizard

5.4.2.5 Configure Onboard I/O

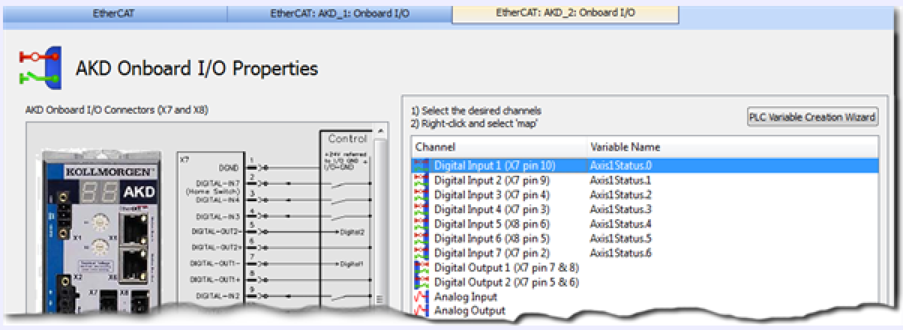
The procedure to define the local I/Os of the AKD drive is very similar to the one for I/O slices, with the following exceptions:

- Channel column also contains in brackets the connector and pin number
- PLC variable selection applies to digital inputs as well as analog inputs and outputs.

5.4.2.6 Digital Input Mode

NOTE

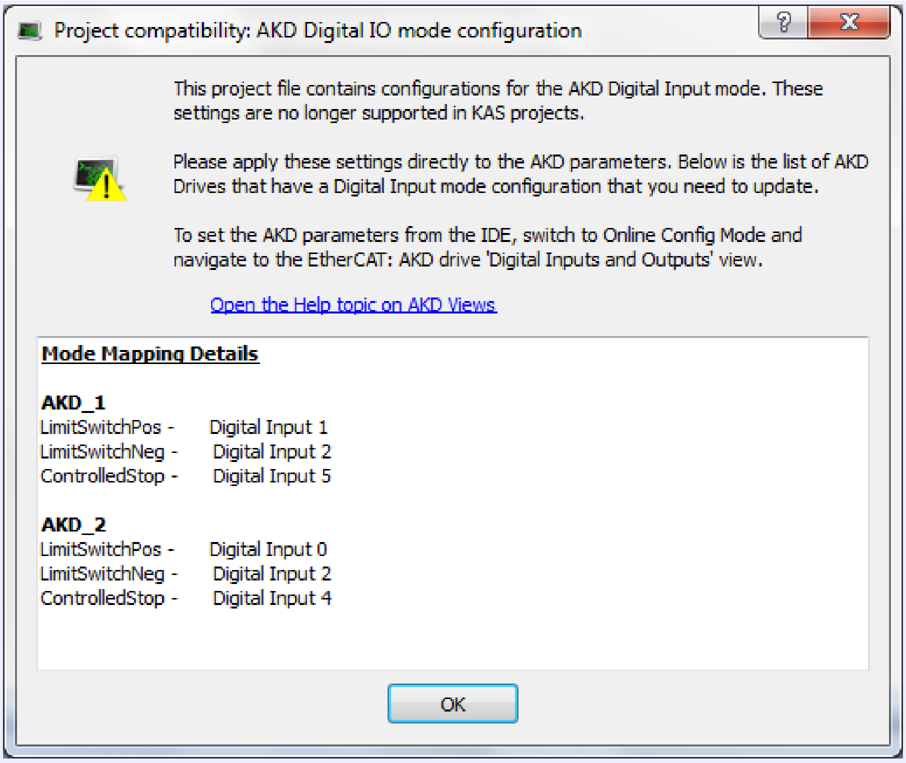
The way to set the Digital Input mode on an AKD drive has changed with KAS 2.6. Previously, this setting was done in the Onboard I/O view (seen below) where the IO was mapped to PLC variables.



The 'Digital Inputs and Outputs' are now accessed using the new AKD GUI integration inside KAS. This will allow you to save this setting inside the drive instead of pushing it every time the application is started.

Files created with KAS 2.5 or earlier, with specific mode settings for any Digital Input of an AKD drive, will automatically be converted to the new project format and these settings will be removed. A dialog box will pop up prior to removal to inform you which settings will be removed from the project.

NOTE

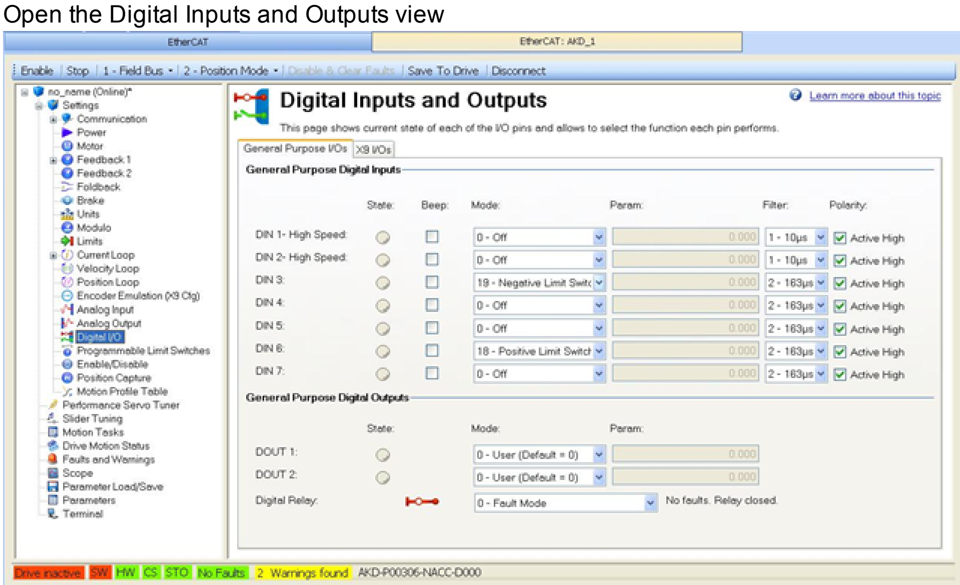


For each drive you will find the list of digital inputs with their specific mode settings. You can copy/paste the content of this text area to a text editor to keep track of them before closing the dialog box.

How to configure Digital Input mode setting inside AKD GUI

To set the mode of a Digital Input

1. Connect to the controller
2. Go to Online Configuration Mode
3. Open the desired drive's AKD GUI by double-clicking on the AKD drive node in the project tree view.
4. Open the Digital Inputs and Outputs view



On this screen, you can set the mode of each Digital IO on the drive using the dropdown list in the Mode column. For more information about this view see .

5.4.3 Step 3 of 15 - Add and Configure I/O Terminal

For local I/O, refer to paragraph "Configure Onboard I/O" on page 193

5.4.3.1 Add the Standard I/O Coupler

1. In the Project Explorer, right-click the EtherCAT node to open the menu
2. Select the **Add Standard I/O Coupler** command
   (this option is only enabled when you are **not** connected to the controller)
3. In the Project Explorer, right-click the Standard I/O Coupler node and select the **Rename** command
4. Click **OK**

NOTE The KAS IDE only supports I/O slices for Standard I/O Couplers. For other devices, you have to manually edit the XML file generated by an external tool (see also paragraph "Motion Bus and Fieldbuses" on page 514)

5.4.3.2 Add the I/O Slice

1. In the Project Explorer, right-click the Standard I/O Coupler node to open the menu
2. Select the **Add I/O Slice** command
3. Choose the I/O slice from the list

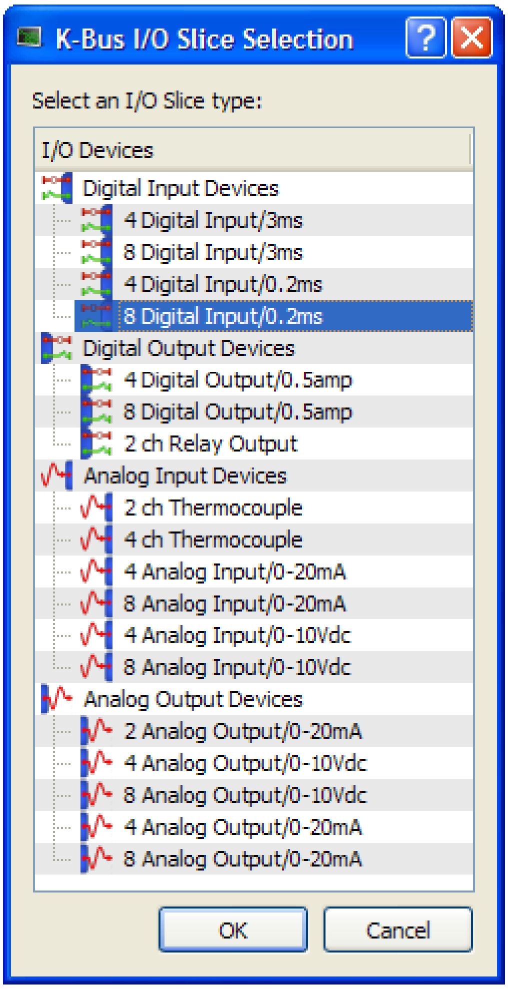


Figure 4-10: Add I/O Slice

4. Click **OK**

5.4.3.3 Configure the I/O Slice

For more details, refer to paragraph "Step 11 of 15 - Map Input and Output to Variables" on page 270

5.4.4 Step 4 of 15 - Configure EtherCAT Motion Bus

TIP Before configuring your EtherCAT settings, you may wish to add third party EtherCAT devices, not including drives. See "Add Third Party EtherCAT Devices" (see page 205) for more information.

Double-click the **EtherCAT** node in the Project Explorer to open the EtherCAT properties dialog in the workspace. This window is composed of three different tabs:

Tab	Description
EtherCAT devices	Displays all the E-Bus devices present in the project tree
"EtherCAT Master Settings" (see page 214)	Allows you to configure the EtherCAT bus master
"ENI File tab" (see page 215)	Allows you to use an external configuration file
"ESI Files" (see page 216)	Display, add, and remove available ESI files

KAS includes an integrated tool to configure the EtherCAT master and start up the fieldbus operation.

The configuration tool enables you to:

- Describe your motion topology as a configuration tree (see procedure in paragraph "EtherCAT Devices" on page 196)
- Associate variables to the I/O channels of devices (see procedure in paragraph "Step 11 of 15 - Map Input and Output to Variables" on page 270)

About Slave devices

Slave devices can support several PDOs (for the list, see page 167). Some of them are mandatory; others are optional.

One of the main tasks of the EtherCAT configuration is to select the PDOs used by each slave (see also "Figure 4-11: EtherCAT Summary Form " on page 197) and group them all in the EtherCAT image.

NOTE PDOs contain real-time cyclic data which is deterministic. Non-cyclic data is not deterministic and is defined by Service Data Objects ("SDO" (see page 718)).

As explained in the introduction, input and output parameters are grouped in predefined blocks called PDOs.

5.4.4.1 EtherCAT Devices

The EtherCAT Devices tab lists all the EtherCAT devices and provides for the ability to discover and map their use.

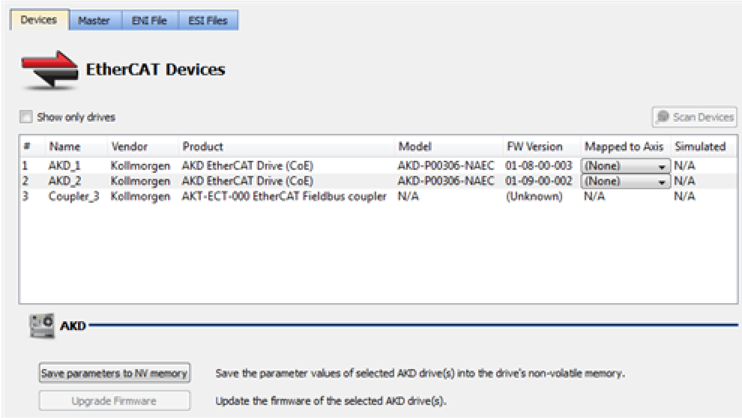


Figure 4-11: EtherCAT Summary Form

Item	Description
Show only drives	This option hides from the list all EtherCAT slaves that are not drives
Scan Devices	The KAS Runtime sends EtherCAT messages to discover the devices present in the network (see explanation below)
Name and Model	The name and model for each device is displayed and ordered by the position in the tree. The model (when available) includes the extension and connectivity options, `NAEC` for example.
FW Version	For the drives, the firmware version is displayed on the conditions that: • The “Scan Devices” routine was run successfully on the EtherCAT network (with AKD drives with version 01-08-000-00 firmware or later) • Firmware was downloaded to the AKD drive(s) while the KAS project was open in the KAS IDE. Otherwise the text displayed: **(Unknown)**
Mapped to Axis	For each drive, it is displayed if it is: • Unassigned: from the drop-down menu, you can choose an axis that has not been assigned (it is applicable either for PLCopen or Pipe Network motion engines). • Already mapped to a physical device: the mapping operation is done using the **Scan Devices** command. See details in paragraph "EtherCAT Mapping Device" on page 200

<table>
<tr><th>Item</th><th>Description</th></tr>
<tr><td>Simulated</td><td>Select this option when you want to simulate the device, which means that the device is not used and no communication to this device is performed through the fieldbus.

For Drives:
<table>
<tr><th>Mapped to an Axis</th><th>Simulated State</th></tr>
<tr><td>No</td><td>Simulation is not applicable</td></tr>
<tr><td>Yes</td><td>If Drive is mapped to a physical drive, then the simulation is Enabled, so you can set state to Yes/No --> Display checkbox
If Drive is not mapped to a physical drive, Simulation is forced to Yes</td></tr>
</table>
</td></tr>
<tr><td>Save parameters to NV memory</td><td>Allows you to save the drives' parameters to the NVRAM of each drive currently selected in the list. This action is enabled only when the Online Mode is activated

TIP: To save a configuration for a specific drive only, right-click on it in the Project Explorer and select the Load/Save Parameter... command in the drive's menu

You will be automatically be prompted with the option to save modified drive parameters if this action has not been performed prior to the following circumstances.
Exiting Online Configuration Mode
Disconnecting from the controller
Closing the project
Exiting the IDE</td></tr>
<tr><td>Upgrade Firmware</td><td>This command triggers a firmware upgrade for the selected drives (you can use Ctrl+A shortcut to select all drives).

For more details, refer to FAQ section.</td></tr>
</table>

Table 4-1: EtherCAT Devices

Scan Devices

The scan process allows the following tasks:

- Discover the devices physically present in the fieldbus network (see "Figure 4-12: EtherCAT Network - Physical View " on page 199)
- Map them to items in the EtherCAT node of the Project Explorer (see "Figure 4-13: EtherCAT Network - Logical View " on page 199)
  Note that the order of the devices in the tree is the same as in the real fieldbus network.

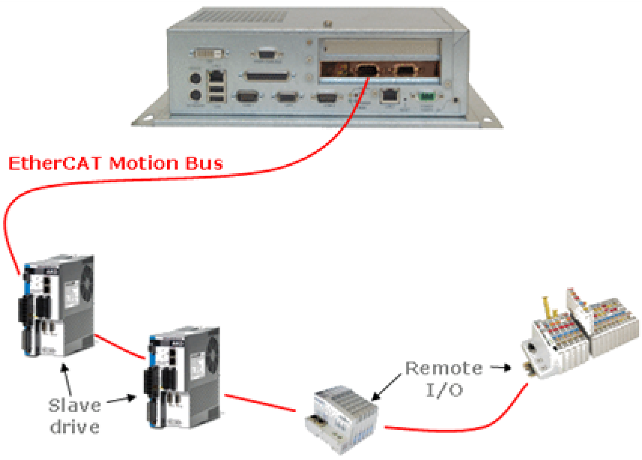


Figure 4-12: EtherCAT Network - Physical View

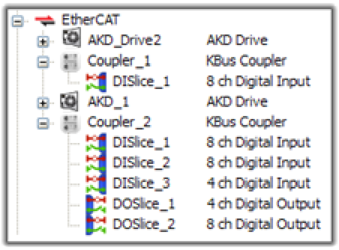


Figure 4-13: EtherCAT Network - Logical View

During the scan operation, all services connected to the EtherCAT network will be discovered and identified. The KAS IDE will list the devices in the order they are connected. Kollmorgen devices, which include AKD and S300 drives, Standard I/O Couplers, and remote I/O terminals (for a list, see "Remote Input/Output (I/O Terminals)" on page 676) will include detailed information for each device.

The status of devices can be determined by their icon in the logical view.

Icon	Meaning	Description
(device icon)	Normal	The device has been added by scanning the system. The associated ESI file has been found. The icon varies by device, but is the icon set by the manufacturer in the ESI file.
	Excluded	The device has been added manually or a discovered device has the Mapped to Axis item set to **None**.
	ESI missing	The ESI file is missing.
	Error	For AKD and S300 devices , shown when there is an error.

Table 4-2: EtherCAT device icon descriptions.

Scan Limitations

- I/O slices for Standard I/O Coupler do not reveal their Device IDs.
- If you plug the EtherCAT cable to the "OUT" port of your IPC (instead of to the "IN"), no error is reported during the scan operation.
- The discovery feature does not differentiate between AKT-DN-004-000 and AKT-DNH-004-000 I/O terminals. Nor between AKT-DN-008-000 and AKT-DNH-008-000.
- Devices other than those made by Kollmorgen will be identified by the Vendor Name (or ID number) and Product Name (or ID number). If the device is missing an ESI file, then you will need to import the ESI file supplied by the device Vendor. TheESI file is

required by the IDE to decode and display the Product Name, Device Description, and other details.

EtherCAT Mapping Device

Below are the procedures to scan and map the physical EtherCAT devices to your project:

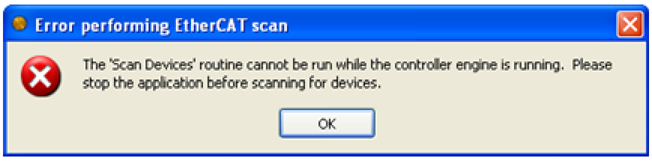
- "EtherCAT Mapping Device" (see page 200) (1st time scan, do devices in the project)
- "EtherCAT Mapping Device" (see page 200) (add or change existing devices in the project)
- "EtherCAT Mapping Device" (see page 200)

When the motion application is started, the network is scanned and the discovered device topology is compared to the project's expected topology. An error is reported if the topologies do not match, and the PLC program will not run. The physical topology must match the project's expected topology for the PLC application to operate the devices properly.

Some important error messages

Virtual Machine Running

If the controller is running a program when the scan process is executed, the following message appears:



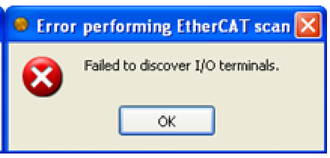
When connected to the Controller, the **Scan Devices**button is disabled if a program is running. However, it is possible to start the scan process when disconnected from the Controller even if the Controller is running a program. It is because the KAS IDE does not know if a program is running until it connects to the Controller.

No EtherCAT Devices Found

If the scan process cannot find any EtherCAT nodes (i.e. the scan process does not encounter errors but finds no EtherCAT device because it cannot communicate to hardware), then the following message appears:

Device Scanning Process Failures

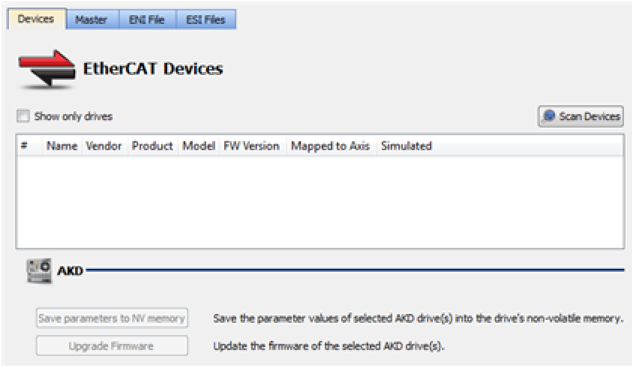
If the scan process fails, one of the following messages (indicating which part of the scanning process encountered an error) appears:



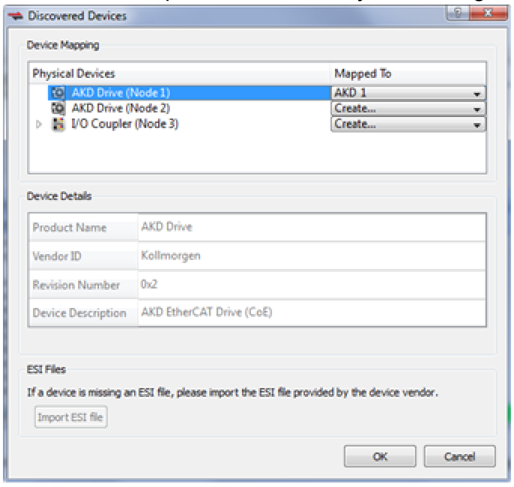
Scan and Map Network Devices

To perform the mapping, follow these steps:

1. In the Project Explorer, double-click the **EtherCAT** node to open its Properties
2. In the Devices tab, click the **Scan Devices** button
   (the topology discovery is only enabled when the controller is **not** running an application)
   If the scan process fails, refer to the error messages



3. If the scan completes successfully, the configuration form appears:



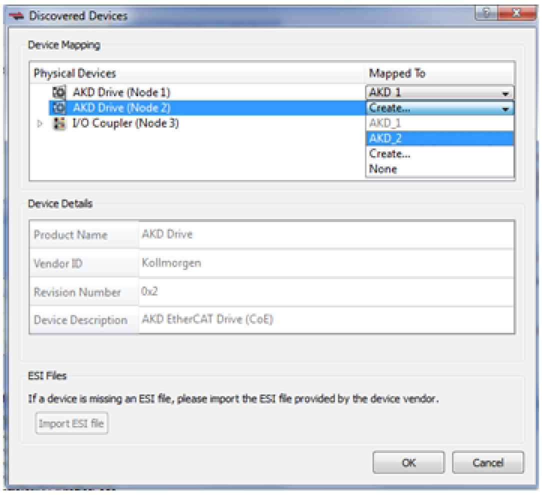
NOTE The order corresponds to the device's position on the physical topology.

Description of the form:

Item	Description
Physical Devices	Displays the devices found during the topology scan operation. To view the I/O terminals, click on the **[+]** box next to the I/O coupler device. All the devices are listed according to the position in the network, including the I/O terminals in the couplers
Mapped To	Displays the item in the project to which the physical device is mapped (or **None** when no mapping is done) See also the explanations in the table below.

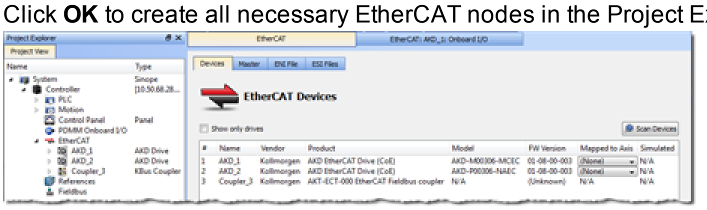
Table 4-3: Mapping Devices - Form Description

4. Select each device not already mapped and choose one of the following options in the **Mapped To** drop-down menu:
   By default, all unmapped devices are set to **Create…**

Mapped To	Description
Create...	Enables you to create a new project node and associate this node to the physical device on the network
None	Indicates that no project node is created or associated with the selected physical device **IMPORTANT** If a physical device is not associated with a project node, it is not possible to start the runtime. If you want a physical device not to be used at run-time, you have to set the Simulated mode in the Devices tab. **Why mapping a device to None?** You have to map a device to None in the following cases: • When you manually add a EtherCAT device in the Project Explorer which is not yet associated to a physical item • When you have removed a physical device from the network and want to keep its node in your project
<Names of existing devices>	List of devices' names that already exist in the project (only devices of the same type and not yet mapped are listed in the drop-down menu) 

5. Choose the **Create...** option to map the physical device to a new device. By default, all unmapped devices are set to **Create…**

6. Click **OK** to create all necessary EtherCAT nodes in the Project Explorer



7. Compile the project to create the EtherCAT "ENI File tab" (see page 215).

Creating the ENI file is necessary to enable the AKD **Setup Wizard...** For more details, see page 192

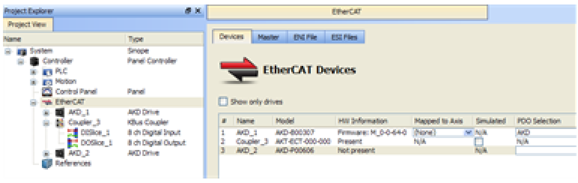
NOTE After the Scan operation, you need to ensure that the selected motor is the correct one

Re-Scan and Re-Map Network Devices

This procedure allows you to update the network topology when some EtherCAT devices were already mapped.

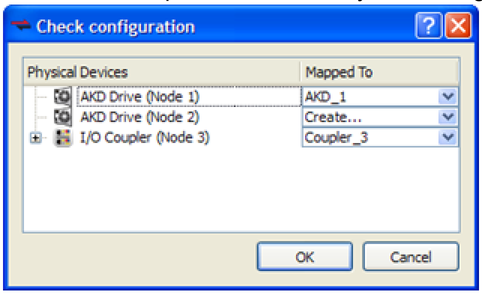
For example, suppose you have the following project configuration:

- The first AKD Drive and a I/O Coupler were created during a previous scan
- The second AKD was added to the project manually, and is not yet associated with any physical drive



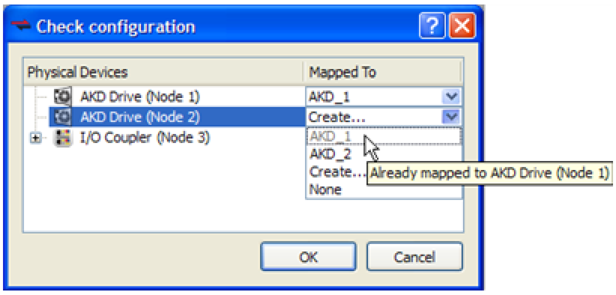
To perform the topology update, follow these steps:

1. In the Project Explorer, double-click the EtherCAT node to open its Properties
2. In the Devices tab, click the **Scan Devices** button
3. If the scan completes successfully, the configuration form appears as follows:



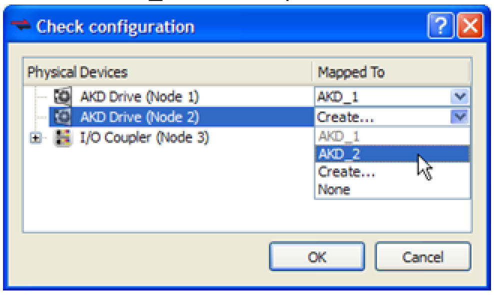
NOTE Each physical device already associated with an EtherCAT node already has its device's name listed in the **Mapped To** column.
AKD Drive (Node 2) is not mapped to any physical device, but the project already contains an unmapped AKD drive that was manually added to the project.

4. To map the physical device **AKD Drive (Node 2)**, open its drop-down menu to list all the devices that are valid for it



NOTE All AKD EtherCAT nodes are listed in the drop-down menu. However, **AKD_1 is disabled** because it is already mapped to the first AKD drive.
If you position the mouse over a disabled item, a tooltip indicates which physical device is currently mapped to that node.
To remove the mapping, select the **None** option in the drop-down menu.

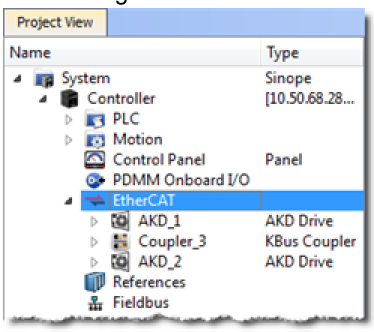
5. Choose **AKD_2** in the drop-down menu

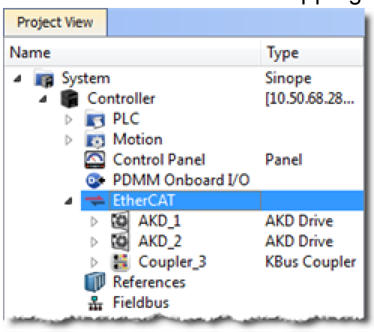


6. Click **OK** to confirm the mapping of the new drives

If, after the mapping process, the KAS IDE detects that the order is not the same, it automatically re-orders the EtherCAT nodes and the I/O terminals in the Project Explorer to match the physical order on the network.

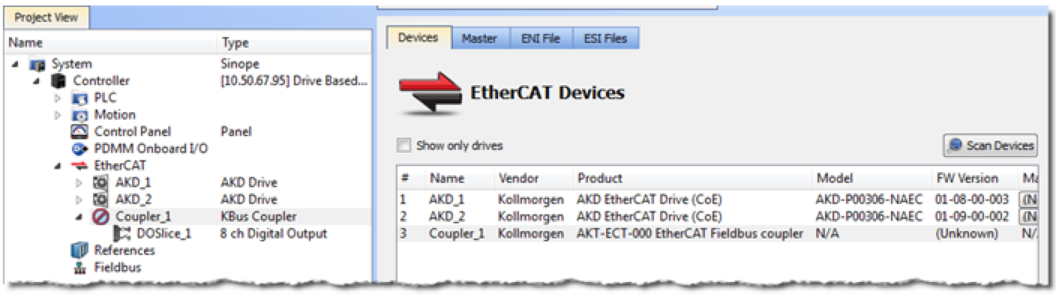
The two figures below show the tree before and after the mapping procedure.



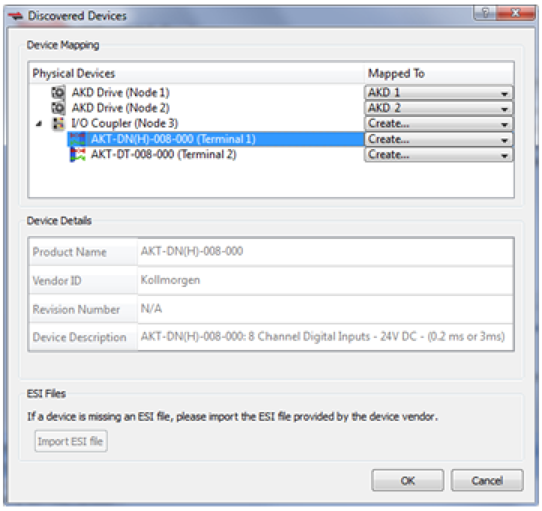


Filling in I/O Terminals

If an I/O coupler is not associated with a physical device, but the I/O terminals are already defined in the project tree, then the KAS IDE automatically associates the terminals.

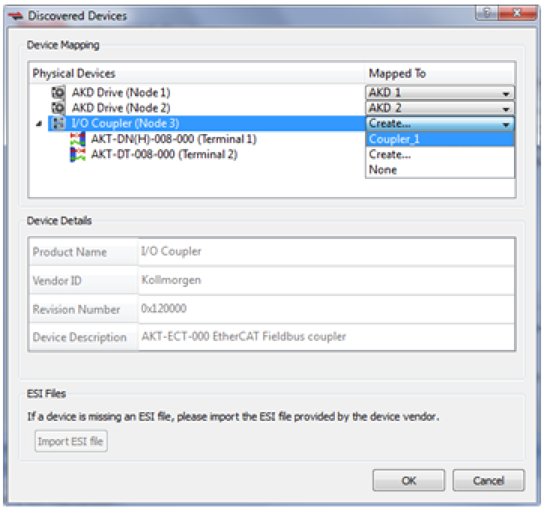


After you click the **Scan Devices** button, the configuration form appears:



NOTE The terminal 2 for the physical device I/O Coupler (Node 3) is identical to what is already declared in the project tree.

If you now map I/O Coupler (Node 3) to Coupler_1 (see figure below), then all the I/O terminals linked to this coupler are automatically associated on a first-match basis as follows:



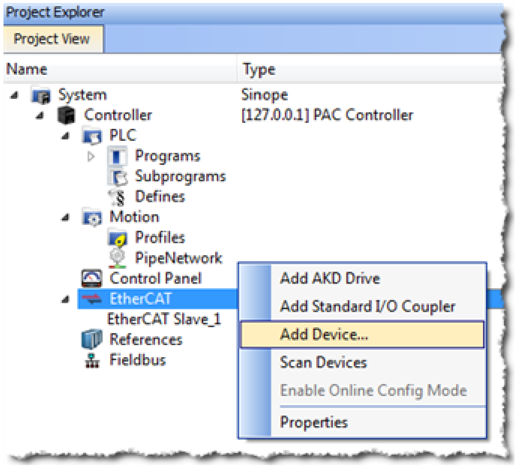
IMPORTANT After changing the configuration of an EtherCAT device, you have to recompile the project and download this new version to save your modifications on the target.

Add Third Party EtherCAT Devices

Third party EtherCAT devices may be added to the EtherCAT node in the Project view. This helps to preconfigure the EtherCAT network in the project before connecting to the controller.

NOTE KAS IDE does not support third party drives. The system scans discovered devices and ESI files to check for compatibility. You will be alerted if a device is found to be a third party drive and that it is not supported.

1. Right-click on the EtherCAT node and select "Add Device..." from the menu.



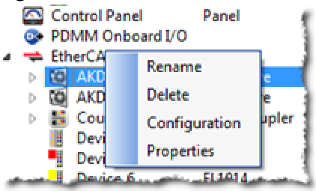
2. Select a device from the EtherCAT Device Selector dialog box.
3. Click OK.

The device is added to the project tree in the EtherCAT node.

General Properties Tab

This tab provides information about the selected EtherCAT device and the associated ESI file. It also provides the ability to import an ESI file if one is not present. There are two ways to access this tab:

- double-click on an EtherCAT device
- right-click on an EtherCAT device and select Properties.



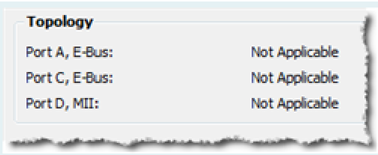
There are two main sections in the dialog: Information, and Topology.

Information

This section details the device's basic information. It also describes the path to the ESI file associated with the device. The **Import ESI File** button is available if the ESI file is missing. This lets you select a file to associate with the device. Lastly, this section lists the **EtherCAT Address** for devices which have been scanned.

Topology

This section lists the device's ports and assignments. There are three possible states for this information.

Condition	Appearance
Complete	
Virtual Device	

Condition	Appearance
ESI file is missing	

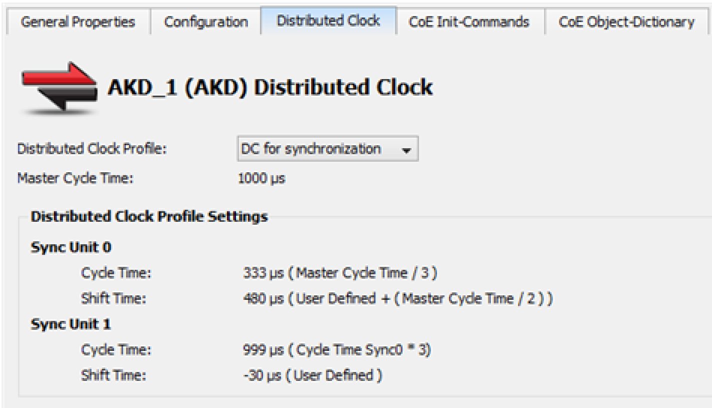
Configuration Tab

Double-clicking on the AKD drive or right-clicking on the drive and selecting Configuration opens this tab. The contents of this tab allow you to set the parameters for the AKD drive.

See "Configure the AKD Drive" (see page 190) for more information.

Distributed Clock tab

The contents of this tab allow you to change settings related to the Distributed Clock for all Kollmorgen and third-party devices (both discovered and manually added).



Element	Description
Distributed Clock Profile	Select the Distributed Clock (DC) operation mode. These modes cannot be edited.
Master Cycle Time	Base interval in microseconds, which will be used by the master. This is changed and automatically updated by changing the **Cycle Time** value on the "EtherCAT Master Settings" (see page 214) tab.
Sync Unit 0	Cycle Time: • Sync Unit Cycle: Unit is synchronized relative to the Master Cycle Time • User defined: Unit has its own interval Shift Time: • Unit is adjusted by the shift time
Sync Unit 1	Cycle Time: • Sync Unit Cycle: Unit is synchronized relative to the Master Cycle Time • Sync 1 Cycle: Unit is synchronized relative to the First Sync Unit • User defined: Unit has its own interval Shift Time: • Unit is adjusted by the shift time

NOTE Some or none of the content will be available under the following scenarios:

- The Sync 0 or Sync 1 parameter is not present
- Distributed Clock is not supported by the device

NOTE
- The ESI file is missing.

Oversampling devices

Some EtherCAT devices have oversampling features. An oversampling device is typically able to record (input) or provide (output) signals at a higher rate than the EtherCAT cycle time. This rate is called the *oversampling factor*. For example, with an oversampling factor of 10 and an EtherCAT cycle time of 1ms (1Khz), an input device can record values every 100µs (1000 divided by 10).

Oversampling devices have as many PDO objects in their cyclic frames as the oversampling factor in order to achieve the higher rate. Each of these PDO objects corresponds to one sample. For example, an output device with an oversampling factor of 4 will typically have 4 PDO objects: Output 1, Output 2, Output 3 and Output 4.

The oversampling factor is tied to the Distributed Clock Profile. A warning is displayed next to the DC profile selection box in the Distributed Clocks tab when a device has oversampling features, as seen below.

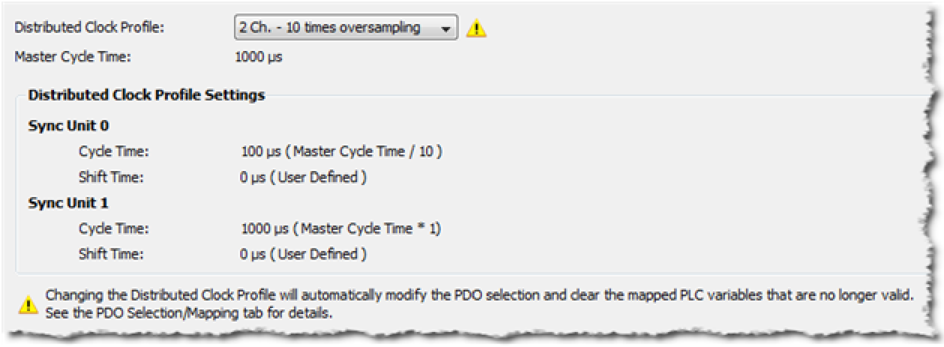


Figure 4-14: Example of a device with oversampling.

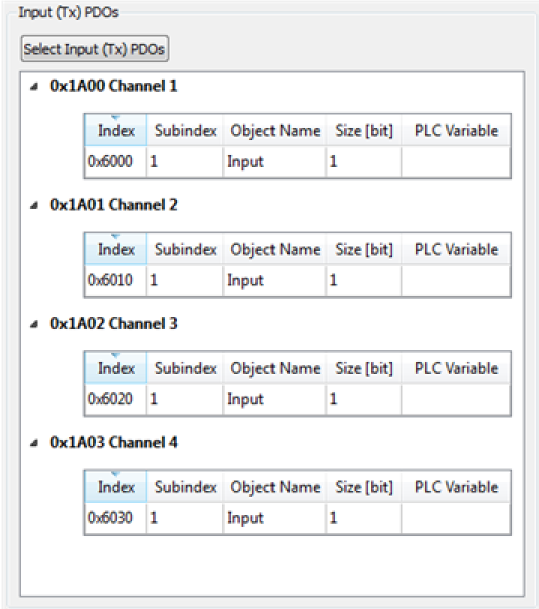
Changing the Distributed Clock Profile will automatically change the PDO selection. All PDOs corresponding to the selected Distributed Clock Profile and its according oversampling factor will be selected. These maps will be discarded if one or several PLC variables were mapped to a PDO that is no longer selected.

PDO Selection/Mapping

This tab includes the PDO configurations for an EtherCAT device.

The assigned PDOs and their objects are viewable for the Inputs (Tx) and Outputs (Rx). The PDOs become active when the EtherCAT network is initialized to operation mode.

Each assigned PDO is listed by its Index (hex) and Name. The objects associated with each PDO are listed below the name. The objects are identified by their object dictionary **Index** and **Subindex**. The **Object Name** provides a simple description. The **Size** determines the data length. The **PLC Variable** mapped to the PDO objects is also listed.



Select Input and Output PDOs

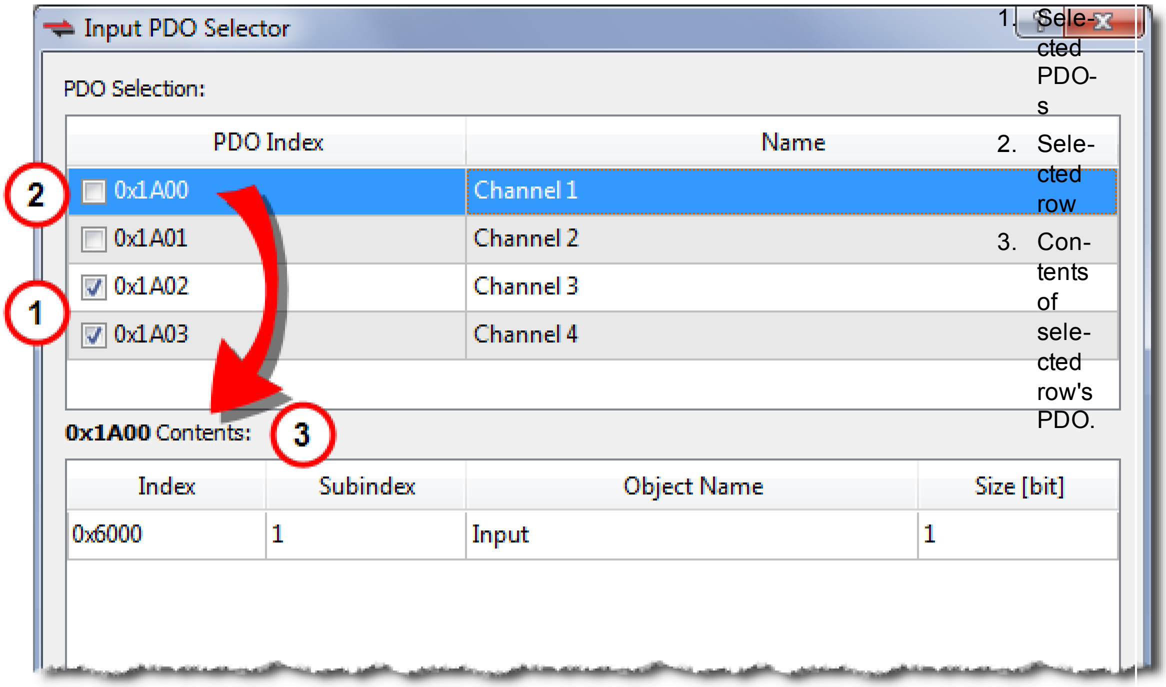
Press the **Select Input (Tx) PDOs** or **Select Output (Rx) PDOs** button to choose the Input or Output PDOs. The appropriate dialog box will open.

- The upper portion contains checkboxes to select specific PDOs for the EtherCAT network cyclic data.
- The lower portion contains the list of object(s) included within a specific PDO.

(S300/700 content)

Viewing the contents of a PDO

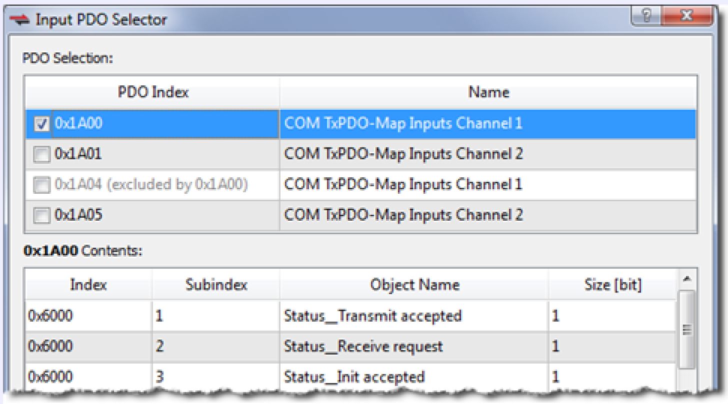
Click on a row in the upper portion of the PDO Selector dialog box to view the contents of that PDO. The lower portion will update to list the associated objects. Using the following example, the PDOs 0x1A02 and 0x1A03 are selected for the input objects but the contents for PDO 0x1A00 are listed because that row is selected..



1. Selected PDOs
2. Selected row
3. Contents of selected row's PDO.

NOTE

- Some EtherCAT devices may not have selectable input and/or output PDOs.
- Some PDOs allow you to select more than one PDO at a time while others are exclusive.
- Exclusive PDOs prevent simultaneously selecting certain other PDOs. Using the following image as an example, choosing PDO 0x1A00 excludes selecting PDO 0x1A04.



- Device vendors determine the PDO(s), content and possible selection exclusivity. This information is defined inside the device vendor's ESI file. Please contact the device vendor for details about a specific device.

Map PLC Variable to PDO Object

PLC variables can be mapped to PDO objects by:

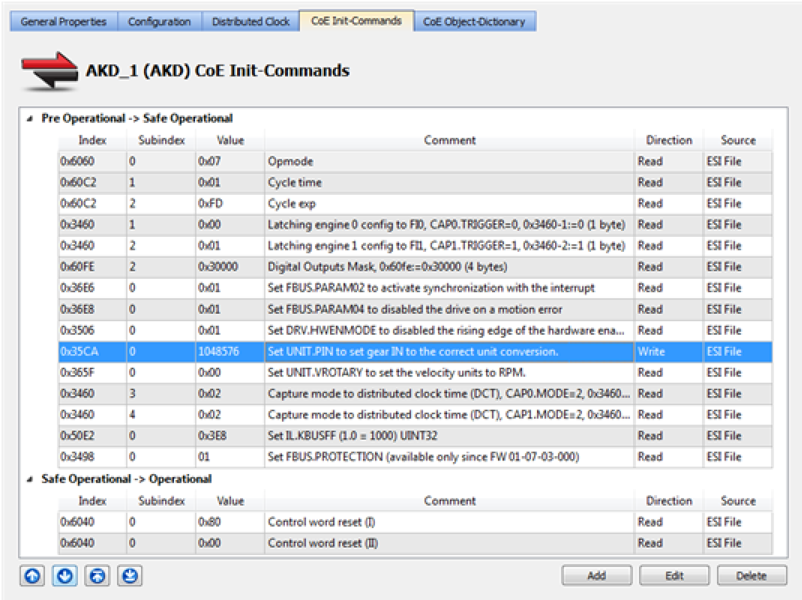
- Double-clicking in a PLC variable cell to open the PLC Variable Selector.
- Right-clicking in a PLC variable cell and select **Map** or **Unmap**.
- Drag-and-drop a variable from the Dictionary to a PLC variable cell.

For more details, please see "Step 11 of 15 - Map Input and Output to Variables" (see page 270).

CoE Init Commands tab

This tab displays the EtherCAT device's CoE Init commands. The Init commands are grouped based on the EtherCAT transition state. The sequence within each group defines the order in which the commands are executed on the device.

The order of CoE Init Commands can be changed to define the correct command sequence required for device operation. Selecting a command enables buttons to move a command Up , Down , to the Top of the table , and the Bottom of the table .

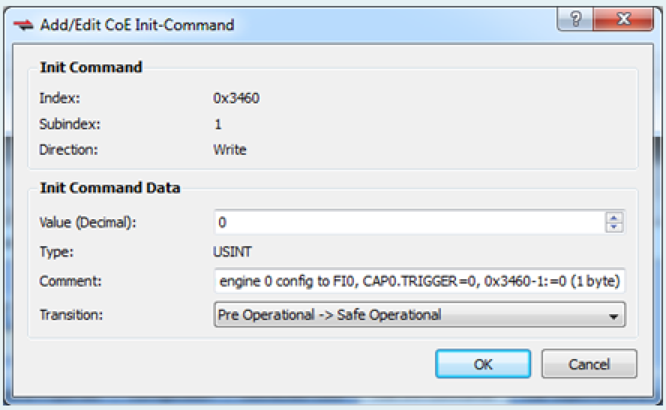


NOTE

- This tab is grayed out if the device does not support the CoE Protocol or the ESI file is unavailable.
- Fixed Init-commands have been deprecated and will not be displayed.

Field	Description
Index	The hex value of the CoE-Index
Subindex	The CoE-Subindex
Value	Value of the init command
Comment	Description of the init command
Direction	Specifies if the command is **Read** or **Write**.
Source	The source of the init command. There are two possible values. • ESI File denotes that the command comes from the ESI file • User denotes that the command is created by the user.

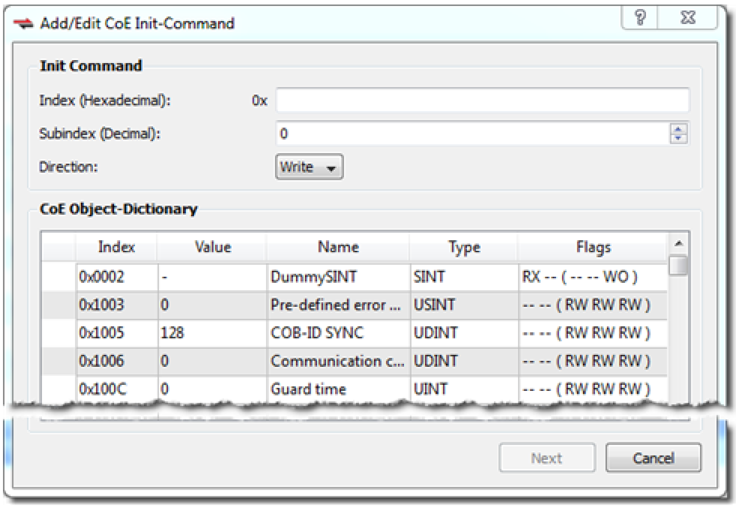
Function	Description
Add	Clicking Add opens a dialog which allows you to create a new Init Command. See "Adding CoE Commands" (see page 212) for more information.
Edit	This function allows you to edit an existing Init Command's attributes. Select an entry and click this button or double-click an entry. The dialog allows you to edit the command data including the **Value**, **Comment**, and **Transition**.



Function	Description
Delete	Delete the selected Init Command row.

Adding CoE Commands

This command allows you to construct a new Init Command by supplying Index, Subindex, Value, Comment, Transition, and Direction attributes. The dialog lists the device's object dictionary and the objects are filtered based on the **Direction** currently selected, e.g. if "Write" is selected then the Read-only objects are not shown.



NOTE

- The **Index** field accepts four characters at most.
- The **Subindex** has a range of 0 - 255.
- When setting the Direction you may select **Read** or **Write**.
- Selecting an item in the CoE Object-Dictionary auto-populates the **Index** and **Subindex** entries.

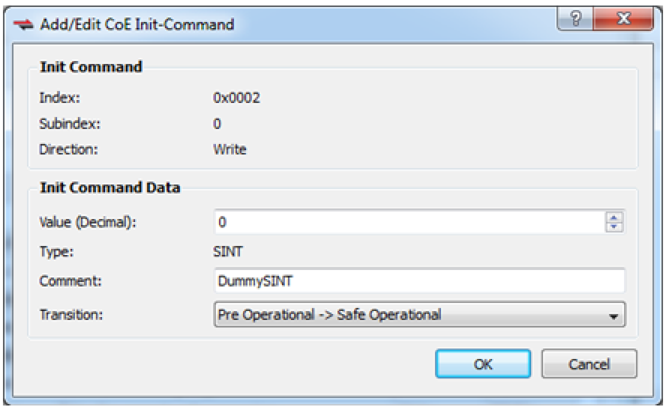
The **Next** button is available once the **Index** and **Subindex** fields have values.

If the Init Command is already present in the dictionary:

If the Init Command is already in the dictionary (i.e. the Index and Subindex you specified are already used) then you are presented with a dialog which allows you to specify the **Value**, a **Comment**, and the **Transition**.

- The Transition may be **Pre Operational -> Safe Operational** or **Safe Operational -> Operational**.
- If the **Type** is "STRING (20)" or "STRING (50)", you may only enter up to 20 or 50 characters.

Clicking **OK** adds the command to the appropriate transition group in the "CoE Init Commands tab" (see page 210) and setes the Source attribute to **User**.

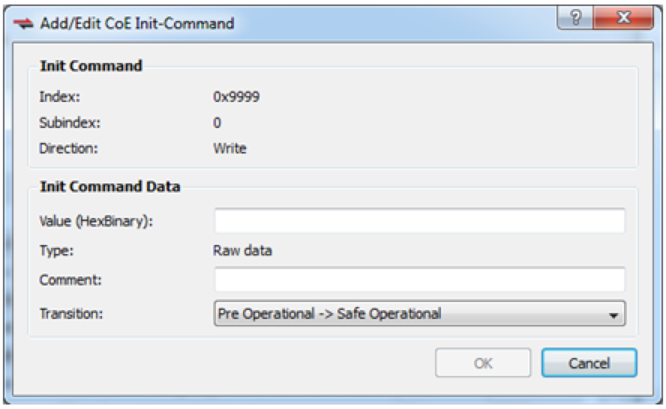


If the Init Command is not present in the dictionary:

Adding a new command (i.e. the Index and Subindex are not in the dictionary) presents you with a dialog which allows you to specify a **Value**, **Comment**, and **Transition**.

- The Value must be entered in HexBinary format.
- The Transition may be **Pre Operational -> Safe Operational** or **Safe Operational -> Operational**.

Clicking **OK** adds the command to the appropriate transition group in the "CoE Init Commands tab" (see page 210) and setes the Source attribute to **User**.

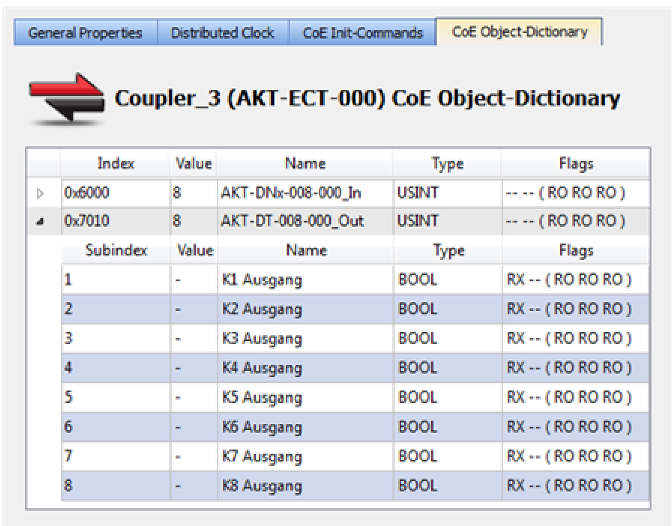


CoE Object Dictionary Tab

This tab displays the all the CoE (CAN over EtherCAT) objects associated with the EtherCAT device. It is used as reference to add new CoE Init Commands for the EtherCAT device. The fields in the table are described below. The CoE objects can be used for three different actions, depending upon the access flags.

- PDO mapping for cyclic communication during application execution
- CoE Init commands during application start-up
- SDO communication during application execution

NOTE This tab is grayed out if the ESI file does not have CoE object information, or if the file is unavailable.



CoE objects can be composed of a simple data type or a complex type. In case of a complex data type, each simple data type composing it has a SubIndex . As shown in the example below, the CoE object can be expanded to show all the subindex's details that are the same as the details for a simple type CoE object.

0x10F3	Diagnostics: History buffer	5	USINT	-- -- (RO RO RO)
Subindex	Name	Value	Type	Flags
1	Max. number of messages	32	USINT	-- -- (RO RO RO)
2	Latest message	0	USINT	-- -- (RO RO RO)
3	Latest acknowledged message	0	USINT	-- -- (RW RW RW)
4	New active message	0	USINT	-- TX (RO RO RO)
5	Control bits	0	UINT	-- -- (RW RW RW)

Field	Description
Index / SubIndex	CoE Index or SubIndex number
Name	Name of the CoE object
Value	Default value of the CoE object
Type	Data type of the CoE object
Flags	This column contains 5 values in the format **XX YY (AA BB CC)**, each representing an access option. The values of XX and YY provide PDO mapping options: XX Specifies if the CoE object can be mapped as RxPDO. It is represented as **RX** if it can be mapped. YY Specifies if the CoE objecct can be mapped as TxPDO. It is represented as **TX** if it can be mapped. The values for AA, BB, and CC provide the CiE Access type. The values can be read only (**RO**), read-write (**RW**), or write only (**WO**) AA Access type of the object in PREOP state BB Access type of the object in SAFEOP state CC Access type of the object in OP state

5.4.4.2 EtherCAT Master Settings

This tab includes configurations for the EtherCAT bus master.

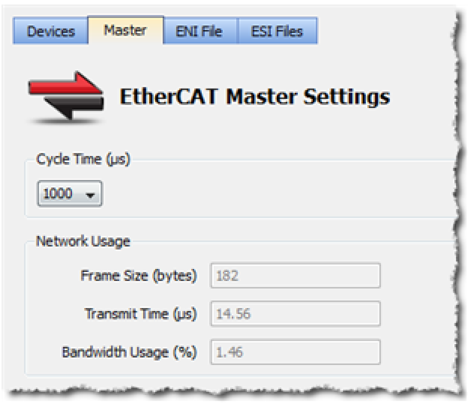


Figure 4-15: EtherCAT Master Settings

Item	Description
Cycle Time	Duration of one cycle in microseconds (time = 250, 500, 1000 µs) to define the time base period for scheduling the motion and the PLC programs (for more details on scheduling, see page 177)
Frame Size	It is the total size (in bytes) of the EtherCAT frame which is sent cyclically. The more EtherCAT slaves (and consequently PDOs) are used in your application, the larger this number is
Transmit Time	It is the time (in microseconds) that it takes to send a frame
Bandwidth Usage	It is an estimation of the percentage of the cycle time used to transmit a frame of data. Bandwidth value goes up when cycle time decreases (see calculation below)

Table 4-4: EtherCAT Cycle Settings - Form Description

The three read-only fields display **(unknown)** when the **Use imported file** option is selected (see "ENI File tab" (see page 215) tab). Otherwise, they are recalculated and refreshed each time that:

- A device is added or removed
- A device simulation state changes
- The **Use imported file** check box is cleared

Bandwidth calculation algorithm

The Bandwidth (BW) usage calculation takes into account the calculated frame size and the Ethernet speed (100 Megabits per second).

```
BW% = Transmission time / Cycle Time
```

With Transmission time (µsec) = (Frame Size in bytes * 8) bits / 100 * 10^6 bps

For example:

If Frame Size = 100 bytes
then Transmission Time = 100*8 / (100*10^6) = 8 µsec

If cycle time = 1000 µsec
then BW% = 8/1000 = 0.8 %

5.4.4.3 ENI File tab

During the compilation, the KAS IDE generates the ENI (EtherCAT Network Information) file based on the EtherCAT devices defined in your project.

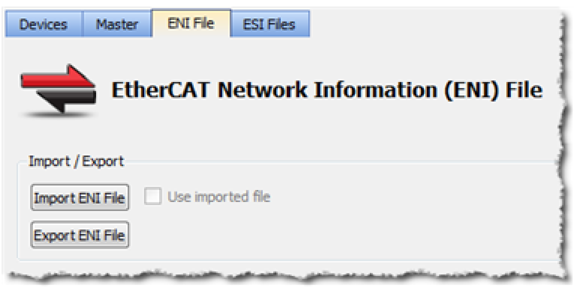


Figure 4-16: ENI File tab

Item	Description
Import ENI File	Enables you to browse and select an ENI file to be imported. If the file is successfully imported into the project, the **Use imported file** option is automatically selected. Once imported, the ENI file is added to your project. This enables you to include EtherCAT devices in your project that are not natively supported by KAS. For more details, refer to "Add Unsupported EtherCAT Device " (see page 520).
Use imported ENI file	Allows you to specify whether or not to use the imported ENI file. See also the paragraph below.
Export ENI File	Enables you to export the ENI file generated by the KAS IDE. You can specify the name and directory for the file. Only the logical devices in the project tree that are mapped to a physical device (and **not** simulated) are taken into account when generating the ENI file. This export can be useful if you want to use the file in another context or with another program.

Table 4-5: ENI File - Form Description

IMPORTANT Importing an external ENI file overrides all EtherCAT project device information and configuration settings in the IDE. The following views and configurations are *not* applicable when using an imported ENI file:

- Project View: All devices located under the EtherCAT node
- EtherCAT Device View tabs:
    - General Properties
    - PDO Selection/Mapping
    - Distributed Clock
    - CoE Init-Commands
- Slice I/O Properties
- Mapping PLC Variables to Slice I/O or PDO objects

Information displayed in the views may not match the imported ENI file.

Using an ENI file

- The KAS IDE works in a degraded mode when using an ENI file, and the Mapped to Axis settings are disabled. This is because the information about the devices in the project tree and the EtherCAT widget table is no longer relevant.
- When using an imported configuration file the following parameters must be manually set for each axis:
    - the type of motion bus
    - its address on the fieldbus ring

  This is done by right-clicking on the Axis Pipe Block and selecting the **Properties** command.
- Scan Devices must be run from EtherCAT Devices before downloading the application to the controller.

5.4.4.4 ESI Files

This tab lists the available ESI (EtherCAT Slave Information) files and provides the ability to add and remove files.

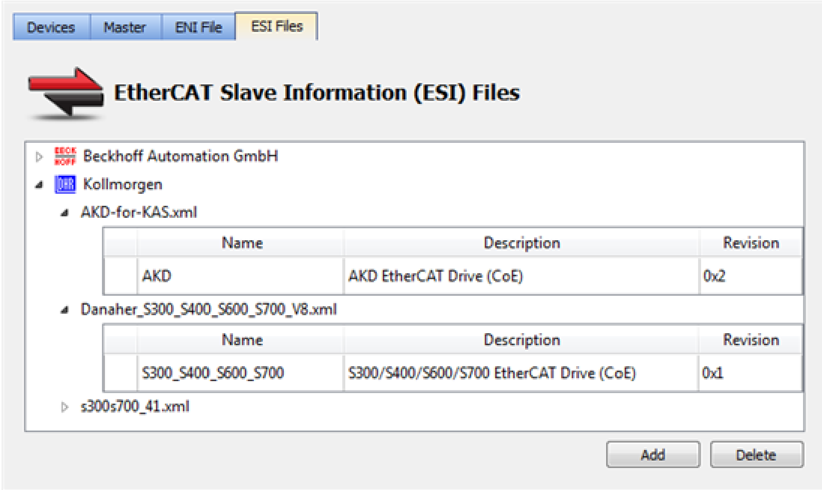


Figure 4-17: The ESI Files tab

NOTE There are two methods for adding ESI files.

- The **Add** button on this tab
- The **Import** button on the Scan Devices dialog (found on the "EtherCAT Devices" (see page 196) tab)

ESI files are sorted by vendor and then the files provided by the vendor. Each file lists the device it supports including

- Device name
- Device description
- Device revision number

ESI files are easily added to the list by clicking the *Add* button and navigating to where the file is saved. The *Delete* button will remove a file from the list but not your hard drive.

IMPORTANT KAS manages AKD devices and requires the default ESI file for AKDs that is pre-installed. Do not import ESI files defined as "Kollmorgen AKD device" or named "AKD-for-KAS.xml".

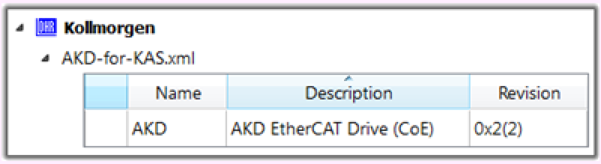


Figure 4-18: Do not overwrite this file.

IMPORTANT **WARNING:** Replacing an ESI file sets the configuration of any devices linked to the ESI file to the new file's defaults. You will be prompted to confirm or cancel replacing an ESI file and the device's configuration. EDI files can be replaced by:

- Pressing the Add button in the ESI Files tab view and:
  - importing an ESI file name that already exists in the ESI file library.
  - importing an ESI file that contains duplicate device information that already exists in the ESI file library.
- Opening a project file that contains an ESI file which is different from the file in the ESI file library.

ESI File Management

ESI files which are referenced from a project's EtherCAT devices are stored in the project.

- Upon opening a project the ESI files are copied to the local folder.

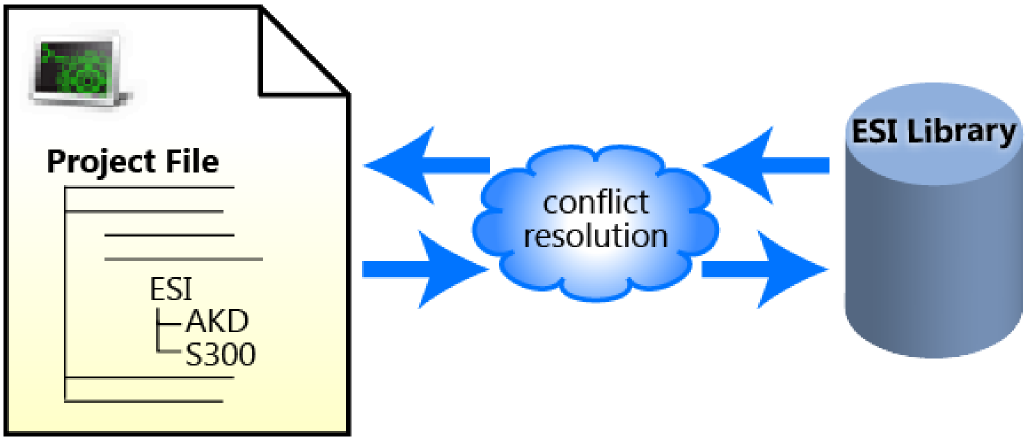


Figure 4-19: Opening — Upon opening a KAS project, the project's ESI files are compared to the internal library. If there are conflicts, you are prompted to resolve them.

- Adding or deleting ESI files from KAS IDE affects the internal library

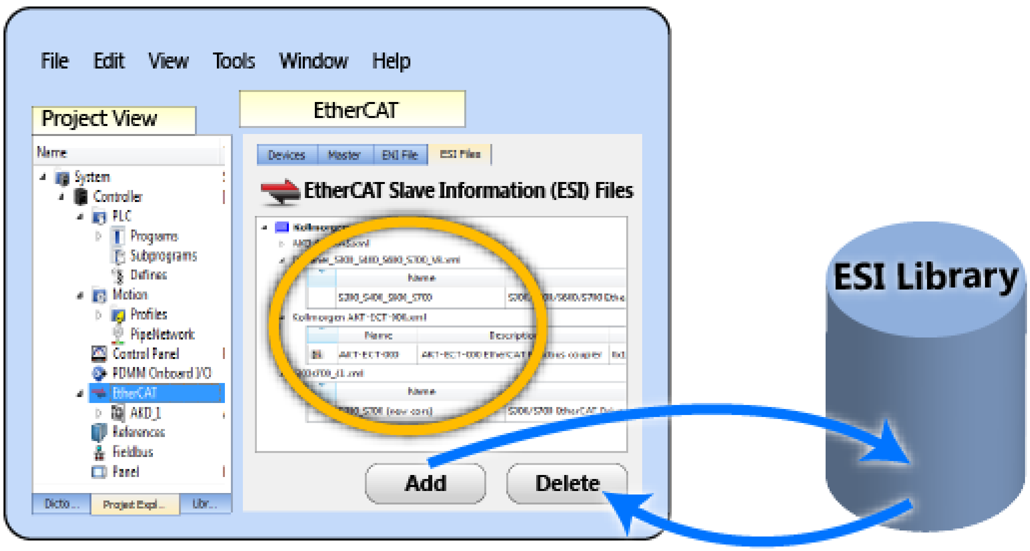


Figure 4-20: Adding/Deleting — Adding or deleting an ESI file from the KAS IDE affects KAS's internal library of ESI files.

- Upon saving a project the ESI files are copied to (project folder)\Controller\ESI.

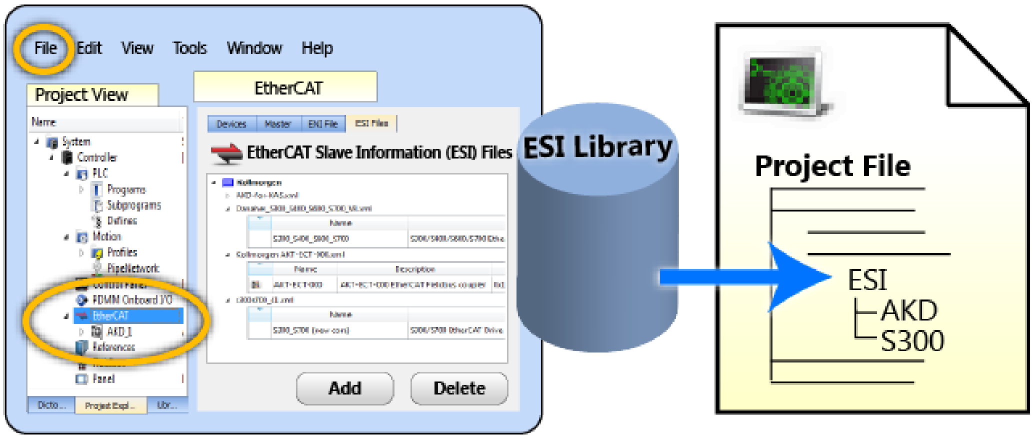


Figure 4-21: Saving — When a KAS project is saved, a copy of the ESI file(s) is included in the project file.

TIP The ESI files are stored along with the project so the project can easily be moved to another location and it will still compile. It still compiles because the ESI files are copied back to the local folder.

NOTE There is the possibility that the names and/or contents of files can be conflicting. The system checks the files before adding them. You will be prompted to chose the correct file should any conflicts be found.

IMPORTANT **WARNING:** Replacing an ESI file sets the configuration of any devices linked to the ESI file to the new file's defaults. You will be prompted to confirm or cancel replacing an ESI file and the device's configuration. EDI files can be replaced by:

- Pressing the Add button in the ESI Files tab view and:
  - importing an ESI file name that already exists in the ESI file library.
  - importing an ESI file that contains duplicate device information that already exists in the ESI file library.
- Opening a project file that contains an ESI file which is different from the file in the ESI file library.

5.4.5 Step 5 of 15 - Create Programs

This chapter provides details on the syntax, structure and use of the declarations and statements supported by the KAS IDE application language.

5.4.5.1 Project Structure

Structuring the application with care is important in creating your project (see "Project Structure Guidelines" (see page 558) in "Advanced Topics" (see page 459)).

5.4.5.2 IEC 61131-3 Editors

The KAS IDE programming environment provides language dedicated editors for:

- Sequential Function Chart (SFC)
- Function Block Diagram (FBD)
- Free Form Ladder Diagram (FFLD)
- Structure Text (ST) and Instruction List (IL)

When SFC must be used?

- SFC must be used when you need to manage sequences of stable process states.
- Using SFC avoids complex switches and the declaration of multiple flags in programs.

When SFC must not be used?

- SFC must never be used as a decision diagram or flow chart for describing an algorithm (i.e. when you think “If / Then / Else...”). This leads to complex SFC charts and bad performances at run-time.
- Never use a step to represent an intermediate point within a calculation. Use ST in this case.

See also "Program Limitations" (see page 87) and the "PLC Online Change" (see page 491) feature.

5.4.5.3 Some Tips...

About Drag-and-Drop

The editor provides you with an ideal programming environment, including drag-and-drop features:

- Drag a variable from Dictionary and drop it into the program to insert it
- Drag a definition from Libraries and drop it into the program to insert its name
- Drag a block and drop it into the program to insert it (you can even select the block from an external text file).
- Drag a function block to the variable list to declare an instance

About Autocompletion

When you type the name of a function block instance (use either as an instance or a data structure), pressing the point "." after the name of the instance opens a pop-up list with the names of possible elements. Click the relevant element and validate it with the check mark.

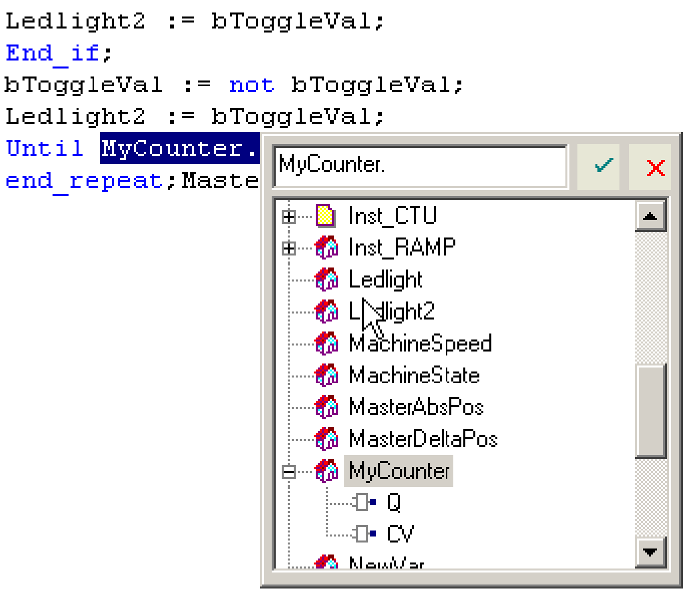


Figure 4-22: Autocompletion

See also "Auto-completion of words" on page 238

About tooltip on variable

When you leave the mouse cursor on a variable in Editors, a tooltip is displayed to give you more details on the item.

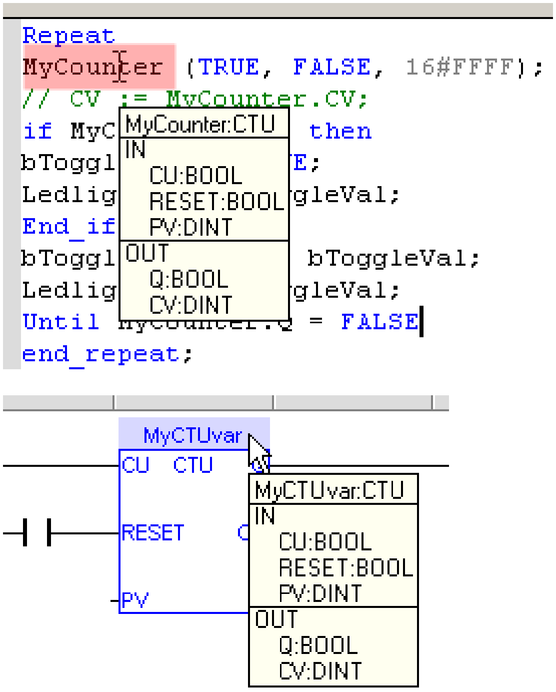


Figure 4-23: Tooltip on Variable

The header of the tooltip displays the name of the variable and its type.

About Bookmarks

See "Bookmarks" (see page 671)

5.4.5.4 Select Function Blocks

All available Operators, functions and function blocks are listed in the Libraries toolbox. The list of available blocks is sorted into categories. The “(All)” category enables you to see the complete list of available blocks.

To insert a block in a program, select it and drag-and-drop it to the desired position in the Editor.

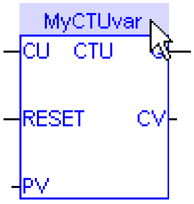
TIP First drag a function block from the Libraries and drop it in the variable list (Dictionary) to declare a new instance. Then drag this instance from the Dictionary and drop it in the program.

5.4.5.5 Select Variables and Instances

Symbols of variables and instances are selected using the variable list in the **Dictionary**. Selecting variables is available from all editors:

- In FBD diagrams, double-click on a variable box, an FB instance name, a contact or a coil to select the associated variable.

- In FFLD diagrams, double-click on a contact, a coil or a block input or output to select the variable. Double-click on the top of an FB rectangle to select an instance.



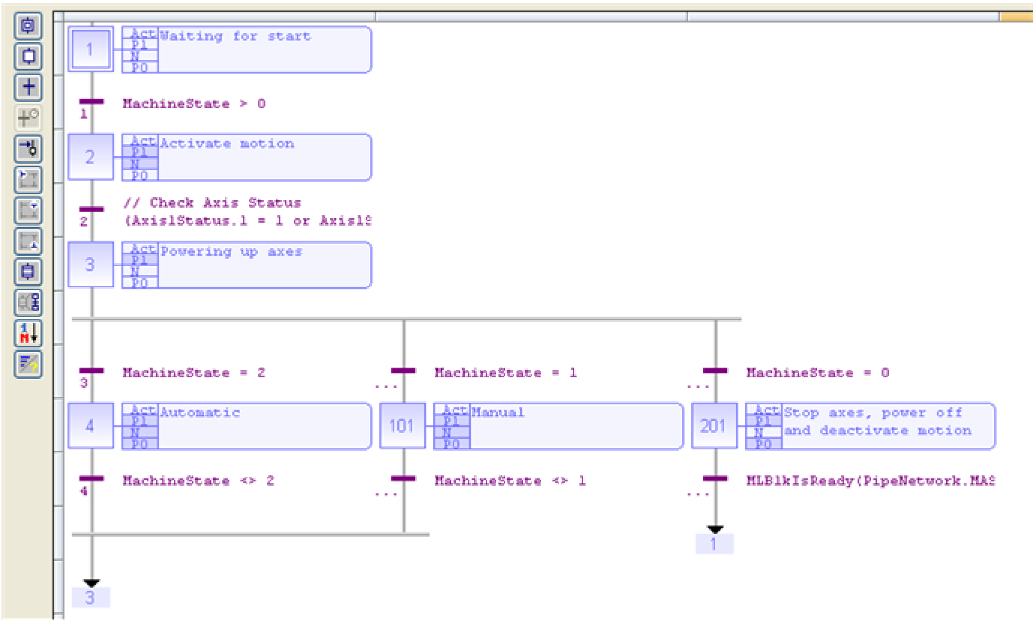
- When the variable editor is visible in the editor window, you can drag a variable from the list and drop it in the program to insert it.

How to access a single bit of an Integer variable?

<variable>.<Bit number> (e.g. `MachineState.7`)

5.4.5.6 Sequential Function Chart (SFC) Editor

The SFC Editor is a powerful graphical tool that enables you to enter and manage Sequential Function Chart according to the IEC 61131-3 standard. The editor supports advanced graphic features such as drag-and-drop, so that you can freely and rapidly arrange the elements of your diagram. It also supports automatic chart formatting when inserting or deleting items, and thus enables quick input using the keyboard.



NOTE For each step, the cells referring to P1, N and P0 actions are colored when they are defined.

SFC diagram components:

Related sections:

TIP

- To change the number of a step, transition or jump, select it and press the **Ctrl+ENTER** keys.
- Hit **Spacebar** on the main corner (on the left) of a divergence or convergence, to set either double or single horizontal line style.

Using the SFC toolbar

The vertical toolbar on the left side of the editor contains buttons for inserting items in the chart. Items are always inserted before the selected item, and the chart is automatically re-arranged when a new item is inserted.

Icon	Description
	Insert an initial step
	Insert a step
	Insert a transition
	Insert a jump to a step
	Insert the main (left side) corner of a divergence or convergence
	Insert a divergence corner
	Insert a convergence corner
	Insert a macro-step
	Insert the body of a macro-step

Table 4-6: SFC Toolbar - List of Icons

Use the following keyboard commands when an item is selected:

- **ENTER**: edit the level 2 of a step or transition
- **Ctrl+ENTER**: change the number of a step, transition or jump

The last button of the toolbar enables you to switch between possible displays:

Swap between possible overviews of level 2 in the level 1 chart:
- display code of actions and conditions
- display notes attached to steps and transitions

Draw SFC divergences

When using the SFC editor, you just need to place items in the grid. The editor calculates and draws lines automatically to link the steps, transitions, and adjusts your place in the chart.

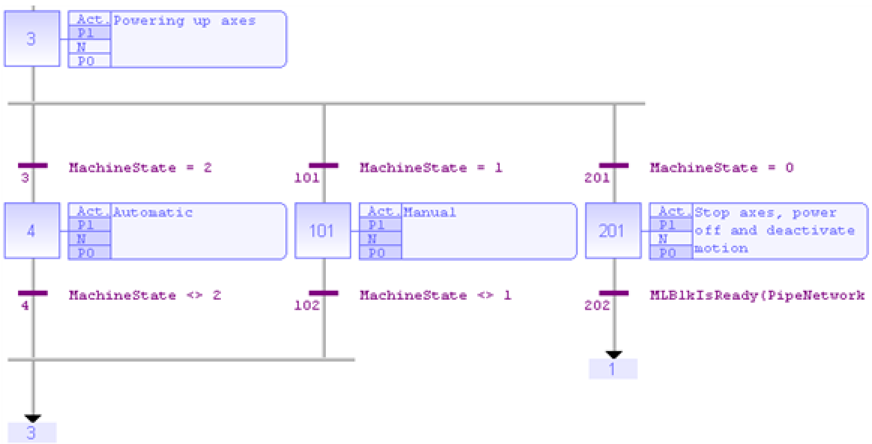
The same method is used for drawing divergences: you just need to place the "corners" that identify divergences, convergences and branches. The editor takes care of drawing vertical and horizontal lines. Use the following buttons in the SFC toolbar:

Insert the main (left side) corner of a divergence or convergence

Insert a divergence corner

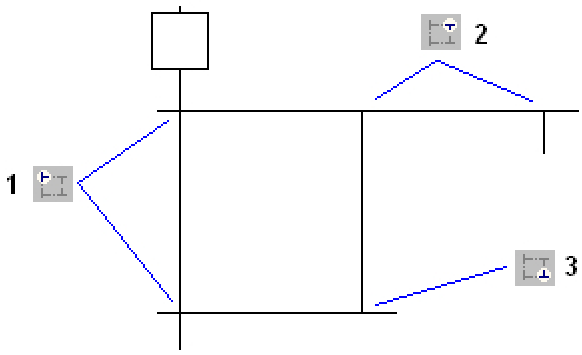
Insert a convergence corner

IMPORTANT Divergences are always drawn from left to right. The first branch, on the left, contains the "corners" that identify the divergence. It must be aligned with the preceding step or transition:



How to proceed?

1- Insert the main corner (on the left-hand side branch) of the divergence and the convergence
2- Insert corners at the top of each branch (divergence)
3- Insert corners at the bottom of the branches where a divergence is required



Simple or double divergence lines:

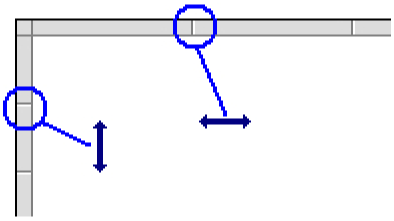
You can change the drawing of a divergence or convergence horizontal line, for drawing simple or double lines according to the SFC definition. To do this, move the selection on the main corner (on the left) and press the **Spacebar**.

View SFC charts

The chart is entered in a logical grid, and all objects are snapped to the grid. You can use the commands of the "View" menu for displaying of hiding grid lines. The (x,y) coordinates of the mouse cursor are displayed in the status bar. This helps you to locate errors detected by the compiler, or to align objects in the chart.

At any moment you can use the commands of the "View" menu for zoomingzoom in or out of the edited diagram using a Ctrl + mouse-wheel operation. You can also press the [+] and [-] keys of the numerical keypad to zoom the diagram in or out.

You can also drag the separation lines in vertical and horizontal rulers to resize the cells of the grid:



The SFC Editor adjusts the size of the font according to the zoom ratio. When a cell is wide enough, a text is displayed with the contents of the step or transition (level 2). The last button of the toolbar enables you to switch between displays:

Swap between possible overviews of level 2 in the level 1 chart:
- display code of actions and conditions
- display notes attached to steps and transitions

Move or copy SFC charts

The SFC Editor fully supports drag-and-drop for moving or copying items. To move an item, select and drag it to the desired position.

To copy an item, do the same, and just press the **Ctrl** key while dragging. It is also possible to drag pieces of a chart from one program to another if both are open and visible on the screen.

At any moment, while dragging items, you can press ESCAPE to cancel the operation.

Alternatively, you can use the Copy / Cut / Paste commands from the Edit menu. The Paste action is performed at the current position.

Enter SFC macro-steps

A macro step is a special symbol that represents, within an SFC chart, a part of the chart that begins with a step and ends with a step. The body of the macro-step must be declared in the same program. The body of a macro-step begins with a special "begin" step with no link before, and ends with a special "end" step with no link after. The symbol of the macros step in the main chart has double horizontal lines.

Use the following buttons of the SFC toolbar to enter macro-steps:

Insert a macro-step

Insert the body of a macro-step

IMPORTANT The symbol of the macro-step and the first step of its body must have the same number. Press Ctrl+ENTER when a macro-step symbol or a first step is selected to change its number.

Renumber steps and transitions

Each step or transition is identified by a number. A jump to a step is also identified by the number of the destination step. The SFC Editor allocates a new number to each step or transition inserted in the chart.

To change the number of a step, transition or jump, select it and press **Ctrl+ENTER**.

It is not possible to change the number of a step or a transition if its level 2 is currently open for editing. The number is used for identifying the step or transition in the level 2 editing window.

In compiler reports, a step is identified by its number prefixed by "GS". A transition is identified by its number prefixed by "GT".

Enter actions of a step

Actions and notes attached to a step (level 2) are entered in a separate window. To open the level 2 editing window of a step or transition, double-click on its symbol in the chart, or select it and press ENTER.

The level 2 editing window proposes five views for entering different types of level 2 information:

- simple actions entered as text
- P1 actions than can be programmed in ST/IL text, FFLD or FBD
- N actions than can be programmed in ST/IL text, FFLD or FBD
- P0 actions than can be programmed in ST/IL text, FFLD or FBD
- text notes

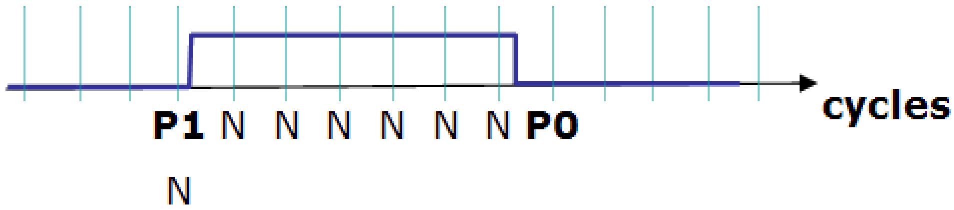
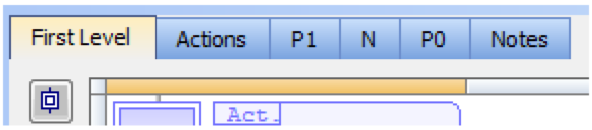


Figure 4-24: SFC Step Action Blocks

Use the tab buttons in the level 2 editing window to select a view:



When editing P1, N or P0 actions, use the radio buttons to select the programming language. This command is not available if the action block is not empty.

The first view ("Action") contains all simple actions for controlling a boolean variable or a child SFC chart. However, it is possible to directly enter action blocks programmed in ST together with other actions in this view. Use the following syntax for entering ST action blocks in the first pane:

ACTION (*qualifier* **) :**
statements...
END_ACTION;

Where *qualifier* is "`P1`", "`N`" or "`P0`".

Enter the condition of a transition

The conditions and notes attached to a transition (level 2) are entered in a separate window. To open the level 2 editing window of a step or transition, double-click on its symbol in the chart, or select it and press ENTER.

The level 2 editing window proposes two views for entering different types of level 2 information:
- condition programmed in ST/IL text or FFLD
- text notes

Use the tab buttons in the level 2 editing window for selecting a view:

When editing the condition, use the "Edit / Set Language" menu command to select the programming language. This command is not available if the condition is not empty. FBD cannot be used to program a condition.

Enter notes for steps and transitions

The SFC editor supports the definition of text notes for each step and transition. The notes are entered in the level 2 editing window of steps and transitions. Refer to the following topics for further information about the level 2 editing window:

- entering Level 2 for steps
- entering Level 2 for transitions

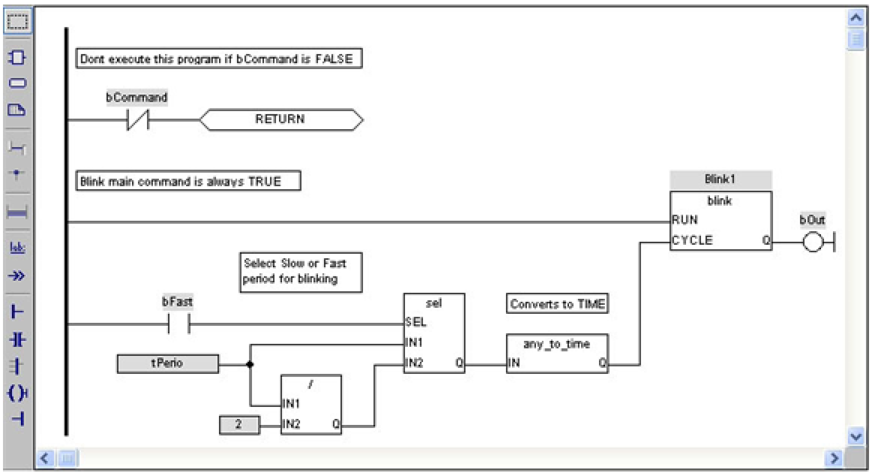
Notes can be displayed in the chart. The last button of the toolbar enables you to switch between possible displays:

 Swap between possible overviews of level 2 in the level 1 chart:
- display code of actions and conditions
- display notes attached to steps and transitions

Notes have no meaning for the execution of the chart. Entering notes for steps and transitions enables you to enhance the auto-documentation of your programs. It also provides an easy way to write and exchange specifications of an SFC program before actions and conditions are programmed.

5.4.5.7 Function Block Diagram (FBD) Editor

The FBD Editor is a powerful graphical tool that enables you to enter and manages Function Block Diagrams according to the IEC 61131-3 standard. The editor supports advanced graphic features such as drag-and-drop, object resizing and connection line routing features, so that you can rapidly and freely arrange the elements of your diagram. It also enables you to insert in a FBD diagram graphic elements of the FFLD (Ladder Diagram) language such as contacts and coils.



FBD diagram components:

Related sections:

Function blocks
Variable tags
Comment texts
Corners
Network breaks
Labels
Jumps
Use of ST instructions

FFLD components:

Contacts
Coils
"OR" vertical rail
Power rails

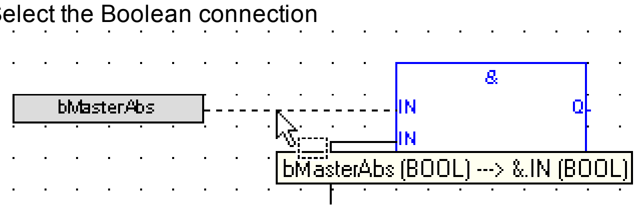
Using the FBD toolbar
Selecting function blocks
Drawing connection lines
Selecting and entering variables and FB instances
Viewing the diagram
Moving or copying parts of the diagram
Inserting an object on a line
Resizing objects
Bookmarks

NOTE When a contact or a coil is selected, you can press the **Spacebar** to change its type (e.g. normal, negated, pulse)

Boolean connections can be negative at the entry of a block.

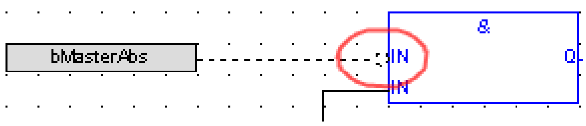
How to toggle the connection to make it negative?

1. Select the Boolean connection



Connections in FBD Programs

2. Press the **Spacebar** (a small circle is displayed)



Toggle Connection in FBD Programs

Execution order can be displayed.

How to display the execution order?

Data flow is executed from top left to bottom right (**CTRL+d** shows the execution order)

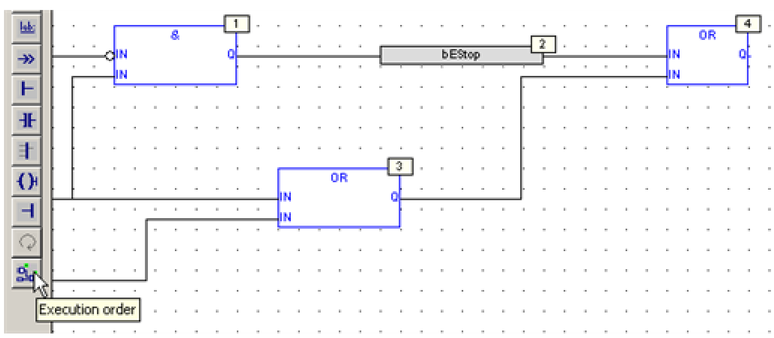


Figure 4-25: Execution Order on FBD

Using the FBD toolbar

The vertical toolbar on the left-hand side of the editor contains buttons for all available editing features. Push the desired button before using the mouse in the graphic area.

Icon	Description
	**Selection**: In this mode, you cannot insert any elements in the diagram. The mouse is used to select object and lines, select tag name areas, or move or copy objects in the diagram. At any moment you can press the ESCAPE key to go back to the Selection mode.
	**Insert Block**: In this mode, the mouse is used for inserting blocks in the diagram. Click in the diagram and drag the new block to the desired position. The type of block inserted is the one currently selected in the list of the main toolbar.
	**Insert variable**: In this mode, the mouse is used for inserting variable tags. Variable tags can then be wired to the input and output pins of the blocks. Click in the diagram and drag the new variable to the desired position.
	**Insert comment text**: In this mode, the mouse is used for inserting comment text areas in the diagram. Comment texts can be entered anywhere. Click in the diagram and drag the text block to the desired position. The text area can then be selected and resized.
	**Insert connection line**: In this mode, the mouse is used to wire the input and output pins of the diagram objects. The line must always be drawn in the direction of the data flow: from an output pin to an input pin. The FBD editor automatically selects the best routing for the new line. You can change the default routing by inserting corners on lines. (see below).You also can drag a line from an output pin to an empty space. In this case, the editor automatically finishes the line with a user-defined corner so that you can continue drawing the connection to the desired pin and force the routing while you are drawing the line.
	**Insert corner**: In this mode, the mouse is used for inserting a user-defined corner on a line. Corners are used to force the routing of connection lines, as the FBD editor imposes a default routing only between two pins or user-defined corners. Corners can then be selected and moved to change the routing of existing lines.
	**Insert network break**: In this mode, the mouse is used for inserting a horizontal line that acts as a break in the diagram. Breaks have no meaning for the execution of the program; they just help in understanding big diagrams, by splitting them into a list of networks.
	**Insert label**: In this mode, the mouse is used for inserting a label in the diagram. A label is used as a destination for jump symbols (see below).
	**Insert jump**: In this mode, the mouse is used to insert jump symbols in the diagram. A jump indicates that the execution must be directed to the corresponding label (having the same name as the jump symbol). Jumps are conditional instructions. They must be linked on their left-hand side to a Boolean data flow.

Icon	Description
	Insert left power rail: In this mode, the mouse is used to insert a left power rail in the diagram. A left power rail is an element of the FFLD language, and represents a "TRUE" state that can be used to initiate a data flow. Power rails can then be selected and resized vertically according to the desired network height.
	Insert contact: In this mode, the mouse is used to insert a contact in the diagram, as in Ladder Diagrams.
	Insert "OR" rail: In this mode, the mouse is used to insert a rail that collects several Boolean data flows for an "OR" operation, in order to insert parallel contacts, as in Ladder Diagrams.
	Insert coil: In this mode, the mouse is used to insert a coil in the diagram, as in Ladder Diagrams. It is not mandatory that a coil be connected on its right-hand side.
	Insert right power rail: In this mode, the mouse is used to insert a right power rail in the diagram. A right power rail is an element of the FFLD language, and is commonly used for terminating Boolean data flows. However, it is not mandatory to connect coils to power rails. Right power rails have no meaning for the execution of the diagram.
	Swap item style: change the text justification
	Execution order: the data flow can be displayed

Table 4-7: FBD Toolbar - List of Icons

FBD variables

All variable symbols and constant expressions are entered in FBD diagrams using small boxes.

1. Press the following button in the FBD toolbar to insert a variable tag:

   Insert variable: In this mode, the mouse is used for inserting variable tags. Click in the diagram and drag the new variable to the desired position.

2. Double-click on a variable tag to open the variable selection box
3. Either select the symbol of the desired variable or enter a constant expression

Variables tags must then be linked to other objects such as block inputs and outputs using connection lines.

4. You can resize a variable box vertically in order to display, together with the variable name, its tag (short comment text), its description text, plus its I/O location if the variable is mapped to an I/O channel.

The variable name is always displayed at the bottom of the rectangle:

tag
description
% location
name

For more details on Variable Tag and Description, see page 605

FBD comments

Comment text area can be entered anywhere in an FBD diagram.

Press the following button in the FBD toolbar to insert a new comment area.

In this mode, the mouse is used to insert comment text areas anywhere in the diagram.

Double-click on the comments area to enter or change the attached text. When selected, comment texts can be resized.

NOTE You can insert hyperlink on external files as shown below. Only TXT and BMP extensions are allowed. When the link is valid, the hyperlink is replaced with the file's content.

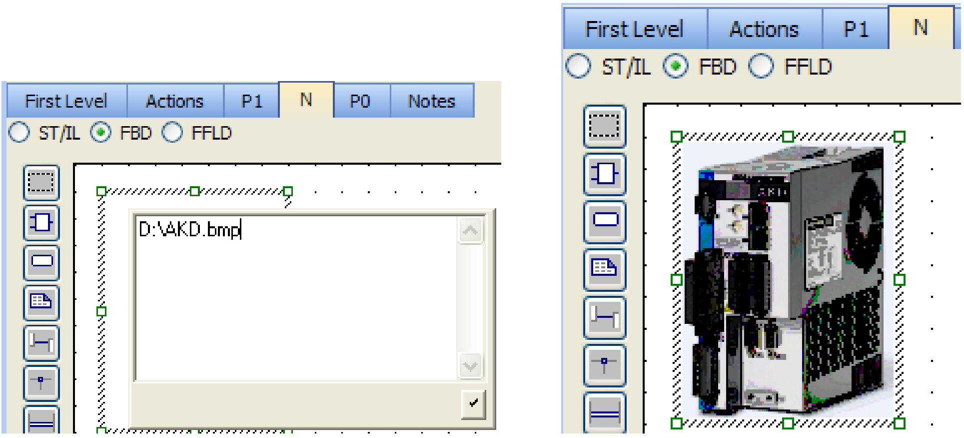


Figure 4-26: FBD Comments - Inserting Graphic

FBD corners

Corners are used to force the routing of connection lines, as the FBD editor imposes a default routing only between two pins or user-defined corners. All variable symbols and constant expressions are entered in FBD diagrams using small boxes.

Press the following button in the FBD toolbar to insert a corner on a line:

Insert corner: In this mode, the mouse is used to insert a user-defined corner on a line.

You can drag a new line from an output pin to an empty space. In this case, the editor automatically finishes the line with a user-defined corner, so that you can continue drawing the connection to the desired pin and force the routing while you are drawing the line.

Corners can then be selected and moved to change the routing of existing lines.

FBD network breaks

Network breaks can be entered anywhere in an FBD diagram. Breaks have no meaning for the execution of the program; they just help in understanding big diagrams, by splitting them into a list of networks. Press the following button in the FBD toolbar to insert a new break:

Insert network break: In this mode, the mouse is used for inserting a horizontal line that acts as a break in the diagram.

The break line is drawn on the whole diagram width. No other object can overlap a network break. Break lines can then be selected and moved vertically to another location.

Network breaks can also be used to browse the diagram. Press the **Ctrl+Page Up** or **Ctrl+Page Down** keys to move the selection to the next or previous network break.

FBD "OR" vertical rail

The FBD Editor enables the drawing of FFLD rungs. The "OR" rail can be inserted on a rung in order to connect parallel contacts together.Press the following button in the FBD toolbar to insert a new "OR" rail:

Insert "OR" rail: In this mode, the mouse is used for inserting a rail that collects several Boolean data flows for an "OR" operation, in order to insert parallel contacts, as in Ladder Diagrams.

The "OR" rail has exactly the same meaning as an "OR" block regarding the execution of the diagram.

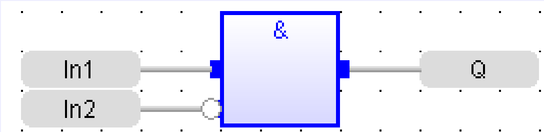
Draw FBD connection lines

Press this button before inserting a new line.

NOTE

As shown below, the editor enables you to terminate a connection line with a boolean negation represented by a small circle:

(* use of a negated link: Q is IN1 AND NOT IN2 *)



To set or remove the boolean negation, select the line and press the **Spacebar**.

Connection lines must always be drawn in the direction of the data flow: from an output pin to an input pin. The FBD editor automatically selects the best routing for the new line. Connection lines indicate a data flow between the following possible objects:

NOTE

Line is colored in red when the two linked items are not the same type.

Block: Refer to the help on the block for the description of its input and output pins, and the expected data types for the coherence of the diagram.

Variable: A variable can be connected on its right-hand side (to initiate a flow) or on their left-hand side to force the variable, if it is not "read only". The flow must fit the data type of the variable.

Jump: a jump must be connected on its left-hand side to a Boolean data flow.

Left power rail: Left power rails represent a TRUE state and can be connected to a non limited number of objects on their right-hand side.

Contact: A contact must be connected on its left-hand side and on its right-hand side to Boolean data flows.

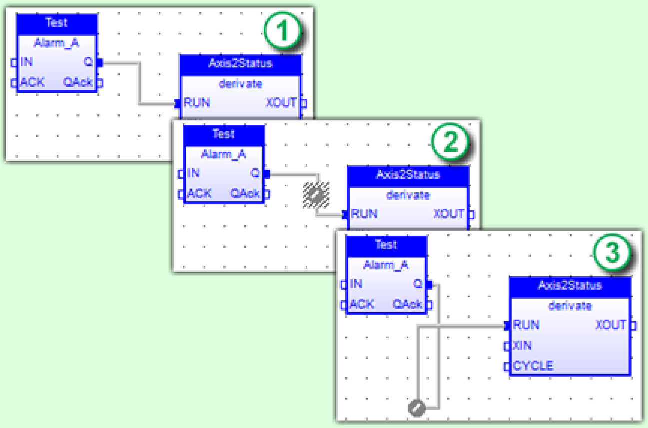
"OR" rail: Such a rail collects several Boolean data flows for an "OR" operation, in order to insert parallel contacts, as in Ladder Diagrams. It may have several connections on its left-hand side and on its right-hand side. All connected data flows must be Boolean.

Coil: A coil must be connected on its left-hand side to a Boolean data flow. It is not mandatory that a coil be connected on its right-hand side.

Right power rail: A right power rail is an element of the FFLD language, and is commonly used for terminating Boolean data flows. It has an unlimited number of connections on its left-hand side. It is not mandatory to connect coils to power rails.

TIP

Connection lines automatically move and follow FBD elements (1) and you can also manually specify the corners. Simply double-click on the line to show a "handle" (2) which can be repositioned (3).



Select FBD variables and instances

Press this button or press ESCAPE before any selection.

To select the name of the declared variable to be attached to a graphic symbol, you must be in "Selection" mode. Simply double-click on the tag-name gray area. The following types of object must be linked to valid symbols:

Block: If it is a function block, you must specify the name of a valid declared instance of the corresponding type.

Variable: Must be attached to a declared variable. Alternatively, a variable box may contain the text of a valid constant expression.

Label: Must have a name. The name must be unique within the diagram.

Jump: Must have the same name as its destination label.

Contact: Must be attached to a declared Boolean variable.

	**Coil**: Must be attached to a declared Boolean variable.

Symbols of variables and instances are selected using a variable list, that can be used as the variable editor. Simply enter a symbol or constant expression in the edit box and press OK. You can also select a name in the list of declared objects, or declare a new variable by pressing the "Create" button.

For more details, see page 221

View FBD diagrams

The diagram is entered in a logical grid. All objects are snapped to the grid. You can use the commands of the **View** menu to display of hiding the points of the grid. The (x,y) coordinates of the mouse cursor are displayed in the status bar. This helps you to locate errors detected by the compiler, or to align objects in the diagram.

At any moment you can use the commands of the "View" menu for zoomingzoom in or out of the edited diagram by means of a Ctrl + mouse-wheel operation. You can also press the [+] and [-] keys of the numerical keypad to zoom the diagram in or out.

Move or copy FBD objects

	Press this button or press ESCAPE before selecting objects

The FBD editor fully supports drag-and-drop for moving or copying objects. To move objects, select and drag them to the desired position.

To copy objects, you can do the same, and just press the CONTROL key while dragging. It is also possible to drag pieces of diagrams from one program to another if both are open and visible on the screen.

At any moment, while dragging objects, you can press ESCAPE to cancel the operation.

Alternatively, you can use the Copy / Cut / Paste commands from the Edit menu. When you run the Paste command, the editor changes into "Paste" mode, with a special mouse cursor. Click in the diagram and move the mouse cursor to the desired position for inserting pasted objects.

Using the keyboard

When graphic objects are selected, you can move them in the diagram by pressing the following keys:

`Shift + Up`	Move to the top
`Shift + Down`	Move to the bottom
`Shift + Left`	Move to left
`Shift + Right`	Move to right

When an object is selected, you can extend the selection by pressing the following keys:

`Shift + Control + Home`	Extend to the top: select all objects before the selected one
`Shift + Control + End`	Extend to the bottom: select all objects after the selected one

To insert or delete space in the diagram, you can simply select an object, press Shift+Ctrl+End to extend the selection, and then move selected objects up or down.

Auto alignment

When objects are selected, the following keystrokes automatically align them:

`Control + Up`	To the top
`Control + Down`	To the bottom
`Control + Left`	To left
`Control + Right`	To right

Insert FBD objects on a line

The FBD editor enables you to insert an object on an existing line and automatically connect it to the line. This feature is available for all objects having one input pin and one output pin, such as variable boxes, contacts and coils. This feature is mainly useful when entering elements of Ladder Diagrams. Just draw a horizontal line between left and right power rails: it is the rung. Then you can simply insert contacts and coils on the line to build the FFLD rung.

Resize FBD objects

Press this button or press ESCAPE before selecting objects.

When an object is selected, small square boxes indicate how to resize it with the mouse. Click on the small square boxes to resize the object in the desired direction.

Not all objects can be resized. The following table indicates possible operations:

Variable	Horizontally and vertically (*)
Block	Horizontally
Labels and jumps	Horizontally
Power rails	Vertically
OR rail	Vertically
Comment area	In all directions

(*) Resizing a variable box vertically enables you to display together with the variable name its tag (short comment text), its description text, plus its I/O location if the variable is mapped to an I/O channel. The variable name is always displayed at the bottom of the rectangle:

% location
description
tag
name

5.4.5.8 Structured Text (ST) / Instruction List (IL) Editor

The ST / IL editor is a powerful language-sensitive text editor dedicated to IEC 61131-3 languages. The editor supports advanced graphic features such as drag-and-drop, syntax coloring and active tooltips for efficient input and test of programs in ST and IL.

```
Blinker (TRUE, t#2s);
Trigger (Blinker.Q);

bSig := Trigger.Q;

Counter (
    bSig,           (* blinking input *)
    not bCommand,   (* reset the counter if command *)
    255
    );
iValue := Counter.CV;
```

Related sections:

Language selection
Syntax coloring
Autocompletion of words
Drag-and-drop
Active tooltips
Selecting function blocks
Inserting variable and FB instances symbols
Reading output of a FB instance
Bookmarks

The ST / IL editor also supports context sensitive help. Place the caret on a keyword or on the name of function or function block and hit F1 key to get help about the text.

TIP Ctrl + **Spacebar** opens the Variable Editor dialog box

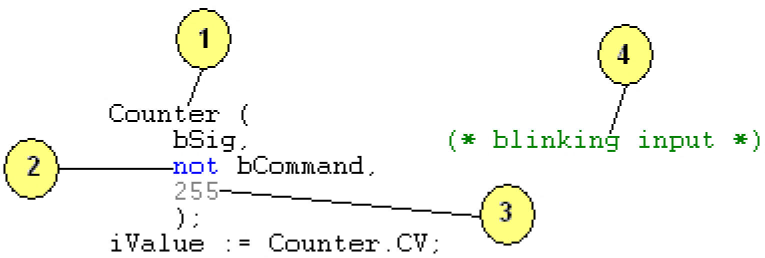
ST / IL Language selection

The KAS IDE allows you to mix ST and IL languages in textual program. ST is the default language. When you enter IL instructions, the program must be entered between "BEGIN_IL" and "END_IL" keywords, such as in the following example

```
BEGIN_IL
FFLD    var1
ST    var2
END_IL
```

ST / IL Syntax coloring

The ST / IL editor supports syntax coloring according to the selected programming language (ST or IL). The editor uses different colors for the following kinds of words:



1- Default (identifiers, separators)
2- Reserved keywords of the language
3- Constant expressions
4- Comments

Intellisense

The following features are available with Intellisense in ST and FBD programs:

NOTE They do not apply to actions in an SFC step.

Conditional compiling coloring

Parts of conditional compiling code (declared with #ifdef pragmas) that are not validated are grayed

```
#define CONDITION


#ifdef CONDITION


if tryGetSpike = true then
   MachineState := 2;
   MachineSpeed := 2000;
end_if;

#else


Printf('Manual mode', 0, 0, 0, 0);

// Start motion
MLMstRun(PipeNetwork.Master, TravelSpeed);


#endif
```

Commenting the CONDITION changes the active part of the program

```
// #define CONDITION


#ifdef CONDITION


if tryGetSpike = true then
   MachineState := 2;
   MachineSpeed := 2000;
end_if;

#else


Printf('Manual mode', 0, 0, 0, 0);

// Start motion
MLMstRun(PipeNetwork.Master, TravelSpeed);


#endif
```

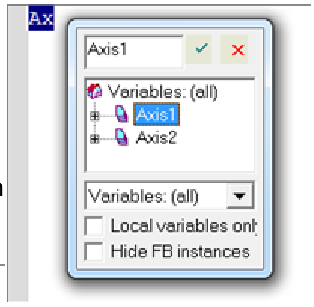
NOTE Save your project to have the code with the correct colors.

Auto-indentation

Lines are automatically indented on the left when you enter structured ST statements

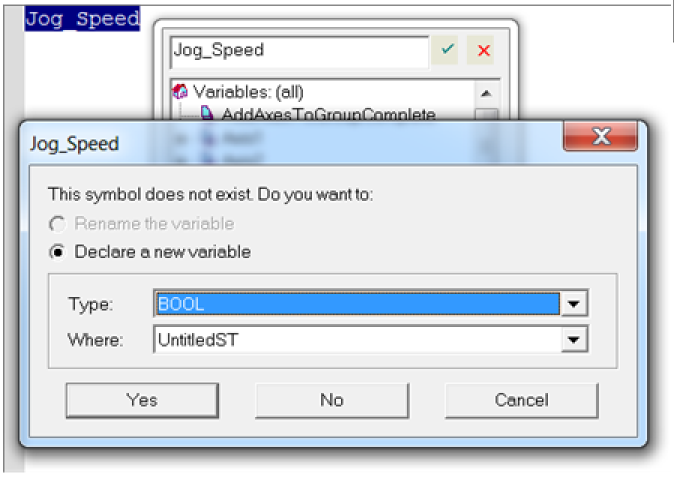
Autocompletion

Auto-completion of words



The ST / IL editor includes powerful commands for automatic completion of typed words, according to declared variables and data types. Pressing Ctrl-space (or alternatively, Ctrl-J) activates the auto-completion.

If you use auto-complete and there is no match for a variable name in the Dictionary, pressing ENTER will open the variable creation box where you can define a new variable.



The following features are available:

Auto-completion of:	Action
Variable Name	If you enter the first letters of a variable name, you can press **CTRL+J** to automatically complete the name. A pop-up list is displayed with possible choices if several declared variable names match the typed characters.
Missing Symbols	When you press **ENTER** at the end of a line containing an unknown variable symbol, you are prompted to declare it immediately.
FB Member	When you type the name of a function block instance (used either as an instance or a data structure), pressing the point "**.**" after the name of the instance opens a pop-up list with the names of possible members.

<table>
<tr><th>Auto-completion of:</th><th>Action</th></tr>
<tr><td>FB Call</td><td>Type the name of a function block followed by an opening parenthesis
<pre>MLMstRun(</pre>
Press the ENTER key to complete the instructions with the appropriate argument list, including comments and possibly default values so that you are guided through the list of values to be passed to the called function.
<pre>MLMstRun(
    (* BlockID : DINT *) ,
    (* Speed : LREAL *)
);</pre></td></tr>
<tr><td>ST Block Statement</td><td>On an empty line, enter the main keyword of a ST statement such as "for", "if"...
<pre>For</pre>
Press the ENTER key to complete the whole statement, including comments that will guide you through the syntax.
<pre>FOR (* DINT var *) := (* minimum : DINT *) TO (* maximum : DINT *) BY 1 DO

END_FOR;</pre></td></tr>
</table>

Other syntax related commands

When lines are selected, you can automatically indent them. Press **TAB** or **Shift+TAB** to shift the lines to the left or right, by adding or removing blank characters on the left.

ST / IL Drag-and-drop features

The ST / IL Editor supports powerful drag-and-drop features that help you to develop and test your programs. You can:

- Drag text (words or lines) from the ST / IL editor to another application (such as a text editor)
- Do the opposite
- Drag a variable symbol from the variable editor to the ST / IL editor
- Drag a variable symbol from the ST / IL editor to the watch list (*)

(*) When dragging the symbol of an array to the watch list, all items in the array are added to the watch list.

How to Read Output of a MC Function Block in ST

In the following example:

```
A6_Inst_MC_MoveRelative( 1, Axis6, -90, 5, 300000, 300000,
0, 0 );
```

A6_Inst_MC_MoveRelative is an instance of MC_MoveRelative PLCopen Motion function block.

The values given in parenthesis correspond to the 8 inputs of this FB.

The syntax to read one of the outputs in ST for this instance is:

```
<FUNCTION BLOCK NAME>.<OUTPUT>
```

This FB has the following 5 outputs: Busy, Active, CommandAborted, Error, and Error ID

So for instance, the **Active** output has the following ST expression: A6_Inst_MC_ MoveRelative.**Active**

Example 1:

```
UserVariable1 := A6_Inst_MC_MoveRelative.Error ;
```

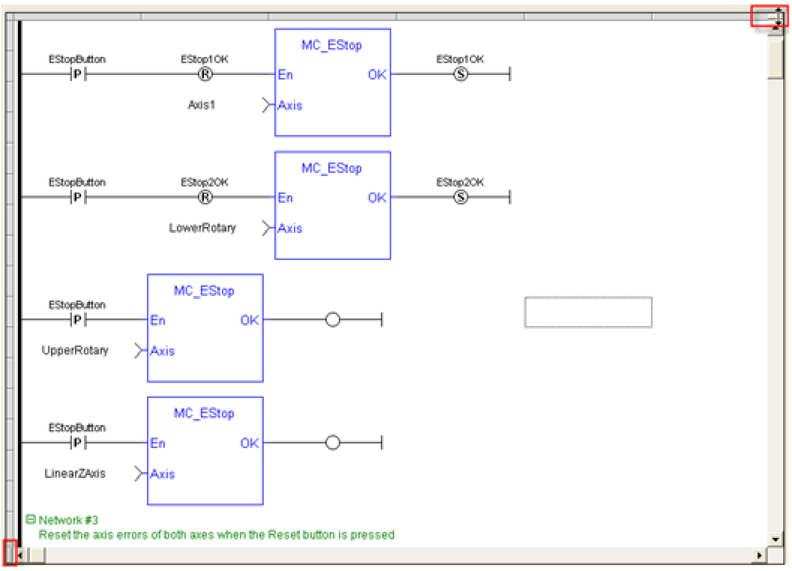
Example 2:

```
IF A6_Inst_MC_MoveRelative.Active THEN
UserVariable2 := 1 ;
ELSE
UserVariable2 := 0 ;
END_IF;
```

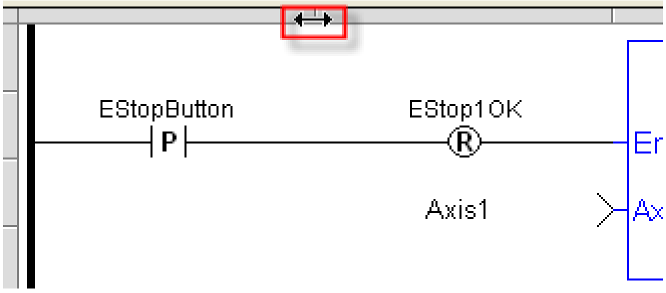
5.4.5.9 Free Form Ladder Diagram (FFLD) Editor

The FFLD Editor is a powerful graphical tool that enables you to enter and manage Ladder Diagrams according to the IEC 61131-3 standard. This Editor enables free drawing and arrangement of FFLD items, and supports advanced graphic features such as:

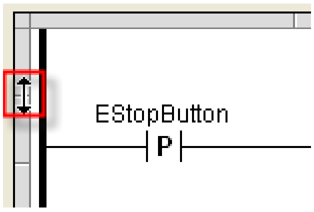
- Split window capability:
  Allows multiple views of the same ladder program to be displayed simultaneously. You can drag the two splitters located in the vertical and horizontal scroll bars to organize your windows.



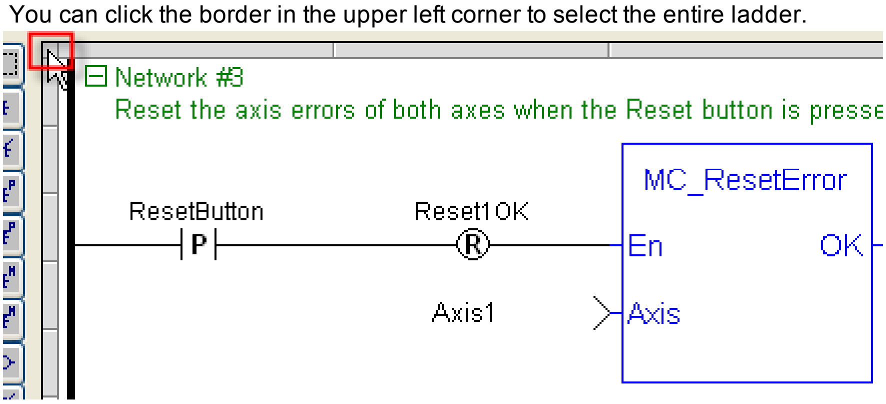
- Change the size of the Ladder Diagram:
  You can drag the column separator to increase or decrease the size of the columns.



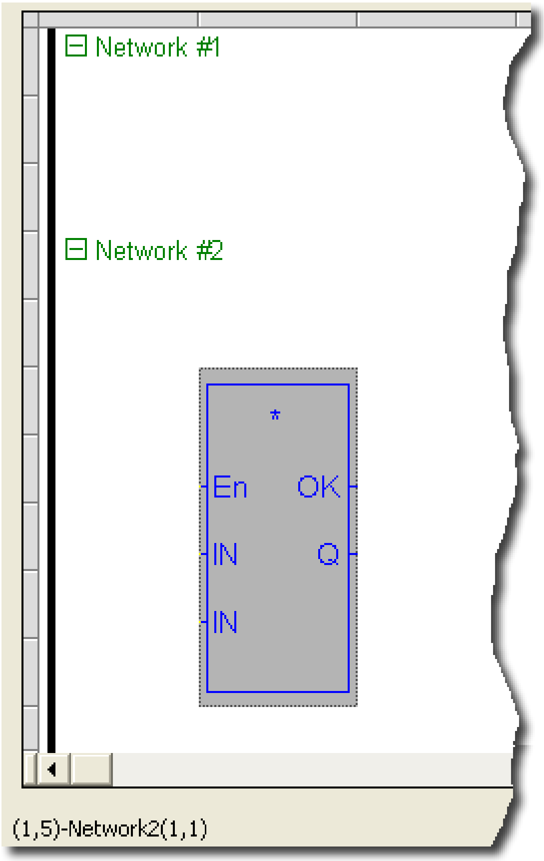
  You can drag the row separator to increase or decrease the size of the rows as well as the texts.



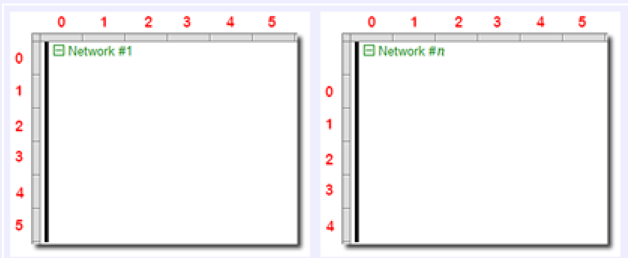
- Drag-and-drop operation
- Select all the Ladder Diagram:
  You can click the border in the upper left corner to select the entire ladder.



- The coordinates for a selected function are displayed at the bottom-left corner of the editor. The coordinates are shown as (absolute coordinates)-(network number).(relative coordinates) in (column, row) format.



NOTE Relative coordinates are measured from below the Network label.



FFLD diagram components:

Networks
Power rail and lines
Contacts and coils
Function blocks
Data In/Out
Jumps and RETURN

Related sections:

Using the FFLD toolbar
Selection grid
Moving and copying items
Run-time

TIP When a contact or a coil is selected, you can press the **Spacebar** to change its type (normal, negated, pulse)

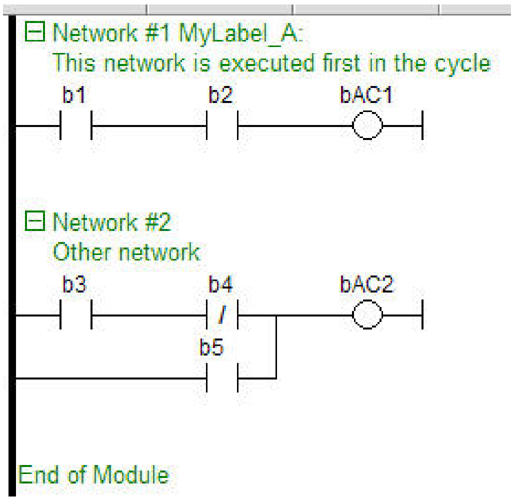
Networks

A program is entered as a list of independent networks. Networks are executed sequentially from the top to the bottom. The head of a network is drawn on a full row in the editor, grouping the following pieces of information:

- The number of the network (from 1 to N)
- (Optional) A label name used as a target for jump operations

- (Optional) A directive for conditional compiling
- (Optional) A multiple line description (comment)

No item can be put on a network header row. No line can go through it. The end of a program is marked with a special "End of module" row. Nothing can be inserted after this row.



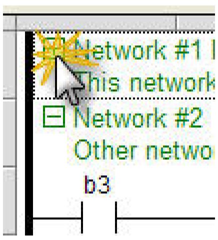
Double-click on the header of a network to enter its label, directive (sometimes called pragma) and description. Network headers are green, but they became blue when a directive is defined (see also paragraph "Conditional Compiling" on page 319 for more details).

New networks can be inserted on empty rows.

When a network is selected, pressing "DEL" merges its content with the previous network. When the first network is selected, pressing "DEL" removes the network and its whole contents.

There cannot be two networks having the same label in a program. If such a situation occurs in the case of a copy operation, you will be prompted to either specify another label name for the new network, or remove the label on the new network.

You can also collapse/expand a network with the minus/plus sign located next to the Network number in the header.



Run-time

When your application is running, you can force and lock a variable or a contact directly in the editor with a double-click operation. For more information, see page 339.

NOTE In FFLD, when a function, function block or UDFB is not connected on the left, then it is ignored (removed at compiling time).

Using the FFLD Editor

This section describes the Toolbar icons and Contextual Menu of the FFLD Editor.

For FFLD accelerator keys, refer to paragraph "FFLD Editor Shortcuts" on page 666

Toolbar

The vertical toolbar on the left-hand side of the Free Form Ladder editor contains buttons for inserting items in the diagrams. Items are inserted at the current position in the diagram.

Icon	Shortcut	Description
		Mode selection
	Ctrl+Shift+O	Insert a contact to the destination cell
	Ctrl+Shift+C	Insert an inverted contact to the destination cell
	Ctrl+Shift+P	Insert a Pulse contact to the destination cell
	Ctrl+Shift+I	Insert an inverted Pulse contact to the destination cell
	Ctrl+Shift+N	Insert a N contact to the destination cell
	Ctrl+Shift+A	Insert an inverted N contact to the destination cell
	Ctrl+Shift+E	Insert a coil to the destination cell
	Ctrl+Shift+D	Insert an inverted coil to the destination cell
	Ctrl+Shift+S	Insert a set coil to the destination cell
	Ctrl+Shift+R	Insert a reset coil to the destination cell
	Ctrl+Shift+K	Insert a positive coil to the destination cell
	Ctrl+Shift+L	Insert a negative coil to the destination cell
	Ctrl+Shift+H	Trace a horizontal line to the destination cell
	Ctrl+Shift+V	Trace a vertical line to the destination cell
	Ctrl+Shift+B	Trace a vertical and horizontal line to the destination cell
		Toggle trace mode: click and move the mouse to draw a line spanning on several adjacent cells
	Shift+Insert	Insert a network
	Ctrl+Shift+J	Insert a jump
	Ctrl+Shift+T	Insert a return
	Ctrl+Shift+F	Insert a data in
	F8	Insert a function block
	Ctrl+Shift+Q	Insert a data out
	Spacebar	Swap item style of the current cell for a contact or coil
		Define a network label
		Define a network pragma
		Define a network comment

Table 4-8: FFLD Toolbar - List of Icons

Contextual Menu

A right-click in the FFLD workspace gives you access to the following commands:

- Insert Network
- Insert Row
- Delete Cell
- Delete Network
- Delete Row

Power rail and lines

Vertical power rails are used in FFLD language to represent the limits of a rung.

The power rail on the left represents the TRUE value and initiates the rung state. Any object connected to this rail is always powered.

Horizontal lines always represent a data flow from the left to the right.

If a vertical line has several items connected on the left, then it represents an OR operation.

You can insert a segment of horizontal line at any location in order to freely draw flow lines. The "vertical line" button enables you to set or remove (toggle) a segment of vertical line on the right of the selected cell.

Contacts and coils

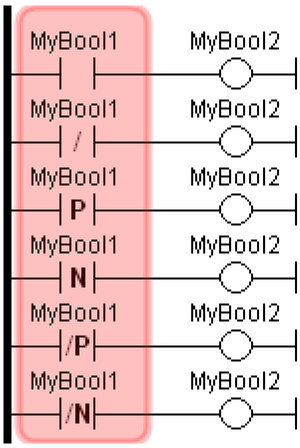
The table below contains a list of the contact and coil types available:

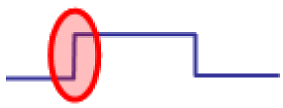
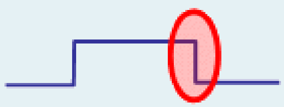
Contacts	Coils
Normally Open -| |-	Energize -( )-
Normally Closed -|/|-	De-energize -(/)-
Positive Transition -|P|-	Set (Latch) -(S)-
Negative Transition -|N|-	Reset (Unlatch) -(R)-
Normally closed positive transition -|/P|-	Positive transition sensing coil -(P)-
Normally closed negative transition -|/N|-	Negative transition sensing coil -(N)-

Contacts

Contacts are basic graphic elements of the FFLD language. A contact is associated with a boolean variable which is displayed above the graphic symbol. A contact sets the state of the rung on its right-hand side, according to the value of the associated variable and the rung state on its left-hand side.

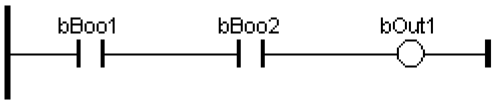
Below are the six possible contact symbols and how they change the flow:



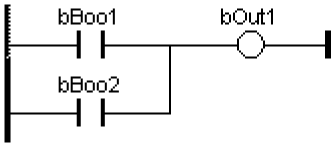
Contacts	Description
`boolVariable` `-] [-`	**Normal**: The flow on the right is the boolean AND operation between: (1) the flow on the left and (2) the associated variable.
`boolVariable` `-]/[-`	**Negated**: The flow on the right is the boolean AND operation between: (1) the flow on the left and (2) the negation of the associated variable.
`boolVariable` `-]P[-`	**Positive Transition**: The flow on the right is TRUE when the flow on the left is TRUE and the associated variable is TRUE and was FALSE the last time this contact was scanned (rising edge) 
`boolVariable` `-]N[-`	**Negative Transition**: The flow on the right is TRUE when the flow on the left is TRUE and the associated variable is FALSE and was TRUE last time this contact was scanned (falling edge). 
`boolVariable` `-]/P[-`	**Normally Closed Positive Transition**: The flow on the right is TRUE when the flow on the left is TRUE and the associated variable does not change from FALSE to TRUE from the last scan of this contact to this scan (NOT rising edge).
`boolVariable` `-]/N[-`	**Normally Closed Negative Transition**: The flow on the right is TRUE when the flow on the left is TRUE and the associated variable does not change from TRUE to FALSE from the last scan of this contact to this scan (NOT falling edge).

Serialized and Parallel contacts

Two serial normal contacts represent an AND operation.



Two contacts in parallel represent an OR operation.

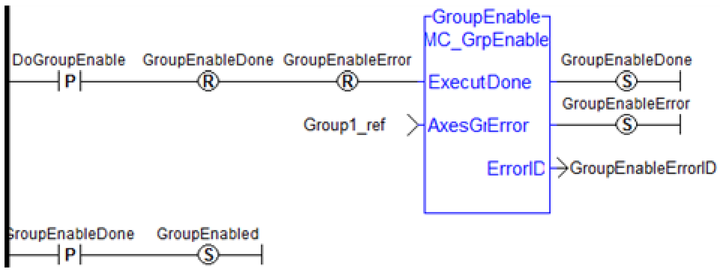


Transition Contacts

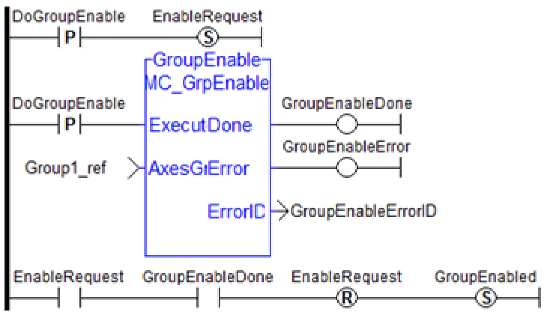
The transition contacts -|P|-, -|N -|/P|-, and -|/N|- compare the current state of the boolean variable to the boolean's state the last time the contact was scanned. This means that the boolean variable could change states several times during a scan, but if it's back to the same state when the transition contact is scanned, the transition contact will not produce a TRUE. Also, some function blocks can complete immediately. Therefore a different approach, other than using transition contacts, is needed to determine if a function block completed successfully.

For example:

MC_GrpEnable executes and turns on its Done output immediately. In the following code, the GroupEnableDone positive transition contact will only provide a TRUE the first time MC_GrpEnable is executed. For all subsequent executions, the positive transition contact will not provide a TRUE since GroupEnableDone will be TRUE every time the contact is scanned.



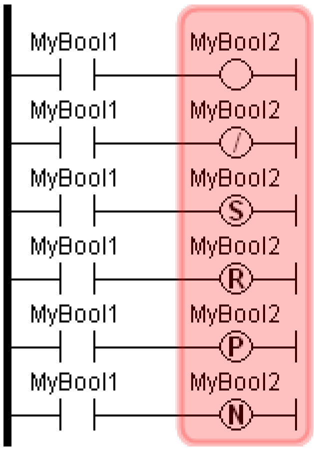
To remedy this, the following code uses the SET and RESET of a boolean (i.e. EnableRequest) to provide a way to detect each successful execution of the function block:



Coils

Coils are basic graphic elements of the FFLD language. A coil is associated with a boolean variable which is displayed above the graphic symbol. A coil performs a change of the associated variable according to the flow on its left-hand side.

Below are the six possible coil symbols:



Coils	Description
`boolVariable -( )-`	**Normal**: the associated variable is forced to the value of the flow on the left of the coil.
`boolVariable -(/)-`	**Negated**: the associated variable is forced to the negation of the flow on the left of the coil.

`boolVariable -(S)-`	**Set**: the associated variable is forced to TRUE if the flow on the left is TRUE. (no action if the flow is FALSE)
	Rules for Set coil animation:
	• Power Flow on left is TRUE:
	  ◦ The horizontal wires on either side of the (S) are red
	  ◦ The variable and the (S) are red
	• Power Flow on left is FALSE and the (S) variable is Energized (ON)
	  ◦ The horizontal lines on either sided of (S) are black
	  ◦ The variable and the (S) are red
	• In all other cases:
	  ◦ The horizontal wires are black
	  ◦ The variable and the (S) are black

Coils	Description

`boolVariable`
`-(R)-`
	Reset: the associated variable is forced to FALSE if the flow on the left is TRUE. (no action if the rung state is FALSE)
	
	Rules for Reset coil animation:
	• Power Flow on left is TRUE:
	  ◦ The horizontal lines are red
	  ◦ The variable above (R) is black
	  ◦ The R and the circle around the R are black
	• Power Flow on left is FALSE and variable above reset coil is NOT Energized (OFF)
	  ◦ The horizontal lines are black
	  ◦ The variable above (R) is black
	  ◦ The R and the circle around the R are black
	• Power Flow on left is FALSE and variable above reset coil is Energized (ON)
	  ◦ The horizontal lines are black
	  ◦ The variable above (R) is red
	  ◦ The R and the circle around the R are red

`boolVariable` `-(P)-`	**Positive transition**: the associated variable is forced to TRUE if the flow on the left changes from **FALSE to TRUE**(and forced to FALSE in all other cases)
`boolVariable` `-(N)-`	**Negative transition**: the associated variable is forced to TRUE if the flow on the left changes from **TRUE to FALSE**(and forced to FALSE in all other cases)

TIP: When a contact or coil is selected, you can press the **Spacebar** to change its type (normal, negated...)
When your application is running, you can select a contact and press the **Spacebar** to swap its value between TRUE and FALSE

IMPORTANT: Although coils are commonly put at the end, the rung can be continued after a coil. The flow is **never changed** by a coil symbol.

Function blocks

Functions and function blocks can be used in FFLD diagrams. Blocks are always connected to the flow line (powered) by their first input and first output. If the first input of a block is not boolean, a special input called "EN" is added, and means that the block is not executed if the input flow is FALSE. If the first output is not boolean, a special output called "OK" is added. The special "OK" output always has the same state as the first input (the flow).

In the case of a function block, the instance of the block must be specified and is shown on the top of the block. Double-click on the top of the block to select the instance. You can also double-click elsewhere in the block to change its type.

Boolean inputs and outputs of blocks can be directly linked to contacts and coils. Block inputs and outputs can also be specified using specific data in/out items (see below).

NOTE: Function and function blocks cannot be put in column 1 of the grid. This would

NOTE not make sense because data inputs require a column.

You cannot change a function block after it has been inserted.

When a Function is not connected on the right, then it is ignored (removed at compiling time).
It is the case for Functions only - **not** for function blocks.
A Function is just part of an expression (same as a contact) and is just intended to provide a result. In case of FFLD, KAS accepts that the output is not connected because it accepts pending "dead" expressions to be removed at compiling time (same as contacts with no coil or FBs after).

TIP If you want another function block, you first have to select it in the Libraries toolbox before inserting it.

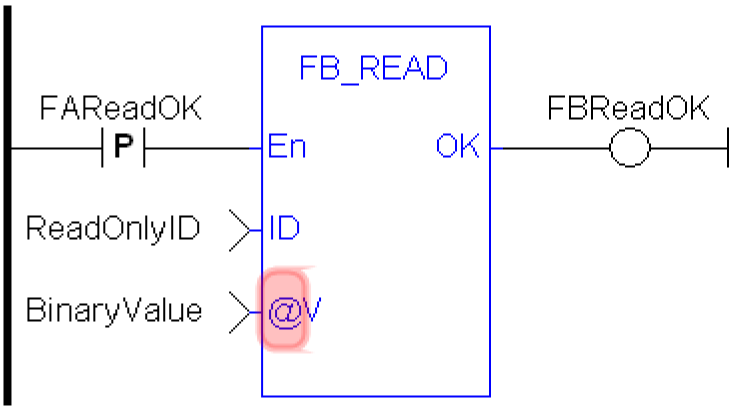
Data In/Out

The "data in" and "data out" items are used to initiate a flow (line) with the value of a variable, or to force a variable on output with the value of a flow:

```
VarIn>-  ..........  ->VarOut
```

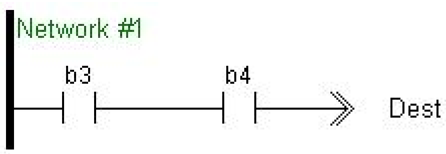
When used with a block, the "data in" and "data out" items can be put close to the block, without any line in-between to connect a variable to an input or output of the block.

In the following example, the **@** symbol in front of the V variable indicates it is used for input and output.



Jumps and RETURN

A jump to a label branches the execution of the program after the specified label. In FFLD language, the ">>" symbol (followed by the target label name) is used as a coil at the end of a rung.



The jump is performed only if the rung state on input is TRUE. The destination label must be specified on a network of the same program.

To specify the destination, double-click the cell to display a drop-down menu that lists all the available labels.

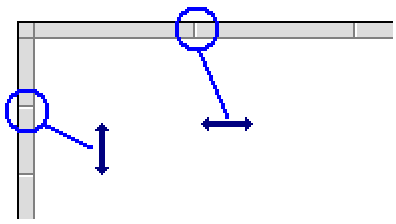
NOTE The special "<RETURN>" destination specifies a jump to the end of the program.

Selection grid

The diagram is entered in a logical grid, and all objects are snapped to the grid.

You can use the commands of the "View" menu for displaying of hiding grid points. This helps you locating errors detected by the compiler, or aligning objects in the diagram.

At any moment you can use the commands of the "View" menu for zoomingzoom in or out of the edited diagram (for shortcuts about zooming, see page 243).You also can press the [+] and [-] keys of the numerical keypad for zooming the diagram in or out. You can also drag the separation lines in vertical and horizontal rulers to freely resize the cells of the grid:



NOTE If a split window is in use, the zoom applies only to the currently selected split window (each split window can be zoomed to different levels).

The current position in the grid is always highlighted by a dotted cell and its coordinates (row, column) are displayed at the bottom left-hand corner of the editor.

If you click on the current position, then the cell is drawn as gray, meaning that it can be dragged somewhere else in the diagram (see below). You can also select multiple cells with the mouse, or use the arrows of the keyboard with the SHIFT key pressed.

Click on the power rail (gray ruler at the left border) to select a full row.

Other selection commands are available from the keyboard:

Home	moves the caret to the left of the line if pressed again, moves the caret to the head of the network
End	moves the caret to the end of the line if pressed again, moves the caret to the end of the network
Ctrl + Page Up Down	moves the caret to previous or next network header
Ctrl + Home/End twice	moves the caret to the beginning or the end of the program
Ctrl + A	selects the whole network if pressed again, selects the whole program
Page Up / Down	scroll 1 page
Shift-Page Up / Down	selection page up or down
Return	equivalent to a double-click
Space	change contact or coil
Tab	move focus cell right
Shift-Tab	move focus cell left
Arrows	move focus cell or scroll through ladder
Shift-Arrow	multi-select cells

Ctrl + F	performs a Search and Replace (+ add hyperlink on the topic) within the whole program
Ctrl-Shift-F2	go to previous bookmark
Esc / Shift-Esc	close the rename widget

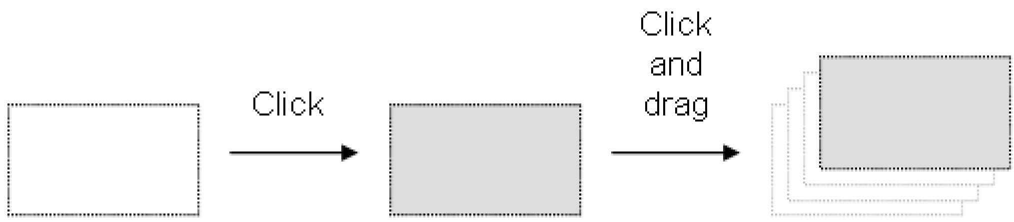
View FFLD diagrams

The FFLD Editor adjusts the size of the font according to the zoom ratio, so that the name of variables associated with contacts and coils are always visible.

When a cell is high enough, variable names are completed with other pieces of information about the variable:
- its tag (short description)
- its description text
- its I/O name (%...) if the variable has a user-defined name.

Move and copy items

When you click on the current position, then the cell is drawn as gray, meaning that it can be copied or moved. Click again on the selection to drag it with the mouse.



Dragging the selected items moves them to the specified location. If you press the **CTRL** key while dragging, then items are copied (for shortcuts, see page 243).

To move a function block, you must select it entirely.

If you move or copy items on a non-empty area, you will be prompted to confirm the overwriting of items in the area.

When you move or copy items only on a network header, the network is automatically moved in order to make the required extra space for moved items.

The "Copy / Cut / Paste" commands can also be used as an alternative to drag-and-drop.

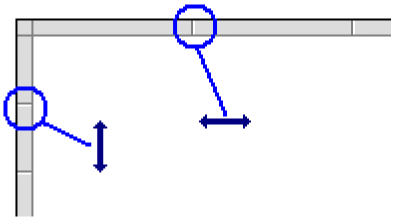
A rectangular selection within the diagram cannot cross a network header, i.e. all selected items must be within the same network. To select a complete network or more, you must select complete rows. To do this, move the caret to the left border or click on the left-hand side ruler (gray).

View FFLD diagrams

The diagram is entered in a logical grid. All objects are snapped to the grid. You can use the commands of the "View" menu for displaying of hiding grid lines. The (x,y) coordinates of the mouse cursor are displayed in the status bar. This helps you to locat errors detected by the compiler, or to align objects in the diagram.

At any moment you can use the commands of the "View" menu for zoomingzoom in or out of the edited diagram by means of a Ctrl + mouse-wheel operation. You also can press the [+] and [-] keys of the numerical keypad to zoom the diagram in or out.

You can also drag the separation lines in vertical and horizontal rulers to freely resize the cells of the grid:



The FFLD editor adjusts the size of the font according to the zoom ratio so that the name of variables associated with contacts and coils are always visible. If cells have sufficient height, variable names are completed with other pieces of information about the variable:
- its tag (short description)
- its description text
- its I/O name (%...) if the variable has a user-defined name.

Manage comment texts

Multiple line comment texts can be entered on any network header.

Commands are available for importing or exporting comment texts to/from text files. This feature enables easy localization of programs.

When exporting comment texts, each comment block will be identified in the text file by a number. You have the selection to use for this number:

- the internal "index number" of networks,
- or the visible network number of networks.

The first method using internal index numbers must be preferred, as such numbers are kept when networks are moved or removed.

When importing comment texts you have the selection of either updating only comment texts of networks found in the import text file, or cleaning all comment texts not found in the import file.

5.4.6 Step 6 of 15 - Create Variables

TIP: As a naming convention for variables, it is recommended to use the initial to reflect the variable type
(e.g. Boolean with **b**; long integer with **L**)

5.4.6.1 Use the Dictionary

For explanations on dictionary usage, including how to create and rename variables, see page 606

5.4.6.2 Create Variables from the Editors

You can create variables directly from the IEC 61131-3 editors, as follows:

FBD editor

1. Click the dedicated button

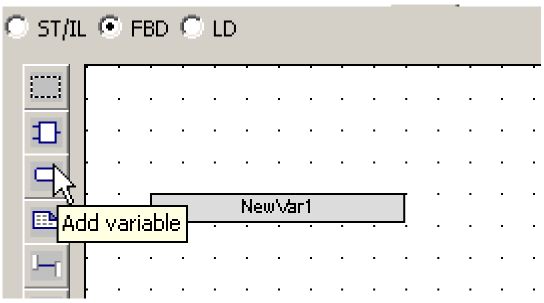


Figure 4-27: Add Variable in FBD Editor

2. Click a location in the editor (or double-click the variable if it is already created)
3. Edit the name in the Variable Editor (or select an existing variable within the list which is already filtered according to their relevant data type)

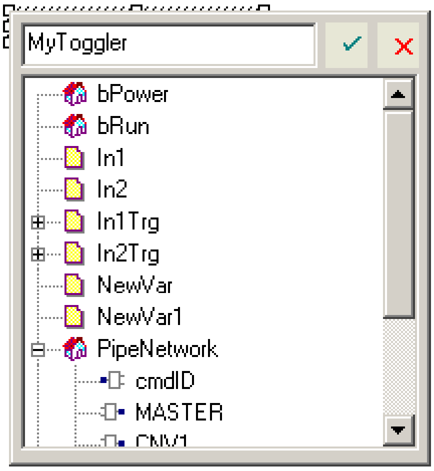


Figure 4-28: Define Variable Name in FBD Editor

4. The KAS IDE automatically checks if the variable already exists. If it is new, you have to:
   - Select its type in the drop-down menu: for FBD and FFLD, it is set by default according to the In or Out data type of the function block
   - Specify where it is defined: the default is the current PLC program, but you can choose to make the variable Global or declared as a retain variable

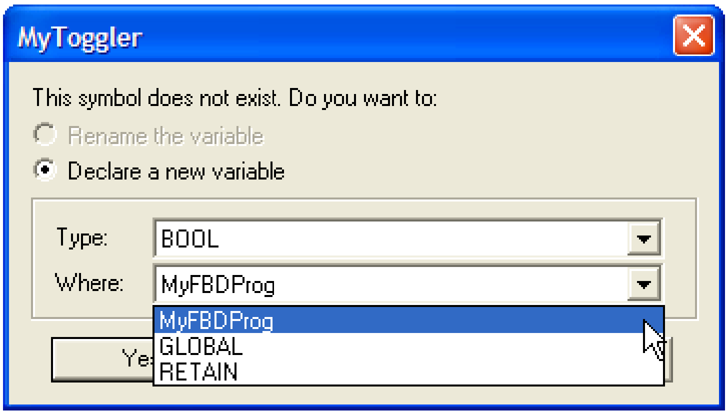


Figure 4-29: Define Variable Type in FBD Editor

See also "FBD variables" on page 230

FFLD editor

1. Double-click the in or out pins of the function block

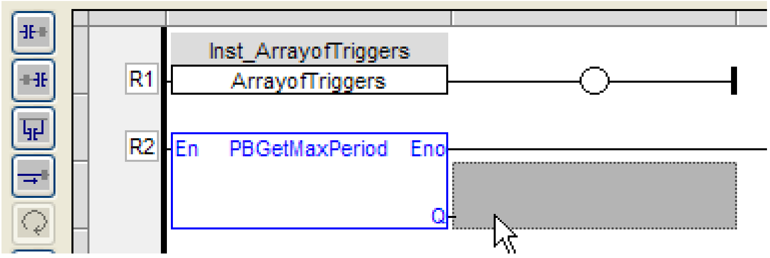


Figure 4-30: Add a Variable in the FFLD Editor

2. Edit the name (or select an existing variable within the list which is already filtered according to their relevant data type)

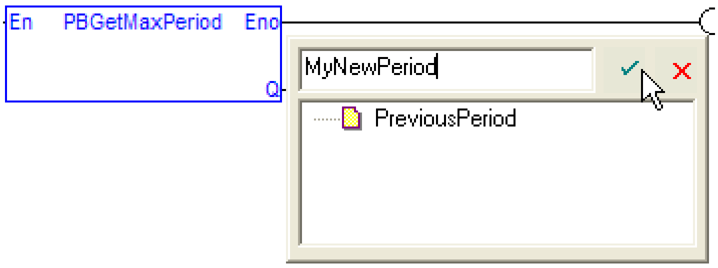


Figure 4-31: Define a Variable Name in the FFLD Editor

3. The KAS IDE automatically checks if the variable already exists. If it is new, you have to:
   - Select its type in the drop-down menu (by default, it is set according to the In or Out data type of the function block)
   - Specify where it is defined

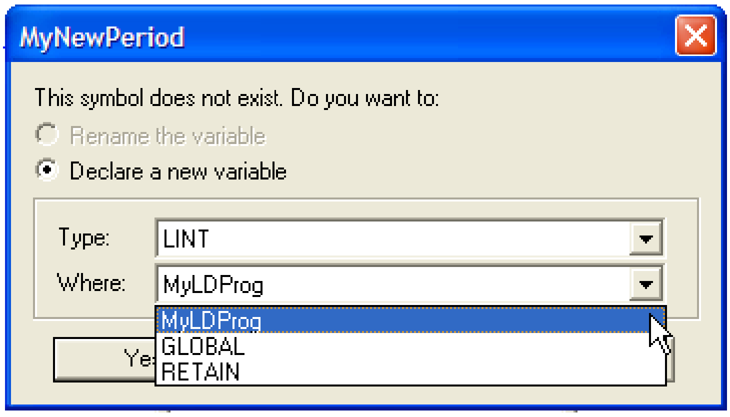


Figure 4-32: Define a Variable Type in the FFLD Editor

5.4.6.3 Data Types

You can create a variable of available Data Types.

How to declare an array?

1. Double-click in the corresponding cell of the variable editor (i.e. the **Dim.** column)
2. Enter its dimension (**Note**: for a multi-dimension array, enter dimensions separated by commas (ex: 2,10,4)

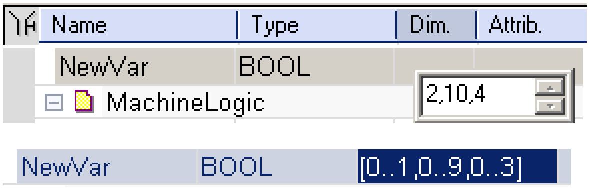


Figure 4-33: Declare an Array for an Internal Variable

See also "Arrays" on page 76

5.4.6.4 Complex Structures

Complex variables are arrays, structures, and instances of function blocks. The following features are allowed for programming:

- Use arrays of structures
- Use arrays of FB instances
- Pass any complex data (array, structure, instance) to a UDFB or sub-program

There is almost no limitation in the amount of complex data declared (theoretically up to 4GB, but practically limited by the memory available in the runtime)

For more explanations on the **Structure** concept, refer to paragraph "Structures" on page 75

Declare the structure

1. Right-click in the Dictionary to open the menu
2. Select the **Add structure** command

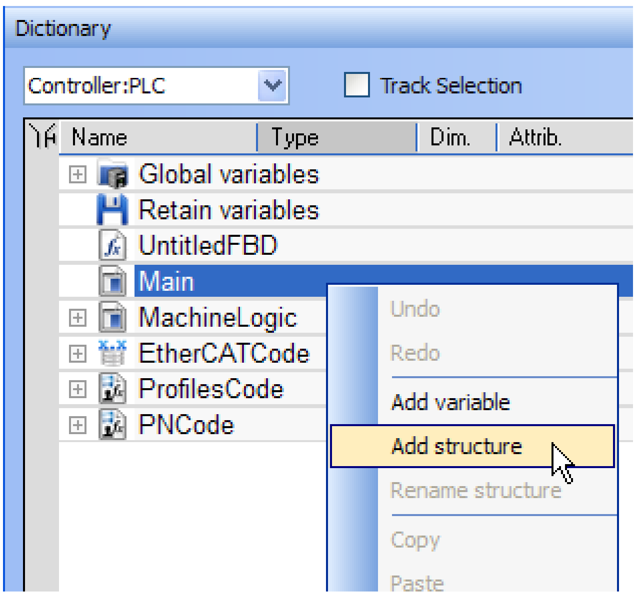


Figure 4-34: Add a Complex Structure

3. Right-click on the newly created structure and select the **Rename structure** command

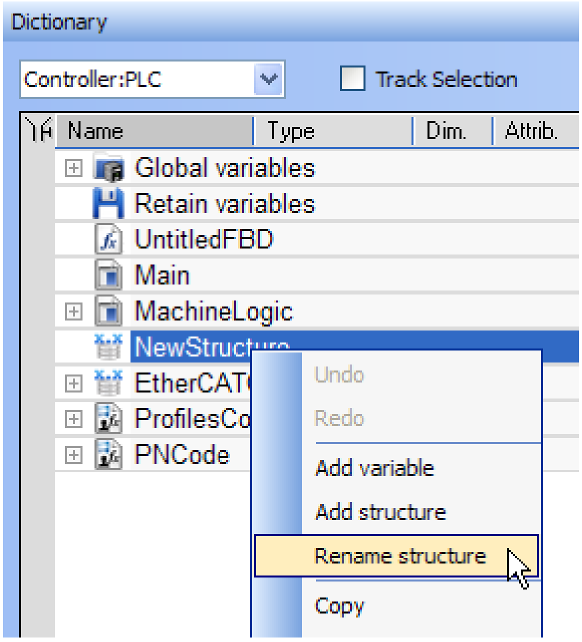


Figure 4-35: Rename Complex Structure

4. Right-click on the new structure and select the **Add variable** command

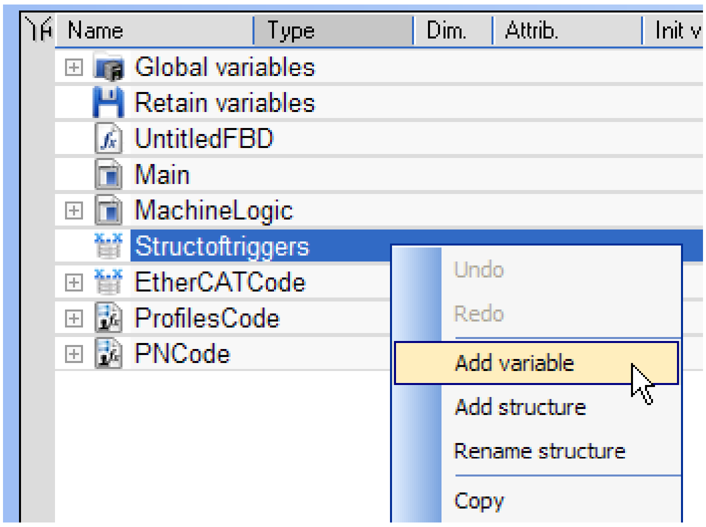
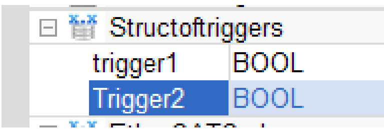


Figure 4-36: Add Variable to a Complex Structure

5. Expand the new structure
6. Double-click on the new nested variable and define its name and type



7. Repeat steps 5 and 7 to add all the requested variables

Create an instance of the structure

When finalized, you can drag-and-drop the structure from the library in the **(Project)** node to a program just like any other function block. A new instance is automatically created.

1. Select the new structure and move it with a drag-and-drop operation to the program declaration within the Dictionary

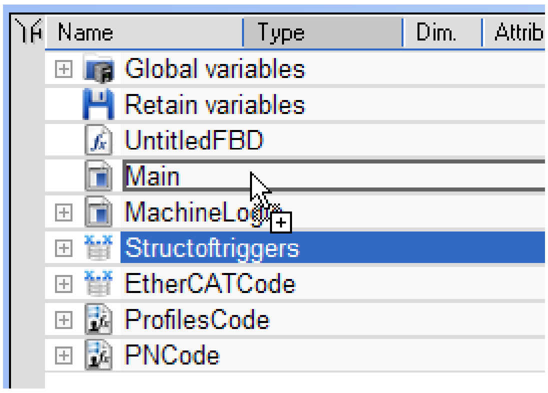
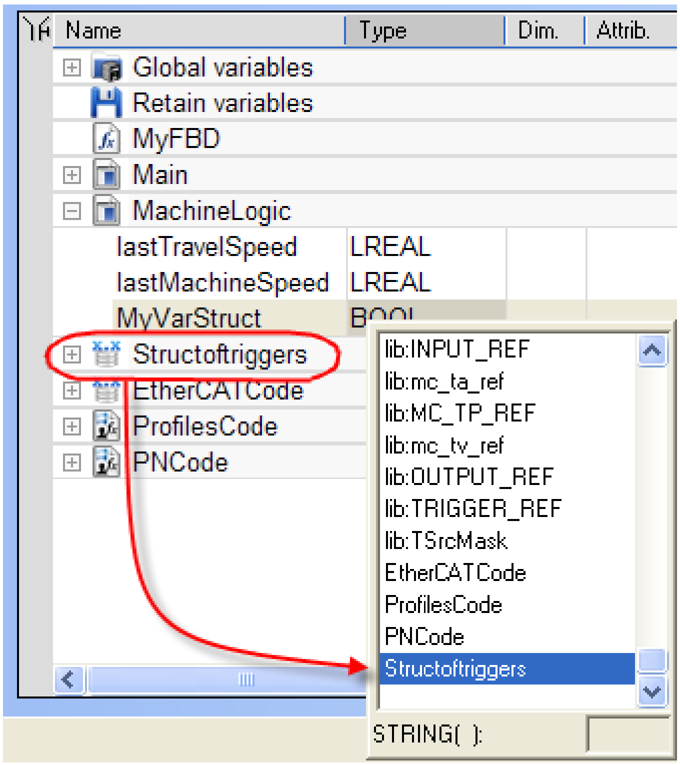


Figure 4-37: Create an Instance of the Structure

2. You can also add a variable in the Dictionary with the **Add variable** command. Then double-click on the new variable to define its type by selecting the structure type which is displayed in the Type drop-down menu.



3. Then you can drag this new instance and drop it in your program like any other variable

5.4.6.5 Variable Editor

You can edit variables directly from each IEC 61131-3 editor.

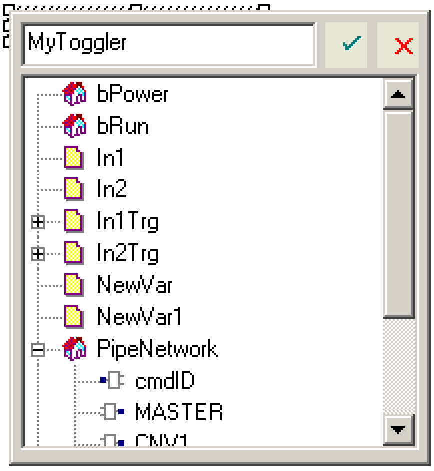


Figure 4-38: Edit the Name in the Variable Editor

TIP Ctrl + **Spacebar** opens this dialog box

KAS IDE automatically checks if the variable already exists. When the variable is new, you have to:

- Select its type in the drop-down menu: for FBD and FFLD, it is set by default according to the In or Out data type of the function block
- Specify where it is defined: the default is the current PLC program, but you can choose to make the variable Global or declared as a retain variable

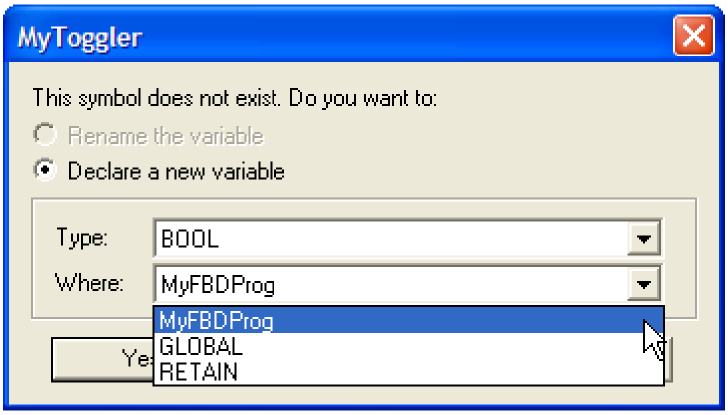


Figure 4-39: Define Type and Scope of the Variable

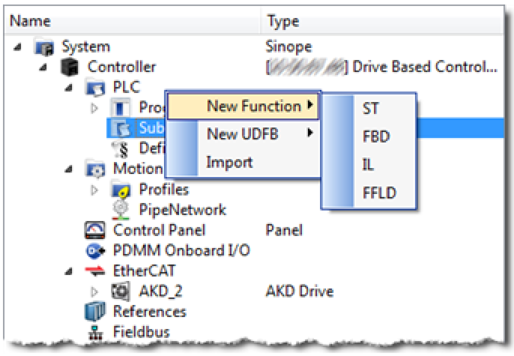
5.4.7 Step 7 of 15 - Create Functions and Function Blocks

For explanation about the difference between functions and function blocks, refer to "Program Organization Units" (see page 84).

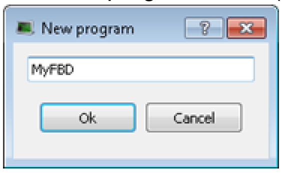
5.4.7.1 Declare Functions or Function Blocks

This section explains how to create a new function or UDFB.

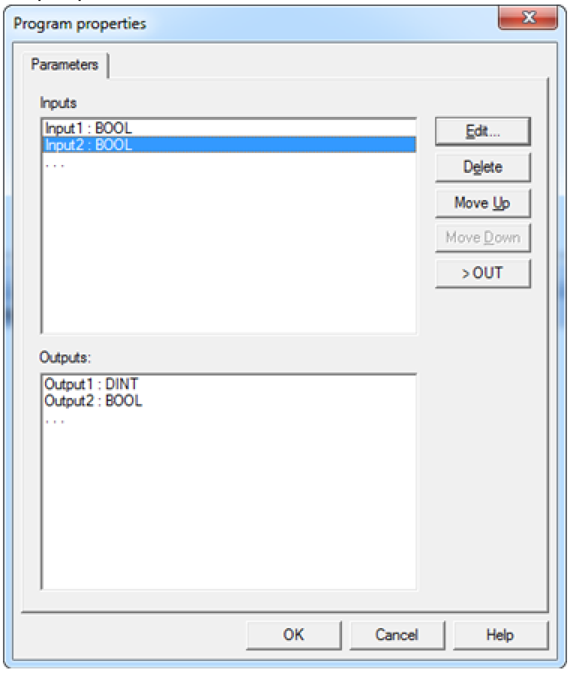
1. Expand the PLC node in the Project Explorer, and right-click on the "Subprograms" (see page 598) item and select the action you wish to perform (i.e., *New Function* or *New UDFB*). The New Program dialog box will open.



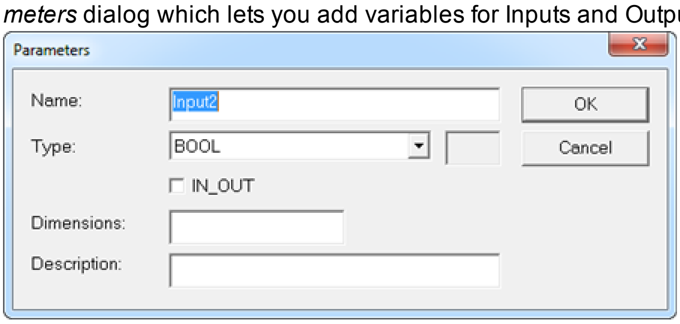
2. Enter the program name (MyFBD, for example), and click OK.



3. The *Program Properties* dialog box opens. This is where you will enter the input and output parameters for the function or UDFB.



4. Clicking on the editable space ("...") in Inputs or Outputs enables the **Edit** button. Clicking the Edit button or double-clicking on the editable space opens the *Parameters* dialog which lets you add variables for Inputs and Outputs.



Item	Description
Edit	Press this button to change the definition of the selected parameter. Pressing this button when the last line (...) of the list is selected lets you add a new parameter.
Delete	Press this button to remove the selected parameter.
Move Up/Down	Press these buttons to move the selected parameter in the list to arrange the order of parameters. The order is very important as it defines the calling prototype of the UDFB or sub-program.
Swap	Pressing this item moves the selected parameter between the Inputs and Outputs lists. When an item is selected the button's text changes to show where the parameter will be moved.

NOTE New UDFBs are added to the **(Project)** node in the Library toolbox

5.4.7.2 Define Parameters and Private Variables

For a Function or UDFB, input and output parameters (as well as private variables) are declared in the Dictionary toolbox as local variables of the item. The **Add variable** command let you add the following:

- Input[1] Parameter
- Output[2] Parameter
- Private[3] Variable

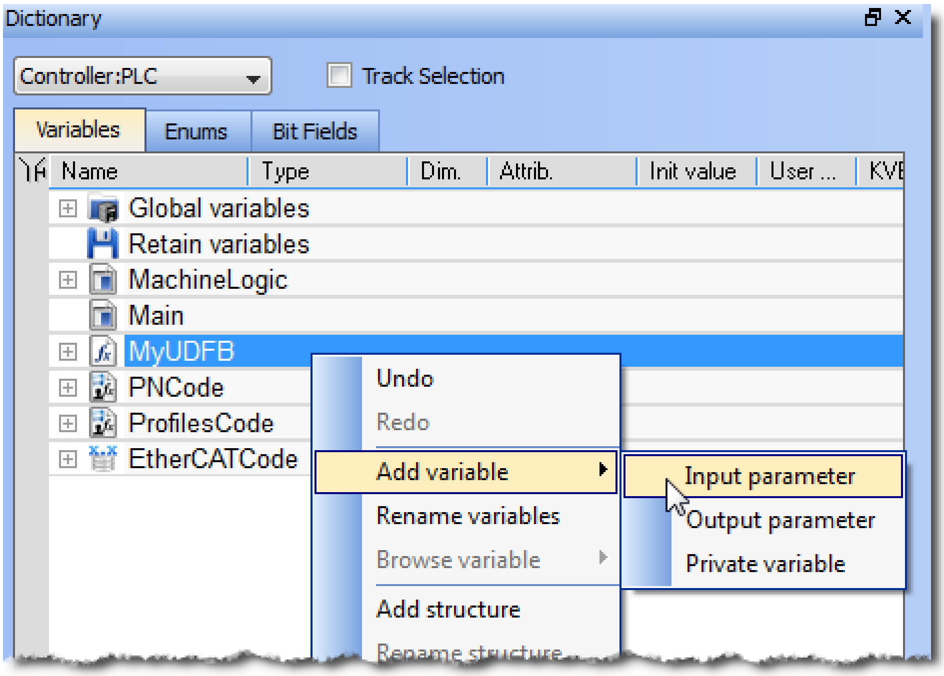


Figure 4-40: Parameters and Private Variables

Input and Output parameters always appear at the beginning of a UDFB group.

Pressing the **INSERT** key when the item is selected adds a private variable.

NOTE There are some things to be aware of with UDFB parameters:

- UDFB cannot contain parameters being both for Input and Output
- Simple parameters (scalar type) can be either IN, OUT, or IN_OUT

---

[1]Externally supplied, not modifiable within the organization unit
[2]Supplied by the organization unit to external entities
[3]Supplied by external entities - can be modified within organization unit

NOTE

- Complex parameters (an array or structure) should be declared as IN but is systemically considered to be IN_OUT.
- UDFB cannot have more than 32 input parameters or 32 output parameters
- Output parameters can only be simple data type

5.4.7.3 Finalize Functions or Function Blocks

Double-click the item in the Project Explorer to open and complete it in its corresponding editor.

5.4.7.4 Call Functions or Function Blocks

When finalized, you can drag-and-drop UDFBs from the library in the **(Project)** node to a program just like any other function block. A new instance is automatically created.

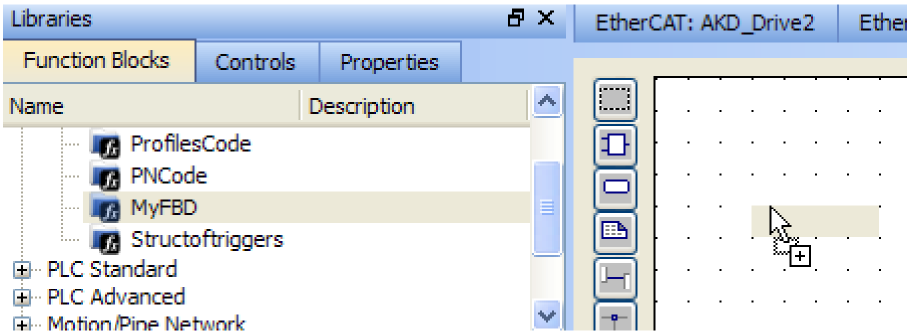


Figure 4-41: Create an Instance of UDFB in a Program

NOTE

- A single data type parameter defined as IN is passed by the calling program to the UDFB and the body UDFB cannot change its value
- A single data type parameter defined as OUT is set in the body UDFB and always actuated in the calling program after the call
- A parameter which is an array or a structure is always declared as IN (visible on the left of the block). Both the calling program and the body of the UDFB can read and write such a parameter

5.4.8 Step 8 of 15 - Use the Defines List

The Defines list consists of defined constants, (an expression with a fixed value). Defines are both pre-defined (internal) and user-created (global and local). Defines are used to determine which parts of a program's code will be compiled using an `ifdef` statement (see "Conditional Compiling" (see page 319)). This creates more efficient code for a given machine type. For example, you can write a program that covers many machine types but compile for a specific machine with more efficient code.

Defined constants have three levels of scope:

Level	Scope
"Internal Defines" (see page 263)	All the projects present on your machine
"Global Defines" (see page 264)	All the programs within your project. These are user-defined.
"Local definitions" (see page 264)	Only the current program currently open

NOTE **Important!** To guarantee precision when evaluating the expression, you need

NOTE to pay special attention to the data types of variables used in the expression. For example mixing **Lreal** and **real** can divide precision by two.

KAS IDE supports the definition of aliases. An alias is a unique identifier that can be used in programs to replace other text. See "Alias Definitions" (see page 89) for more information.

5.4.8.1 Internal Defines

These are pre-defined, common constant definitions which are declared for all projects.

NOTE **Important!** To ensure consistency, you should not modify these declarations.

To see the set of declarations currently installed on your machine, you can view the file (named: *lib.eqv*) located under: C:\Program Files\Kollmorgen\Kollmorgen Automation Suite\Astrolabe\Bin\HwDef (the folder location differs if you chose another location when installing KAS). Below is a an example of pre-defined constants that you may find in your system.

```
#define MLPN_CREATE_OBJECTS 1 (* Creation of blocks and pipes *)
#define MLPN_ACTIVATE 2 (*Activation of pipes*)
#define MLPN_CONNECT 3 (*Connections from convertors to axes*)
#define MLPN_POWER_ON 4 (*Power ON of axes*)
#define MLPN_POWER_OFF 5 (*Power OFF of axes*)
#define MLPN_DEACTIVATE 6 (*Deactivation of pipes*)

#define MLSTATUS_NOT_INITIALISED 0 (*Motion not initialised*)
#define MLSTATUS_RUNNING 1 (*Motion is running*)
#define MLSTATUS_STOPPED 2 (*Motion is stopped*)
#define MLSTATUS_ERROR 3 (*Motion is in error*)
#define MLSTATUS_INITIALISED 2 (*--DEPRECATED-- Motion is ini-
tialised*)

#define MLPR_CREATE_PROFILES 1 (* Creation and initialization of
profiles *)

#define MLFI_FIRST 0 (* ID of the first FastInput of an axis *)
#define MLFI_SECOND 1 (* ID of the second FastInput of an axis *)

#define MLFI_DISABLE 0 (* configures a FastInput as disabled *)
#define MLFI_RISING_EDGE 1 (* FastInput is sensible to rising edges
*)
#define MLFI_FALLING_EDGE 2 (* FastInput is sensible to falling
edges *)

#define PB_EXCHANGE_PRIORITY_NORMAL 0 (* Profibus exchange thread
priority lower than VM thread priority *)
#define PB_EXCHANGE_PRIORITY_HIGHER 1 (* Profibus exchange thread
priority equal to VM thread priority *)

#define PI 3.1415926535897932

#define EC_POSITION_DEMAND_VALUE 10000
#define EC_VELOCITY_DEMAND_VALUE 10001
#define EC_TORQUE_DEMAND_VALUE   10002
#define EC_ADDITIVE_TORQUE_VALUE 10003
#define EC_MAX_TORQUE            10004
#define EC_OPERATION_MODE        10005
#define EC_CONTROL_WORD          10006
```

```
#define EC_LATCH_CONTROL_WORD     10007
#define EC_ANALOG_OUTPUT          10009
```

NOTE The exact contents of the list depend on the version of the KAS IDE.

5.4.8.2 Global Defines

Global Defines are user-generated constants to be used in a program. Global Defines let you write code and add an `ifdef` statement to call the Define only if it is used for a particular machine. They are created and edited from the Project Explorer toolbox under **PLC**.

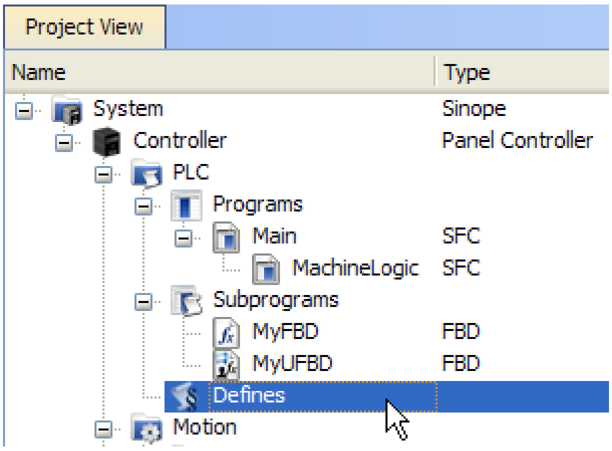


Figure 4-42: Global Defines

Double-click the **Defines** item to open your global definitions file (named: *appli.eqv*) in a text editor as follows:

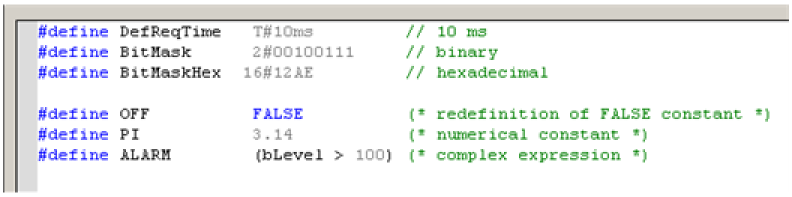


Figure 4-43: Edit the Global Definitions

Each definition must be entered on one line of text according to the following syntax:

```
#define   Identifier   Equivalence   (* comments *)
```

You may use a definition within the contents of another definition. The definition used in the second must be declared first. See example below:

```
#define PI       3.14
#define TWOPI    (PI * 2.0)
```

5.4.8.3 Local definitions

Local definitions are user-created defines that are being used within the corresponding program through an `ifdef` statement.

TIP Using definitions disturbs the program monitoring and makes error reports more complex. It is recommended to restrict the use of definitions to simple expressions to avoid misunderstandings when reading or debugging a program.

5.4.9 Step 9 of 15 - Use Pre-defined Libraries

The Libraries toolbox allows you to select the functions.

TIP

- The **(All)** category at the top enables you to see the full list of available blocks.
- You can access a specific function by entering its initial letters on the keyboard (if the elapsed time between two strikes is greater than 1 second, the KAS IDE considers the last letter as the new initial).

Drag-and-drop into the editors

1. When the function is selected, move it with a drag-and-drop operation in the program editor
2. In the editor, right-click on a function to set the number of input pins if the block allows an extension.

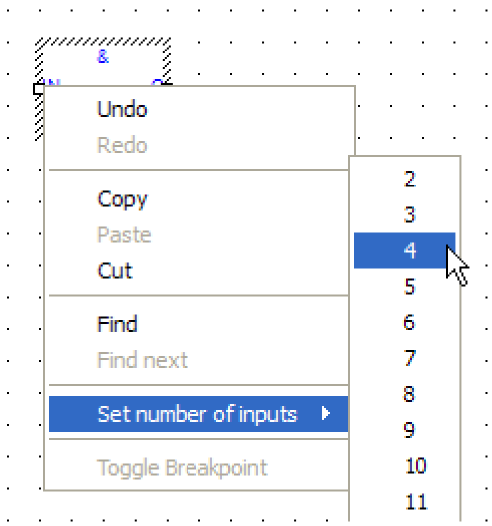


Figure 4-44: Set the Pins Number of the Block

Drag-and-drop into the dictionary

If you have selected a function block, you can drag-and-drop it in the program declaration within the Dictionary toolbox, to create an instance of that object.

5.4.10 Step 10 of 15 - Create and Use Custom Libraries

You first need to create a custom library before you can use it to define a new item: function, function block or variable (for more details on library usage, refer to paragraph "Use the Custom Library" on page 267).

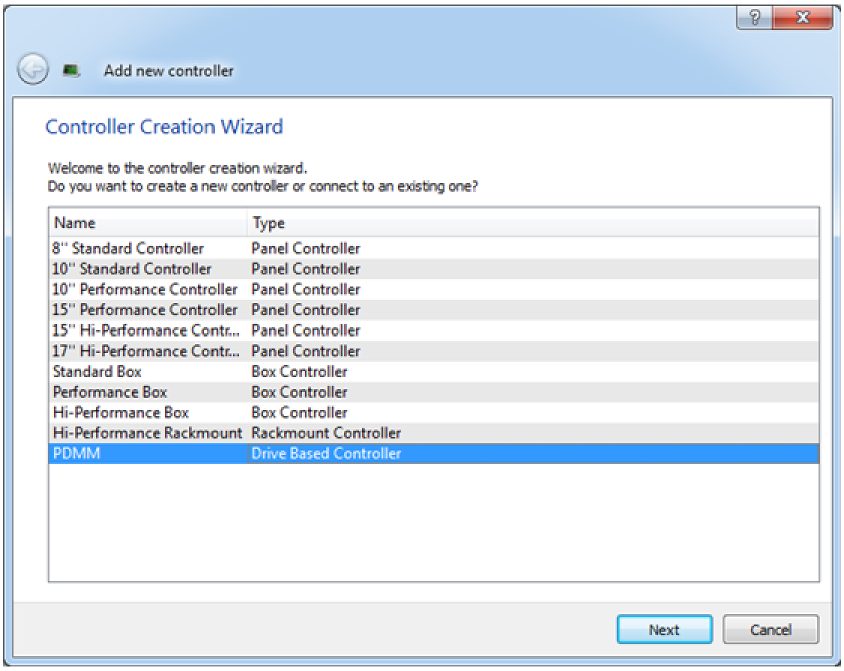
NOTE

There is a difference between **Libraries'** usage and the **Import / Export** commands related to PLC programs.

- **Import/export** is equivalent to a copy and paste operation of programs: when you update the source of your UDFB, the other programs are not updated because the code has been duplicated.
- **Library** is a unique source that can be shared between different projects (like a dll in C): when you modify the library, all the linked projects are impacted.

5.4.10.1 Create the Custom Library

1. In the **File** menu, click the **New** command (save your current opened project if necessary)
2. Select the controller name within the list



3. Click the **Next** button
4. Select the **Library** application template

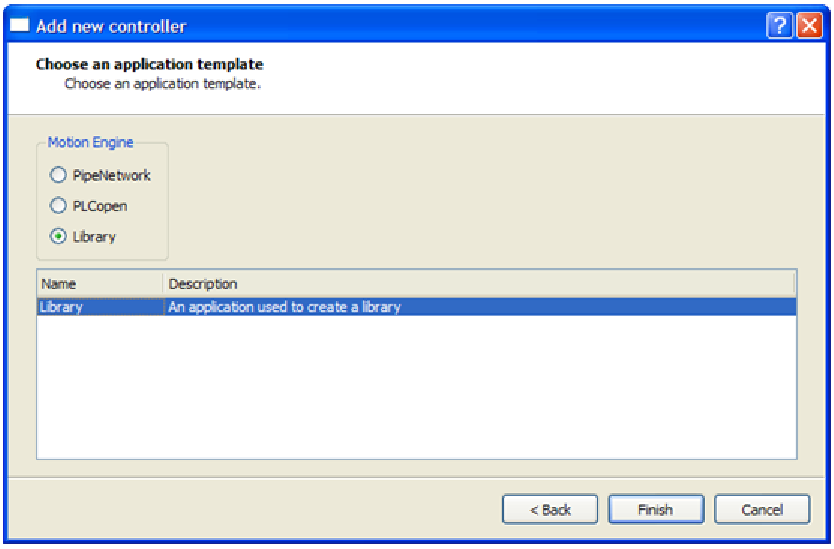
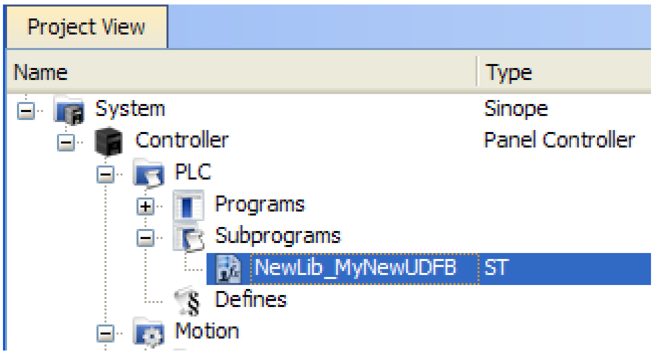


Figure 4-45: Create a Custom Library - Select the Library Template

5. Click the **Finish** button
6. Click the **Save As** command in the **File** menu
7. Define the Library Name (extension *.kal) and its Location
8. Click **OK**
9. In the Project Explorer, expand the **Controller** and **PLC** nodes

10. Right-click on **Subprograms** and choose **New UDFB** in the contextual menu, then select the type of programming language
11. Expand the **Subprograms** node and rename the new UDFB



NOTE It is the name of the variable type which is displayed in the dictionary if you use this library in another project.

IMPORTANT Duplicate UDFB names in a library are not possible. Only the first instance found is kept when importing the library definitions in a project.

TIP To avoid this situation when designing your libraries, use a prefix to identify the library for all UDFBs and functions in the libraries (in the current procedure, the prefix is: NewLib_).

12. Create the UDFB program (for more details, refer to paragraph "Step 5 of 15 - Create Programs" on page 219)
13. In the **File** menu, click the **Save** command

5.4.10.2 Use the Custom Library

1. Open the project where you want to use a library
2. In the Project Explorer, expand the **Controller** node
3. Right-click on **PLC** and choose **Libraries** in the menu
4. Click **Add**

NOTE You can add as many external libraries as you want

5. Select the *.kal file already created before and click **Open**

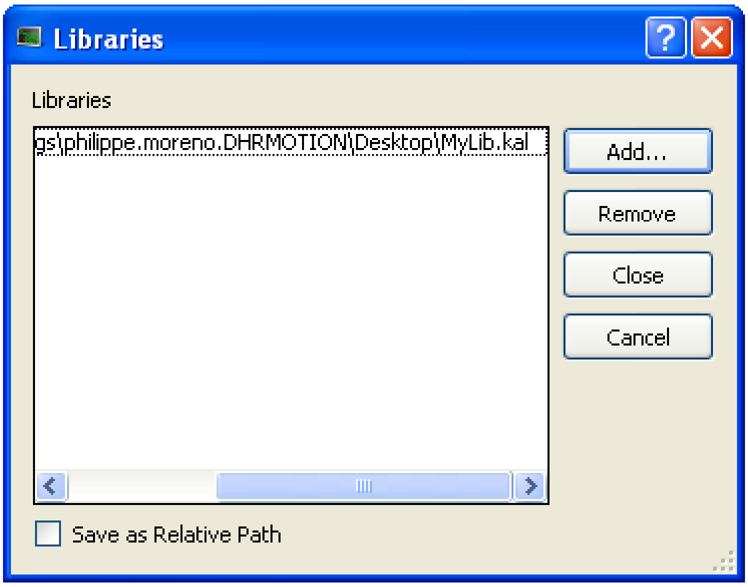


Figure 4-46: Use a Custom Library - Select the Library

> **TIP** You can use the **relative** path to specify the path relative to the working directory where your project is saved. This ensures consistency when you move your project and your library.
>
> Conversely, the **absolute** path points to the same location on your file system regardless of your project directory.

6. Click **Close**
7. The library is displayed in the Library widget and you can now drag-and-drop the UDFB (or any subprogram) of this library in any editor

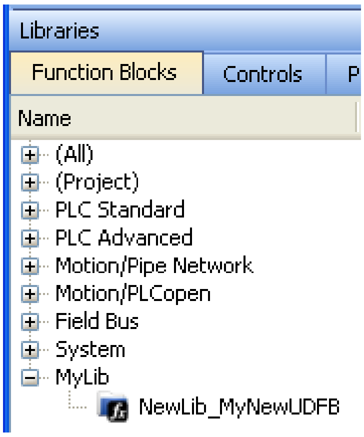


Figure 4-47: Use a Custom Library - Display the Library

8. In the Dictionary toolbox, right-click on the program and choose **Add variable** in the menu

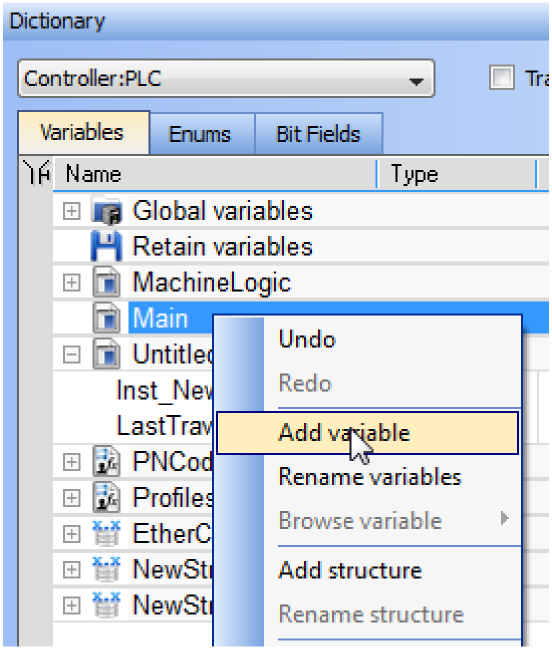


Figure 4-48: Use a Custom Library - Add a Variable

9. In the Type drop-down menu, select the type defined in the external library (it can be at the bottom of the list)

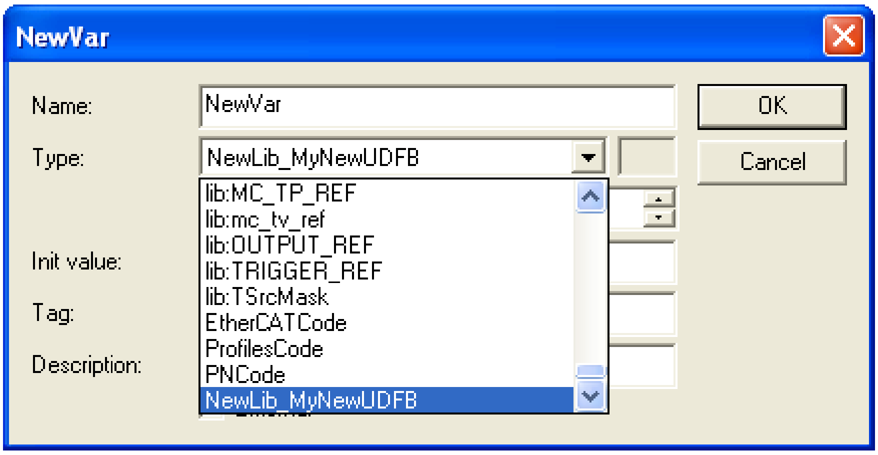


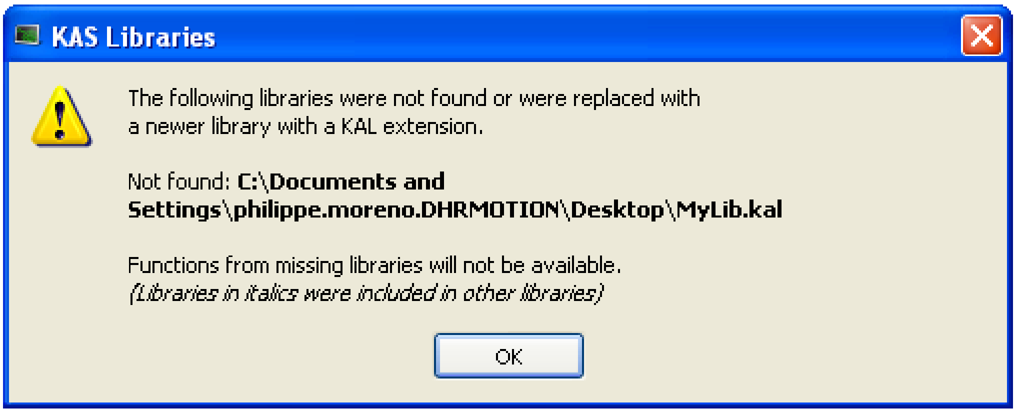
Figure 4-49: Use a Custom Library - Select the Type

What happens when you remove a library from your project?

If you remove a library from your project, all its types are removed from your project and all variables based on the library are displayed in the dictionary in red with question marks

What happens when a library no longer exists?

If you open a project containing a link on a library which is no longer available, a warning is displayed:



To recover the libraries, you have two options:

- Enter the new path to this library (assuming it still exists on your machine) using the library dialog (see "Figure 4-46: Use a Custom Library - Select the Library " on page 268).
- Find the missing library and copy the library back to the path originally specified. Note that the project has to be closed and re-opened for the library to be read again.

Broken link displayed in *Italics*

If a library references another library which is no longer available, a dialog with the library link that causes the problem is displayed in italics.

It can happen for example if your project has referenced LIB-4, which in turn references LIB-1-ND, but LIB-1-ND does not exist.

To recover your project, you have to open LIB-4 and fix the issue (i.e. LIB-1-ND broken link), then re-open this project again.

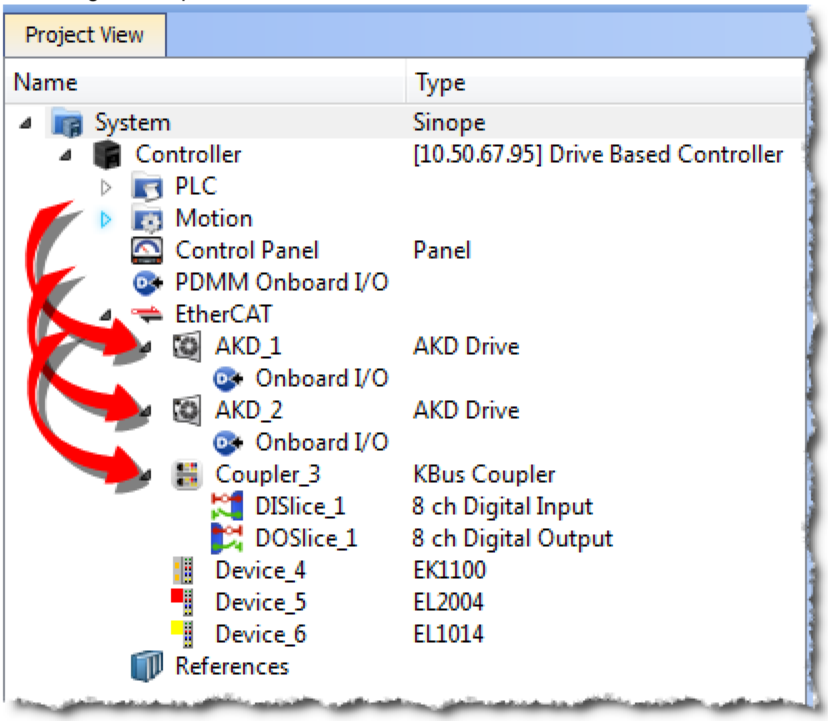
5.4.11 Step 11 of 15 - Map Input and Output to Variables

This procedure describes how to map EtherCAT motion bus I/O or AKD PDMM Onboard I/O to PLC variables.

> **NOTE**
> This operation is disabled when the controller is running.
> For Profibus fieldbus, you have to do the I/O mapping directly from the Dictionary. For more details, refer to "I/O Mapping (for Profibus Fieldbus)" (see page 516)

5.4.11.1 Map from the Project Explorer

1. In the Project Explorer, expand the:
   - Kollmorgen **AKD** to access the AKD's **Onboard I/O**.
   - Kollmorgen Coupler to access the Slice I/O.



2. Double-click the PDMM Onboard I/O, the AKD's Onboard I/O, or I/O Slice to open its properties window.

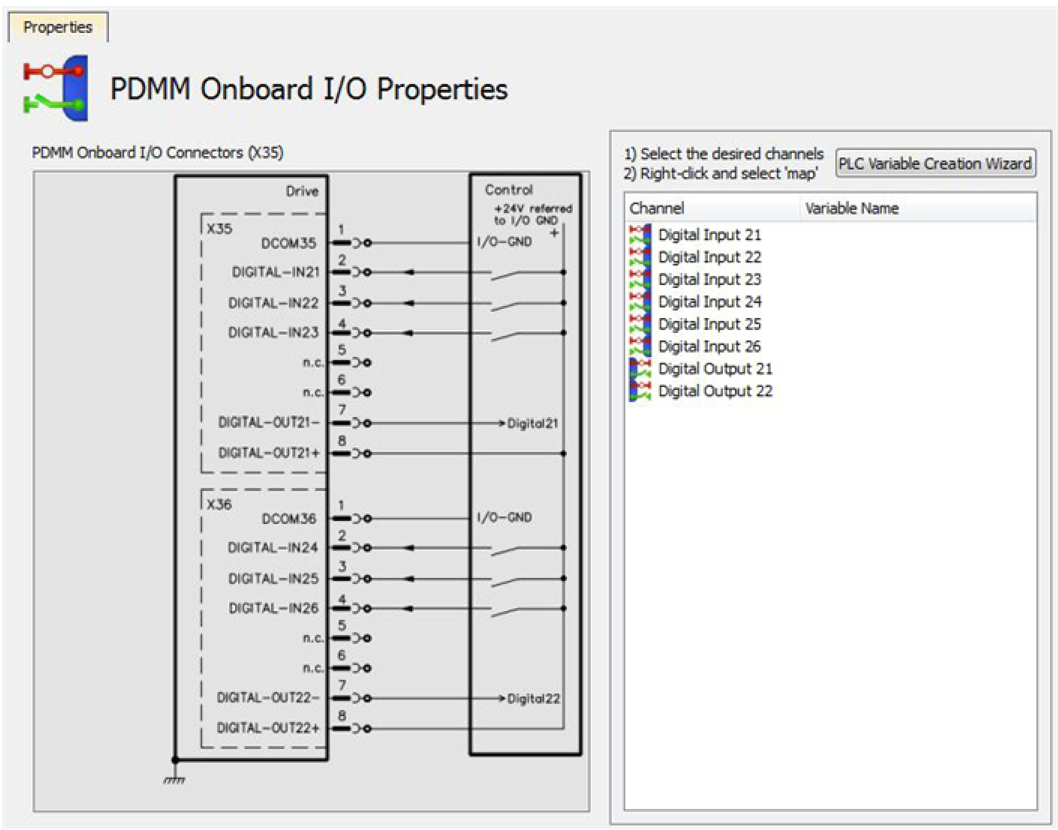


Figure 4-50: PDMM Onboard I/O

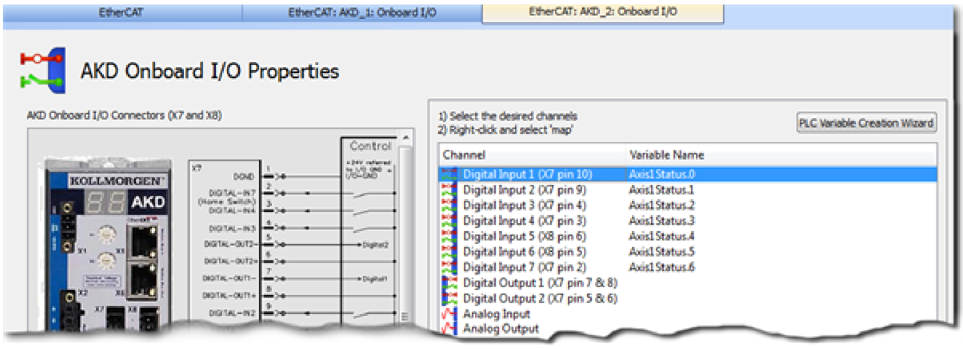


Figure 4-51: AKD's Onboard I/O

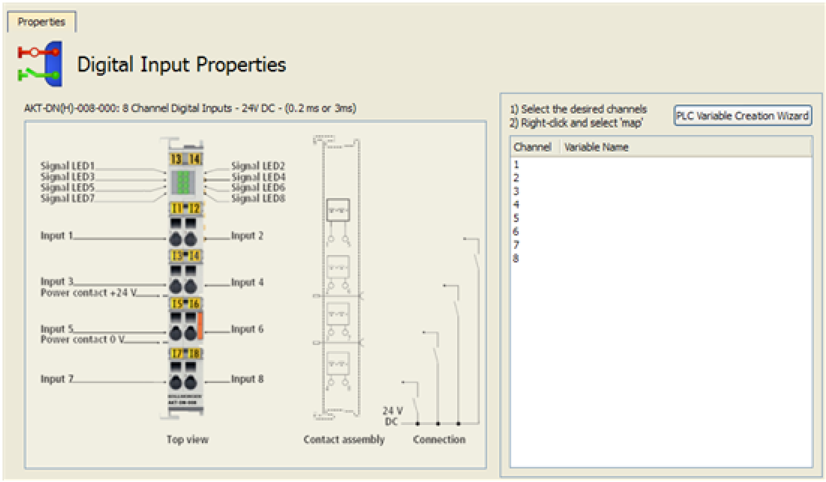
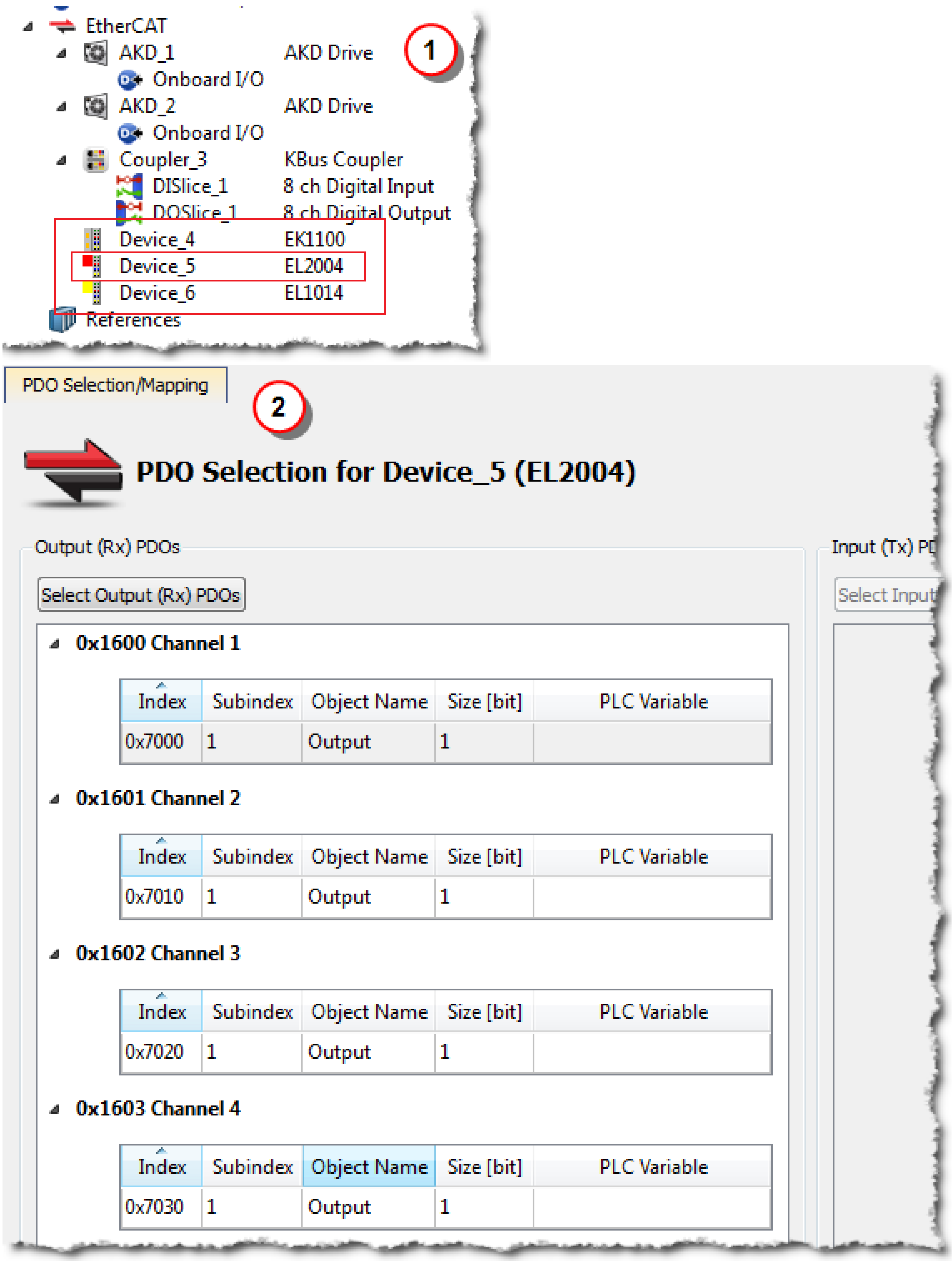
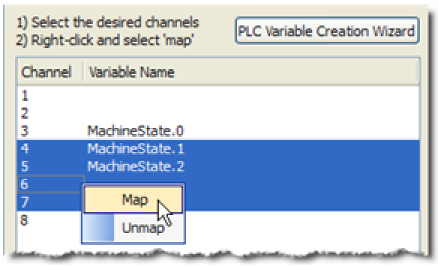


Figure 4-52: I/O Slice

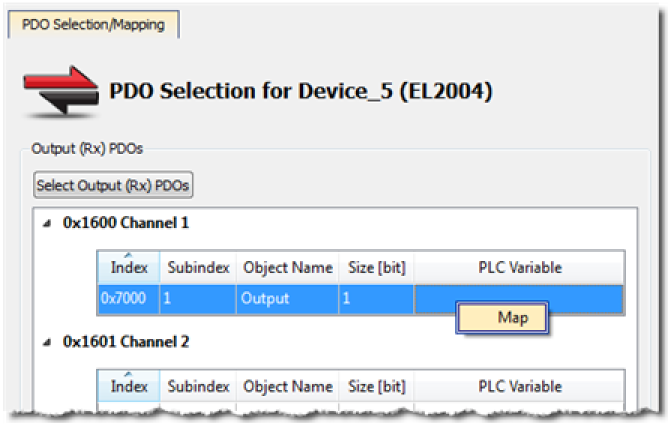
3. For Third Party devices, double-click the device (#1 below) and select the PDO Selection/Mapping tab (#2).



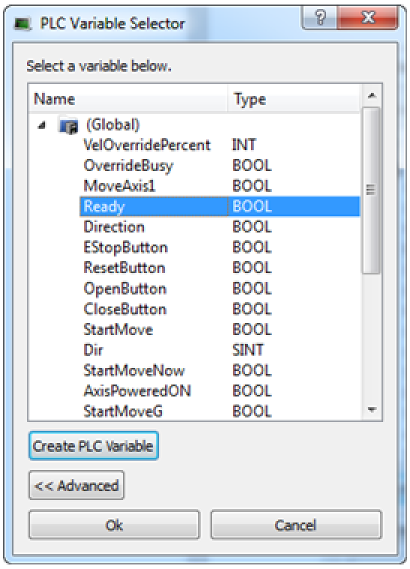
4. Use the "PLC Variable Creation Wizard" (see page 275) for Kollmorgen devices.
5. You may directly map the Inputs/Outputs to PLC variables.
   - Kollmorgen devices: Select the channel(s) you want to map. Multiple channels may be selected by click-dragging or by pressing Shift+Up/Down Arrow.



- Third-party devices: Select a PLC Variable cell associated with the PDO object index.



6. Select the **Map** command in the menu to open the "PLC Variable Selector" (see page 276).
7. Choose the variable to be linked to the channel(s) or PDO object.



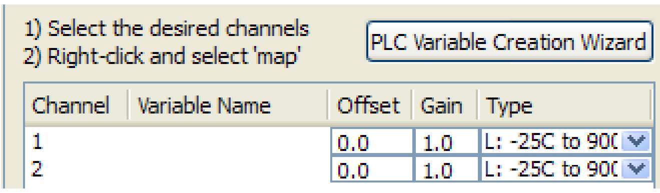
NOTE

- The list of variables is filtered to display only those with relevant types.
- For analog I/O, only variables with integer types are displayed.
- Because a variable can only be mapped to one channel or PDO object, when you link a variable to a new channel or PDO object, the previous mapping is removed (even if linked to another slice or device).
- PLC variables cannot be mapped to multiple PDMM Onboard I/O, AKD Onboard I/O, or Slice I/O.
- Read Only PLC variables should not be mapped to inputs because the value will not be able to change to match the input state.

NOTE

- *Do not map PLC variables to third-party drive PDO objects.* They may conflict with the KAS Runtime's internal motion engine read/write operations.
- Double-check before any confirmation because there is no possibility to Undo this operation.
- For details on the **Create PLC Variable** and **Advanced** buttons, see "PLC Variable Selector" (see page 276).

8. Click **OK**
9. For analog I/O and thermocouples, you also have to define offset and gain parameters



- For more details on parameters, see "Analog I/O Parameters" (see page 278).
- For more information on the AKD Onboard EtherCAT I/Os, see "Configure Onboard I/O" (see page 193).
- For more information on the AKD PDMM local digital I/Os, see "Configure AKD PDMM Onboard I/O" (see page 370).

The **Unmap** command in the contextual menu (see figure in step 4 above) allows you to remove the link between the variable and the associated channel(s). In addition, deleting a variable from the dictionary which is mapped to the channel(s) also removes the link(s).

NOTE

Important Note About PLC Variable Mapping

Please be aware of the following limitation if PLC variables.

Each PLC variable can be mapped to an EtherCAT I/O and exclusively to either:

- Modbus for an HMI
- a PDMM onboard I/O
- an external driver such as Profibus

For example, the same PLC variable cannot be mapped to both Modbus and an onboard PDMM I/O but it is possible with a regular EtherCAT I/O.

5.4.11.2 PLC Variable Creation Wizard

This wizard allows you to automatically create a list of variables used for the mapping.
The variable type is **Boolean** for digital I/Os and **UINT** for analog I/Os.

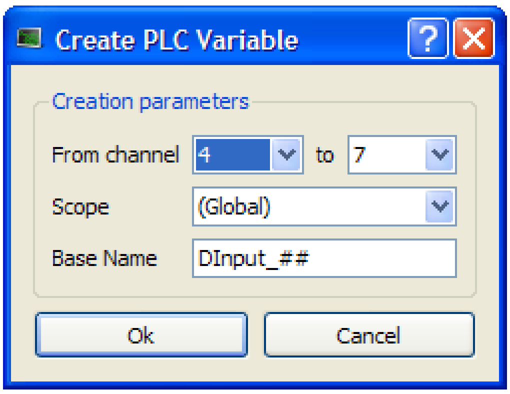


Figure 4-53: Wizard to Create PLC Variable - Parameters

Field	Description
From channel	Defines the range of channels you want to map automatically
Scope	Defines where the variables are created (if you select the **Global** scope, then the variables are created under the Global node in the Dictionary)
Base Name	Pattern used for variable naming where **##** are replaced with the channel number

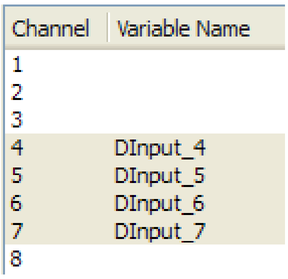


Figure 4-54: Wizard to Create PLC Variable - Mapped Channels

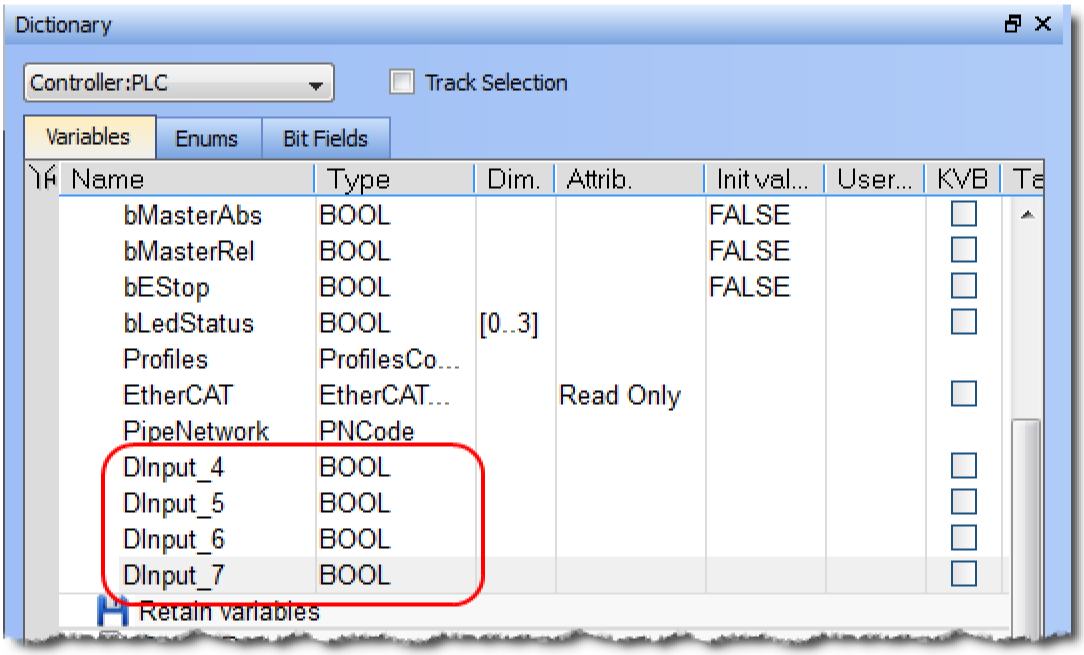


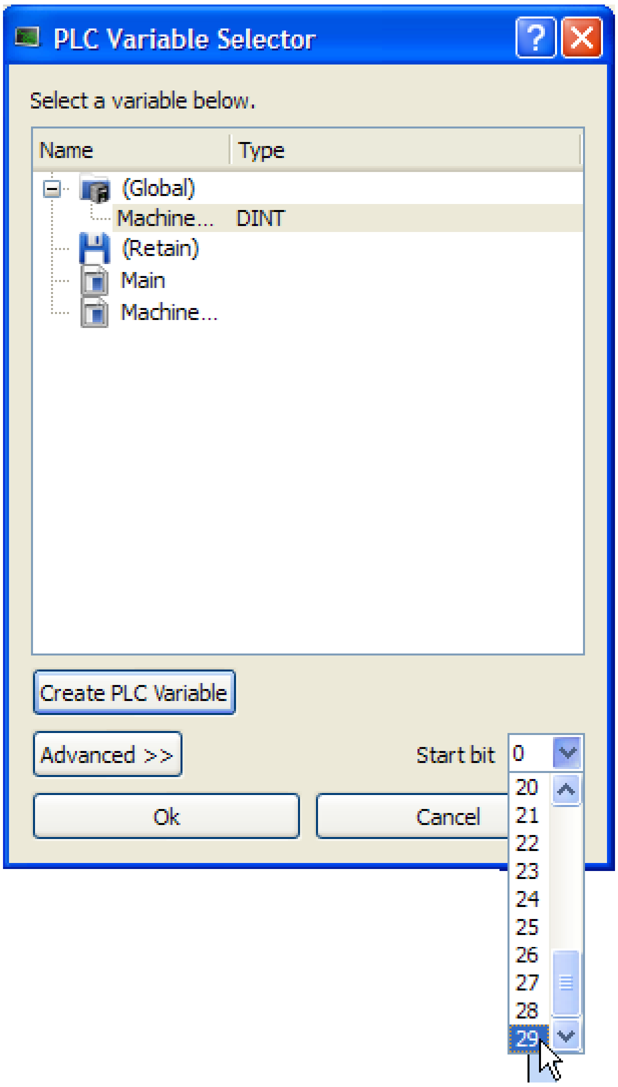
Figure 4-55: Wizard to Create PLC Variable - Variables in the Dictionary

5.4.11.3 PLC Variable Selector

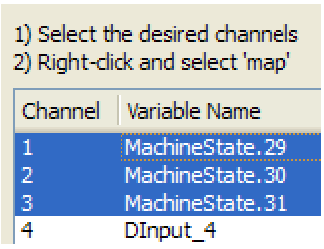
Advanced Button

For integer variables with types stored on several bits, the Advanced button gives access to the **Start bit** definition. This allows you to link a set of channels to a specific range of bits within an integer variable.

For example, when you select three channels ranging from 1 to 3 and map them to a DINT variable (stored on 32 bits ranging from 0 to 31), the first channel can be linked to position ranging from 0 to 29.

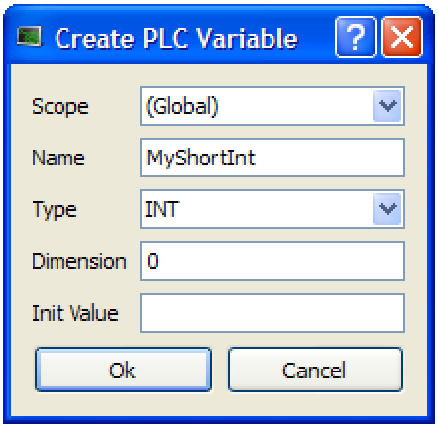


The three channels are mapped to the last three bits ranging from 29 to 31.



Create PLC Variable Button

This button allows the creation of a new variable to be linked to the selected channels.



Field	Description
Scope	Defines where the variable is created
Name	See "Name a variable" (see page 614)
Type Dimension	You can define the Type of the variable, and its Dimension if the variable is an array
Init Value	See "Initial Value of a Variable" (see page 618)

See also "Step 6 of 15 - Create Variables" (see page 253)

5.4.11.4 Analog I/O Parameters

Input Terminals

The process data that are transferred to the Bus Coupler are calculated using the following equation:

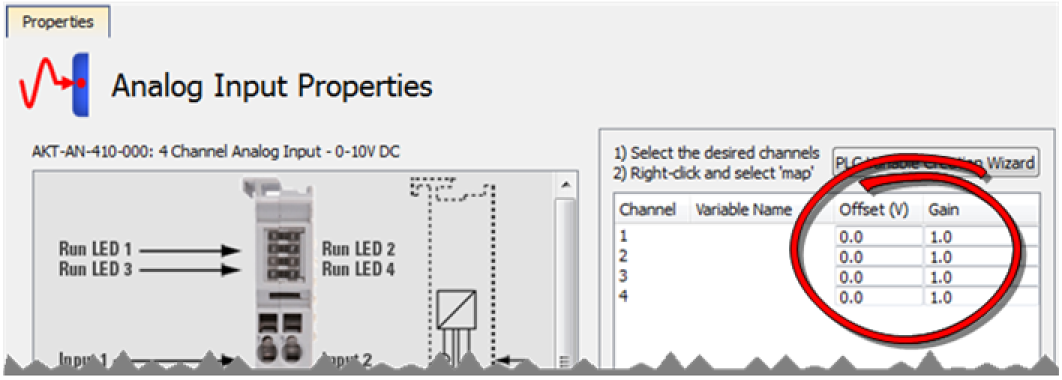
$$Y_a = (B_a + X_{ADC}) \times A_a$$

$$Y_{out} = B_w + ((A_w + A_h) \times Y_a)$$

With the following parameters:

X_{ADC}	Output values of the Analog Input Modules A/D converter
Y_{out}	Process data to the controller
B_a , A_a	Manufacturer offset and gain compensation ‡
A_h	Manufacturer scaling: default gain ‡
B_w , A_w	User scaling: Offset and Gain as set in the *Analog Input Properties* (see image below).

‡ For the thermocouple input terminals, AKT-AN-200-000 and AKT-AN-400-000, the manufacturer default gain is 160. For all other supported terminals, the manufacturer default gain is 1. The manufacturer default offset is zero for all supported terminals.



Output Terminals

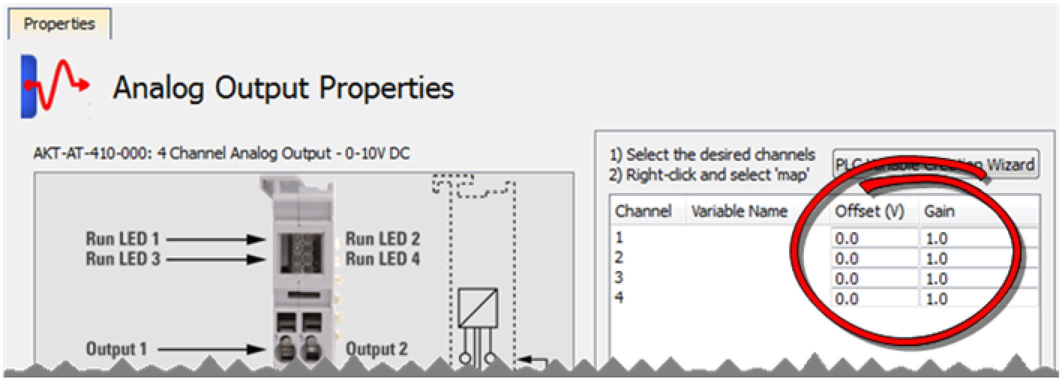
The process data that are transferred to the Bus Coupler from the controller are calculated using the following equations:

$$Y_2 = B_w + ((A_w \times A_h) \times X)$$

$$Y_{dac} = Y_2 \times A_a + B_a$$

X	Controller Process data
Y_{dac}	Controller data to analog output module D/A converter
B_a , A_a	Manufacturer offset and gain compensation ‡
A_h	Manufacturer scaling: default gain ‡
B_w , A_w	User scaling: offset and gain as set in the *Analog Output Properties* (see image below).

‡ The manufacturer default offset is zero for all supported terminals. The manufacturer default gain is 1 for all supported terminals.



5.4.12 Step 12 of 15 - Adding Motion

There are two ways to create motion, depending on the motion engine:

- For Pipe Network, refer to "Design Motion with Pipe Network" (see page 279)
- For PLCopen, refer to "Design Motion with PLCopen Axis" (see page 289)
- If you are not sure which engine to use, refer to "Pipe Network or PLCopen" (see page 94)

For high-level discussions about motion, refer to the sections within "Motion Concepts" (see page 91).

5.4.12.1 Design Motion with Pipe Network

The contents of this section detail how to create and modify a Pipe Network.

Create the Pipe Network

To create the Pipe Network, do as follows:

1. In the Project Explorer, double-click the **PipeNetwork** button to open the graphical Pipe Network Editor

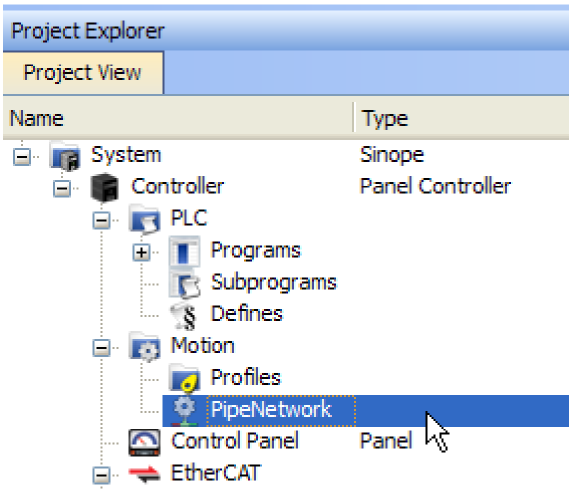


Figure 4-56: Pipe Network - Open Editor

NOTE If you have created a project from a template (for instance the standard two-axis template) there is already a Pipe Network in the editor.

2. To add a new Pipe Block, right-click on the editor's background and select the **Add Pipeblock** command in the menu
3. Choose in the drop-down menu the type of Pipe Block you want to add

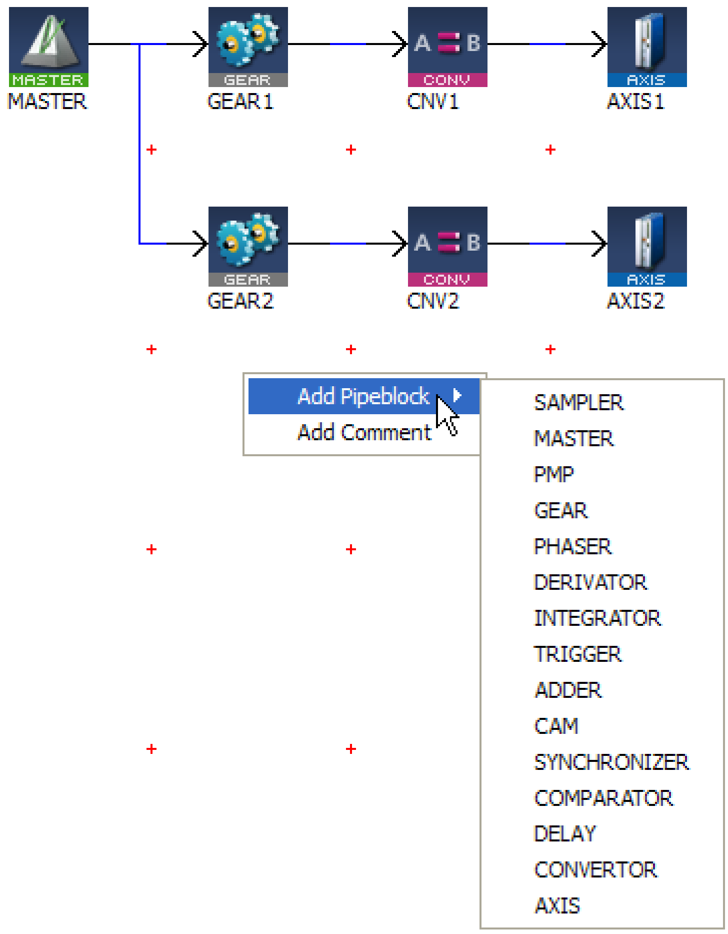


Figure 4-57: Pipe Network - Add Pipeblock

4. To link the newly created Pipe Block, move the arrow to the corresponding Pipe Block with a drag-and-drop operation

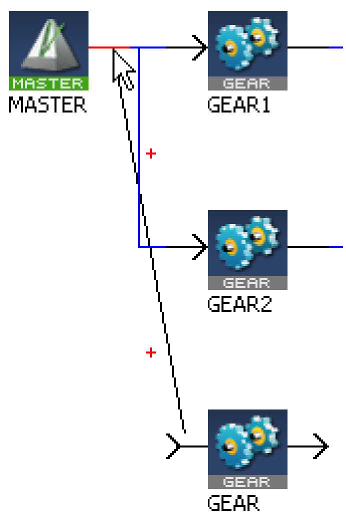


Figure 4-58: Pipe Network - Create a Link

How to delete a Pipe Block?

Right-click on the Pipe Block and select the **Delete** command in the contextual menu.

How to change a link?

1. Select the link so that it becomes Red

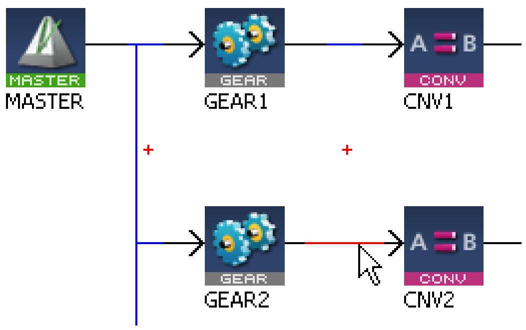


Figure 4-59: Pipe Network - Edit a Link

You can either:

- Right-click and select the **Delete** command if you want to remove the link

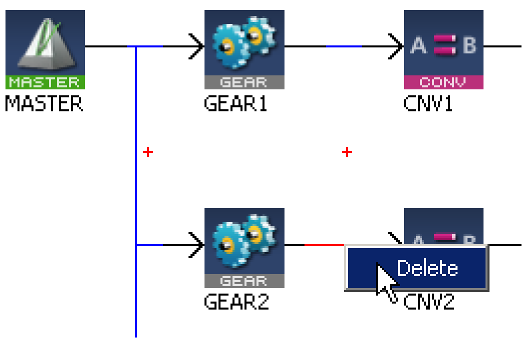


Figure 4-60: Pipe Network - Delete a Link

- Move the arrow to another Pipe Block with a drag-and-drop operation

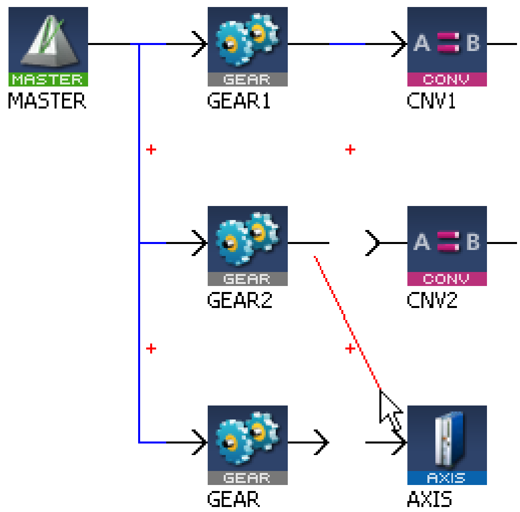


Figure 4-61: Pipe Network - Move a Link

See also §O.3: Application Notes for application examples

Edit Properties of Pipe Blocks

Initial values for Pipe Network blocks are entered in the parameter screen for each block. To get to the parameter screen, right-click on a Pipe Block and select the **Properties** command in the contextual menu.

- Right-click on the Pipe Block and select the **Properties** command in the menu

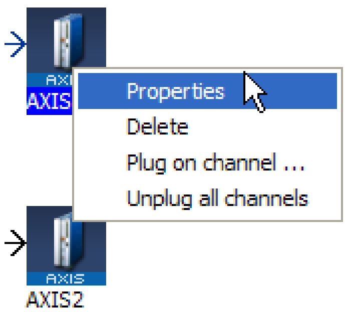


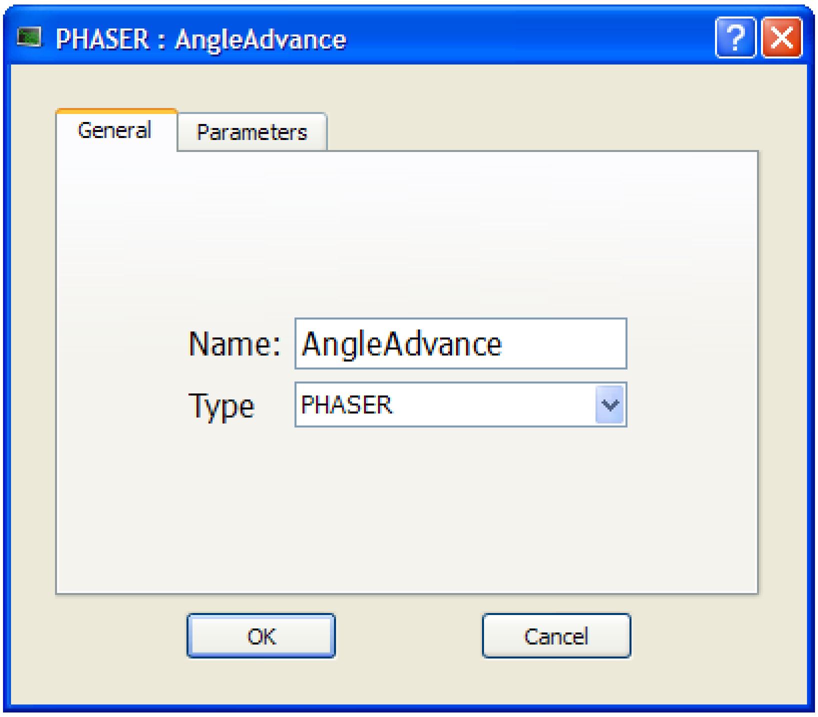
Figure 4-62: Pipe Network - Pipe Block Properties

You can change the name (or even the type of Pipe Block) in the **General** tab.

The **Parameters** tab gives access to properties related to the type of Pipe Block .

See example

In this example, the selected name “AngleAdvance” would be used in the PLC application program for this Pipe Network block.



Map the Axis to the Drive

To link the axis to an EtherCAT drive, you have to do the mapping as described in paragraph "Mapped to Axis" on page 197.

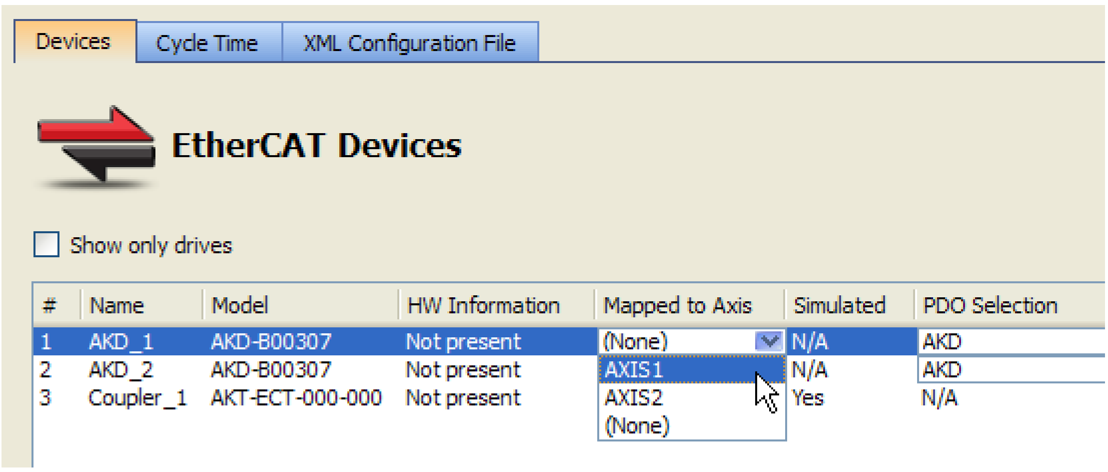


Figure 4-63: Pipe Network - Mapping Axis to Drive

Add Comments

To add a comment:

1. Right-click on the editor's background and select the **Add Comment** command in the menu
2. Right-click on the comment opens the contextual menu to let you edit (**Properties** command) or delete the comment

Set the Position Units

You can set up the position units in the parameter screen of the Axis block.

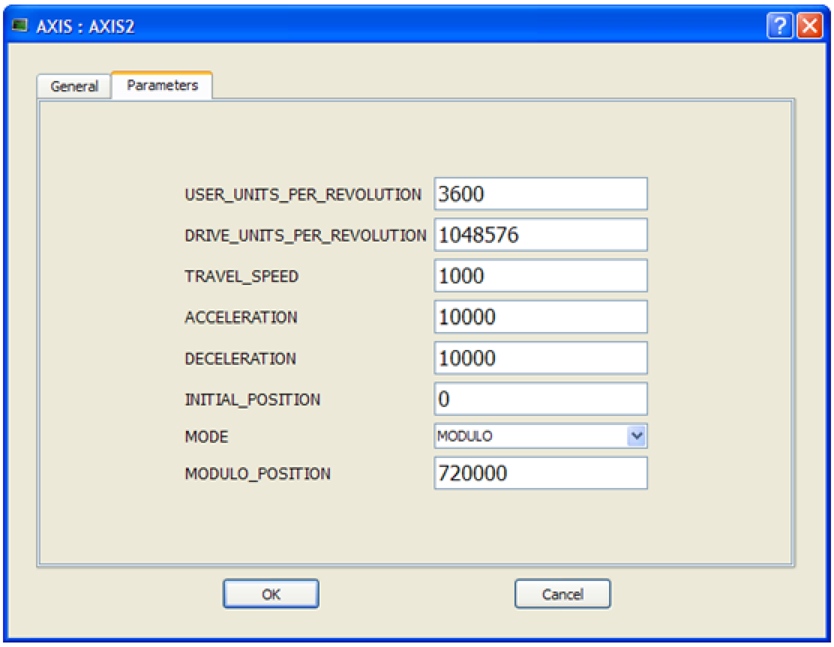


Figure 4-64: Setting Axis Units

Some guidelines for suitable settings advises for a good choice is given below:

- The unit is adapted for the machine
- The unit must be meaningful for the user
- The same unit must be used for all related axes, for reasons of simplicity
- The unit must be set as soon as possible and must not be changed during the program lifetime, for reasons of consistency
- **Speed** is defined in User Units for position / second
- **Acceleration** in User Units for position / second^2
- The unit must be related to the final moving object, instead of any intermediate part (e.g. the driven belt rather than the motor or axis shaft, which are intermediate parts)

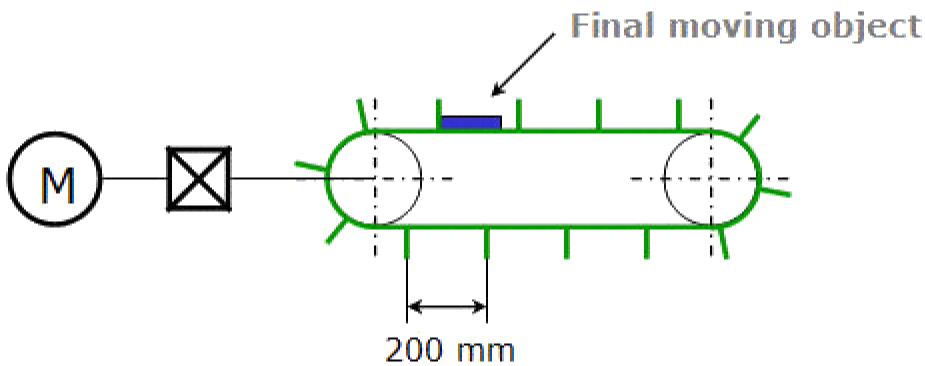


Figure 4-65: Setting the Units - Example

A User Unit = 0.1mm could be selected for this transportation system

Show Pipe Network and Profiles-Generated Code

You can access the code equivalent to the graphical representation with the contextual menu of the Pipe Network item in the Project Explorer as follows:

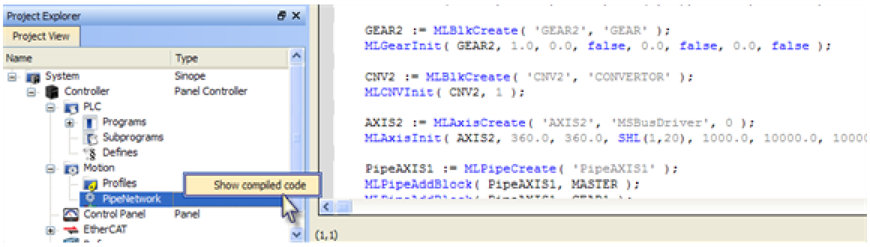


Figure 4-66: Display Source Code of the Pipe Network

The KAS IDE provides a set of Functions and function blocks for each of the Pipe Blocks. These function blocks allow the logic part of the application to control and interact with the motion engine.

Pipe Network Functions for the PLC

After creating the Pipe Network, the complete project has to be compiled before you can use the Pipe Network in your PLC Programs. Compiling creates a list of Functions that can be used in the PLC Program. These Functions simplify programming by combining the same function block for all axes in the Pipe Network:

Pipe Network Function	Function Blocks included (for 2 axis system)
MLPN_ACTIVATE :	MLPipeAct(PipeAXIS1); MLPipeAct(PipeAXIS2);
MLPN_CONNECT :	MLCNVConnect(CNV1, AXIS1); MLCNVConnect(CNV2, AXIS2);
MLPN_POWER_ON :	MLAxisPower(AXIS1); MLAxisPower(AXIS2);
MLPN_POWER_OFF :	MLAxisPowerOFF(AXIS1); MLAxisPowerOFF(AXIS2);
MLPN_DEACTIVATE :	MLPipeDeact(PipeAXIS1); MLPipeDeact(PipeAXIS2);

For more details on all constant definitions related to Pipe Network, see page 262

TIP To see how these functions are used, open a project, go to the Project Explorer, right-click on PipeNetwork and select the **Show compiled Code** command

Initialize and Start up a Pipe Network

See Motion State Machine

The Motion State Machine is driven by the IEC 61131-3 application with the help of dedicated function blocks.

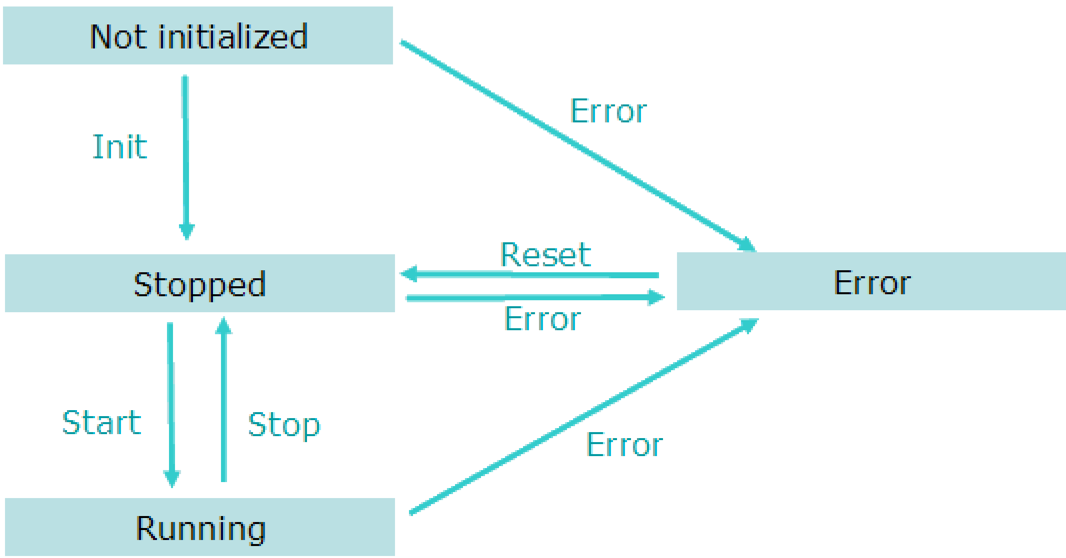


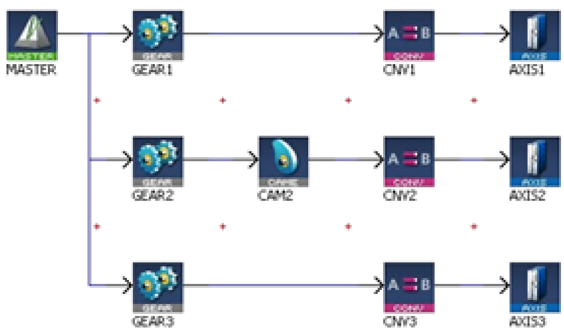
Figure 4-67: Motion State Machine

Each arrow represents a transition from one State to another one.

To start-up a Pipe Network in your IEC 61131-3 application program, you have to perform the following steps with their respective functions:

Step	ML function blocks	Description
Motion Init	MLMotionInit	Initialization of the Motion is done with this dedicated function Set the Motion engine update rate. Wait for acknowledgement: MLMotionStatus() = MLSTATUS_INITIALISED to continue program operation
Create Cam Profiles	Profiles(MLPR_ CREATE_ PROFILES);	Create Cam Profiles from cam files
Create Pipe Network	PipeNetwork (MLPN_ CREATE_ OBJECTS);	
Motion Start	MLMotionStart	Starts the motion engine, motion bus driver, and initializes EtherCAT network to operational mode, then waits for acknowledgement: MLMotionStatus() = MLSTATUS_RUNNING to continue program operation
Power on all axes	PipeNetwork (MLPN_ POWER_ON);	
Activate the pipes	PipeNetwork (MLPN_ ACTIVATE);	

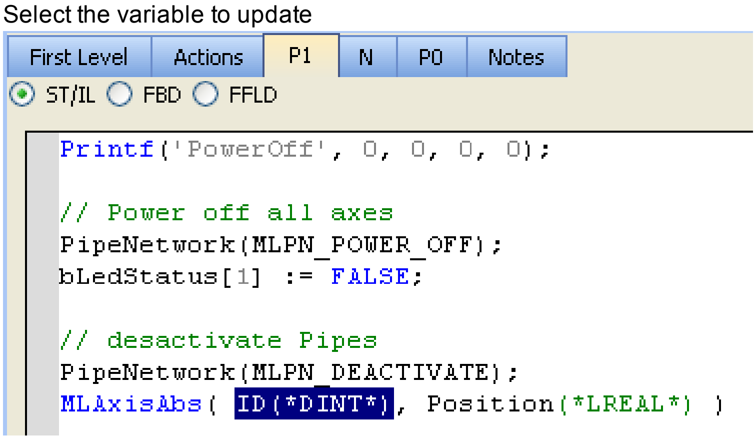
Step	ML function blocks	Description
Connect the axes to the pipes	PipeNetwork (MLPN_ CONNECT);	For example: in the following Pipe Network this function connects the Converter blocks (CNV1, CNV2 and CNV3) to the Axis blocks



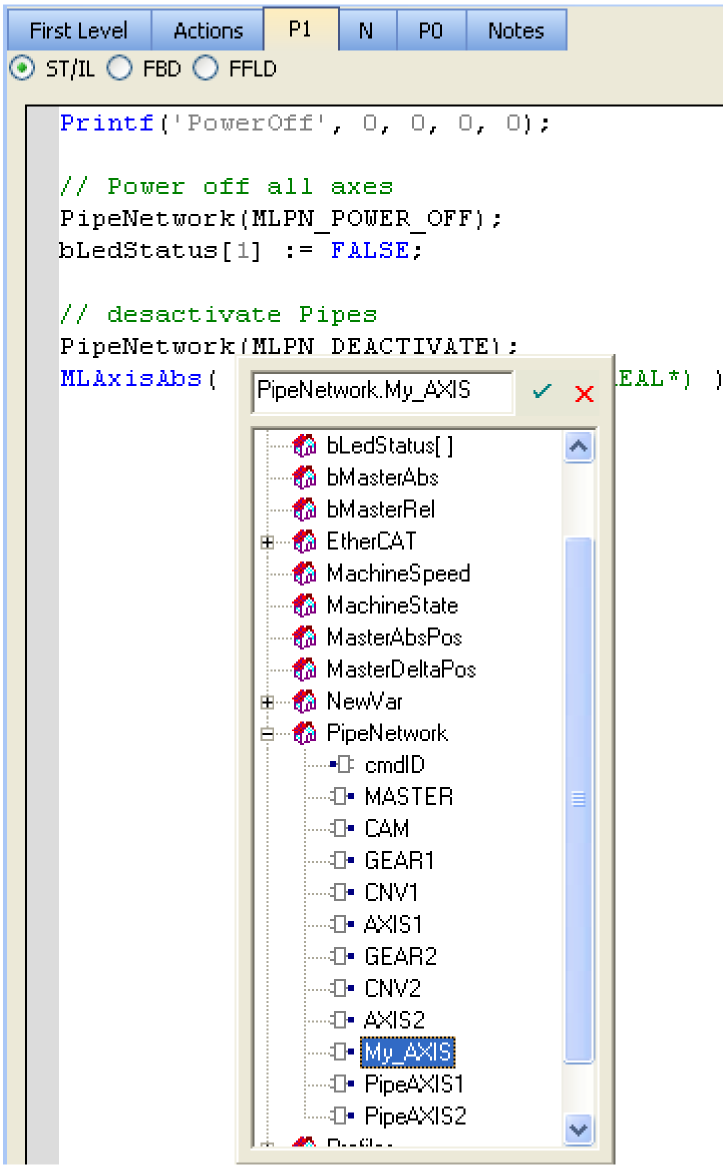
How the Pipe Network interacts with PLC programs

Each Pipe Block is supported by several ML function blocks in the function block Library. As soon as you add a Pipe Block, it is included as well in the Variable Editor.

- Add the FB into your program (see procedure here)
- Select the variable to update



- Press **CTRL+SPACE** to open the Variable Editor
- Expand the PipeNetwork node and select the name of the Pipe Block in the list (all the Pipe Blocks created in the Pipe Network are listed)



Then your ST instruction is updated

```
// desactivate Pipes
PipeNetwork(MLPN_DEACTIVATE);
MLAxisAbs( PipeNetwork.My_AXIS, Position(*LREAL*) )
```

NOTE When you add a new Block in the Pipe Network, you first need to compile your project to make the block visible in the list of items.

- Click the ✔ icon to update your code

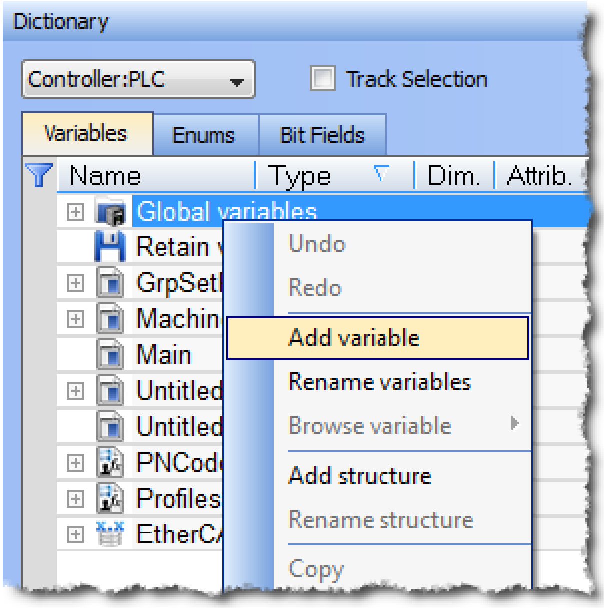
5.4.12.2 Design Motion with PLCopen Axis

This chapter explains how to modify an existing PLCopen Axis, and how to create a new one.

Create PLCopen Axis

To create a new PLCopen axis, follow these steps:

1. In the Project Explorer, right-click on the PLCopen item and select the **New Axis** command in the menu
2. Fill in the PLCopen Axis Data dialog
3. In the Dictionary, right-click on the **Global variables** node and select the **Add variable** command in the menu.



4. Create a new instance of the AXIS_REF data structure

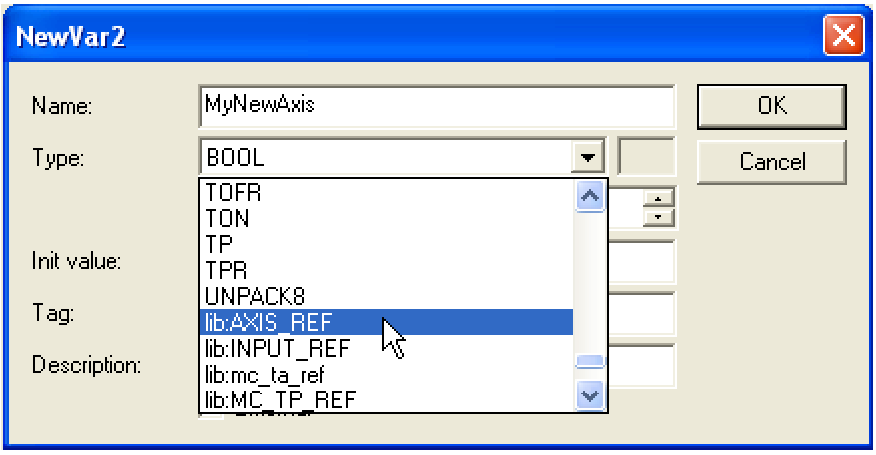
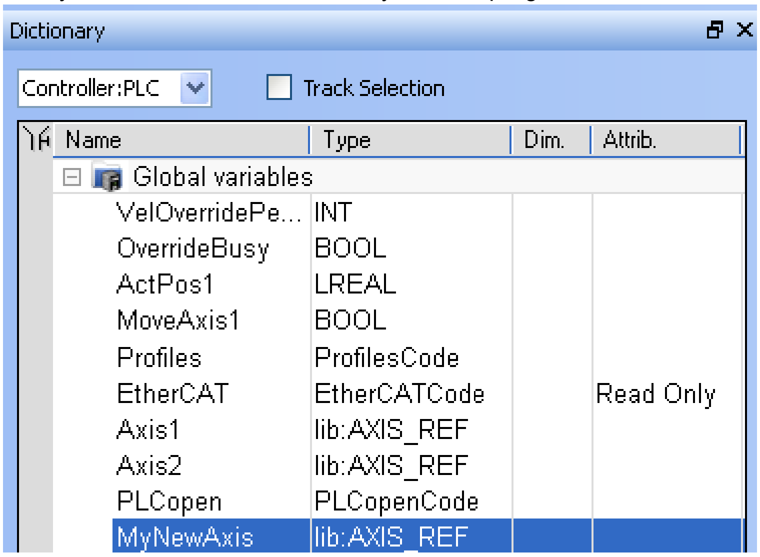


Figure 4-68: PLCopen Axis - New Instance of AXIS_REF

NOTE The name must be the same as the **Name** field defined in the PLCopen Axis Data dialog. The KAS IDE already contains the AXIS_REF data structure when you choose the PLCopen motion engine.

5. Then, this Axis Name (**MyNewAxis** in our example) is an instance of an AXIS_REF library function that can be used in your PLC programs



TIP

- In FFLD, the **Copy** function block is needed to load the Axis Number (defined in the PLCopen Axis Data dialog) into the new data structure.
- In ST, use a statement (Example: Axis10.AXIS_NUM := 10;)

Modify PLCopen Axis

A PLCopen axis can be modified by using the PLCopen Axis Data dialog. To display this dialog you can:

- Double-click on a PLCopen axis in the Project Explorer
- Right-click the PLCopen axis in the project manager and select **Properties** in the menu as shown below

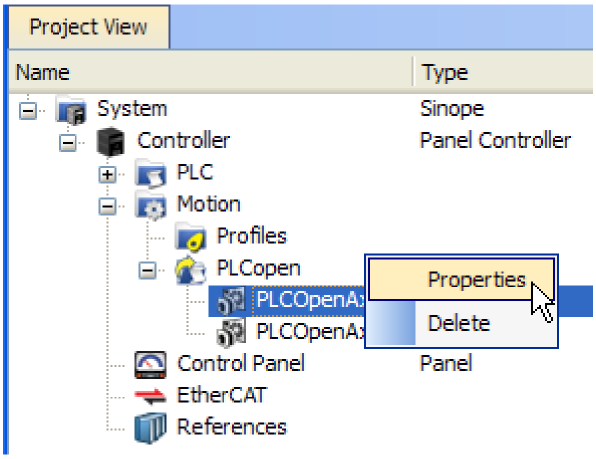


Figure 4-69: PLCopen Axis Context Menu

The PLCopen Axis Data dialog is displayed as follows:

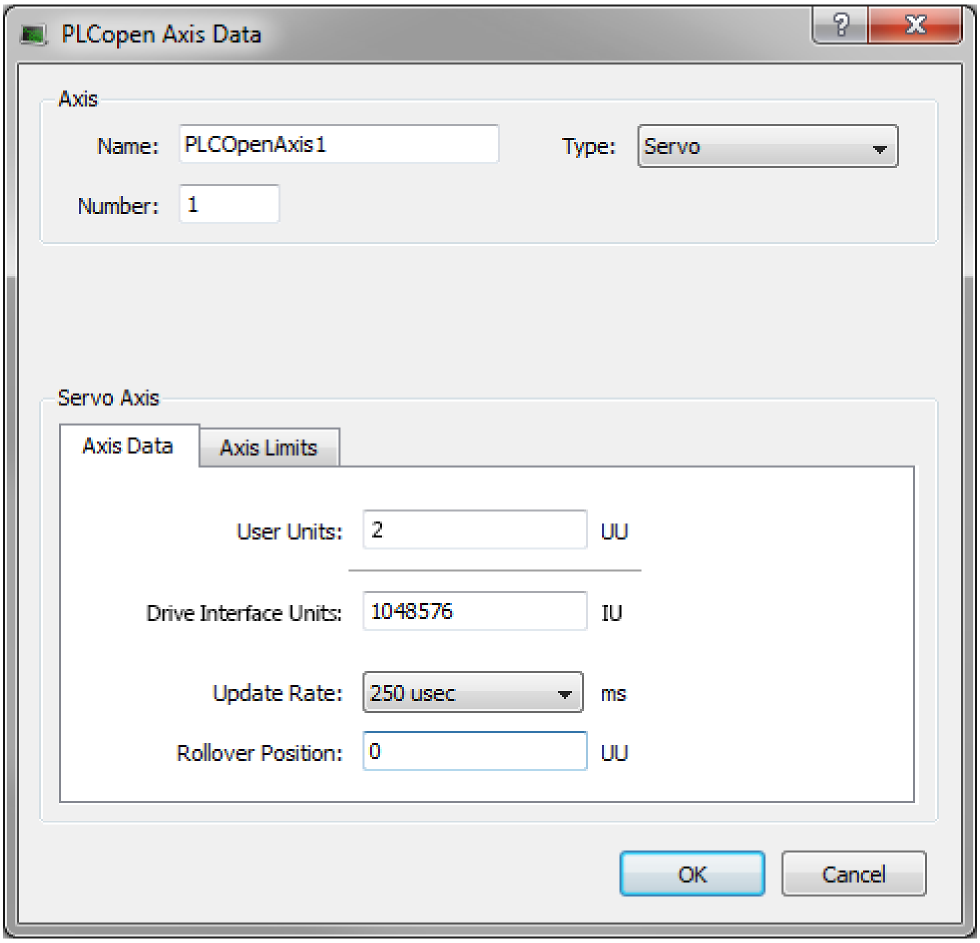


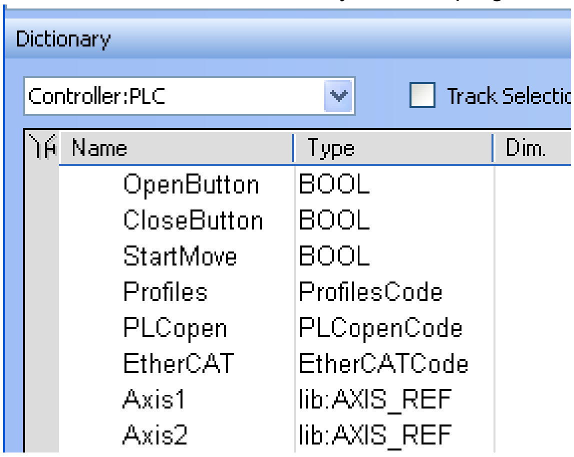
Figure 4-70: PLCopen Axis Data Dialog

About Axis Name and Number

NOTE AXIS_NUM is the same number as the one used in the PLCopen Axis Data dialog (see field **Number** in the Axis frame).

The **Copy**Copy function block is needed to link the Axis Number defined in the PLCopen Axis Data dialog (**1** in the figure above) to the Axis Name (**Axis1** in our example)

Then, this Axis Name (**Axis1** in our example) is an instance of an AXIS_REF data structure that can be used in your PLC programs.



Common Axis Parameters

Three types of axes are available: *Servo*, *Digitizing* and *Virtual Servo*. All types have common parameters related to an axis.

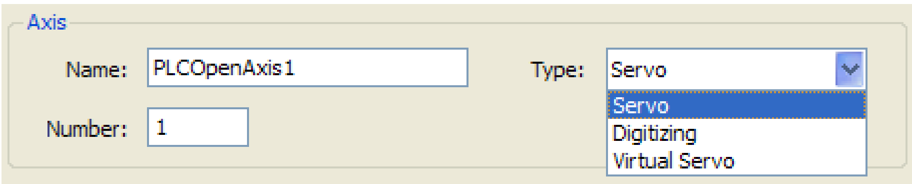


Figure 4-71: PLCopen Axis Parameters

Parameter	Description
Name	The user-defined name of the axis. The name can consist of 1-16 alphanumeric characters. Spaces are not allowed in the name. The Axis Name identifies the axis displayed on the KAS Simulator.
Type	A *Servo* axis is closed loop: commands are sent to the axis and feedback is read from the axis. A *Digitizing* axis is read-only, open loop: only feedback is read from the axis. A *Virtual Servo* is a servo axis with no feedback or drive hardware. The feedback for a virtual servo axis is automatically generated from the command position. There is no limit to the number of virtual axes that may be used in an application.
Number	The axis number (1-256) specifies the axis for PLCopen motion function blocks.

The Digitizing axis type has some additional Bus parameters to define the fieldbus.

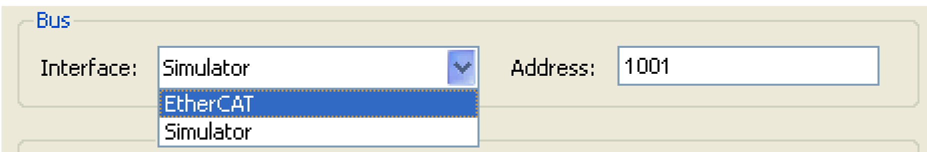


Figure 4-72: PLCopen Axis - Bus Parameters

The bus parameters are:

Parameter	Description
Interface	The type of bus interface. The choices are: • EtherCAT • SynqNet • Simulator Since the EtherCAT setup does not support a digitizing axis, you have to specify the bus interface so the KAS IDE can create the axis correctly.
Address	The 4-digit node address of the servo drive on the bus. This address is required to assign a digitizing axis to an EtherCAT node that already has a servo axis assignment.

NOTE The bus parameters are also displayed when you choose to import an external XML file to describe the EtherCAT Motion Bus.

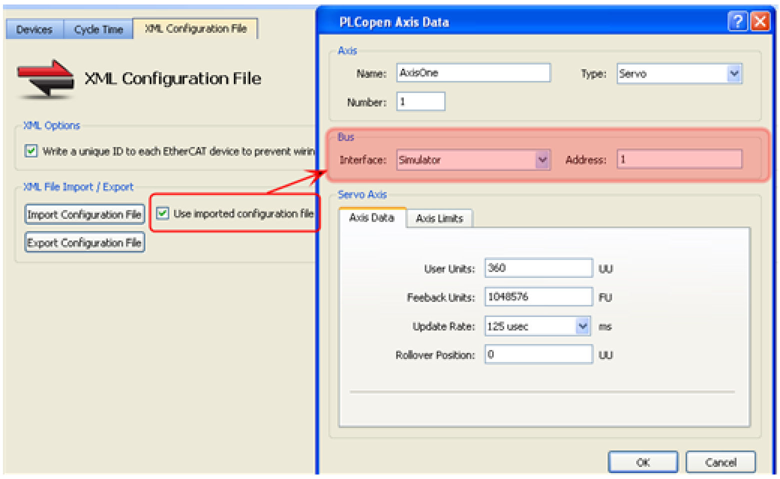


Figure 4-73: PLCopen Axis Parameters with Imported XML

Axis Data

If a Servo axis is selected, two tabs are available: Axis Data and Axis Limits. If a Digitizing axis is selected, only the Axis Data tab is available.

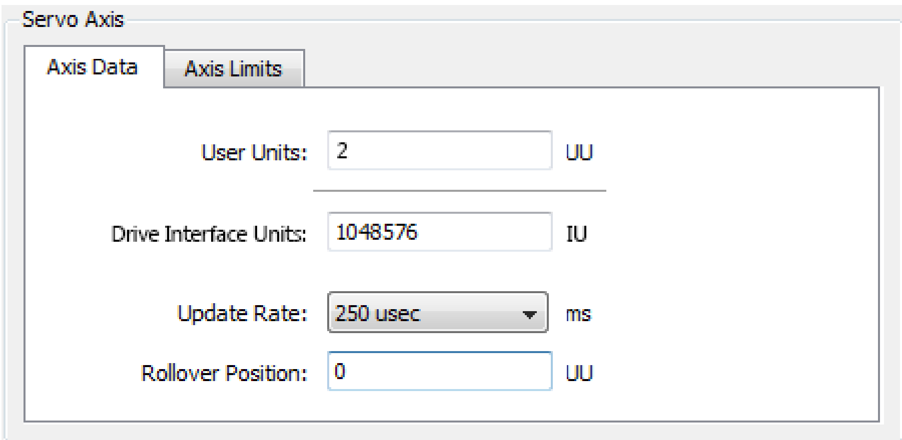


Figure 4-74: Servo Axis - Axis Data

The Servo Axis - Axis Data parameters are:

<table>
<tr><th>Parameter</th><th>Description</th></tr>
<tr><td>User Units</td><td>The User Units portion of the User Units / Interface Units ratio.

The application program specifies positions in User Units. Positions are commanded to, and read from the drive bus interface in Interface Units. The User Units default value is 360. The default ratio is 360 User Units / 1048576 Interface Units.</td></tr>
<tr><td>Drive Interface Units</td><td>The Interface Units portion of the User Units/ Interface Units ratio.

The drive provides an axis feedback resolution of 2^20 or 1048576 Interface Units per motor revolution. If the application programmer desires to work in User Units of degrees, the User Unit field of the PLCOpen Axis Data dialog typically would be set to 360, to define a 360 degree to 1048576 interface unit ratio. Similarly, if the axis scaling is 1000 user units per motor revolution, the ratio would be entered as 1000 User units to 1048576 Interface Units in the Axis Data dialog.

There may be times where it is desired to work with an integer number of User Units per partial revolution of the motor, or a non-integer number of User Units per revolution of the motor. To accommodate this, the ratio of User Units to Interface Units must be computed such that both terms of the ratio are integers. See the examples below.

NOTE
Why enter the axis scaling as a ratio of integers rather than a decimal number?
Real numbers cannot exactly represent repeating decimals like 1/3.
An entry field must have a limit to how many significant digits are entered. The entry field may not be large enough to exactly enter the decimal.
Most machines are designed with lead screws and gear boxes that are typically represented in ratios.
This makes representing the UU to IU scaling as a ratio of integers the most exact and easily represented method.</td></tr>
</table>

Parameter	Description
Update Rate	The rate at which the axis's feedback is read and a new command position is generated. The choices are: 125 µsec 250 µsec 500 µsec 1 msec 2 msec 4 msec This rate can be slower or equal to the EtherCAT Cycle Time The EtherCAT Cycle Time specifies the rate at which data is transferred between the control and the drives. The axis Update Rate is the rate at which the PLCopen code reads the feedback, runs its interpolation, and generates a new command position. By allowing some axes to run at a slower rate and staggering the updates on which these axes are interpolated, more axes and/or quicker execution times can be achieved since every axis does not have to be interpolated every update. If you select an axis Update Rate which is faster than the EtherCAT Cycle Time, the axis is set to run at the EtherCAT Cycle Time.
Rollover Position	The value at which the axis position rollovers to zero. Rollover Position is specified in User Units. For example: If the rollover position is 1000, the axis position counts up from 0 to 999 and then rollover back to 0. In the reverse direction, the axis position counts down to 0 and then rollover to 999. If Rollover Position is 0, no rollover occurs. Axis positions become negative values when counting down below 0.

Example 1

It is desired to work in inches for a linear axis where the axis is driven with a rotary motor and lead screw of 20mm/revolution pitch. We must calculate a User Unit to Interface Unit ratio where both terms of the ratio are integers. (ratio shown below is optionally simplified)

```
20mm      1 in        1 Rev          20 in          200 in         25
* 8 in        25 in
----- * ------- * ---------- = ------------- = ------------ = -----
-------- = -----------
1 Rev   25.4 mm   1048576 IU   26633830.4 IU   266338304 IU
33292288*8 IU   33292288 IU
```

We would enter 25 User units, and 33292288 Interface units in the PLCOpen Axis Data dialog.

Example 2

It is desired to work in degrees for a rotary axis where the axis is driven with a gear ratio of 1/3 degree rotary motion per motor revolution. We must calculate a User Unit to Interface Unit ratio where both term of the ratio are integers.

```
1/3 deg     1 Rev          1 deg          1 deg
------- * ---------- = -------------- = ----------
 1 Rev    1048576 IU   1048576 * 3 IU   3145728 IU
```

We would enter 1 User Unit, and 3145728 Interface units in the PLCOpen Axis Data dialog.

Axis Limits

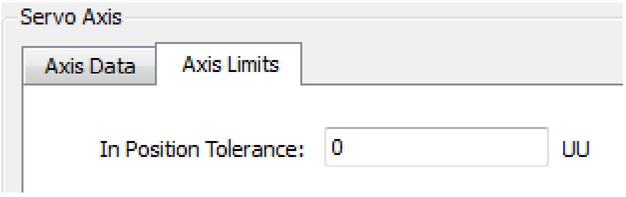


Figure 4-75: Servo Axis - Axis Limits

The Servo Axis - Axis Limits parameters are:

Parameter	Description
In Position Tolerance	The maximum distance between the axis's actual position and its commanded end-point for the axis to be considered “in position”. The In-Position Tolerance is specified in User Units.

5.4.12.3 How-To: Coordinated Motion

This section discusses how to create a coordinated motion application, including adding coordinated motion to existing applications.

For more information on Coordinate Motion an the associated functions and function blocks see:

- "Coordinated Motion" (see page 459) in the Advanced Topics section
- Coordinated Motion Function Blocks

Create a Linear or Circular Coordinated Motion Application

A Coordinated Motion application can be created in one of two ways:

- Use a Coordinated Motion template to create a new application. Two Coordinated Motion templates are currently available.
  - The first template controls two PLCopen axes in coordinated motion.
  - The second template controls two PLCopen axes in coordinated motion plus a third independent Pipe Network axis.
- Modify an existing application to included coordinated motion functions. When modifying an existing application, axes need to be grouped to define the axes that will be active when performing coordinated motion on that group. More information about Axes Groups can be found in the section "What are Axes Groups?" (see page 465).

NOTE Coordinated motion can only be performed with PLCopen axes. Pipe Network axes do not support this feature, although Pipe Network axes can be moved independently from coordinated motion groups. Any synchronization between coordinated motion and Pipe Network axes must be performed by the PLC application.

Related axes are "grouped" in an axes group. Coordinated motion is then performed on an axes group. For more information see "What are Axes Groups?" (see page 465).

Typically, the following set of function blocks should be called before executing coordinated motion.

1. Call MLMotionInit (BasePeriod) to initialize the motion engine. Base period is specified in microseconds.

```
MLMotionInit(1000.0);    // 1000 µSec -> 1 mSec
```

2. Call MC_CreateAxesGrp (Enable, GroupName, UpdateRate, MaxNumberOfAxes, AxesGroupRef) to create a Coordinated Motion Axes Group

   NOTE MC_CreateAxesGrp needs to be called between MLMotionInit() and MLMotionStart().

   ```
   Inst_MC_CreateAxesGrp(TRUE, 'GROUP1', 6, 2, Group1_
   ref);
   ```

   In the example above, the axes group name is 'GROUP1', the update rate is 1 mSec (specified by '6') and the maximum number of axes that can be added to the group is 2. The group reference variable 'Group1_ref' will be used in future coordinated motion function block calls to reference this newly created group.

3. Call MC_InitAxesGrp (Enable, AxesGroup, VelLimit, AccLimit, DecLimit, JerkLimit) to initialize the path limits for velocity, acceleration, deceleration, and jerk.

   ```
   Inst_MC_InitAxesGrp(TRUE, Group1_ref, 100.0, 300.0,
   300.0, 1000.0);
   ```

   In the example above, the kinematic limits for axes group 'Group1_ref' will be set. The velocity limit will be set to 100.0 user units/second, acceleration and deceleration limits will be set to 300.0 user units/second^2 and jerk will be set to 1000.0 user units per second^3 (Jerk will be supported in a future release).

4. Call MC_CreateAxis (AxisName, BusInterface, BusAddress, AxisNumber, AxisType, UserUnits, FeedbackUnits, Rollover, UpdateRate) to create a Coordinated Motion Axis. This function needs to be called for each Coordinated Motion Axis wanted in the application.

   NOTE MC_CreateAxis needs to be called between MLMotionInit() and MLMotionStart().

   ```
   Inst_MC_CreateAxis(TRUE, 'CoordAxis1', 'Ether-
   CATDriver', 1001, CoordAxis1_AxisNum, 0, 360, 1048576,
   0, 6);
   Inst_MC_CreateAxis(TRUE, 'CoordAxis2', 'Ether-
   CATDriver', 1002, CoordAxis2_AxisNum, 0, 360, 1048576,
   0, 6);
   ```

   In the example above:

   - Two axes are created and are named 'CoordAxis1' and 'CoordAxis2'.
   - The bus interface for both is 'EtherCATDriver'.
   - The address of the drive on the bus is 1001 and 1002.
   - The axis numbers are set with variables CoordAxis1_AxisNum and CoordAxis2_AxisNum which is set to an integer value between 1 and 256. Each axis number is unique.
   - The axis type for both, '0', indicates a servo axis.
   - The user units are 360, which is the 'user unit' portion of the 'user unit/feedback' ratio.
   - The feedback units are 1048576, which is the 'feedback' portion of the 'user unit/feedback' ratio.

- The rollover position for both, '0' indicates no rollover.
- The update rate for both, '6', indicates a 1mSec update rate.

5. Call MLMotionStart () to start the Motion and the motion bus driver. This also initializes the EtherCAT network to operational mode.

```
MLMotionStart();
```

6. Call MC_AddAxisToGrp (Execute, AxesGroup, Axis, IdentInGroup) for each axis to be added to the group.

```
Inst_MC_AddAxisToGrp(TRUE, Group1_ref, CoordAxis1_ref,
0);
Inst_MC_AddAxisToGrp(TRUE, Group1_ref, CoordAxis2_ref,
1);
```

In the example above, we are adding two axes, CoordAxis1 and CoordAxis2, to the group referenced by 'Group1_ref'. The axes are stored in the IdentInGroup positions 0 and 1. Note that when the group was created, it was specified that no more than 2 axes will be part of this group. Therefore, valid IdentInGroup locations are 0 and 1.

7. Call MC_Power (Enable, Axis, EnablePositive, EnableNegative, BufferMode) for each Coordinated Motion Axis to enable the drive and close the servo loop.

```
Inst_MC_Power1(TRUE, CoordAxis1_ref, TRUE, TRUE, 0);
Inst_MC_Power2(TRUE, CoordAxis2_ref, TRUE, TRUE, 0);
```

In the example above, drives CoordAxis1_ref and CoordAxis2_ref will be enabled and the position loop will be closed. Note that parameters 'TRUE, TRUE, 0' are place holders for future use and are not currently used.

8. Call MC_GrpEnable (Execute, AxesGroup) to change the state of the Coordinated Motion Axis Group from GroupDisabled to GroupStandby and allow motion to be performed on the group.

```
Inst_MC_GrpEnable(TRUE, Group1_ref);
```

In the example above, 'Group1_ref' state will be changed from GroupDisabled to GroupStandby. The group must be in GroupStandby in order to perform motion.

9. For the examples that follow, we want to set the current location of the axes in the group to 0, 0. This can be done by calling MC_GrpSetPos (Execute, AxesGroup, Position[], Relative, CoordSystem, BufferMode)

```
PosAbs[1]:= 0;
PosAbs[2]:= 0;
Inst_MC_GrpSetPos(TRUE, Group1_ref, PosAbs, 0, MC_
COORDINATE_SYSTEM_ACS, 0);
```

In the example above, the axis positions of 'Group1_ref' will be set to 0, 0. 'PosAbs' specifies the position for each axis in the group. 'Relative' input, '0', uses 'PosAbs' to set the absolute position. The coordinate system is set to ACS . The buffer mode, '0', is a placeholder for future use and is not currently used.

NOTE No motion will be performed when this function block is executed.

After the above function calls have been made, we can start coordinated motion moves.

"Performing a Linear Move" (see page 465)

"Performing a Circular Move" (see page 467)

What are Axes Groups?

Related axes are grouped in an AxesGroup to support interpolation. AxesGroups are accessed via the type AXES_GROUP_REF. The following image shows the relationships between the different CSs and groups.

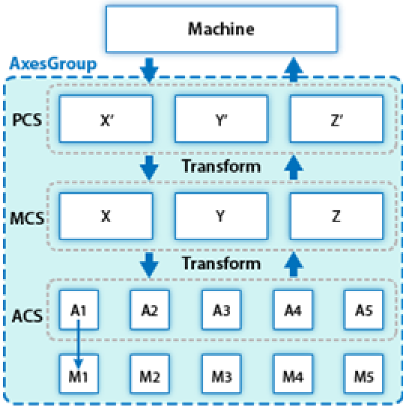


Figure 4-76: Overview of AxesGroup

The AxesGroup, shown in blue above, provides the interface to the user of the group of axes. To access the relevant coordinate system, the relevant function blocks have an input CoordSystem which supports the three levels ACS, MCS, and PCS.

Parameters in the AxesGroupRef can include remaining time and remaining distance before target position (or velocity or equal) is reached.

Performing a Linear Move

Linear moves can be programmed using absolute or relative positions using the following function blocks:

- MC_MoveLinAbs which commands interpolated linear movement on an axes group to the specified absolute positions.
- MC_MoveLinRel which commands interpolated linear movement on an axes group to the specified relative positions.

Prior to performing any coordinated moves, some setup is needed (see "Create a Linear or Circular Coordinated Motion Application" on page 462). Once these steps have been performed, a linear move can be performed.

In the following examples, two linear moves will be performed. The first move is an absolute linear move that goes from (0, 0) to (100, 200). The second move is a relative linear move that goes a distance of (-75, 50) from the end of the first move.The BufferMode input is set to 'Buffered', meaning this move will wait for the first move to complete before it begins executing.

- **To Perform an Absolute Linear Move**

  Call MC_MoveLinAbs (Execute, AxesGroup, PositionArray, Velocity, Acceleration, Deceleration, Jerk, CoordSystem, BufferMode, TransitionMode, TransitionParameter). PositionArray is an array of absolute end positions containing one position for each axis in the group. The inputs velocity, acceleration, deceleration, and jerk establish the maximum values for the move.

In this example, PosArrayAbs[0] represent the x-axis and PosArrayAbs[1] represent the y-axis.

```
PosArrayAbs[0] := 100;
PosArrayAbs[1] := 200;
TransParam[0]  :=   0;
TransParam[1]  :=   0;

Inst_MC_MoveLinRel(TRUE, Group1_ref, PosArrayAbs,
MaxVel, MaxAcc, MaxDec, 0, MC_COORDINATE_SYSTEM_ACS, 1,
0, TransParam);
```

In the example a linear move will be performed on axis group 'Group1_ref'.

- PosArrayAbs contains the absolute end points of the axes in the group. The axis stored in position 0 (IdentInGroup) of the group will be moved to 100.0. The axis stored in postiion 1 of the group will be moved to 200.0.
- The maximum velocity is specified by variable MaxVel and is specified in 'user units/sec'.
- The maximum acceleration and deceleration are specified by variables MaxAcc and MaxDec and are specified in 'user units/sec2'.
- The maximum jerk is currently not supported and can be set to a value of 0.
- The coordinate system is ACS
- The BufferMode is set to 1, indicating the move is buffered. For more information about buffer modes, see the "Buffer Modes" (see page 136) overview.
- The TransitionMode is set to 0, indicating no transition mode will be used. For more information about transition modes, see the "Transition Between Moves" (see page 477) section.
- The TransParam array is required and the contents can be set to 0 since the transition mode is not being used. There has to be one array entry for each axis in the group.

- **To Perform a Relative Linear Move**

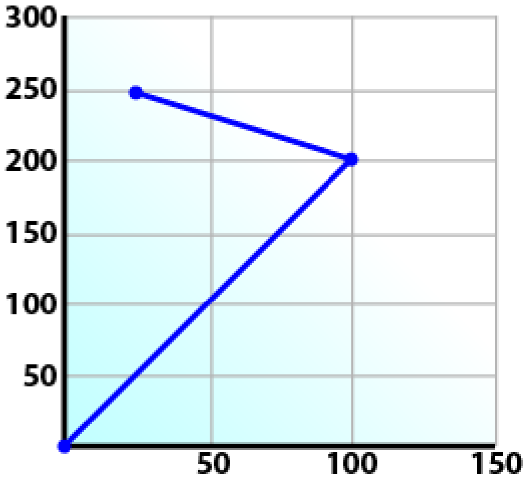
Call MC_MoveLinRel (Execute, AxesGroup, Distance, Velocity, Acceleration, Deceleration, Jerk, CoordSystem, BufferMode, TransitionMode, TransitionParameter). The Distance input is an array of distances, one distance for each axis in the group. The inputs velocity, acceleration, deceleration, and jerk establish the maximum values for the move.

In this example, DistArrayRel[0] represent the x-axis and DistArrayRel[1] represent the y-axis.

```
DistArrayRel[0] := -75.0;   // Start pt 100 - rel 75 ->
 25 absolute end pt
DistArrayRel[1] :=  50.0;   // Start pt 200 + rel 50 ->
250 absolute end pt
TransParam[0]   :=   0;
TransParam[1]   :=   0;

Inst_MC_MoveLinRel(TRUE, Group1_ref, DistArrayRel,
MaxVel, MaxAcc, MaxDec, 0, MC_COORDINATE_SYSTEM_ACS, 1,
0, TransParam);
```

In the example above, all the variables have the same meaning as the absolute linear example except DistArrayRel. DistArrayRel contains the relative distance to move for each axis in the group. The axis stored in position 0

(IdentInGroup) of the group will be moved a distance of -75.0. The axis stored in postiion 1 of the group will be moved a distance of 50.0.



Performing a Circular Move

Circular moves can be programmed using absolute or relative positions using the following function blocks:

- MC_MoveCircAbs which commands interpolated circular movement on an axes group to the specified absolute positions.
- MC_MoveCircRel which commands interpolated circular movement on an axes group to the specified relative positions.

Prior to performing any coordinated moves, some setup is needed (see "Create a Linear or Circular Coordinated Motion Application" on page 462). Once these steps have been performed, a circular move can be performed.

In the following examples, two circular moves will be performed. The first move is an absolute circular move that goes from (0, 0) to (90, 90). CircMode specifies that the aux point (0, 180) will be crossed during the paths start to end. The second move is a relative circular move whose end point is (90, 90) from the end of the first move. In this move, CircMode specifies that the aux point (0, 90) is the relative center of the circle. The BufferMode input is set to 'Buffered', meaning this move will wait for the first move to complete before it begins executing.

- **To perform an Absolute Circular Move:**

  Call MC_MoveCircAbs (Execute, AxesGroup, CircMode, AuxPoint[], EndPoint[], PathChoice, Velocity, Acceleration, Deceleration, Jerk, CoordSystem, BufferMode, TransitionMode, TransitionParameter).

  In this example, PosCircAuxAbs[0] and PosCircEndAbs[0] represent the x-axis. PosCircAuxAbs[1] and PosCircEndAbs[1] represent the y-axis.

```
PosCircAuxAbs[0] :=   0;   // A point on the circle
that is crossed on the
PosCircAuxAbs[1] := 180;   // path from start to end
point.
PosCircEndAbs[0] :=  90;   // Absolute end point.
PosCircEndAbs[1] :=  90;

Inst_MC_MoveCircAbs(TRUE, Group1_ref, MC_CIRC_MODE_
BORDER, PosCircAuxAbs, PosCircEndAbs, MC_CIRC_
PATHCHOICE_CLOCKWISE, MaxVel, MaxAcc, MaxDec, 0, MC_
```

```
COORDINATE_SYSTEM_ACS, MC_BUFFER_MODE_BUFFERED, MC_
TRANSITION_MODE_NONE, TransParam);
```

In the example a circular move will be performed on axis group 'Group1_ref'.

- CircMode is defined as MC_CIRC_MODE_BORDER. This mode indicates that the AuxPoint array input will indicate a point on the circle which is crossed on the path from the starting point to the end point. See "Circular Moves Diagrams" (see page 469) for more information on CircMode movement options.
- The AuxPoint array, 'PosCircAuxAbs', defines an absolute point on the circle which is crossed on the path from the starting point to the end point. The contents of this array are determined by the CircMode variable, MC_CIRC_MODE_BORDER.
- The EndPoint array, 'PosCircEndAbs', contains the absolute end point for each axis in the group. The absolute end point of the axis stored in position 0 (IdentInGroup) of the group will be 90.0. The absolute end point of the axis stored in position 1 of the group will be 90.0.
- PathChoice is only relevant when CircMode is set to MC_CIRC_MODE_CENTER. In this case, this parameter is not used.
- The maximum velocity is specified by variable MaxVel and is specified in 'user units/sec'.
- The maximum acceleration and deceleration are specified by variables MaxAcc and MaxDec and are specified in 'user units/sec^2'.
- The maximum jerk is currently not supported and can be set to a value of 0.
- The coordinate system is ACS
- The BufferMode is set to MC_BUFFER_MODE_BUFFERED, indicating the move is buffered. For more information about buffer modes, see the "Buffer Modes" (see page 136) overview.
- The TransitionMode is set to MC_TRANSITION_MODE_NONE, indicating no transition mode will be used. For more information about transition modes, see the "Transition Between Moves" (see page 477) section.
- The TransParam array is required. The TransParam array is a 2-element array containing the corner distance and velocity for the transition. Transitions are not used in this example and therefore the contents can be set to 0.

- **To perform a Relative Circular Move:**

Call MC_MoveCircRel (Execute, AxesGroup, CircMode, AuxPoint[], EndPoint[], PathChoice, Velocity, Acceleration, Deceleration, Jerk, CoordSystem, BufferMode, TransitionMode, TransitionParameter).

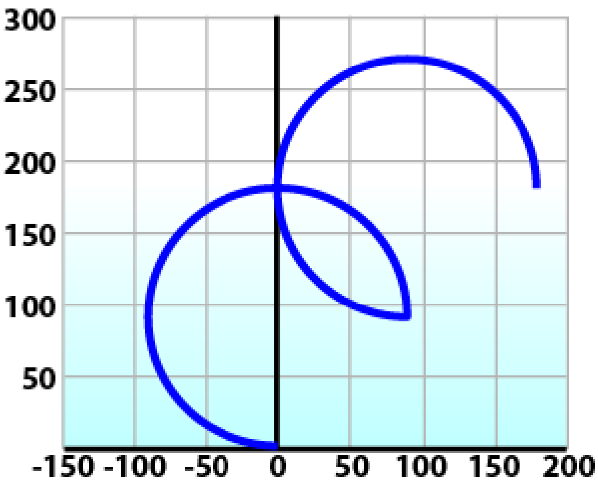
In this example, PosCircAuxRel[0] and PosCircEndRel[0] represent the x-axis. PosCircAuxRel[1] and PosCircEndRel[1] represent the y-axis.

```
PosCircAuxRel[0] :=  0;   // Relative center of the
circle.
PosCircAuxRel[1] := 90;
PosCircEndRel[0] := 90;   // Relative end point.
PosCircEndRel[1] := 90;   // Start pt 90,90 + rel 90,90
-> 180,180 absolute end pt

Inst_MC_MoveCircRel(TRUE, Group1_ref, MC_CIRC_MODE_
CENTER, PosCircAuxRel, PosCircEndRel, MC_CIRC_
PATHCHOICE_CLOCKWISE, MaxVel, MaxAcc, MaxDec, 0, MC_
```

```
RDINATE_SYSTEM_ACS, MC_BUFFER_MODE_BUFFERED, MC_
NSITION_MODE_NONE, TransParam);
```

In the example all the variables have the same meaning as the circular absolute example except:

- CircMode is defined as MC_CIRC_MODE_CENTER. This mode indicates that the AuxPoint array input will indicate the center point of the circle. See "Circular Moves Diagrams" (see page 469) for more information on CircMode movement options.
- The AuxPoint array, 'PosCircAuxRel', defines the relative center point of the circle. The contents of this array are determined by the CircMode variable, MC_CIRC_MODE_CENTER.
- The EndPoint array, 'PosCircEndRel', contains the relative end point for each axis in the group. The relative end point of the axis stored in position 0 (IdentInGroup) of the group will be 90.0. The relative end point of the axis stored in postiion 1 of the group will be 90.0.
- PathChoice is relevant when CircMode is set to MC_CIRC_MODE_CENTER. In this case, PathChoice is MC_CIRC_PATHCHOICE_CLOCKWISE which specifies the direction of the path.

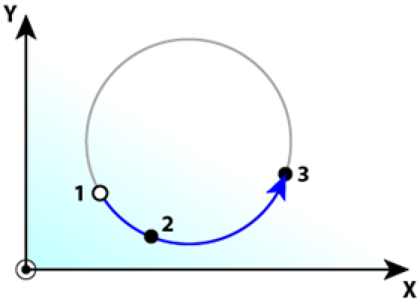


5.4.13 Circular Moves Diagrams

5.4.13.1 CircMode = BORDER

The user defines the end point and a border point (= input 'AuxPoint') on the sector of the circle which the machine will traverse. For Relative mode, both points are defined relative to the starting point.

Advantages	• The border point can usually be reached by the machine, i.e. it can be taught.
Disadvantages	• Restricted to angles < 360° in one single command.

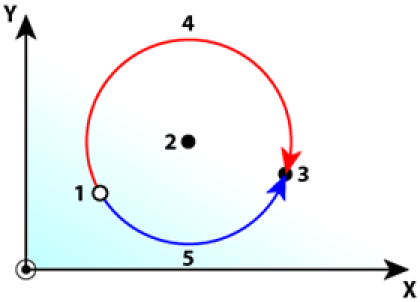


1. Starting point
2. Border point
3. End point

5.4.13.2 CircMode = CENTER

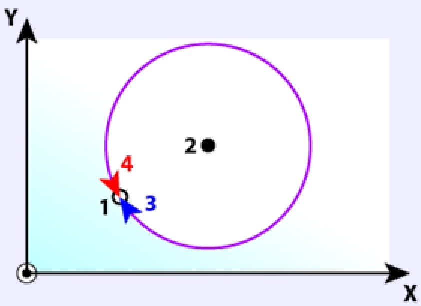
The user defines the end point and center point (= input 'AuxPoint') of the circle. The input 'PathChoice' defines clockwise or counter-clockwise motion. For Relative mode, both points are defined relative to the starting point.

Advantages	• Full 360° moves are possible.
Disadvantages	• Cannot perform zero-distance moves. • Over-determination of the circle equation.



1. Starting point
2. Center point
3. End point
4. Clockwise move
5. Counter-clockwise move

NOTE A 360° move will be performed if the end point is the same as the start point.



1. Starting & Ending point
2. Center point
3. Clockwise move
4. Counter-clockwise move

5.4.14 Step 13 of 15 - Adding Cam Profiles

5.4.14.1 Create Cam Profiles

To create a cam profile, do as follows:

1. In the Project Explorer, right-click the **Profiles** item and select the **New profile** command in the contextual menu

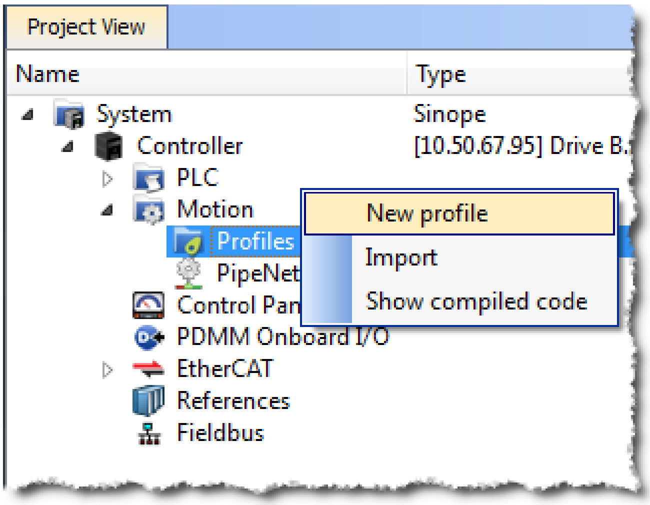


Figure 4-77: Cam - New Profile

2. A new profile named "Profile" is created with default parameters.
3. Right-click on the new profile to rename it or change its properties.

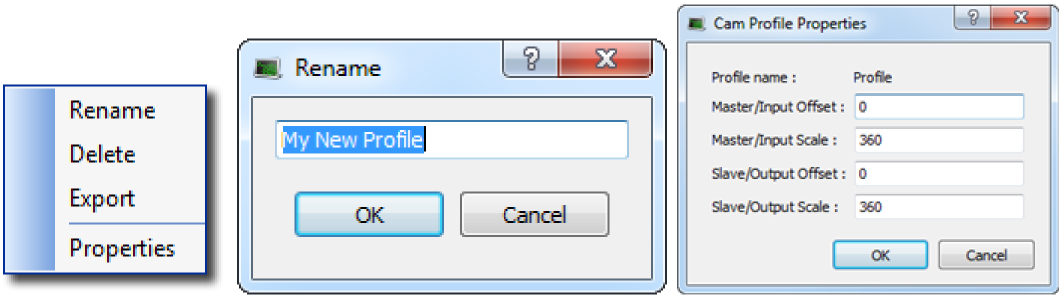


Figure 4-78: Cam - Define Profile Filename

4. Click on the new profile to edit it.

Field	Description
Profile name	The name of the Profile which is: • displayed in the Project Explorer • used in the Properties of the cam Pipe Block

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td>

Master/Input Offset

Master/Input Scale

Slave/Output Offset

Slave/Output Scale

</td><td>

Offsets and scales on the X and Y axes transform the normalized profile into the output profile as shown on the two figures below:

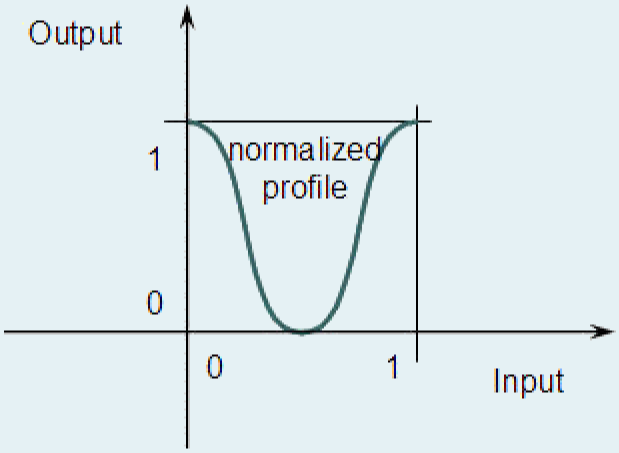


Figure 4-79: Cam - Normalized Profile

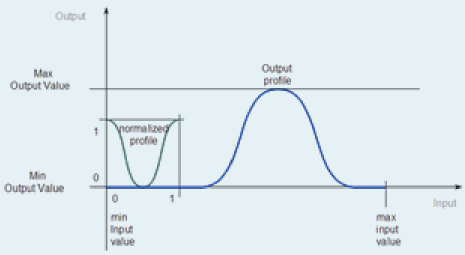


Figure 4-80: Cam - Output Profile

For more details, refer to the paragraph below: Four Parameters Transforming the Cam Profile

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

Table 4-9: Cam Profile Parameters

Four Parameters Transforming the Cam Profile

Master/Input offset

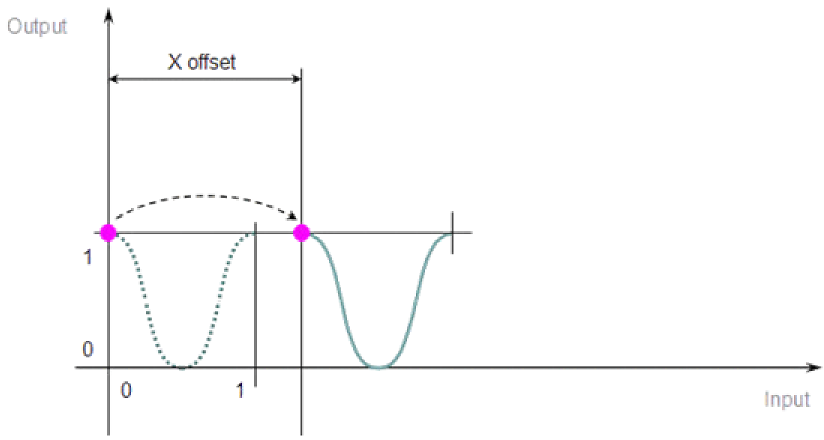


Figure 4-81: Cam Profile Transformation - Step 1

Master/Input scale

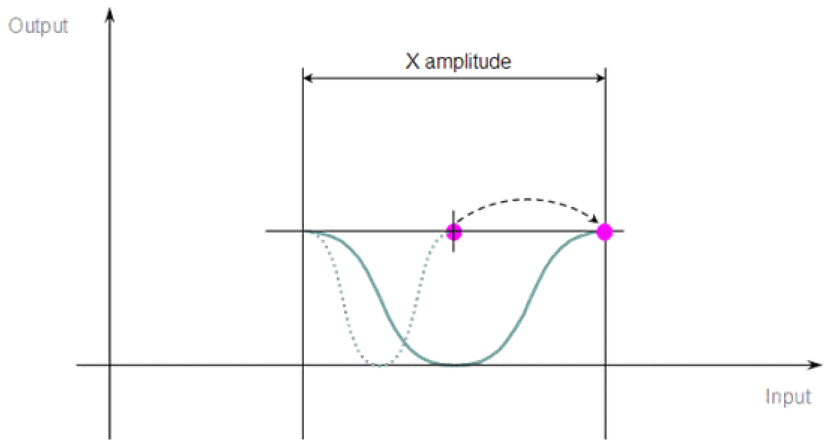


Figure 4-82: Cam Profile Transformation - Step 2

Slave/Output scale

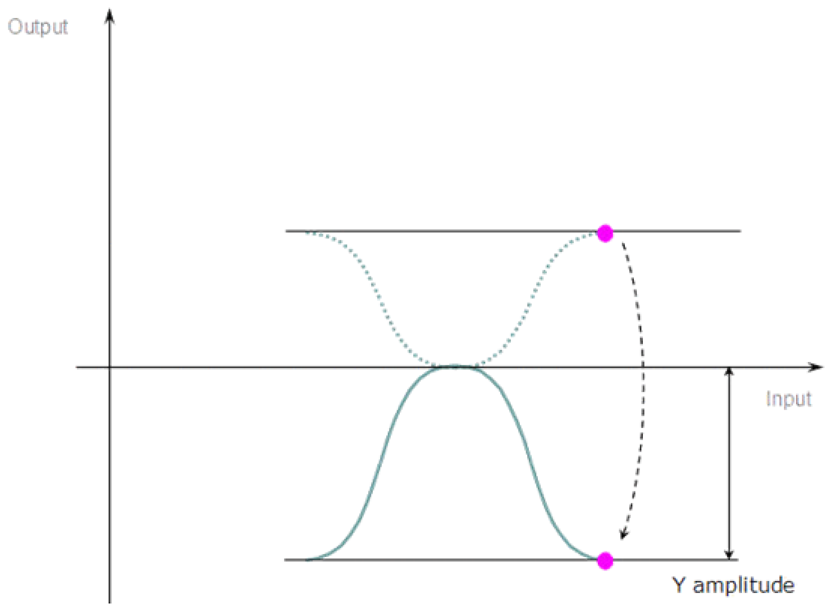


Figure 4-83: Cam Profile Transformation - Step 3

Slave/Output offset

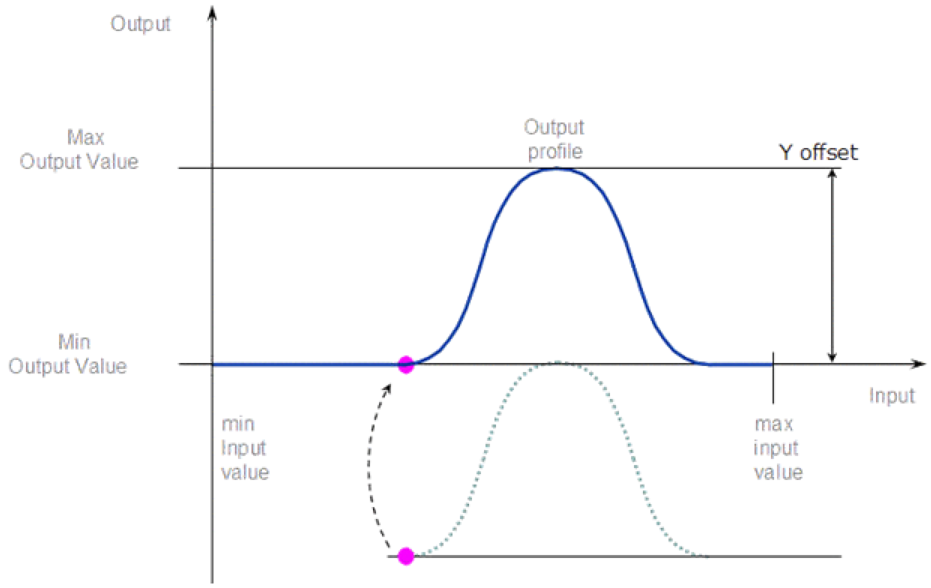


Figure 4-84: Cam Profile Transformation - Step 4

> **NOTE** When you change a Cam Profile property, a dialog box indicates the progression of the operation.

For more details about editing the profile, refer to "Cam Profile Editor" (see page 396).

5.4.14.2 Use Cam Profiles

Once defined, you can associate the cam profile to a cam Pipe Block in the Pipe Network as follows:

1. Right-click on the cam Pipe Block and select **Properties** in the menu
2. In the **Parameters** tab, enter the profile's name

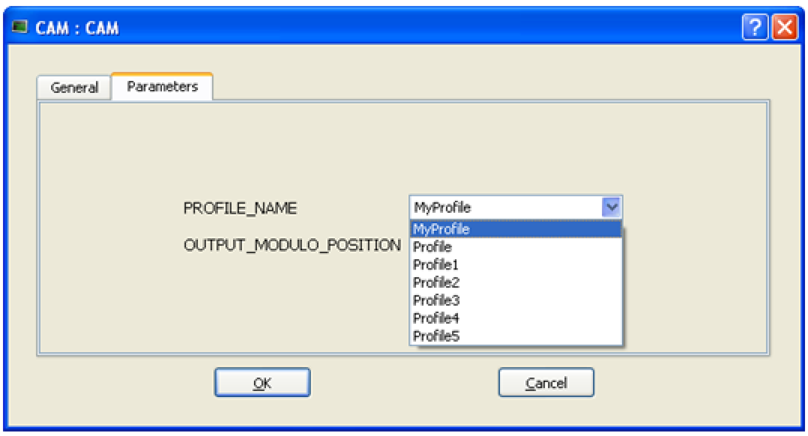


Figure 4-85: Cam - Associate Profile to a Pipeblock

> **NOTE** Separating the declaration of the cam Pipe Block from the cam profile provides the capability to prepare several different cam profiles and then apply one of them to the cam Pipe Block.

> **NOTE** If you change the profile's filename, do not forget to update the cam Pipe Block accordingly.

5.4.15 Step 14 of 15 - Define Scheduling

5.4.15.1 Periodicity

The period of execution of a pipe is the time spent between two successive computations of set values for the same pipe. The period of execution of a pipe is specified by the PERIOD parameter of the input Pipe Block.

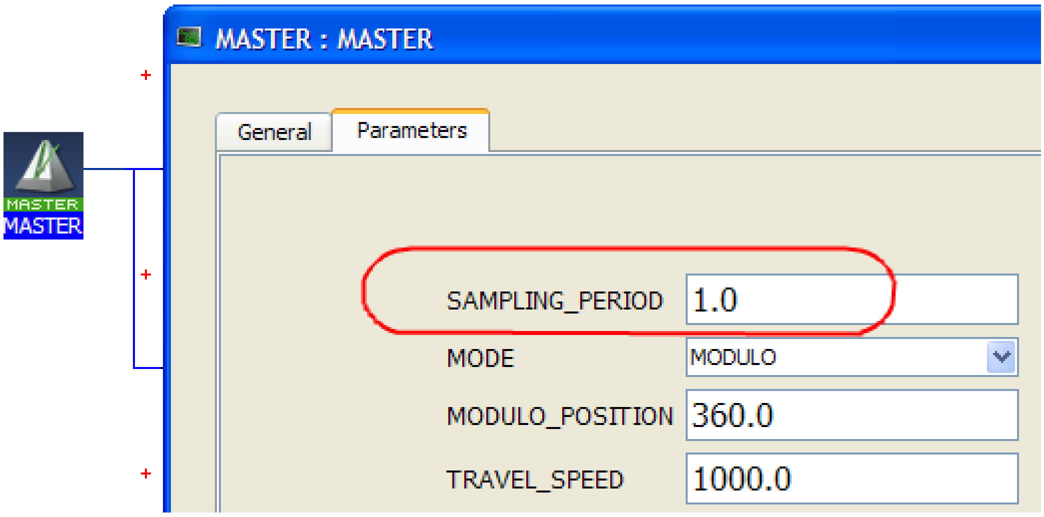
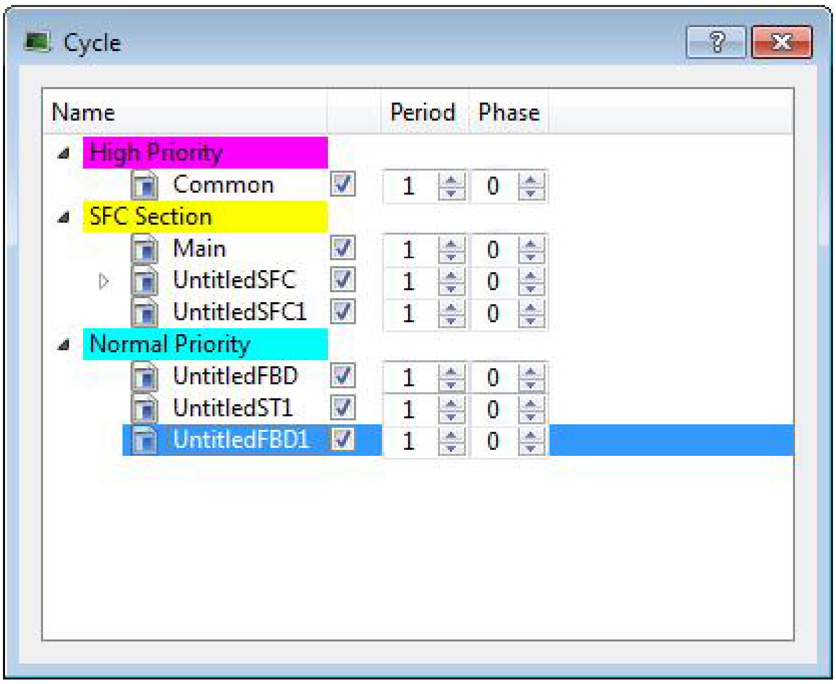


Figure 4-86: Set the Period of Execution

All the pipe values are computed independently of events and sequences execution.

5.4.15.2 Order of Execution

The order of execution of programs within a project is determined from the Cycle window.



The order of execution is always

1. High Priority
2. SFC Section
3. Normal Priority

The order within each grouping is determined by the vertical ordering. Using the image above, the order of execution for Normal Priority programs is:

1. UntitledFBD
2. UntitledST1
3. UntitledFBD1

For more information see "Define the PLC Cycle" (see page 311).

5.4.15.3 Define the PLC Cycle

The cycle specification defines the number of cycles between successive executions of the programs.

1. In the Project Explorer, expand the PLC node and right-click on the Programs item to open the contextual menu and select the **Cycle** command

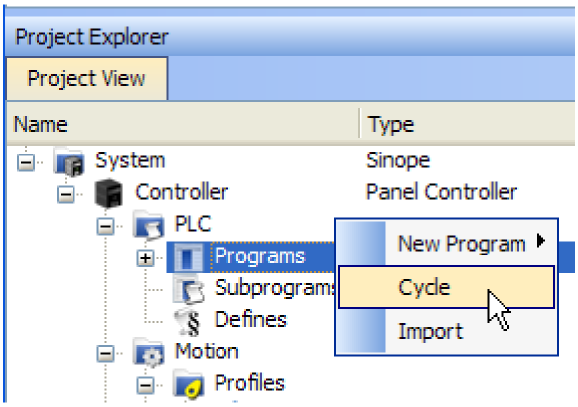


Figure 4-87: Edit the Cycle

The Cycle window allows the regulation of the following parameters: Period and Phase.

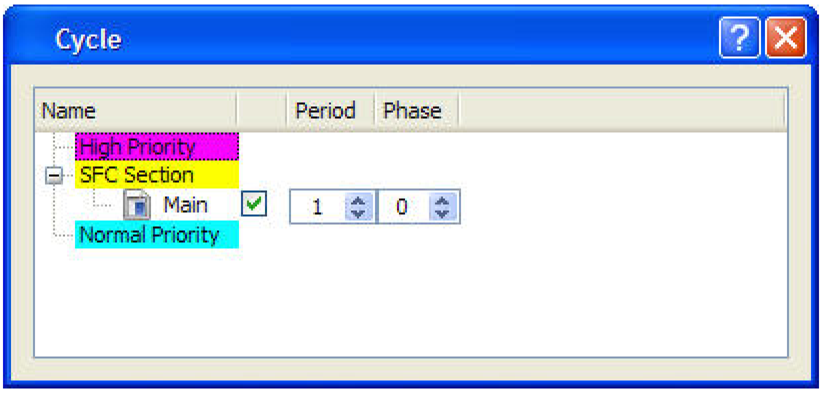


Figure 4-88: Define the Cycle

The cycle configuration dialog box is used to configure the programs priority into the Virtual Machine.

Column	Description
Name	List of PLC programs grouped together by priority level. The priority defines the order of execution. The SFC programs have a specific section as they must be grouped together. *High Priority* means "executed before SFC", and *Low Priority* means "executed after SFC".
Check box	Enables or disables the execution of the corresponding program.
Period	Defines how many cycles are set between two executions of the program. You can define various sampling periods for programs of the application. Default period is "1" (the program is executed on each cycle). Giving a slower period to some programs is an easy way to give higher priority to some other programs.
Phase	Defines an offset that enables you to dispatch slow programs among few cycles. The goal of postponing the program execution is to reduce execution peak loads. Example: a program with period=2 and Phase=1 is executed each **even** cycle a program with period=2 and Phase=0 is executed each **odd** cycle

Table 4-10: Cycle Parameters

In the **High** and **Normal** Priority sections, you can adjust the order of the programs with a drag-and-drop operation according to the expected sequence. In each section, the program on the top is executed first.

Select the program you want to set with a higher priority, then drag and drop it to the relevant priority level.

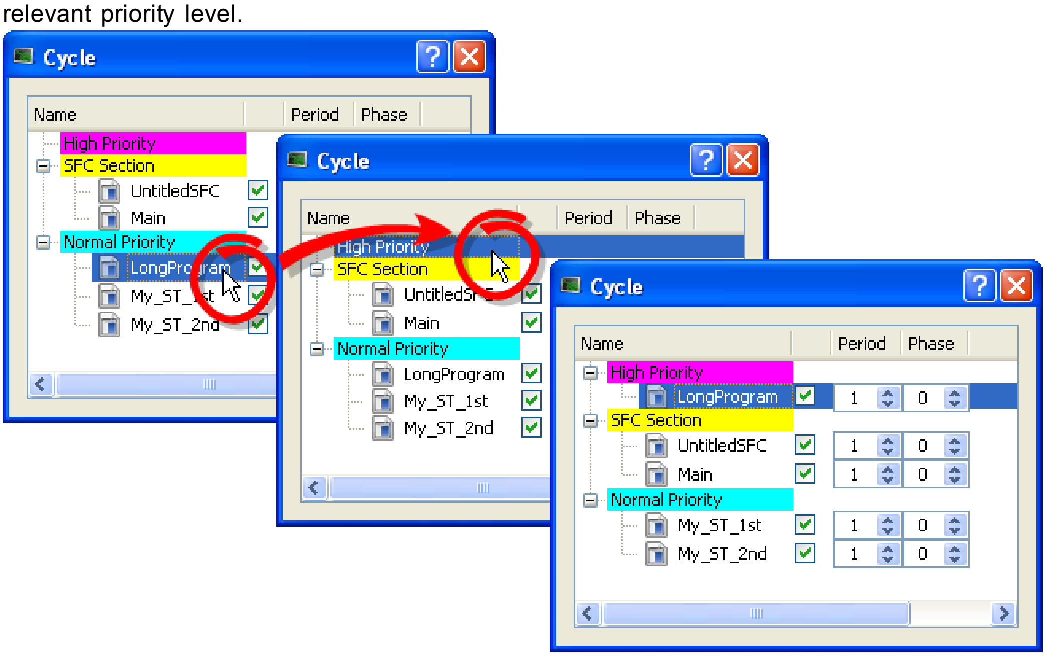


Figure 4-89: Change Priorities by Defining the Cycle

If all programs are with a Period set to 1, the KAS IDE is more loaded. The choice of the Period for the programs gives you the possibility to distribute the load of the application.

See also "Tasking Model / Scheduling" on page 177 and "Order of Execution" (see page 310).

How to specify the duration of a cycle

This parameter is defined in "EtherCAT Master Settings" (see page 214).

Ensuring Variables are Exported

Program Organization Units (POUs) which contain variables (see "Map Variables to HMI" on page 316) must be compiled in order for the variable to be exported. For example, in the following set of images we see a POU (*UntitledST*) with two variables, *NewVar* and *NewVar1* and only NewVar1 is set to be exported (1). The POU, however, is not set to be executed in the Cycle dialog box (2). This will cause a compile error (3).

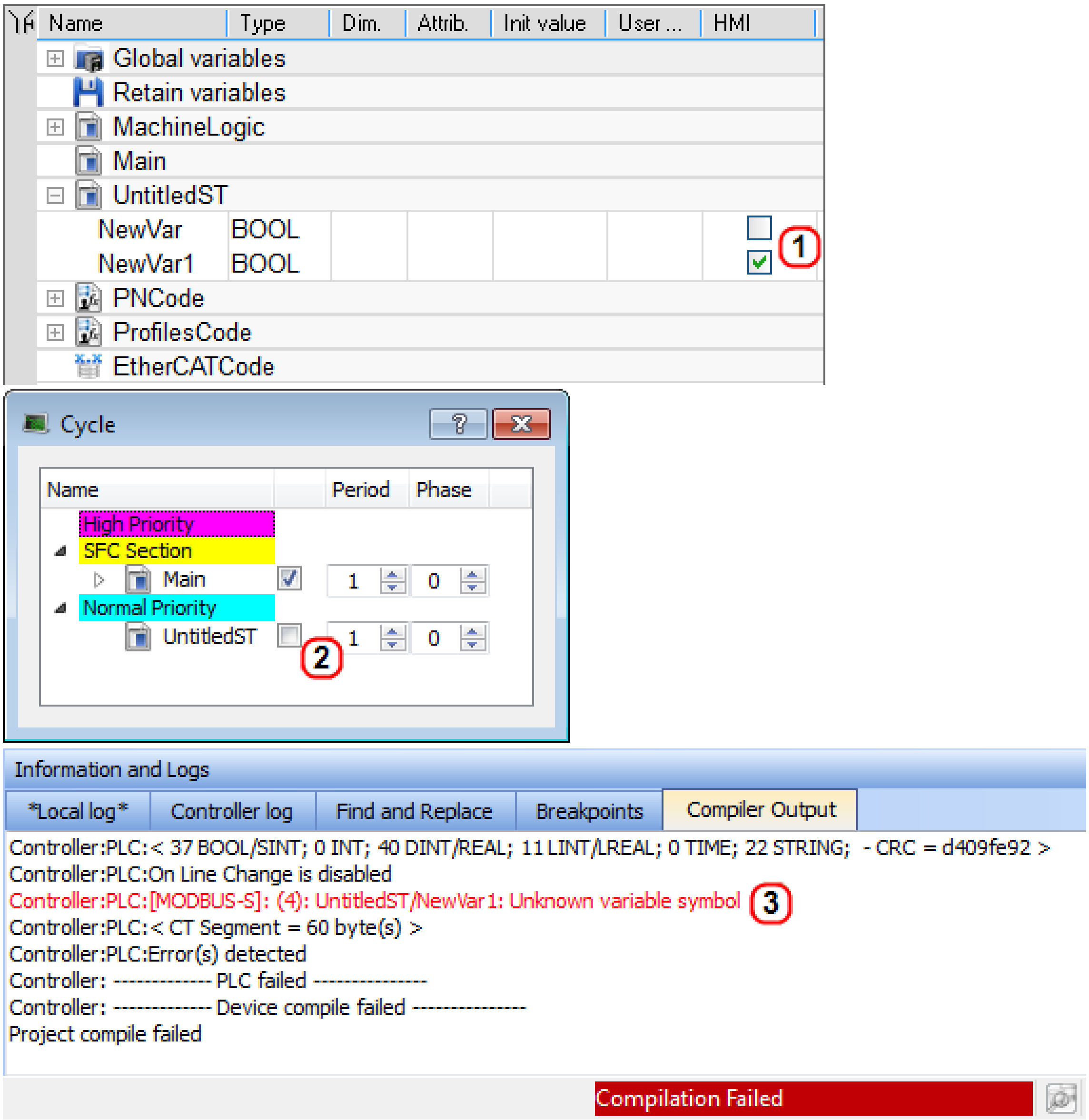


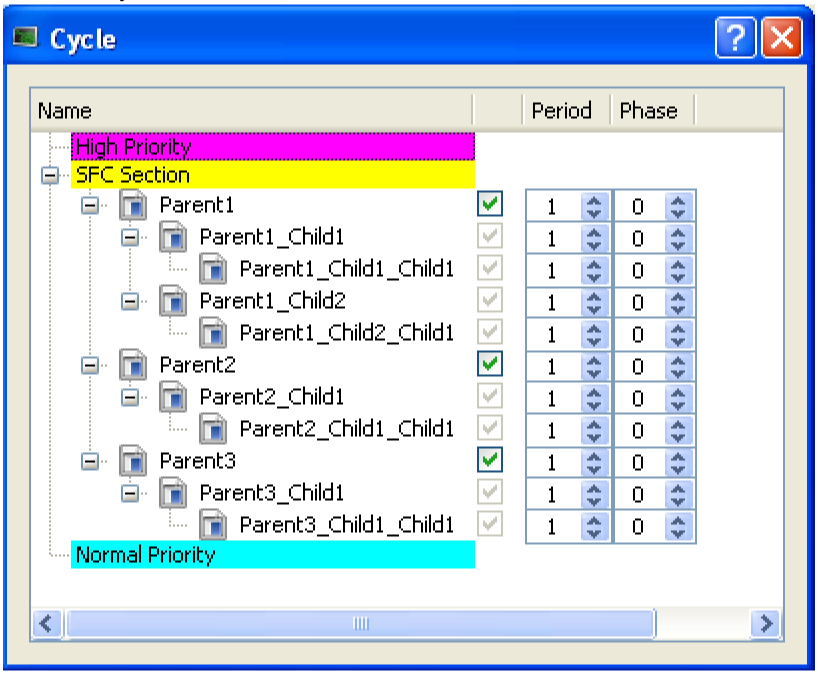
Figure 4-90: Example of a variable not being exported and the resulting compile error.

5.4.15.4 About Parent-Child relationships and execution order

The SFC Section allows for editing the period and phase values of parent and child programs. Parent and child programs follow certain rules:

- A parent program can be enabled or disabled. If a parent is disabled the child will also be disabled. A child program cannot be disabled.
- Parent programs are allowed to move across priorities. Child programs will follow the movement of a parent. Child programs are not allowed to move independently.
- When a child program is created or imported, it will inherit the enabled/disabled state of the parent program.
- The SFCs are executed at the set cycle period and phase. All parent programs will be executed first and then the children programs will be executed in order.

To understand the last rule, consider the following Cycle example. There are three parent programs, each with a number of child programs. All parents are executed, followed by the children, in order. The actual flow is illustrated below the image.




```
Parent1
 Parent2
  Parent3
   Parent1_Child1
    Parent1_Child1_Child1
     Parent1_Child2
      Parent1_Child2_Child1
       Parent2_Child1
        Parent2_Child1_Child1
         Parent3_Child1
          Parent3_Child1_Child1
```

IMPORTANT *Parent SFCs should run faster than their children.* If this is not the case, the

IMPORTANT stop condition can be vague. When a child runs slower than its parent it does not stop when the parent stops, but at the child's next execution. This means the parent could execute more, while the child is still running.

TIP A child program is initiated at Phase 0 in respect to its parent.

5.4.16 Step 15 of 15 - Add an HMI Device

To control your application, HMI panels can be downloaded to a dedicated HMI device (as described in the following procedure), but it can also be embedded into a targeted controller.

When running the KAS Simulator, an internal HMI editor is also available to debug your application (for more details, see page 439)

5.4.16.1 Create KVB Project

KVB projects are managed in the Project Explorer and can be created as follows:

1. In the Project Explorer, right-click on the **System** item to open the contextual menu
2. Select the **Add HMI device** command
3. Select the device name within the list and Click OK

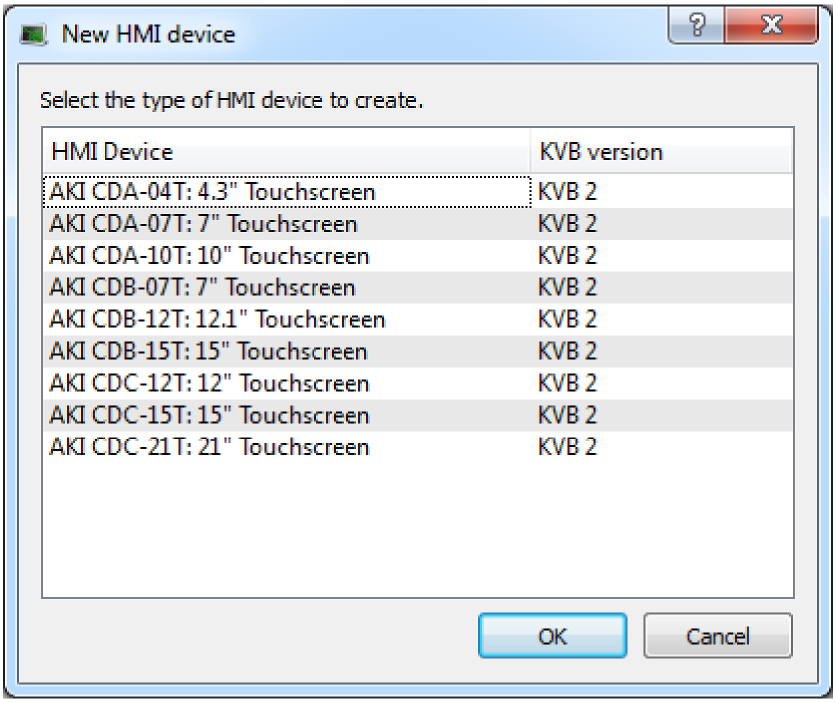


Figure 4-91: Select an AKI to add.

TIP The content will vary depending upon what version of KVB is installed on your system. If you have KVB 1.1 installed you may open projects that have a KVB 2.1 panel, but you will not be able to open the panel in KVB.

4. Right-click on the newly created item and select the **Rename** command to change its name
5. Right-click and select the **Add KVB Project** command

NOTE Note that this command is disabled when a KVB panel is already created for the current HMI device

5.4.16.2 Map Variables to HMI

For HMI, the variable mapping is done in two phases.

- Phase 1 - You first have to tag all the variables that you want to be exported in your HMI project (see procedure below)
- Phase 2 - Then you can use this mapping file when designing the HMI

The tag operation directly takes place in the Dictionary, as follows:

1. Open the Variable list editor available in the **Dictionary** toolbox

   TIP Double-click the Dictionary header to display the widget as a popup window in order to have more space.

2. Develop the nodes to display the list of variables
3. In the HMI column, select the variables you want to map

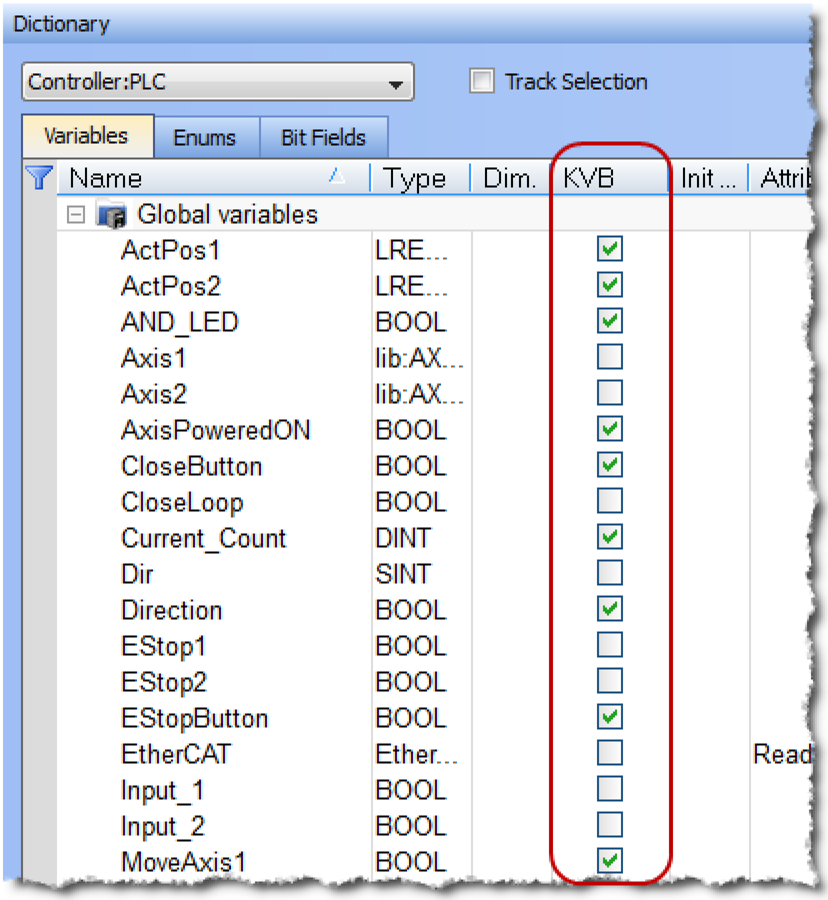


Figure 4-92: Variable Mapping to HMI

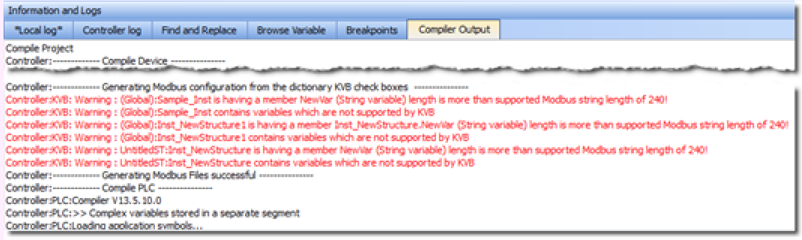
TIP Selecting the variable alone does not guarantee it will be exported. The POU must be set to compile as well. See "Ensuring Variables are Exported" (see page 313) for more information.

IMPORTANT Being based on Modbus, the communication is limited to 32 bits. As a consequence:

- Data type conversion can lead to a loss in accuracy:
  - LREAL variables are saved as REAL
  - LINT variables are converted to DINT
  - ULINT variables are saved as UDINT

IMPORTANT

- LWORD data types cannot be used within the HMI when using KVB. This is because variables of this type are not exported, even when selected.
- STRING variable data types are allowed in the HMI when using KVB 2.0 or later (they are not supported in KVB 1.2). String variables allow up to 240 characters.
- Variables of types "PNCode", "ProfilesCode", or instances of UDFB cannot be used within the HMI.
- String variables whose size exceed the maximum string length of 240 characters generate a warning message in the Compiler Output window when being exported over Modbus. Only the first 240 characters will be exported. See "Compiler Output" (see page 639) for more information.



TIP

Non-primitive data types which are not exported over Modbus generate a warning message in the Compiler Output window. See "Compiler Output" (see page 639) for more information.

4. Compile the application to create the Modbus mapping file

TIP

This text file (named **HMI Variable Import File.txt**) can be located in the folder “C:\Documents and Settings\(user)\Local Settings\Application Data\Kollmorgen\KAS\Project” where “(user)” is the Windows' username you are currently logged in with. This file is used for manually importing variables into KVB.

NOTE

- If you modify the set of tagged variables in the dictionary, you have to update the text file by recompiling the project.
- The Modbus variables defined in KAS IDE are imported in Kollmorgen Visualization Builder only when you start KVB (there is no update in real-time between the two applications).

5. Then you can use this mapping file in your HMI project.

List of variables that you can export

The following types of variables can be exported to the HMI:

- The fundamental data types: BOOL, SINT, INT, USINT, UINT, BYTE, WORD, DINT, UDINT, DWORD, TIME, REAL, LREAL, STRING.
- Arrays of supported data types
- Structures that include members of supported data types

Examples of structures that you can export

- Arrays of BOOLs such as a structure that includes a BOOL array member and instances of UDFB members.
- An array of structures that include INT and LREAL members

- A structure that includes both instances of UDFB member as well as an embedded structure which includes an INT member and instances of a UDFB member. This is due to the embedded structure has an INT member and therefore the outer structure can be exported too.

Examples of structures that you cannot export

- A structure that includes instances of UDFB or LWORD members only.

5.4.16.3 Design KVB Panel with Kollmorgen Visualization Builder

1. Double-click the new KVB panel to open the builder (for more details, refer to "Using Kollmorgen Visualization Builder" (see page 431).)

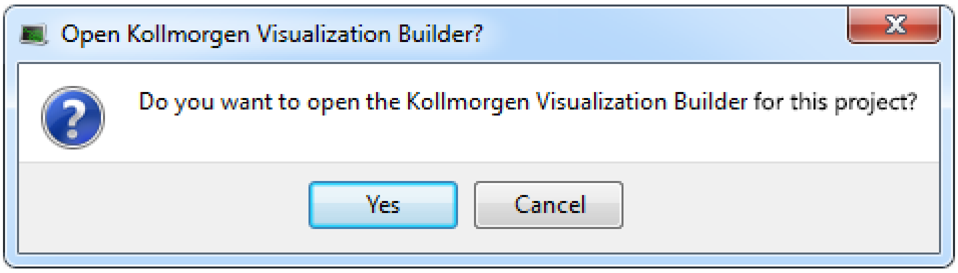


Figure 4-93: Open the HMI Builder

IMPORTANT	You must have the specific application already installed on your machine.
TIP	Be sure to close the Kollmorgen Visualization Builder before deleting the KVB Panel from the IDE.

5.5 Running the Project

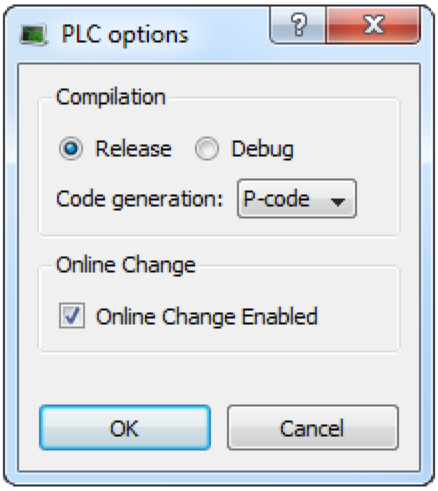
This chapter explains how to build, download and run your project.

Step	Description
1	Set the compilation options to run your project in Debug or Release modes, and choose if you want to activate the Online Changes
2	Compile the application and see all the remaining warnings and errors
3	Connect the KAS IDE to the target device
4	Download the Application compiled on the KAS IDE to the target device
5	Start / stop the device, and control your application with the script commands

NOTE Before step 3, you need to start the KAS software (KAS Simulator or KAS Runtime) on the target device where you want to run your project.

5.5.1 Step 1 of 6 - Set the Compilation Options

You can open the PLC compilation options, as shown below, with the icon.



If you want step-by-step debugging to be available during simulation or online testing, you need to select the "**Debug**" compiling mode. If step-by-step debugging is no longer required, select the "**Release**" compiling mode in order to give highest performance to your application.

When you incorporate additional statements (such as trace outputs) in your code, you must select the "**Debug**" compiling mode so that they are taken into consideration (in RELEASE mode, those statements are not included).

- For Conditional Compiling, see page 319
- For Online Changes, see page 491

Code generation

Applications created with the KAS IDE are first compiled to machine code (P-code) before being downloaded to the target PAC or Simulator.

Select P-code if your runtime system works with a specific P-code instruction set.

Why select P-code?

Size constraints. Since P-code is based on an ideal virtual machine, most of the time the resulting P-code is much smaller than the same program translated to machine code.

For debug purposes. Since P-code is interpreted (which means that the code is read by the KAS Runtime engine that then determines the instructions to run), the interpreter can apply many additional runtime checks that would be harder to implement with native code. .

5.5.1.1 Conditional Compiling

The compiler supports conditional compiling directives in ST, IL, FFLD, and FBD languages. Conditional compiling directives condition the inclusion of a part of the program in the generated code based on pragma. Conditional compiling is an easy way to manage several various machine configurations and options in one unique application project.

Conditional compiling uses definitions as conditions. Below is the main syntax:

```
#ifdef CONDITION
    statementsYES...
#else
    statementsNO...
#endif
```

If CONDITION has been defined using `#define` syntax, then the "*statementsYES*" part is included in the code, else the "*statementsNO*" part is included. The "`#else`" statement is optional.

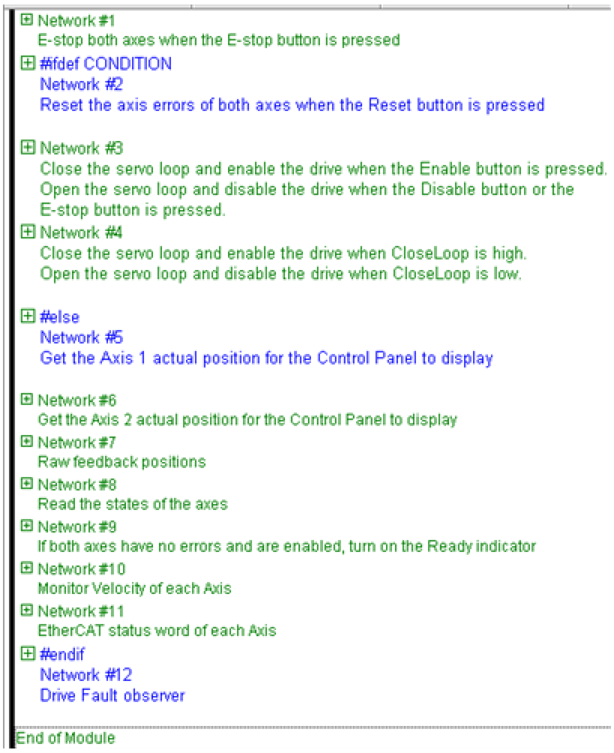
TIP Intellisense facilitates the reading by coloring in gray the part of the program which is not active.

How to define conditional compiling directives?

<table>
<tr><th>Languages</th><th>Description</th></tr>
<tr><td>ST and IL</td><td>Directives must be entered alone on one line of text

<code>#ifdef DEF_A1_PeriodicAxis</code>
<code>MLPhaSetPhase(PipeNetwork.PHASE1,DEF_A1_PosPeriod-A1_RefPos4);</code>
<code>#else</code>
<code>MLPhaSetPhase(PipeNetwork.PHASE1,DEF_A1_LinearPeriod -A1_RefPos4);</code>
<code>#endif</code></td></tr>
<tr><td>FBD</td><td>Directives must be entered as the text of network breaks</td></tr>
<tr><td>FFLD</td><td>Directives must be entered as a network pragma with the icon.

In the example below, if CONDITION has been defined using <code>#define</code> syntax, then the networks 2 to 4 are included in the code, else the networks 5 to 12 are included.

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

NOTE Conditional compilation do not apply to actions in an SFC step.

The condition "`__DEBUG`" is automatically defined when the application is compiled in DEBUG mode. This allows you to incorporate some additional statements (such as trace outputs) in your code that are not included in RELEASE mode.

```
#ifdef      __DEBUG
      Printf('In debug mode', 0, 0, 0, 0);
#endif
```

See also "Running the Project" on page 318

5.5.2 Step 2 of 6 - Compile the Application

After creating all the elements of your project, you are ready to compile it. The project must be compiled before it is simulated or downloaded to the target.

You can compile your project with the compile icon in the toolbar (**Ctrl+B** shortcut).

The compiler reports messages in the Information and Logs toolbox (see **Compiler Output** tab).

No other actions are possible when the compilation is in progress.

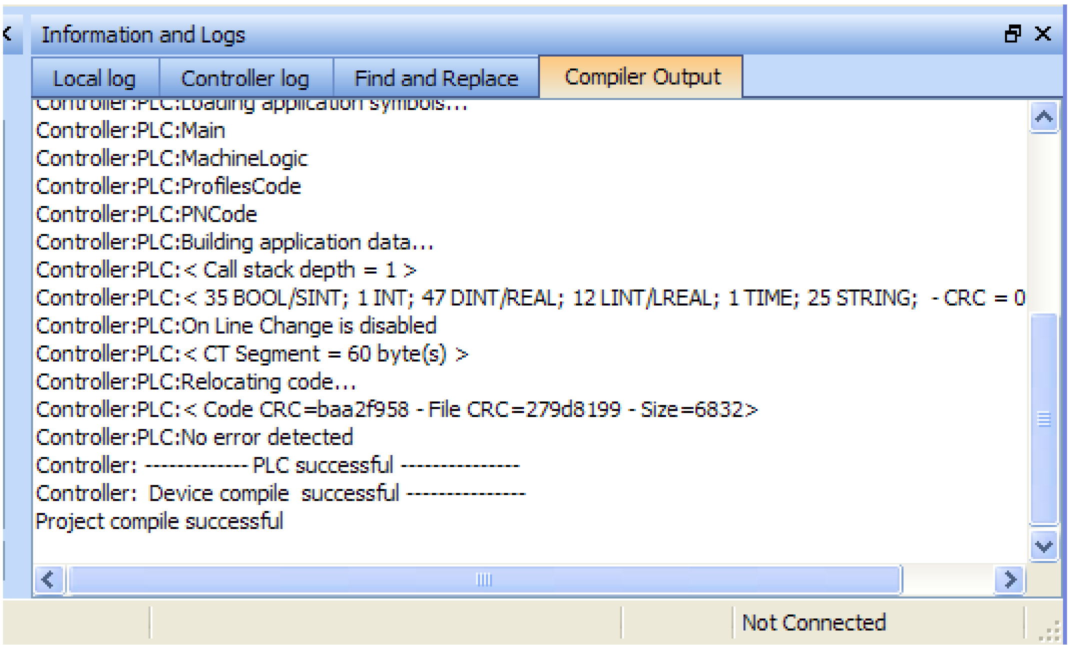


Figure 4-94: Compiler Output

TIP Errors are easily located using the information and logs window as shown below.

Double-click on an error in the list to open the program and jump directly to the relevant location in the editor.

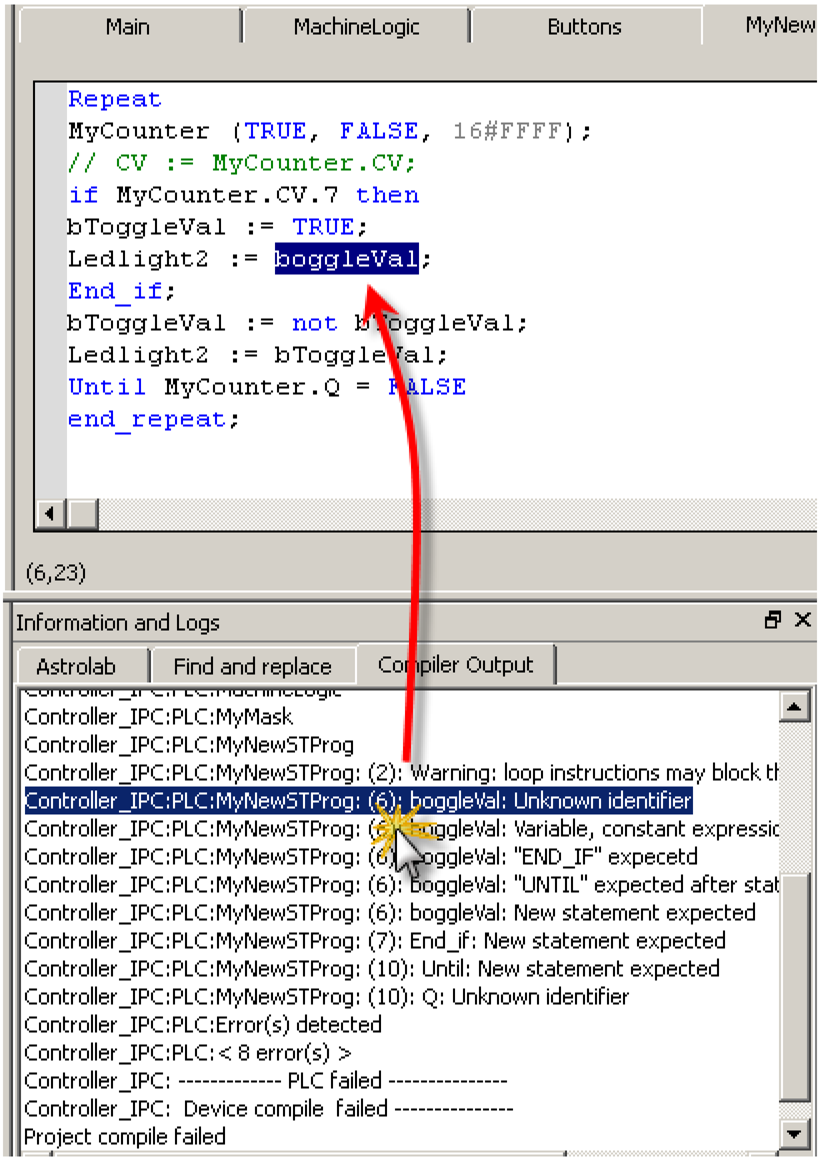


Figure 4-95: Error Location when Compiling

To locate source code, you can also use the **Find and replace** feature (for more information, refer to paragraph "Information and Logs" on page 626)

NOTE

In FFLD, when a function, function block or UDFB is not connected on the left, then it is ignored (removed at compiling time).
This case only applies for functions - **not** for function blocks.

5.5.3 Step 3 of 6 - Launch KAS Simulator

If you want to simulate your application, open **All Programs** on your computer and start the KAS Simulator application located under the **Kollmorgen** folder and the **Kollmorgen Automation Suite** subfolder.

Once the program opens, adjust your desktop preferences (position, size, etc.)

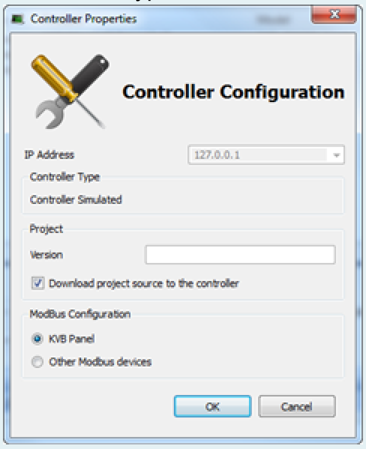
See also "Using the KAS Simulator" on page 345

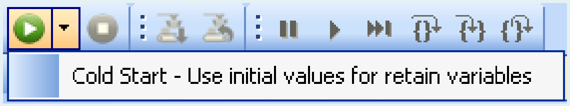
If you want to run your project on a physical device, start the KAS Runtime on the target controller.

5.5.4 Step 4 of 6 - Connect to the Controller

KAS provides all the commands for controlling the target in the **Device** toolbar:

Figure 4-96: The Device Toolbar

Icon	Description
	Show the controller communication properties dialog (PLC options) (see details below)
	Change the controller IP address to connect with KAS Simulator. In this simulated mode, the controller properties change as shown below and the Controller Type includes the IP address `127.0.0.1`.  For more details, refer to "Using the KAS Simulator" (see page 345)
	Compile project
	Compile and download the project. This downloads the application and Project Source files.
	Establish connection with the target controller (see possible statuses here)
	Close connection with the target controller
	Download the application to the targeted controller (note that the application must **not** be running). For more details, refer to "Step 5 of 6 - Download the Application" (see page 325)

Icon	Description
	Start the application. It can be either: **Warm** Start (default mode): retain variables are loaded at the application startup. They are **Not** re-initialized; whereas other variables are started with their initial values. **Cold** Start: use retain variables with their default values. Such starts occurs from time to time but are few. To choose a Cold Start, you have to drop-down the Start button as follows:  **TIP** If the previous download failed (due to events such as the Ethernet being disconnected or the controller being turned off), you will not be able start your application. Additionally: • no version information is shown in the web server or status bar • the Start, Cold Start and Download buttons are not available on the web server • if the AKD PDMM tries to auto-start, an E24 error is flashed on the 7-segment display. Review the Controller logs for more information.
	Stop the application

Ensure the Simulated device mode is active (the icon must be selected)

To establish the connection with the target controller, click the Connect Device icon .

NOTE You need to configure the device before connecting (see "Configure the Controller" (see page 188))

5.5.4.1 Actions to Prevent Compatibility Issues

The software versions of the KAS IDE and the KAS Runtime have to match to avoid compatibility problems. The version consists of a series of four numbers (e.g. 2.1.1.87).

See "KAS IDE to Runtime Compatibility" (see page 182) for more information.

TIP The software versions of the KAS IDE and the KAS Runtime are also available in the local log messages (the level for this message is INFO).

When another KAS IDE is already connected to the controller, a warning is displayed and the connection is discarded to prevent any conflict.

5.5.4.2 Application Status Bar

The status bar provides global information about the target and the name of the running application currently stored in the device.
Text displayed with **orange** background means that the version of the application is different between the KAS IDE and the target.

For more details, see page 648

5.5.4.3 Message Window

Every log message has the following information:

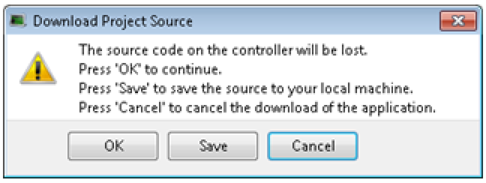
- Timestamp
- ID
- Message

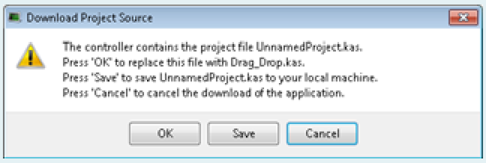
NOTE Once connected to the device, it is no longer possible to edit the PLC programs, unless the Online Change mode is active (see "Step 2 of 6 - Compile the Application" (see page 321))

IMPORTANT Depending on the number of AKD drives physically present in the EtherCAT network, the KAS IDE might slow down when getting data.
The KAS Runtime is **not concerned** with this limitation.

5.5.5 Step 5 of 6 - Download the Application

- Click the *Compile and Download* or the *Download* button.
  The application is compiled and downloaded or just downloaded to the controller.
  - If you enabled the "Download Project Source in the controller" option in the Controller Properties dialog box (see "Configure the Controller" (see page 188)) then the source code is also copied to the controller.
  - If you disabled the "Download Project Source in the controller" option and the controller has project source loaded you will be prompted with a message. The message presented will vary, based on whether your current project and the project on the controller have the same name or not.

Project Names	Message	Actions
Same name		**OK:** Source on controller is deleted. **Save:** Source on controller is overwritten and is also saved to "My Documents". If that source file already exists in My Documents you will be prompted to rename the source file, overwrite that file or not. If you do not rename or overwrite the file then the download to the controller will be canceled. **Cancel:** Download is stopped.

Project Names	Message	Actions
Different name		**OK:** Source on the controller will be replaced. **Save:** Source on the controller will be saved to "My Documents" as above. **Cancel:** Download is stopped.

- You will be prompted with an alert if there is not sufficient space on the for the application. Clearing the "User Data" (see page 383) is one method to create more space on an AKD PDMM.

TIP The versions between the KAS IDE and the KAS Runtime must be the same if you want to be able to debug your application (for example to display the animated values in the editors).

IMPORTANT If the IDE version differs from the runtime after compiling an application, the function blocks defined in the IDE and those implemented in the virtual machine of the runtime can possibly be different. To prevent this potential mismatch, you must compile and download your application again.

NOTE In addition to downloading the application to the controller's flash memory, you can download the project's source code, allowing you to store the project. See "Menus and Toolbar Overview" (see page 651) for information on retrieving saved projects.

5.5.5.1 Application Status Bar

The tooltip of the application status bar gives more information about the application stored in the target: name of the project, name of the device, version of the application, its build number and date of compilation.

To view the tooltip, hold the mouse over the application status bar and wait for 1 or 2 seconds without moving the mouse.

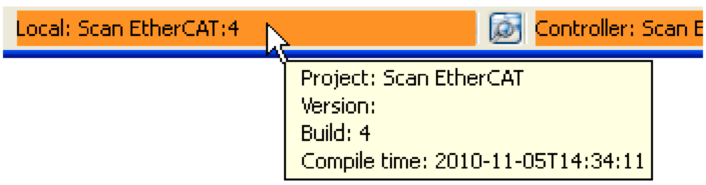


Figure 4-97: Device Tooltip displays Version

5.5.6 Step 6 of 6 - Device Control

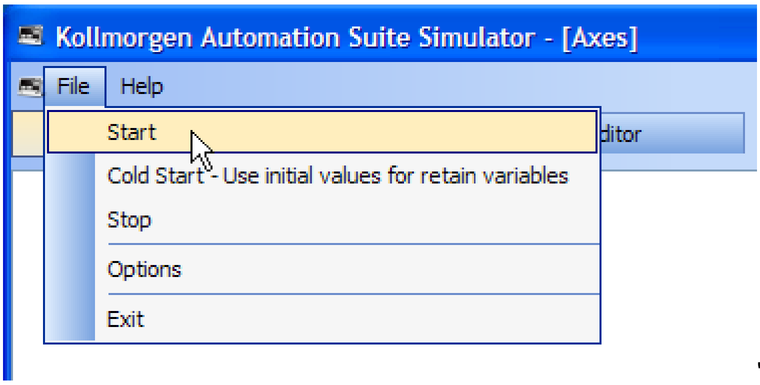
5.5.6.1 Start/stop the Device

With the KAS IDE

You can start / stop the device with the buttons  and 

With the KAS Runtime

In the KAS Runtime menu you can click the **start** / **stop** command.



" />

Figure 4-98: Start Device with the KAS Runtime

5.5.6.2 Log Window

The Log window displays all run-time messages issued by the device or by the KAS Simulator when testing the application.

The log area of the KAS IDE and the KAS Runtime Simulator are the same. It contains the log messages as described in "Information and Logs" (see page 626)

5.6 Testing and Debugging the Project

During system validation it is essential that the KAS IDE allows you to monitor the application program execution and to capture critical events and their data when they occur.

A Control Panel (designed with an internal editor) can be used to provide a basic interface.

5.6.1 Step-By-Step Debugging

To minimize risk, the KAS IDE in conjunction with the KAS Simulator allows checking and validating the application program prior to deployment of the machine/system in production. This is achieved by capturing critical events in a step-by-step mode.

In addition to the cycle-by-cycle execution mode, the debugger has a rich collection of powerful features for making step-by-step debugging in the source code of your application.

NOTE Step-by-step debugging is available only if the project has been compiled with the **DEBUG** option. This option can be selected from the project compiling PLC options dialog box, accessible with the icon.

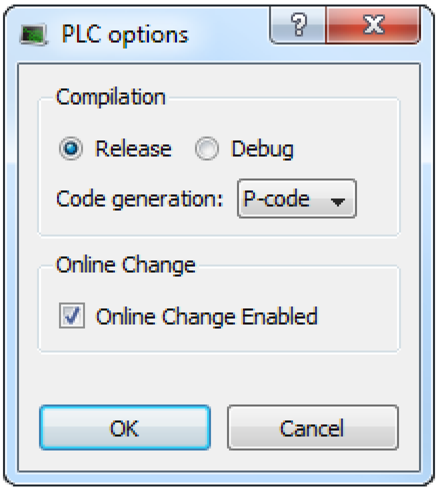


Figure 4-99: PLC Options - Debug Compiling Mode

- An application compiled in **Debug** mode includes additional information for stepping. This leads to bigger code size and reduced performance.
- When debugging is finished, it is recommended to compile your application in **Release** mode to give highest performance to your application.

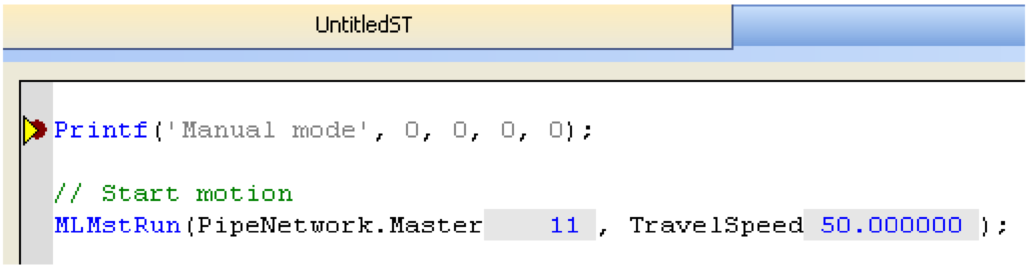
Step-by-step debugging is available:

- In ST and IL text programs (a step is considered as a statement)
- In FFLD programs (a step is considered as a rung)
- In FBD (a step is considered as a graphic symbol corresponding to an action)

IMPORTANT Step-by-step debugging is **not possible in SFC** programs (for note about SFC, see page 329)

There are two possibilities for entering the step-by-step debugging mode:

- Set a breakpoint in a program (for more details, see "Breakpoints" on page 329)



When you start your application and the breakpoint is reached, the execution stops at the specified location and you can run one step further in the program with the stepping commands.

- When the target is in cycle stepping mode (STOP), you can step to the beginning of the first program.

TIP Pausing a program will not interrupt the current VM cycle. The current cycle will finish and execution will be paused before the beginning of the next VM cycle.

In the Debug toolbar, the following commands are available for stepping:

Icon	Description
	Step Over the next instruction: If the next instruction is a call of a function block or a sub-program, the execution passes over to the following instruction.
	Step Into the next instruction: The next step will be at the beginning of the called block (if the next instruction is **not** a call of a function block or a sub-program, then the Step Into behaves like the Step Over)
	Step Out the current block: If the current stepping position is in a called function block or a sub-program, the execution continues up to the end of the current block. Otherwise, the Step out behaves like the Step Over.

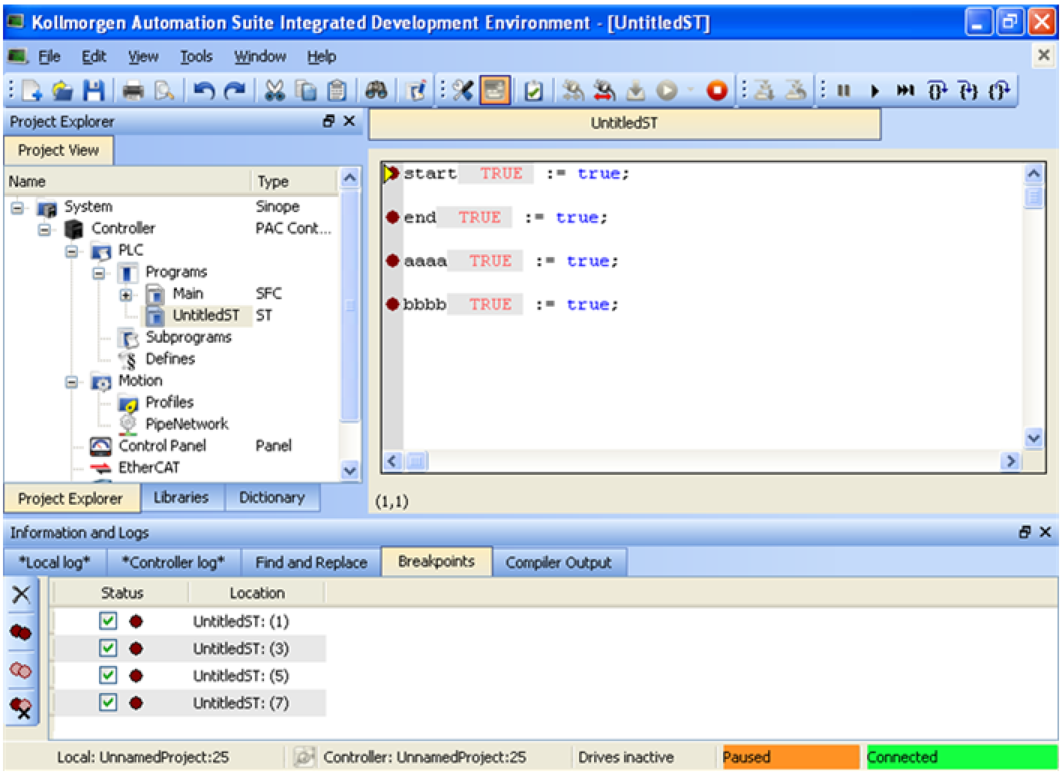
In addition to these commands, you can click at any time:

Icon	Description
	Execute the cycle (from the current position up to the end of the last program)
	Restart the target in "normal" execution mode (RUN)

5.6.2 Breakpoints

The step-by-step debugging feature is enabled by setting breakpoints in the source code of the application.

NOTE This feature is only available when you have chosen the **DEBUG** mode (for more details, see page 318).



5.6.2.1 About Breakpoints

- Breakpoints are a marker that is set in code which, when reached, stops the code's execution stops at that location. This lets you run one step further in the program with stepping commands.

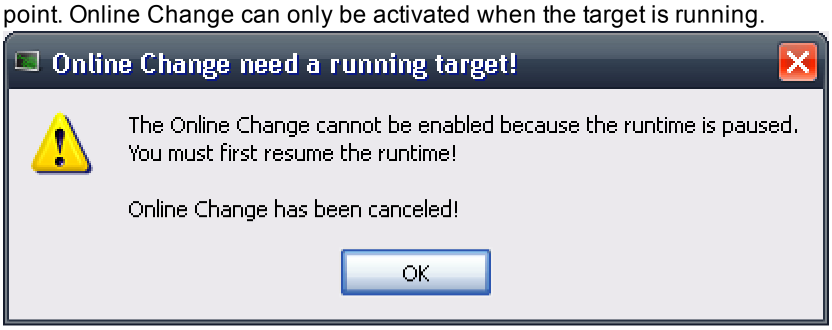
> **TIP** Pausing a program will not interrupt the current VM cycle. The current cycle will finish and execution will be paused before the beginning of the next VM cycle.

- Breakpoints are shown as a red circle (dark or light) in the left margin.
- Breakpoints may be active ( ) or inactive ( ).
- Breakpoints are active only when the IDE is connected to a target running an application that is compiled from the exact code displayed in the editor.
- Breakpoints are inactive if:
  - the IDE is not connected to a target
  - the IDE is connected but not running
  - the IDE is connected to a different version of the code
  - the IDE is connected to the code but a modification has been made in Edit mode.
- Breakpoints will always be applied to the target, based on their position in the editor. If a breakpoint is moved in the editor, then you reconnect to a target, the breakpoint in the target will be moved to the new position.
- A Breakpoint that has been "hit" has a yellow triangle (and) to indicate it has been reached in the code.
- Breakpoints are saved when saving the KAS application and are reloaded when loading a KAS application.
- See "Setting, Removing, Enabling, and Disabling Breakpoints" (see page 331) for information on working with breakpoints.
- See "Breakpoints tab" (see page 637) for information on the **Breakpoints** tab in the **Information and Logs** widget.
- Projects support a maximum of 16 breakpoints. This includes both enabled and disabled breakpoints.

> **NOTE** Breakpoints can significantly increase the PLC cycle time execution. This is due to the fact that the VM must evaluate the breakpoint condition at every cycle.

About Online Change

- Online Change cannot be enabled when the KAS Runtime is paused due to a breakpoint. Online Change can only be activated when the target is running.



- Every breakpoint is activated if an Online Change is performed successfully.

> **NOTE** The breakpoints are not activated synchronously but in a reasonable time.

- All breakpoints become inactive when an Online Change is reverted.

5.6.3 Setting, Removing, Enabling, and Disabling Breakpoints

This section discusses working with breakpoints within the editor. See "Breakpoints tab" (see page 637) for information on the **Breakpoints** tab in the **Information and Logs** widget, including modifying breakpoints in bulk.

5.6.3.1 How to Set Breakpoints

1. Open your program in the IEC 61131-3 Editor.
2. Click on the line (for ST/ IL) or diagram (for SFC [1] , FBD or FFLD) where you want to set the breakpoint.
3. Press **F9** or right-click and select **Set Breakpoint** from the menu.

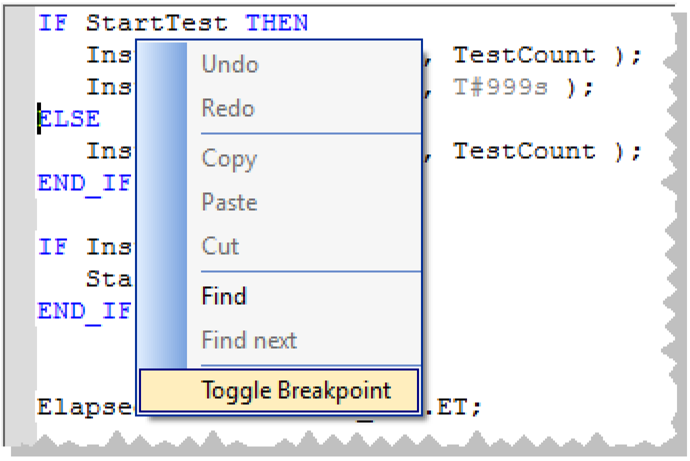


Figure 4-100: Setting Breakpoints

4. A Breakpoint circle is added in the left margin. The Breakpoint will be set as either active ( ) or inactive ( ), based on the IDE's connectivity (see "About Breakpoints" (see page 330)).

Even when you are **not** connected to the Controller, breakpoints can be placed in programs, sub-programs or UDFBs.

NOTE When you start your application, if the current position is not on a valid line for stepping, the breakpoint is automatically moved to the nearest valid position.

IMPORTANT When you close the connection with the target, all the breakpoints are removed in the KAS Runtime.

About SFC

There are several things to note about breakpoints in SFC programs:

- In **SFC** programs, breakpoints can only be set on transitions (i.e. in First Level diagram), and not in steps or conditions. With a breakpoint set on a transition, you can debug cycle-by-cycle. Please remember that P1, N and P0 placeholders are designed to contain very simple code.

  TIP The recommended way to proceed for SFC sub-level programs is to rely on subprograms, where debugging is allowed.

- Breakpoints can be set and removed in SFC programs, they cannot be enabled and disabled.

[1] See limitation explained in paragraph below: **About SFC**

5.6.3.2 How to Remove a Breakpoint

To remove a breakpoint, right-click where the Breakpoint is set and select **Remove Breakpoint** from the menu. Selecting this option will remove the breakpoint from the left margin of the editor. This applies to both active and inactive breakpoints.

5.6.3.3 How to Enable a Breakpoint

To enable a breakpoint, right click on an inactive breakpoint and select **Enable Breakpoint**. This is only available when the IDE and runtime are connected.

5.6.3.4 How to Disable a Breakpoint

To disable a breakpoint, right click on an active breakpoint and select **Disable Breakpoint**. Selecting this option will remove the breakpoint from the runtime; the breakpoint will be remain in the editor and be changed to an inactive state ( ).

5.6.4 Printf Function

You can use the Printf function to display string in debug mode.

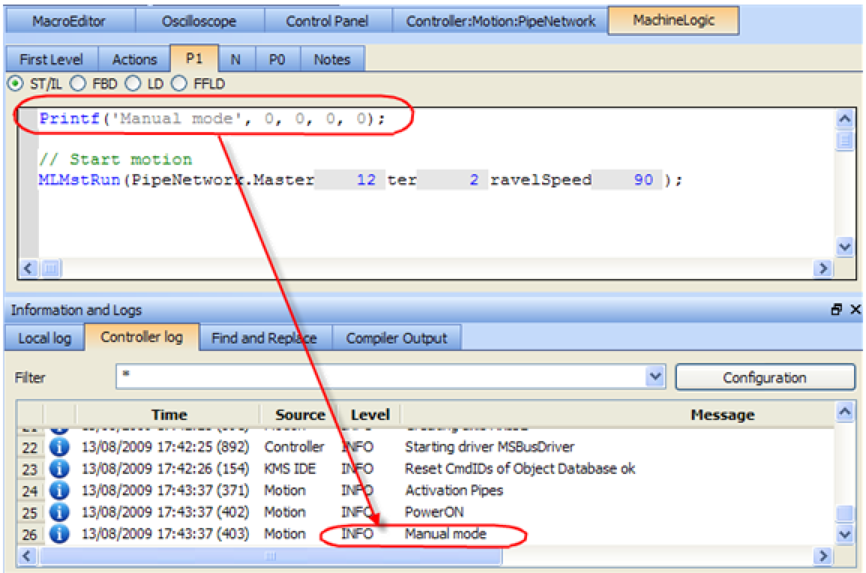


Figure 4-101: Printf Function

It can be a good way to trace your SFC programs.

Note that you can also use the PrintMessage (Function).

How to customize output in the log window?

Raise warnings or errors icons

First column in the log window displays an information icon which can be replaced with a warning or error icon as follows:

```
@W or @E
```

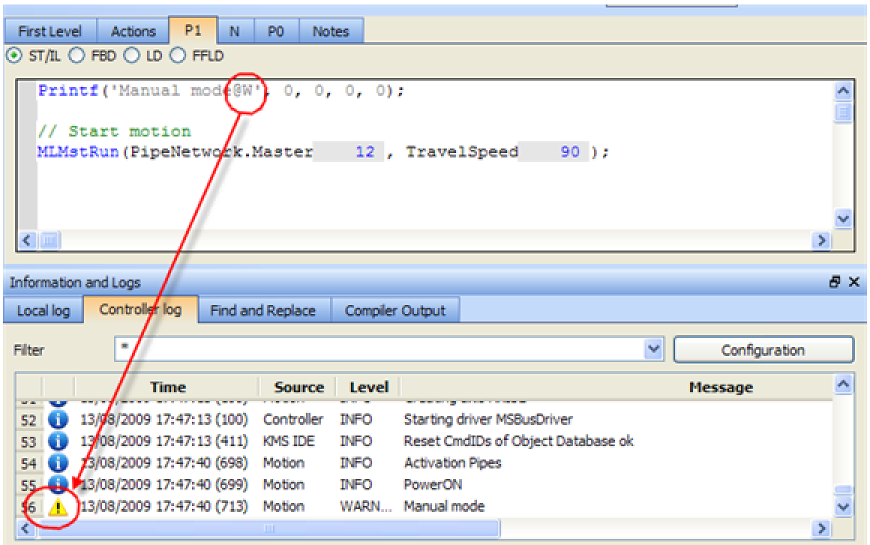


Figure 4-102: Customizing Output for Printf Function

5.6.5 Soft Oscilloscope Debugging

It can be interesting to access the values of the variables of the application. It is particularly important during development and debugging.

There is a way of visualizing and changing variables via the Graphics HMI panel (see paragraph "IEC 61131-3 Editor Debugging" on page 340). You can also access and change variables via the Variable Dictionary (see paragraph "Variable Monitoring" on page 338).

However, these two methods can only access and change variables from the PLC part and not from the Motion part of the application. Furthermore, the temporal evolution of the Motion variables would not be very intuitive. The ideal tool to trace the Motion variables is a softscope.

Other typical areas for using the softscope are:

- Recording when an input is sensed in a cycle
- Recording how much correction is being made in each cycle
- Checking the settling time of an axis

To open the Softscope, click the **Oscilloscope** command in the Tools Menu.

For more details on Softscope description and usage, refer to paragraph "Softscope" on page 409

5.6.5.1

5.6.5.2 How to Plug Motion Variables

NOTE The Softscope retrieves the variable values from the Motion Simulator. You can only plug objects which exist in the Motion Simulator. While the PLC variables exist all the time, the Motion objects are only created after the start of the application.

When your application is running, do the following:

1. Open the **PipeNetwork** of your Controller in the Workspace
2. Right-click on Gear1 to open its menu
3. Choose the command **Plug on channel...**

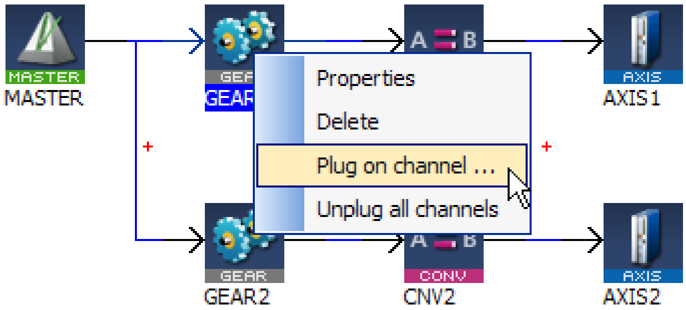


Figure 4-103: Plugging a Motion Variable

NOTE Your application **must be connected and running** to let you plug a channel to a variable

4. Set Channel to **1** and choose the relevant Data

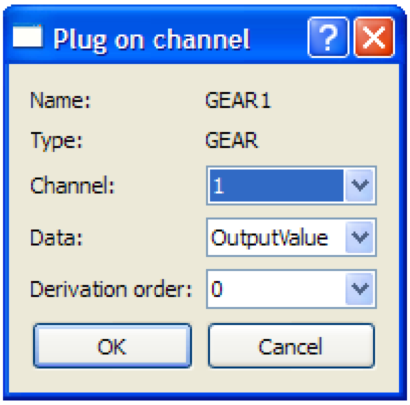


Figure 4-104: Plugging a Motion Variable - Parameters

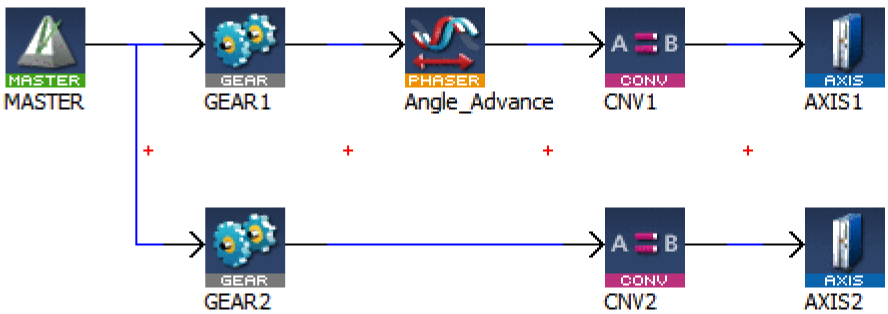
NOTE The complete list of data are only visible when your application is running

For more details on the parameters, refer to paragraph "Plugging Probes" on page 417

Usage example with the Pipe Network

The Softscope allows the recording and display of motion at points any where in a Pipe Network.

The following example shows the difference between the input and output of the Phaser Pipe Block (called AngleAdvance).



The red line is the input, the green line is the output and the blue line shows when the phase advance change was active.

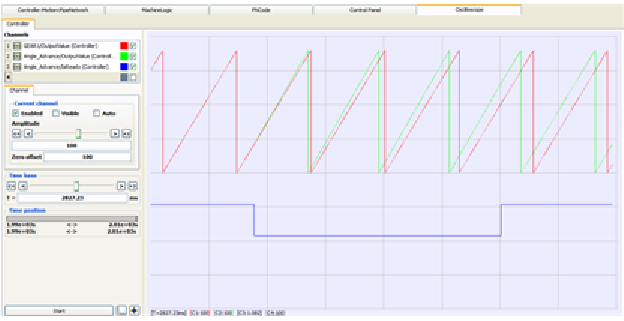


Figure 4-105: Example of Plugging a Pipe Block

See also "How to Plug PLC Variables" on page 335

5.6.5.3 How to Plug PLC Variables

1. In the Variable Dictionary, right-click on the variable **lastMachineSpeed** to open its menu
2. Choose the command **Plug on channel**

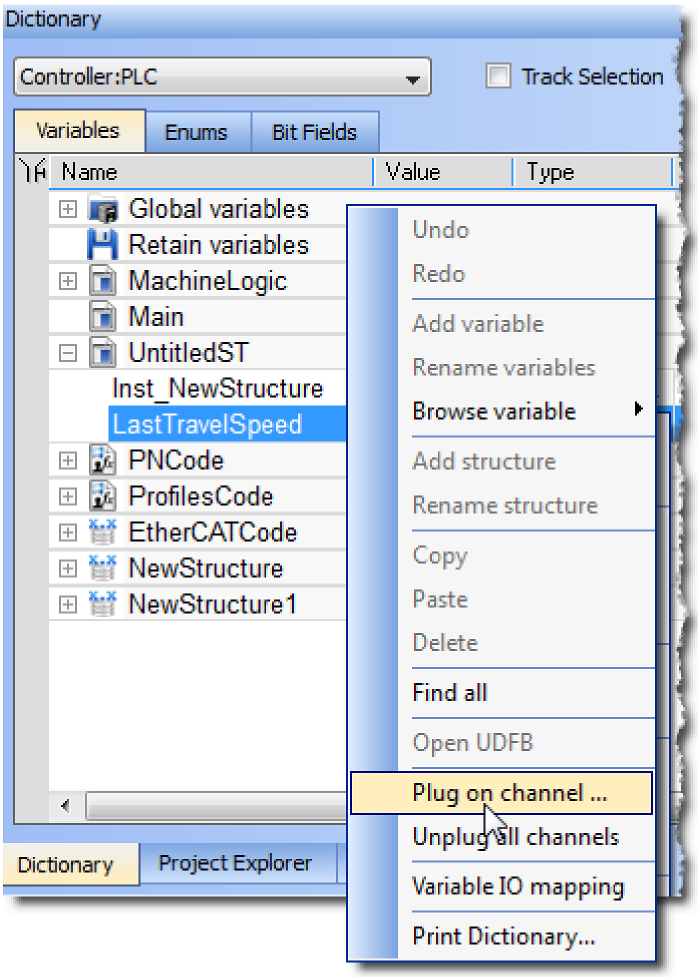


Figure 4-106: Plugging a PLC Variable

3. Set Channel to **2**(because channel 1 is already plugged)

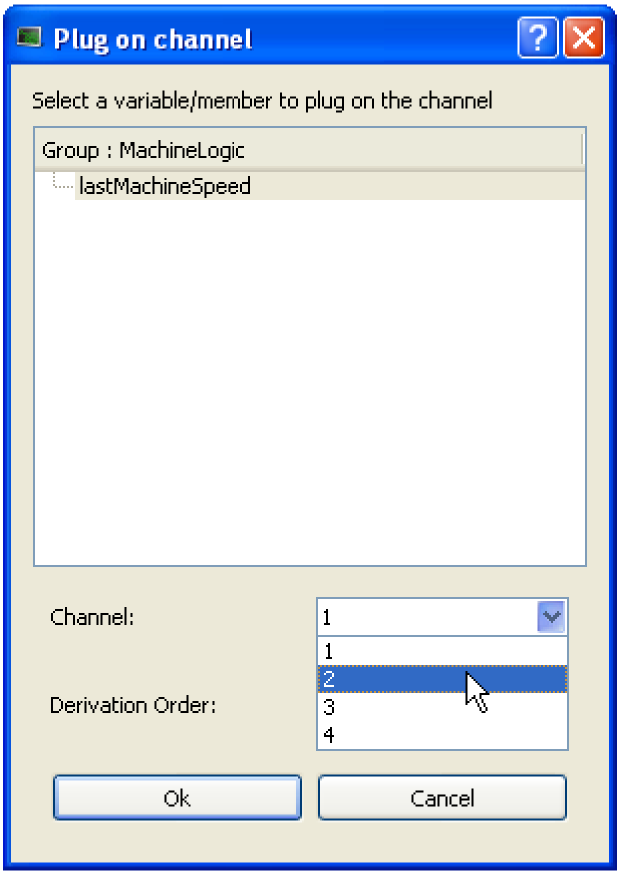


Figure 4-107: Plugging a PLC Variable - Parameters

You can start the Softscope now to see traces, as shown in the following figure:

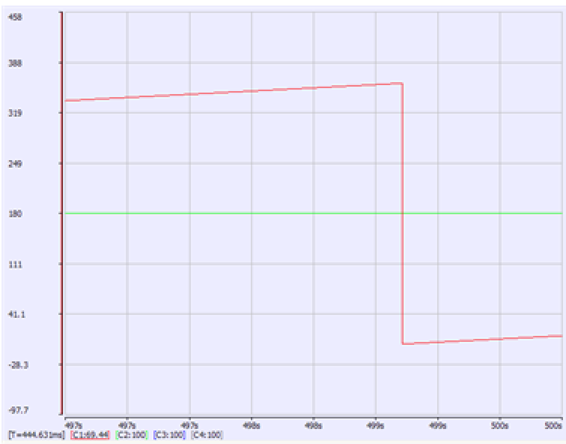


Figure 4-108: Traces Displayed with Soft Oscilloscope

TIP Easy probe plugging is assured since you do not need to unplug a probe from a channel before plugging a new probe into the same channel.

See also "How to Plug Motion Variables" on page 333

5.6.6 Compare PLC Programs

KAS provides a tool to show the differences between the "Local Project" and the project currently on the Controller.

TIP To compare local projects, use the Compare Projects function from the File menu.

This tool, "Compare PLC Programs" is accessed from a button on the Status Bar, between the Local and Controller versions. It is active when KAS is connected to a controller.

Figure 4-109: Difference in Local and Controller Versions

Click the button to open the list of items for both versions. **Red** item indicates where there is a mismatch. Double-click to open an item. The **<<** button brings you back to the list.

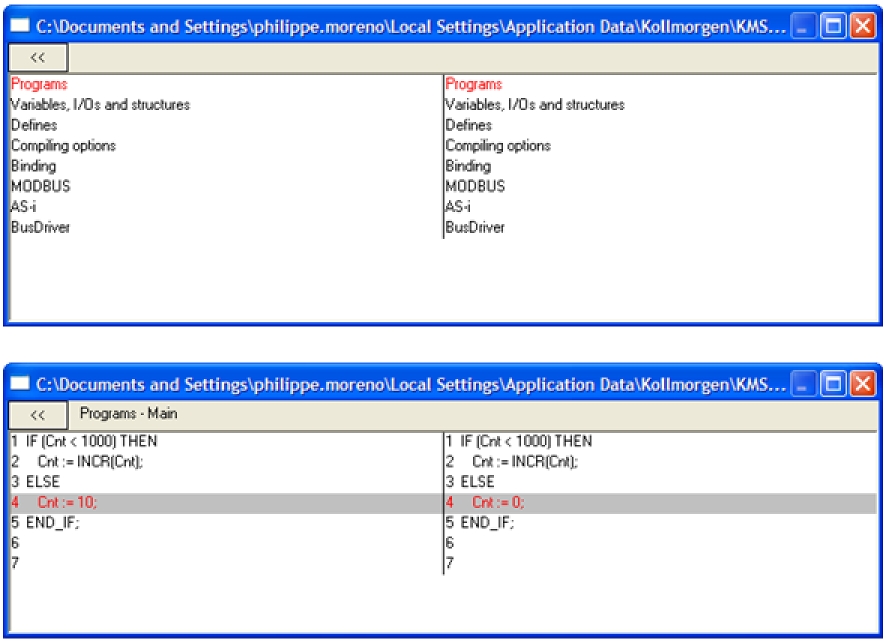


Figure 4-110: Listing the Differences

The following message will be shown if you click on the compare button but the project is not present on the controller.

5.6.7 Variable Animation

When your application is running, all variables in the IEC 61131-3 Editors, in the Dictionary and in the Watch Window are animated. This means that the value of each variable is displayed dynamically.

NOTE When the value of a variable is displayed, only the value computed at the end of the cycle is displayed.
So if the same variable is set in different programs, the animation in all those programs displays the same value for the variable, which corresponds to the

NOTE latest program executed within the cycle.

About Online Change

When Online Change is enabled,the animated values only take place when you are in Debug mode (and not edit).

Limitations

- The versions on the KAS IDE and the KAS Runtime must be the same
- Animation does not apply to actions in an SFC step

5.6.7.1 Variable Monitoring

The Variable Dictionary contains all the IEC 61131-3 variables needed by the application. The variables are listed by categories corresponding to the declared programs, functions and function blocks.

When your application is running:

- all variables in the Dictionary are animated [1] with real-time values displayed in the **Value** column (see call out 1)
- a specific column is used to indicate the initial values of all variables 2

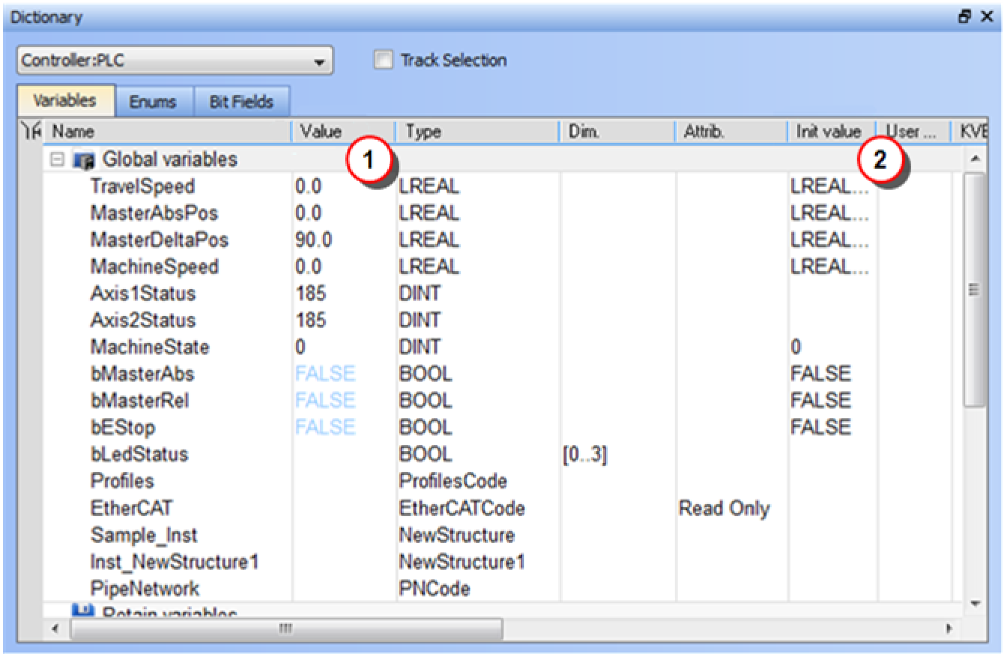


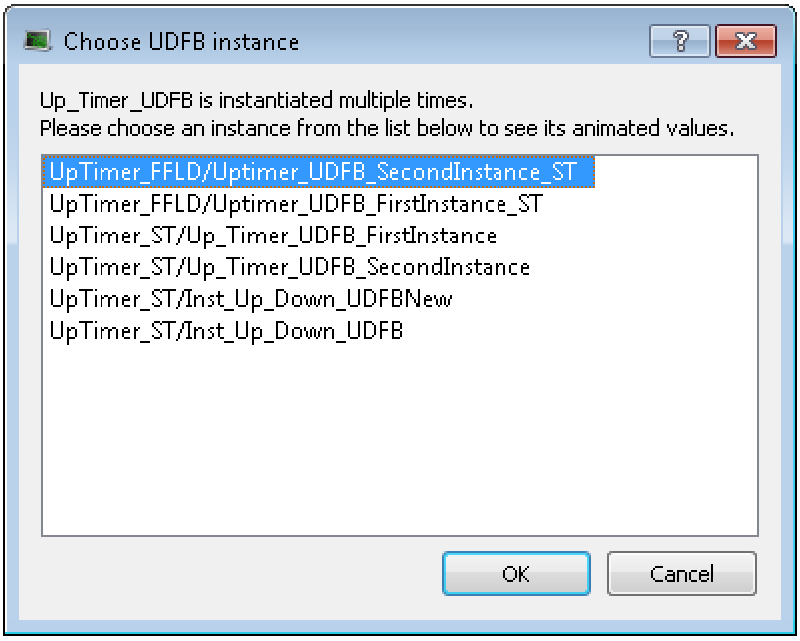
Figure 4-111: Variable Dictionary

About UDFBs

The real time values for UDFBs can be shown in the workspace while the program is running. Double clicking on a UDFB in the Project View's Subprograms list will open the UDFB in the workspace. You will first be presented with a list of the UDFB's

[1]To better track variables and expressions of the PLC programs in Test mode, the KAS IDE dynamically computes their value along with the program execution and display the result in gray boxes beside their usage in the instruction lines of the IEC 61131-3 editor.

instances if it is instantiated more than once or declared inside of a structure.



If the UDFB is open in the Workspace before running the program it will not automatically animate because there may be more than once instance of the UDFB. To start the animation double click the entry in the Project View. If there is only one instance, it will open; if there are multiple instances the list will be presented.

Forcing a variable

At run-time, double-click on the value of the variable in the list or press the **ENTER** key when it is selected. A popup window appears and allows you to:

- **Force**: change the value of the selected variable. Depending on the variable type, you have the possibility to define its value either in the text field or with the check boxes.

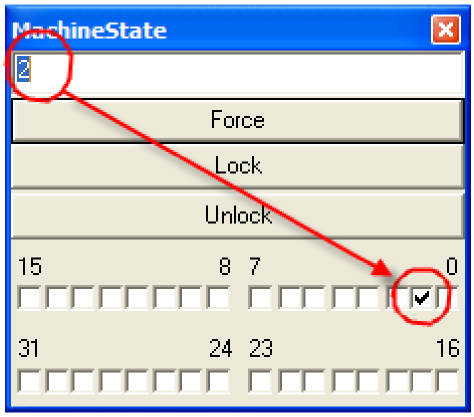


Figure 4-112: Forcing a Variable

- **Lock**: When a variable is locked, its value is no longer changed by the runtime. You can then force its value from the debugger, independently from the runtime operations. Note that all variables can be locked and forced at run-time.

ⓘ **TIP** The value of a locked variable is displayed with square brackets.

```
ActualMachineState [[ -1 ]] := -1;
```

- **Unlock**: Remove the lock on a variable so it can be changed again by the runtime.

5.6.7.2 IEC 61131-3 Editor Debugging

In Test mode (Online or Simulation), all editors are animated [1] with real-time values of the edited objects:

- Values of variables, contacts and coils are displayed in FBD diagrams. Double-click on a variable name to force or lock the variable
- Values of variables, contacts and coils are displayed in FFLD diagrams. Double-click on a variable name to force or lock the variable
- Step activities (tokens) are displayed in the SFC editor
- In the text (ST or IL) editor, place the mouse cursor on a variable name to display its real-time value in a tooltip.
  Double-click on the variable name with the **Shift** key pressed to force or lock the variable

```
Repeat
MyCounter (TRUE, FALSE, 16#FFFF);
// CV := MyCounter.CV;
if MyCounter.CV.7 FALSE then
bToggleVal TRUE := TRUE;
Ledlight2 TRUE := bToggleVal TRUE ;
End_if;
bToggleVal TRUE := not bToggleVal TRUE ;
Ledlight2 TRUE := bToggleVal TRUE ;
Until MyCounter.Q FALSE = FALSE
end_repeat;
```

Figure 4-113: Animation in Editors

See also paragraph "Forcing a variable" on page 339

5.7 Managing a Project

The **New** command in the File menu uses a wizard to help you to define the project.

The **Open...** command opens a window to let you navigate your system and retrieve previous projects.

The **Save** command saves your entire project.

The **Save As...** command allows you to save your project with a custom name and location.

NOTE Choose a safe folder for your project. Never select the Installation repository .

The **Close** command prompts you to save first if some modifications have not been saved.

NOTE When a project is already open, and you try to create or open another one, the KAS IDE proposes you to save your project before it is closed.

The **Print...** command allows you to create documentation containing editors' programs or diagrams.

[1] To better track variables and expressions of the PLC programs in Test mode, the KAS IDE dynamically compute their value along with the program execution and display the result in gray boxes beside their usage in the instruction lines of the IEC 61131-3 editor.

For more details on the File menu, also refer to paragraph "Menus and Toolbar Overview" on page 651.

TIP With the **Recent Projects** command in the File menu, the last four projects can be opened easily.

When editing your project, the KAS IDE has the following restrictions:

- You cannot work with several projects in parallel
- Modifications that impact the project structure cannot be reversed with the **Undo** command (you have to make a backup first using the **Save As** command)
- No guarantee is provided by the KAS IDE with respect to the project file's integrity (this means that if you modify your data from outside the KAS IDE, you can spoiled your project)

Use a Version Control System

To ensure integrity of your project files, you have to rely on tools to control versions.

Generally, such tools also have facilities for:

- Backup management
- Multi-users or multi-site development

5.7.1 Print

5.7.1.1 Printable Elements

The elements that you can print are:

- All PLC programs (see PLC node in the Project Explorer)
- Individual programs
- Level 2 SFC
- Level 2 SFC of single transition/state
- The Pipe Network editor
- The Dictionary

You can either print one specific program or all the project (PLC, Motion, Dictionary variables)

5.7.1.2 Page Setup

This dialog enables you to define the following settings:

Page Setup tab

- Orientation:
  Allows you to choose between portrait or landscape.
  Because the orientation can be set in both the page setup and the printer driver, it is recommended to have both settings synchronized.
- Scaling:
  You can select the **Fit to** option to fit on the specified number of pages. You have to enter one of the two values (either Wide or Tall) and the other are filled in automatically to keep a 1:1 aspect ratio of the print.

NOTE These settings are not applicable when printing a project.

Margin and Header/Footer tab

If you specify new margins or header/footer for a program, it affects the entire project when printed.

About field items used in Header/Footer

Special items can be inserted into the header/footer string as **{@item}**. They are converted to the correct format on printing or for print preview.

About the **Filename** field:

If an SFC level 2 program is being printed, the filename contains the SFC program name, Step or Transition number and the action tab name (e.g. Main, GS3, P1).

NOTE All the settings defined in the Page Setup are saved within your project and are applied to each printed program.

This dialog box also contains two buttons:

- **Print**… displays the Printer dialog box as described below
- **Print Preview** displays a printout on the screen so you can see how it looks like before printing it.

5.7.1.3 Print

This dialog enables you to:

- Set the output (a printer, a PDF)
- Set the output preferences to set-up the printer options
- Look for a printer on the network
- Set the number of copies
- Set the page area to be printed
- Start the print

To print an SFC level 2 program, open it in the SFC editor and click the Print icon (Ctrl + P)

5.7.1.4 Print Preview

This dialog box enables you to display a printout on the screen so you can see how it looks before printing.

NOTE Print preview limits the number of pages to display to the first 30 pages.

5.7.1.5 Print Project

A Print Project dialog displays all the items that are printable. Then you can select those you want to include in your output and click **OK**.

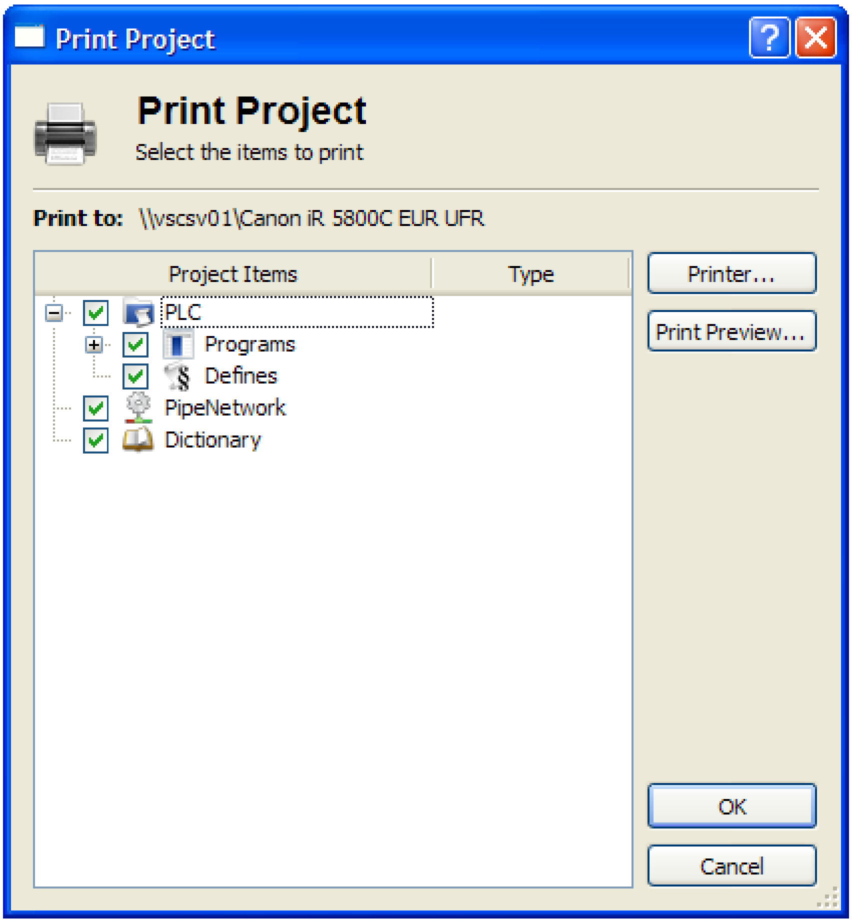


Figure 4-114: Print Project

> **NOTE** Selecting an SFC program prints the SFC chart as well as SFC level 2 programs.
> Automatic scaling is applied for best readability.

5.7.2 Use the Reference Folder

Using the Reference item, you can link as many files as you want to your project.

1. Right-click on the Reference item and select the **Insert Reference** command

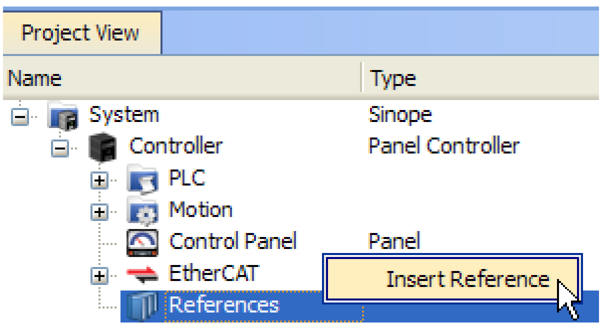


Figure 4-115: Inserting a Reference

2. Define the **Name** and choose a valid **URL**

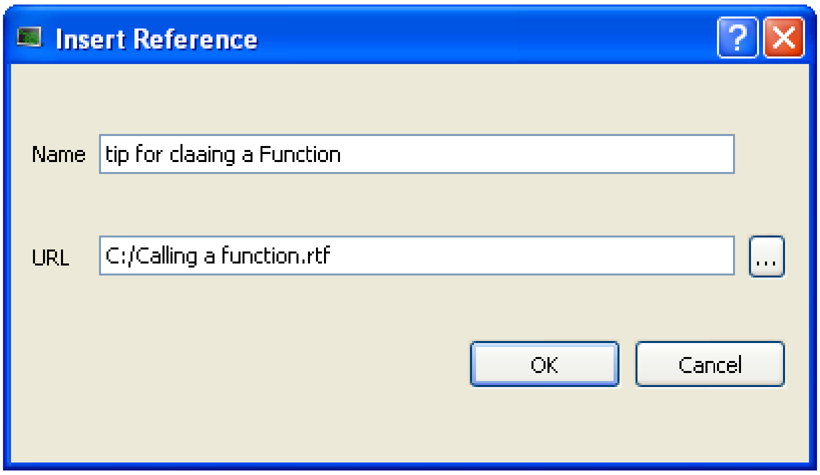


Figure 4-116: Defining the Reference

3. You can double-click the new reference to open it in the workspace

NOTE You can link files that are on your local machine (or to a server shared with a mapped drive) and of the following types: pdf, doc, xls, drawings, etc.. You must ensure the link is not broken if you want the KAS IDE to open it correctly.

6 Using the KAS Simulator

Tasks related to the Runtime are:

- Start and stop the machine
- On-line inspection to check actual parameters during commissioning

6.1 Start KAS Simulator

Open **All Programs** and start the **KAS Simulator** application located under the **Kollmorgen** folder and the **KAS** subfolder.

NOTE Simulator uses port 80 for the web server. This is mandatory for proper communication. Before starting Simulator, please close any application, such as VOIP, Skype, or IIS, that may use port 80. If another service is using port 80, you will receive a prompt to close the application and retry Simulator.

TIP If you are experiencing trouble determining what software is using port 80:

1. Run `netstat -o` from a command prompt. This will output a list of ports and the process ID using the port. In this example, process 4000 is using port 80.

```
C:\Users\Admin>netstat -o

Active Connections
  Proto  Local Address                 For-
eign Address          State            PID
  TCP    xxx.xxx.xxx.xxx:2492
```

TIP

```
blugro5relay:2492      ESTABLISHED
5232
  TCP    xxx.xxx.xxx.xxx:80
173:http               ESTABLISHED
4000
  TCP    xxx.xxx.xxx.xxx:80
173:http               ESTABLISHED
4000
  TCP    xxx.xxx.xxx.xxx:53405
cs115p1:5050           ESTABLISHED
1688
  TCP    xxx.xxx.xxx.xxx:53416      bos-
m001c-rdr2:https   ESTABLISHED     1688
  TCP    xxx.xxx.xxx.xxx:53418      chat-
d03b-rdr2:https   ESTABLISHED     1688
  TCP    xxx.xxx.xxx.xxx:53428
mtnradsvk1200:52230    ESTABLISHED
2076
  TCP    xxx.xxx.xxx.xxx:53442      pb-in-
f125:5222        ESTABLISHED     4868

C:\Users\Admin>
```

2. Open Windows Task Manager, switch to the Processes tab.
3. Click View > Select Columns... and ensure that PID (Process Identifier) is selected.
4. Sort the Windows Task Manager by the PID column to easily find the name of the process which is using port 80.

The first time Simulator is run it will attempt to open some TCP/IP ports to allow communication. Your system's firewall will detect this and prompt for an action. Allow the Simulator to open the ports by selecting Unblock (Windows XP) or Allow Access (Windows 7).

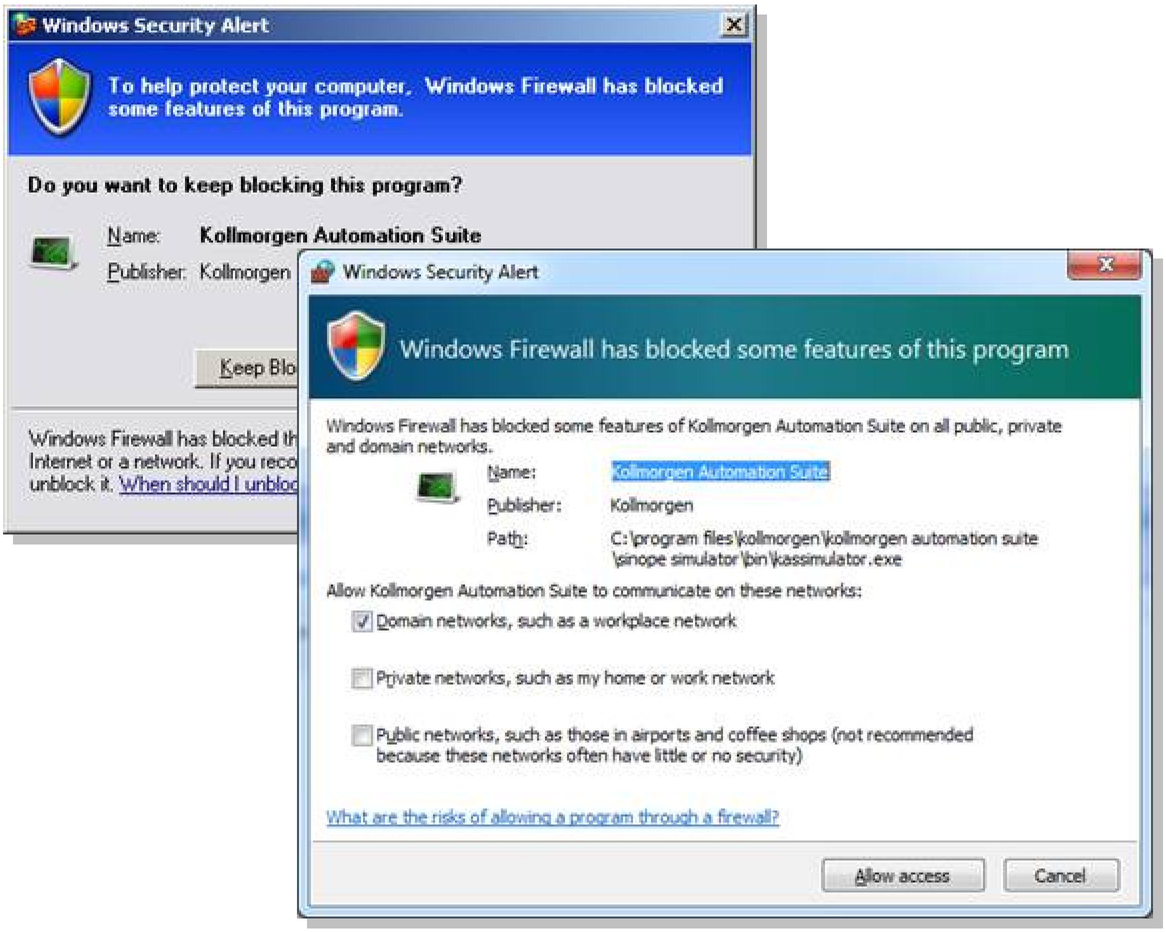


Figure 5-1: Windows XP and Windows 7 Firewall alert dialogs.

Before using the KAS Runtime, you first need to compile and chapter "Step 5 of 6 - Download the Application" on page 325

ⓘ TIP

After the project is debugged using KAS Simulator, it can be downloaded to the real device in production. This operation can be done simply by modifying the IP address of the device.

6.1.1 KAS Runtime Log Window

The KAS Runtime Log window provides a running display of activity related to the execution of the application. Items displayed include application startup and initialization information.

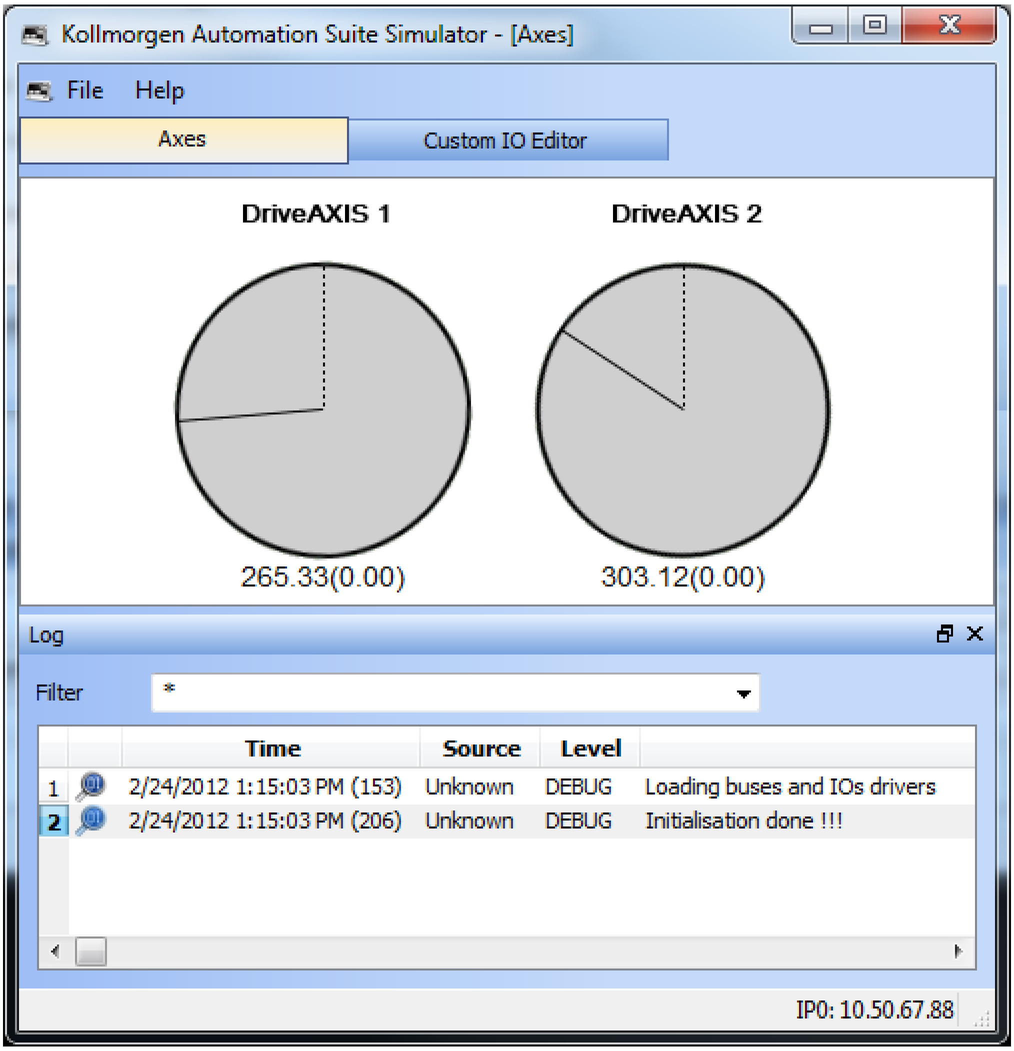


Figure 5-2: KAS Runtime Log Window

NOTE Some of the steps performed during the initialization process can be specified in the **initscript.bin** file located under:
C:\Program Files\Kollmorgen\Kollmorgen Automation Suite\Sinope Simulator\Resources

See also chapter "KAS Simulator log window" on page 352

6.2 Axes Tab

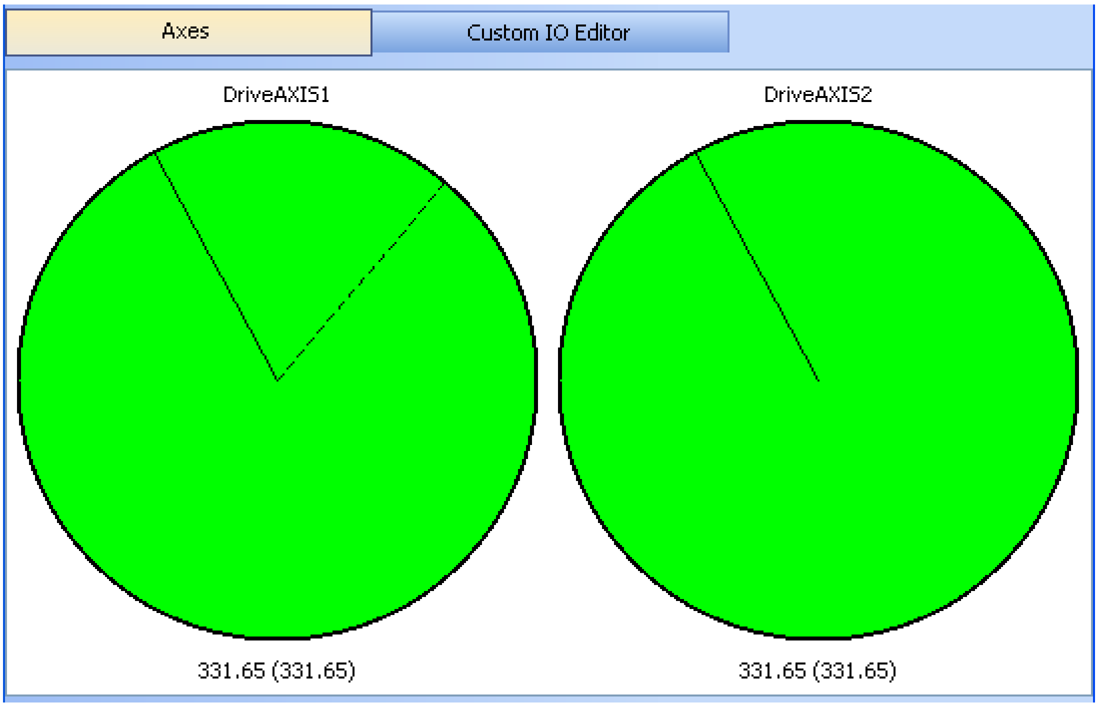


Figure 5-3: Axes Tab

The solid line (or normal line) represents the chapter "Reference Position" on page 101 in User units.
When the dashed line (or dotted line) is visible, it represents the chapter "Actual Position" on page 102 in User units.

Below the disk, the reference position for the associated axis is represented in the following format:

```
Range value (Modulo value according to the periodicity)
```

As shown on the figure below, the **Error** command (in the contextual menu of the axis tab) is used to simulate an error on an axis (then you can see the impact on the HMI and implement counter-measures if necessary).

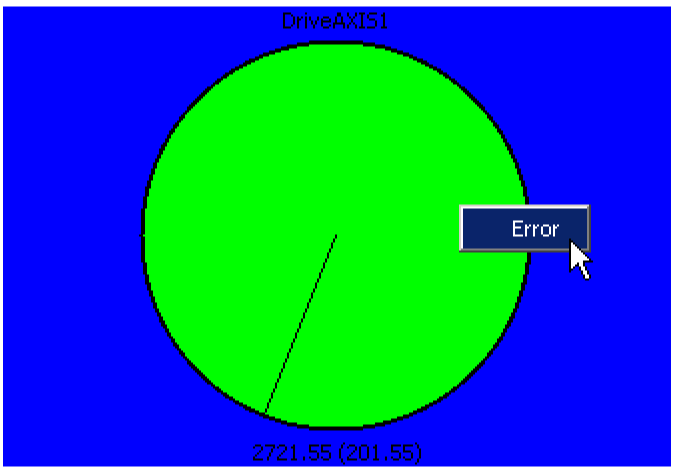


Figure 5-4: Set Axis in **Error** Mode

The drive becomes Red when it is set to **Error** (see also the figure showing the chapter "Design Motion with Pipe Network" on page 279)

To deselect an axis already selected (blue rectangle), click on the white surrounded outside border of the axis tab.

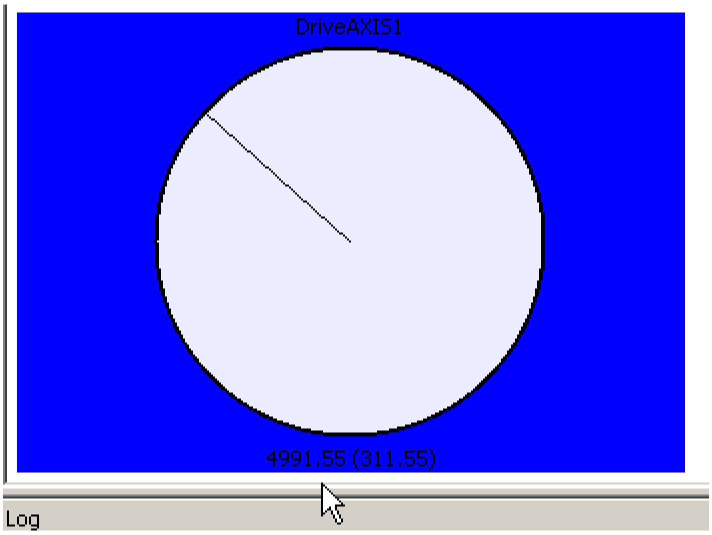


Figure 5-5: Deselect an Axis

6.3 Custom IO Editor

NOTE This tab present in the KAS Simulator is reserved for Profibus fieldbuses only.

Each I/O is displayed based on a tree-structure representation. The structure is the counterpart of the formatting used in the KAS IDE to define I/Os address within the I/O editor (see chapter "Modify Input/Output" on page 456).

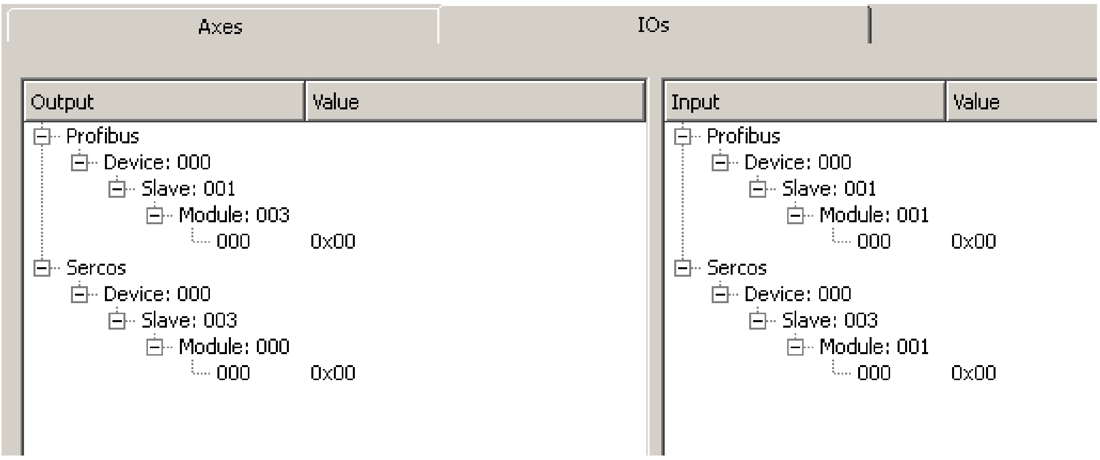


Figure 5-6: I/Os Displayed in Object Tree

I/O value can be displayed according the following formats:

- Byte
- Unsigned Short Integer
- Short Integer

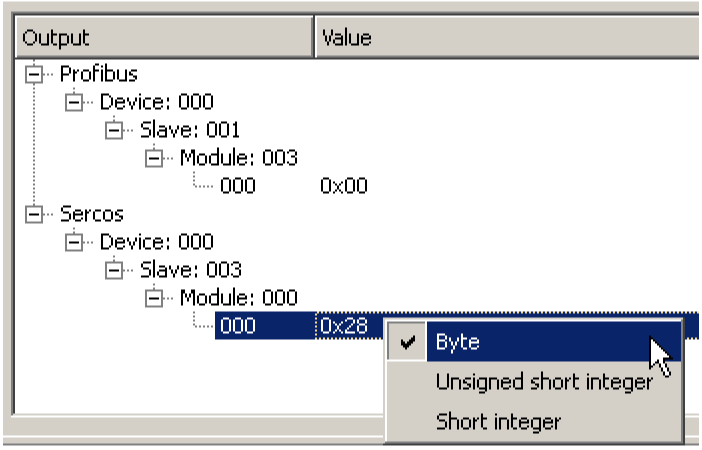


Figure 5-7: I/Os Value

See also chapter "Custom Input/Output Editor" on page 456

6.4 Describing KAS Simulator Graphical User Interface

6.4.1 Windows Overview

6.4.1.1 Main window

KAS Simulator main window contains:

- The menu bar (see call out 1)
- The workspace 2
- The Log window 3

In addition, the workspace contains two tabs to display the **Axis** and the **I/Os**.

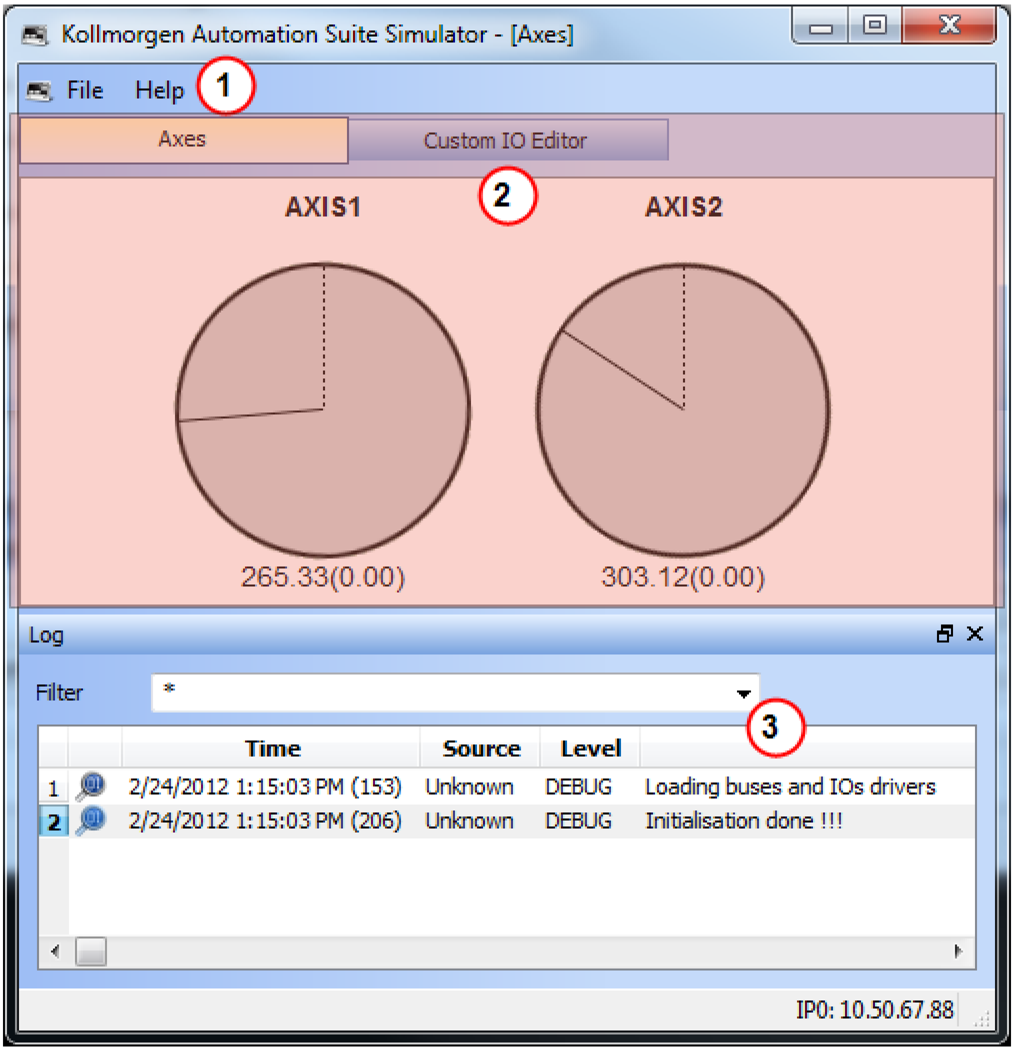


Figure 5-8: KAS Simulator Main Window

6.4.1.2 KAS Simulator log window

This Log window shows all log messages related to the KAS Simulator. Error and warning messages issued from the operating system, as well as chapter "Printf Function" on page 332 instructions, are also placed on this window.

Every log message includes the following:

- Timestamp
- ID
- Message

6.4.2 KAS Simulator Menus Overview

6.4.2.1 File Menu

Command	Description
Start	Start the application with the "Retain Variables" (see page 79).
Cold Start	Start the application with the initial settings

Command	Description
Stop	Stop the application
Option	Set parameters for the KAS Simulator application (see explanations below)
Exit	Leave KAS Simulator application

Option

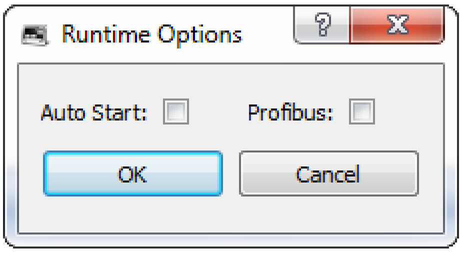


Figure 5-9: Options for KAS Simulator

Option	Description
Autostart	Autostart the application when KAS Simulator is launched **NOTE** You can choose to start the application manually when debugging with the Simulator. Whereas the Autostart mode is recommended when the system is in production, in order to prevent from doing inappropriate actions.
with Retain Variables	Autostart the application with the retain variables When selected, all the variables declared as retain are saved on the chapter "NVRAM" on page 675 before leaving the application. **NOTE** To correctly recover those variables when starting the application again, do not forget to have this check box enabled.
Profibus	When you have Profibus slave devices (e.g. WAGO I/O slices) in your system, you need to enable this flag to make the fieldbus active.

NOTE Parameters are saved in the **Options.bin** file located under:
<user>\AppData\Local\Kollmorgen\KAS\Sinope Simulator\Resources\

Options are slightly different for the IPC

When the KAS Runtime is downloaded on IPC, the Option window contains an additional drop-down menu (named **Main Bus Driver**) that lists all the fieldbuses predefined in KAS.

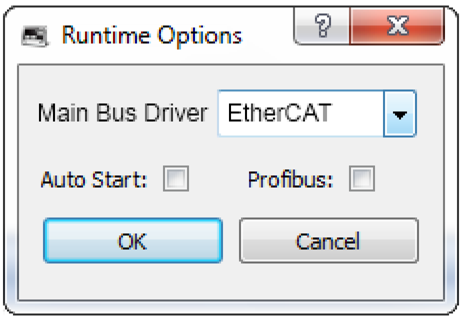


Figure 5-10: Options for KAS Runtime on IPC

From among the fieldbuses you can select the one used as the **main motion bus** (master bus) so that all the motion part is synchronized on its sampling rate frequency.

6.4.2.2 Help Menu

Command	Description
About	Show version numbers and other chapter "View Version Information" on page 183 about KAS Simulator

7 Using the AKD PDMM

Tasks related to the AKD PDMM are:

- On-line inspection to check actual parameters and diagnostic your system
- Configure parameters
- Start and stop your KAS application
- Update the firmware
- Reset to factory settings

Rebooting the AKD PDMM, recovering the firmware, and resetting the AKD PDMM may be performed from the device or, more conveniently, using the web server.

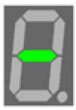
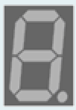
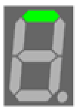
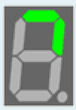
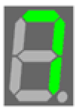
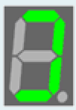
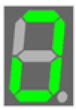
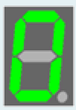
7.1 Booting the AKD PDMM

This topic explains the boot sequence for the AKD PDMM that is based on the RAM and the Flash memory.

The flash memory contains two images:

- Recovery image (4 Mb) contains QNX operating system and the KAS web server
- Regular image (9 Mb) contains QNX operating system, the KAS web server, and the KAS Runtime

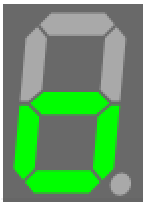
7.1.1 Boot Sequence

State	Display	Description
Hardware power on		Defines the range of channels you want to map automatically
Stage 0		Reached after the i2c is initialized
Stage 1		Reached after the DDR3 ram memory is initialized
Stage 2		Reached just after the RAM memory relocation At this point the boot is running in DDR3 RAM memory
Stage 3		Reached after the flash memory is initialized
"Boot Startup Script" (see page 357)		After all the previous steps, the startup script starts automatically.
QNX startup		Reached after the Boot startup script is finished
Sysinit		Reached after specific configuration parameters of the target are loaded, and after the network is started using the rotary switch

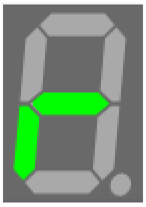
NOTE The AKD PDMM may be booted with or without a ethernet cable attached.

When the AKD PDMM is booted with a cable attached the configured IP address (depending upon the current position of the rotary switch) will be displayed in the 7-segment display (see "Display the PDMM's IP Address" (see page 359)). If the AKD PDMM is started without a network connection then the IP address will not be displayed.

After the boot sequence is successful, the AKD PDMM will be in one of two modes:



Normal operation

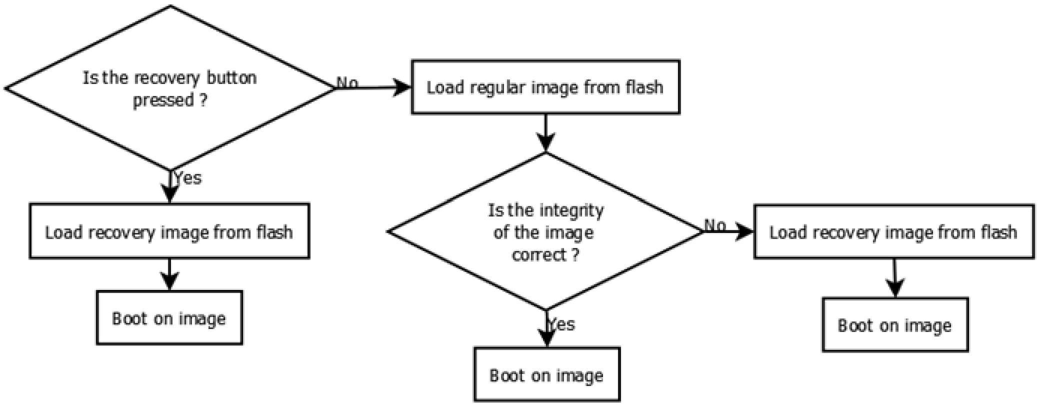


Recovery Mode (if firmware download is permitted)

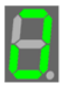
7.1.2 Boot Startup Script

After all the previous steps, the startup script starts automatically.The script first puts the 7-segment display into stage 4.

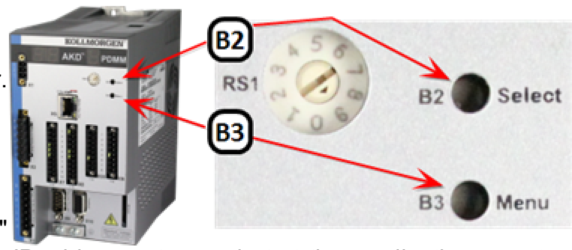
Before the AKD PDMM boots up, the following flowchart applies:



7.1.3 Booting from the Recovery Image

Automatic Mode	The boot from the recovery image is done automatically if the regular image is corrupted.
Manual Mode	If the AKD PDMM starts booting normally but freezes after the startup script (see image to the right), then you have to boot manually from the recovery image by pressing the recovery button (B2). See "About Recovery Mode" (see page 359) for more information. 

7.2 Working with the Hardware



In some cases, using the buttons on the AKD PDMM may be preferable to using the web server. On the front of the AKD PDMM there are two buttons, B2 and B3. B2 is above B3. These buttons may be used to enter Recovery Mode (see "About Recovery Mode" on page 359), display the PDMM's IP address, stop and start the application, reset the control to factory settings (see "Reset the Control to Factory Settings" on page 359), and backup/restore the firmware.

Result	Press and hold
B2	Recovery Mode
B3	Menu access

Table 6-1: B2/B3 button functionality at start-up

Press	Result
B2	"Display the PDMM's IP Address" (see page 359)
B3	Menu access

Table 6-2: B2/B3 button functionality while running

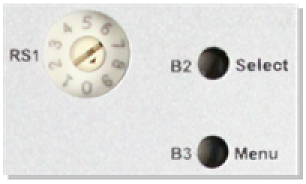
7.2.1 AKD PDMM Memory

The AKD PDMM is equipped with ample memory to handle the most challenging programs.

Memory Type	Amount	Purpose
Flash	64 MB	Non-volatile memory
SD Card slot	2+ GB	Backup and Restore functionality as well as moving data. See "SD Card Support" (see page 360) for more information.
DDR RAM	256 MB	everything else

7.2.2 PDMM B3 Button Menu

The B3 “Menu” pushbutton will cycle through a list of menu items displayed on the 7-segment LED. Each B3 press will advance to the next menu item. The menu item will be displayed for 10 seconds. If no button is pressed within the 10 seconds, the 7-segment display will return to Normal operation.



Pressing and holding the B3 button during the boot sequence (before the Boot Startup Script runs) provides access to a menu of functions.

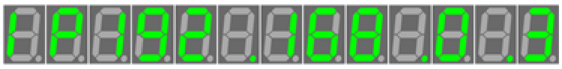
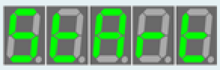
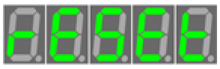
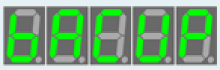
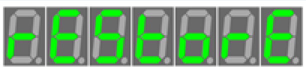
Functionality	Display	Notes
Display the IP		See "Display the PDMM's IP Address" (see page 359)
Start the application		This will start the KAS Runtime.
Factory Reset		See "Reset the Control to Factory Settings" (see page 359)
Backup firmware to SD card		See "Backup and Restore a PDMM" (see page 361).
Restore firmware from SD card		See "Backup and Restore a PDMM" (see page 361).

Table 6-3: Application is not running

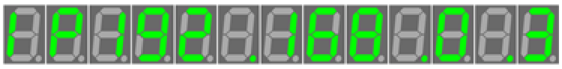
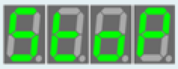
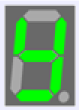
Functionality	Display	Notes
Display the IP		See "Display the PDMM's IP Address" (see page 359)
Stop the application		This will stop the KAS Runtime.

Table 6-4: Application is running

NOTE

Please note that when selected, the Start, Stop, Backup, Restore and Reset functions do not initiate immediately; they require confirmation. The 7-segment displays flashes a "y", prompting for confirmation. Pressing B2 confirms the function and the process begins. If the function is not confirmed within 10 seconds the action is canceled.



7.2.3 Display the PDMM's IP Address

The IP Address assigned to the AKD PDMM can be shown on the 7-segment display. The IP may be displayed at boot and can be accessed from the "PDMM B3 Button Menu" (see page 358). Note that there is a 5 second delay before this function may be used again.

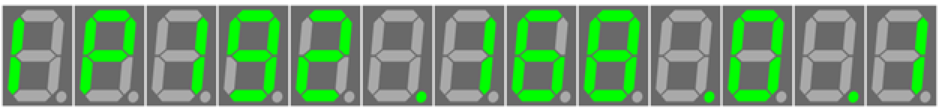
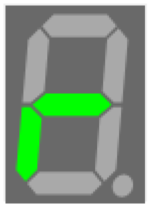


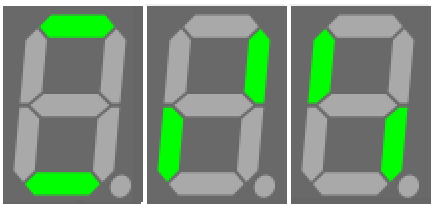
Figure 6-1: Example of the IP sequence by the 7-segment display.

7.2.4 About Recovery Mode



To enter recovery mode you must press and hold B2 during the boot sequence before the Boot Startup Script runs. If the system detects that the button is pressed then it will enter Recovery Mode. The 7-segment display will show a lower-case "r" as seen here.

While in Recovery Mode the AKD PDMM will download the firmware from the recovery image. When the firmware is being written to the flash drive the 7-segment display will animate as seen below. Do not power-off the system during this process.

 (repeats)

When the download is complete the AKD PDMM will go into normal operation. If the download or write to flash fails the 7-segment display will display a numeric error code.

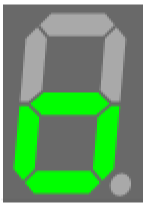
7.2.5 Reset the Control to Factory Settings

The AKD PDMM may be manually ordered to perform a factory reset. The reset is performed using either of two methods:

- Selecting the function from the "PDMM B3 Button Menu" (see page 358). This can be done during the boot sequence or while the drive is running.
- From the "File System Tab" (see page 386) of the web server while the drive is running. This method is recommended due to its ease of use.

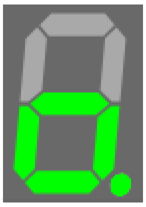
7.2.5.1 Resetting while the drive is running

This may be done any time after the control is powered on and the program is not running. Please note that the reset will be ignored if an application is running on the control.



Normal Operation

Press the B3 button to access the “reset” item in the menu, then press B2 to select “reset” to factory defaults.



Program Running

Reset to factory defaults is not permitted. The “reset” menu item is not available.ss

7.2.6 About the reset

After two seconds have expired (or longer if pressed during power-up), the 7-segment display on the control will change to an animation pattern indicating that the factory reset has started.

The following changes occur during a factory reset:

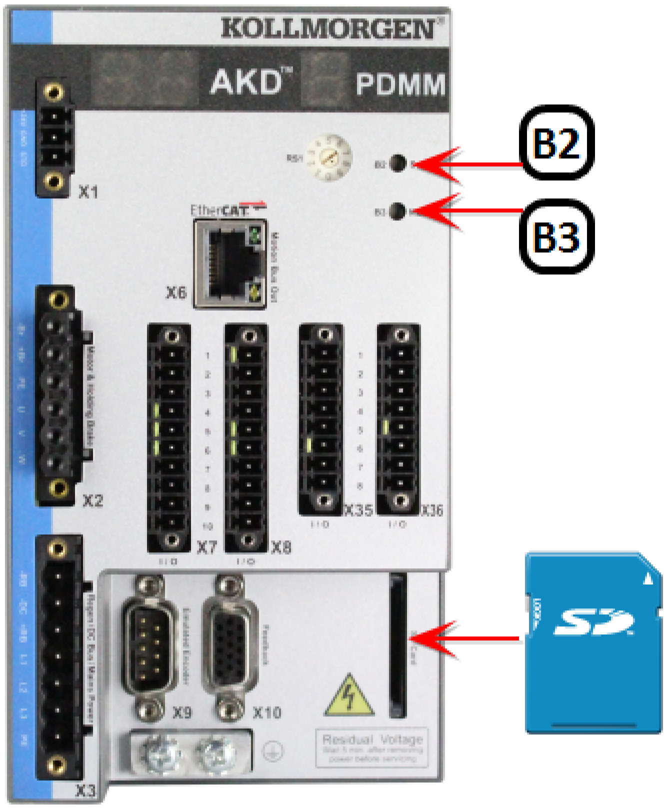
- Reset any application previously download
- Reset IP address, Subnet and Gateway settings
- Reset retained variables
- Reset Auto-Start option

Some important facts to remember:

- Factory reset cannot be performed while an application is running.
- If the control has just been powered up, the B3 button will have to be held down much longer than 2 seconds. In this case, hold down the button until the 7-segment display shows the "PDMM B3 Button Menu" (see page 358).
- Factory reset will take about 4-5 minutes to complete and the 7-segment display on the control will animate during this process. The control should not be turned off during this procedure.
- After the factory reset is complete, the control will be powered down and restarted automatically.

7.2.7 SD Card Support

The PDMM supports using an SD card for backup and restore functionality. This lets you manage the PDMM configuration, application and operation data. The PDMM has a SD card slot and push buttons (B2 and B3) which activate file transfers to and from a SD card.



Using the SD card provides an easy way to

- backup and restore a PDMM configuration
- store and retrieve an application, including source code
- store and retrieve user data from an application or PC

7.2.7.1 Supported SD Card Formats

Format	File System	Capacity
SD (SDSC) 	FAT16‡ or FAT32	2GB
SDHC 	FAT32‡	4-32GB

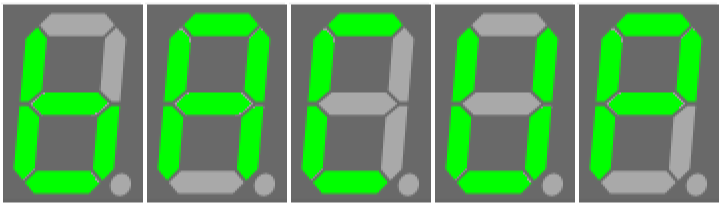
‡ The default file system for the format.

7.2.8 Backup and Restore a PDMM

A mounted SD card can be used to store files, such as a copy of the PDMM's firmware. The Backup and Restore functions may be accessed from the webserver or from the "PDMM B3 Button Menu" (see page 358). Access from the webserver is discussed in the "SD Card Tab" (see page 387) section.

7.2.8.1 Backup

The Backup function will store a copy of the PDMM's data on a SD card. This function is displayed on the 7-segment display as shown here ("bACUP"). Pressing B2 selects the function. This function does not initiate automatically, B2 must be pressed again to confirm the process.



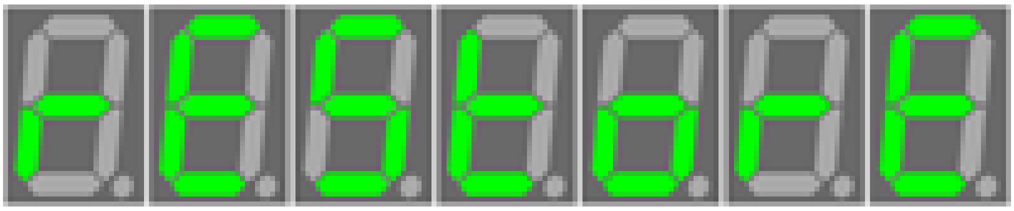
The data that is backed up and copied to the SD card includes:

- PDMM firmware
- Application (including ECAT XML configuration, cam tables, etc.)
- "Retain Variables" (see page 79)
- PDMM configurations (auto-start and IP address)
- Designated user data files

NOTE Log files are not copied to the SD card.

7.2.8.2 Restore

The Restore function will restore and load files onto the PDMM from an SD card. This function is displayed on the 7-segment display as shown here. Pressing B2 selects the function. This function does not initiate automatically. The 7-segment displays flashes a "y", prompting for confirmation. Pressing B2 again confirms the function and the data transfer begins. If the function is not confirmed within 10 seconds the action is canceled.

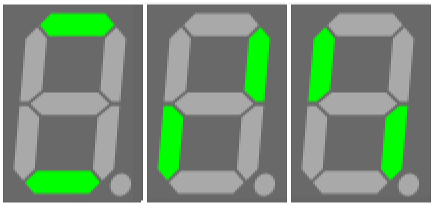


The Restore process will:

- Load PDMM firmware into on-board flash, if version is different
- Load AKD firmware into each drive, replicating the firmware versions for each drive.
- Load AKD parameters into all drives
- AKD unique IDs
- Load PDMM configurations (auto-start and IP address)
- Load "Retain Variables" (see page 79)
- Load user data files
- Re-start KAS runtime using restored firmware

7.2.8.3 About the data transfer

- The 7-segment display will show the chasing lights animation while the backup or restore is occurring.


<repeat>

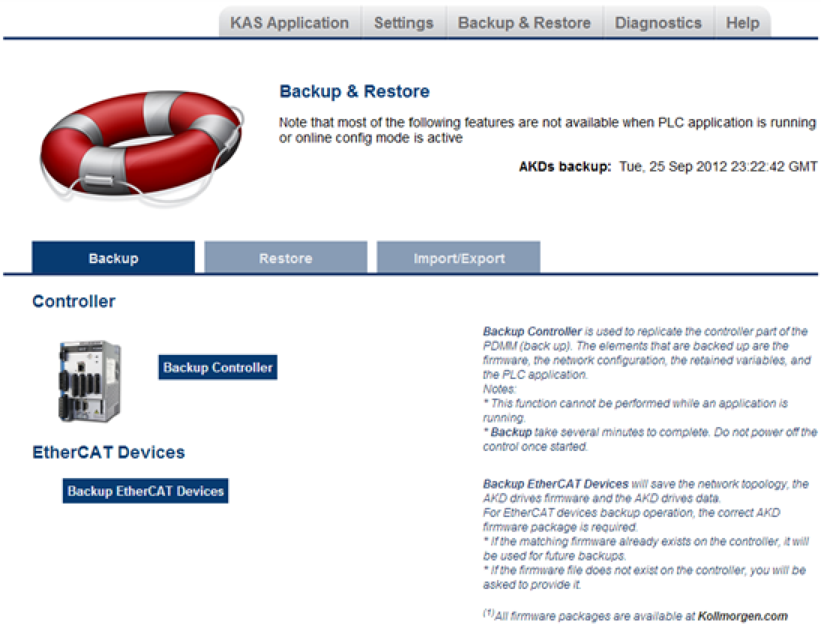
- The Backup and Restore functions have an "all or nothing" behavior. If there is no SD card inserted, if there is not enough space on the card or if files are missing then nothing will be copied and the 7-segment display will show an error.
- If files already exist on the SD card (in the backup directory), then they will be deleted and replaced with the new PDMM backup configuration files. Likewise, the files on the PDMM will be replaced with the SD files.

NOTE **Warning!** Do not modify the files on the SD card as this could result in the Restore function failing.

TIP If you have multiple PDMM backup configurations, you will need to use one SD card per backup configuration.

7.2.9 EtherCAT Devices Backup and Restore

The PAC and AKD PDMM can backup/restore EtherCAT devices (at present, only AKD drives) on an EtherCAT network. This feature is useful as a maintenance operation to replace any AKD drives in an operational machine. This feature reduces the manual steps for saving/loading each AKD drive's firmware and parameters into a few simple automated steps. The Backup/Restore functionality is located in the PAC and AKD PDMM web server and is accessible from a web browser. For details about the web server see "Using the KAS Web Server" (see page 375).



See also "Backup and Restore a PDMM" (see page 361).

7.2.9.1 EtherCAT Devices Backup

The Backup operation discovers all the devices on the EtherCAT network and stores the topology information, AKD firmware files, and AKD parameters to the controller's local storage.

Controller	Local Storage
PAC	Compact Flash Card
AKD PDMM	SD Memory Card (see "SD Card Support" (see page 360) for more information)

Before starting a backup of the EtherCAT Devices, you will need:

- The AKD firmware files package, `AKD-Firmware-for-KAS-Vxx-xx-xx-xxx.tgz`. This package is included in the KAS software installation directory (`\Program Files (x86)\Kollmorgen\Kollmorgen Automation Suite\Astrolabe\DrivesFW`) or is available for download from the Kollmorgen website (www.kollmorgen.com/en-us/website-resources/other/akd-software/).
- An SD memory card must be in the SD slot if you are using an AKD PDMM. PACs have a built-in Compact Flash card.
- All the EtherCAT network devices must be connected to the PAC or AKD PDMM, and configured as necessary for machine operation.

TIP	Make sure the devices are in the order as expected. The topology information is stored in the backup and is used to check for identical topology during a restore function.

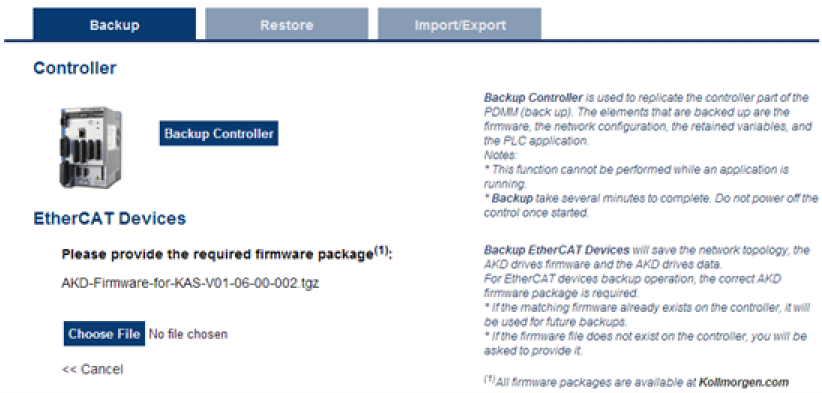
- A PLC application downloaded to the PAC or AKD PDMM, containing the EtherCAT device map.

NOTE	The PLC application cannot be running and the IDE must not be in "Online Configuration Mode" (see page 660). Please stop your PLC application or disable Online Configuration Mode before a Backup or Restore.

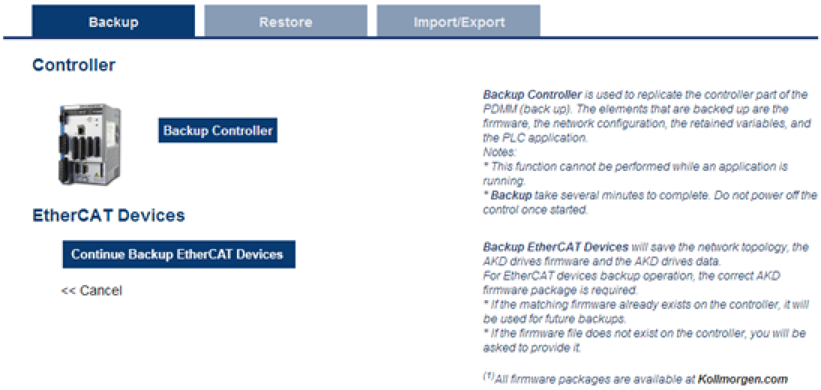
Steps

1. From the web server home page, click on the Backup & Restore tab:
2. Under the Backup tab, press the Backup EtherCAT Devices button.

   The web server will ask you to choose a firmware package file and suggest the filename that matches the firmware version on your drives. Selecting the firmware package file is a one-time event. The controller will remember your selection and will not ask you again for future backups.



3. Press the Continue Backup EtherCAT Devices button. It will take a couple of minutes or longer, depending on the number of AKDs in the system.



When the backup is complete, the web server will indicate whether the backup was successful.

TIP After the backup is complete, it is a good idea to export the backup to an off-site location for safe keeping. See Export/Import below.

7.2.9.2 EtherCAT Devices Restore

The Restore operation discovers the devices on the EtherCAT network and compares the physical topology information to the topology information stored in the backup. A Restore will detect the replaced AKD devices and restore them. Advanced users can manually select the specific AKD devices and restore them. The backup files containing the topology information, AKD firmware file, and AKD parameters are retrieved from the controller's local storage.

Controller	Local Storage
PAC	Compact Flash Card
AKD PDMM	SD Memory Card (see "SD Card Support" (see page 360) for more information)

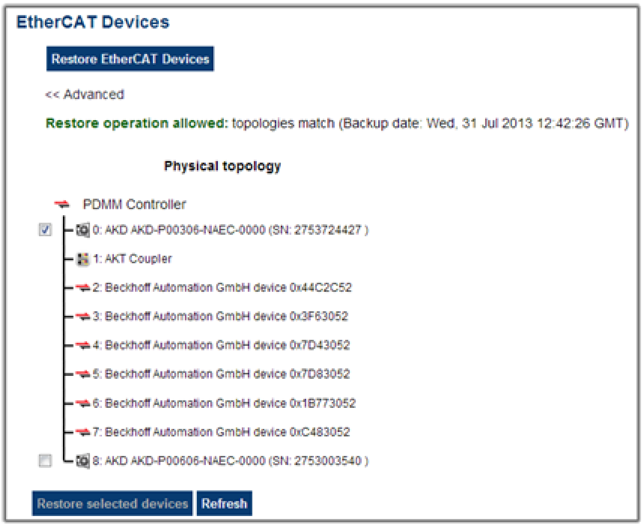
Steps

IMPORTANT The application will start immediately after the Restore operation is complete if the Autostart option is enabled on the controller. Be sure to Restore all of the replaced devices and the EtherCAT device order is correct. *Disable the Autostart option before commanding the Restore operation if you want to check the devices before starting the application.*

NOTE The EtherCAT Devices Backup and Restore feature may be used in EtherCAT networks which have third party devices, but only the Kollmorgen AKDs can be backup and restored. Specific configurations applied to third party devices with non-Kollmorgen tools have to be reapplied when the third party device is replaced.

1. From the web server home page, click on the *Backup & Restore* tab:
2. Under the *Restore* tab, you can choose to restore the replaced EtherCAT AKD devices or manually select the AKD devices with the Advanced view.
   - To restore the replaced AKD devices, press the *Restore EtherCAT Devices* button. When the restore is complete, the web server will indicate whether it was successful.
   - To select the AKD devices and manually restore:
     1. Press the *Advanced* link.
     2. Select the AKD devices you want to restore. The controller will identify the replaced AKDs and pre-select them for you.

3. Press *Restore selected devices*.

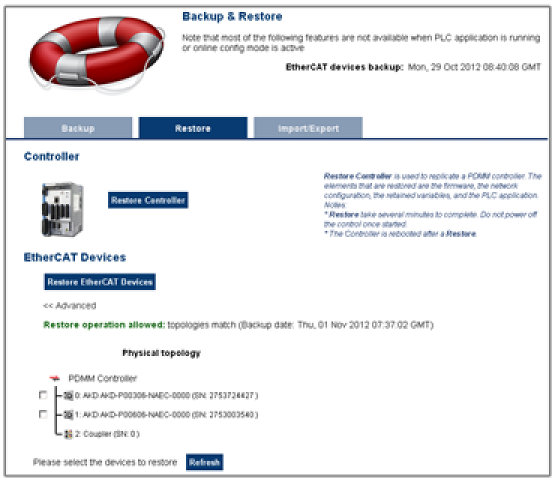


When the restore is complete, the web server will indicate whether it was successful.

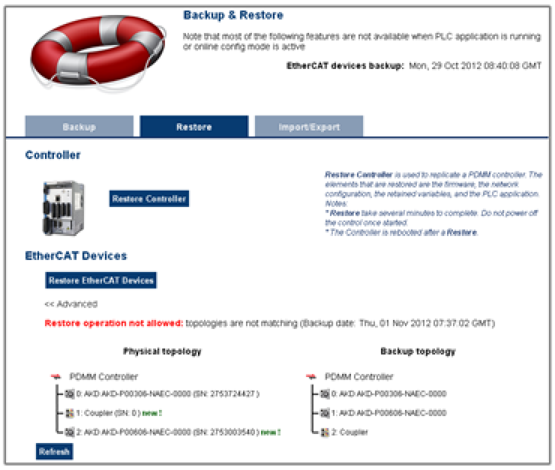
7.2.9.3 AKD Backup/Restore Compatibility

The replacement AKD must have the same model number as the AKD that was originally stored in the backup. The Restore operation compares the model numbers between the backup and the replacement AKD. The restore will not be allowed if they are not compatible.

- To check the model number on your AKD, see the sticker on the side of the drive.
- To check compatibility between your replacement drive and the backup, replace the AKD on the EtherCAT network, and press the Scan network button.
  - The web server displays the physical topology and allows you to restore the selected drives if the backup and replacement drives are compatible.



  - The web server displays the backup and physical topology and indicates the non-compatible drives if the backup and replacement drive are not compatible, .



TIP The serial number (SN) is displayed in the Physical topology web server view and on the sticker affixed to the AKD. You can use the serial number to match the actual hardware with its representation on the web server.

7.2.9.4 Export/Import EtherCAT Devices Backup

A network backup may be exported and imported. The export procedure saves a backup file to the computer running the web browser. The import procedure allows you to transfer a backup file onto the controller to be used later for restoring a previous configuration.

Export Procedure

NOTE The Export button is only displayed if an AKD backup is available.

1. Click on the "Export Backup" button. The browser starts transferring a backup file. Depending upon the browser being used, this may involve a prompt confirming that you wish to receive the file.
2. You may move the file to a different directory once the file is saved. The file may be renamed to help identify the backup file with the machine.

Import Procedure

1. Specify a backup file to import by clicking on the "Browse" button. This backup file will be used to replace the current backup on the controller.
2. After a backup file is specified, click on the "Replace Backup" button. This creates a backup on the controller with the data stored in the specified backup file. Any previously existing backup will be replaced. If the import fails, the previous backup will not be replaced.

NOTE
- The Replace Backup button is disabled until a backup file has been selected.
- On some browsers, the "Browse" button may be labeled "Choose File".

7.2.9.5 EtherCAT Devices Backup/Restore Limitations

- The "EtherCAT Devices Backup" (see page 363) and "EtherCAT Devices Restore" (see page 365) functions are not permitted while a PLC application is running or when the IDE is in "Online Configuration Mode" (see page 660). Please stop your

PLC application or disable Online Configuration Mode before a Backup or Restore.

- Only a single backup is supported in the controller at one time.

NOTE If you want to keep multiple backup configurations, you can perform the backup and then export it to a local PC or USB flash memory stick. Later, when you are ready to restore, you can import the specific backup file to the controller, and then perform the restore.

TIP Beware that an import will overwrite any backup existing in the controller or AKD PDMM SD card. If you are using an AKD PDMM, the alternative method is to use a separate SD card for each backup/restore configuration.

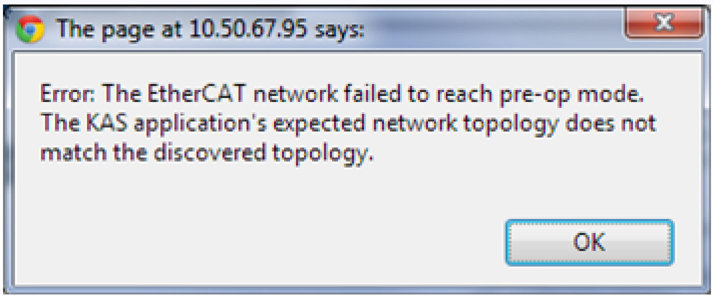
- All AKD drives on the EtherCAT network must have the same AKD firmware version.
- The AKD firmware version must be 01-06-00-003 or higher.
- AKD firmware packages are available for all production releases.
- Only AKD drives are supported for backup/restore. Kollmorgen S300 drives are not supported by backup/restore.
- AKD PDMM system backup or restore is a two-step process:
  1. Backup or Restore the AKD PDMM
  2. Backup or Restore the AKDs (including the AKD drive inside the AKD PDMM).

7.2.9.6 Troubleshooting EtherCAT Devices Backup/Restore

The web server displays an error message if an EtherCAT Device backup or restore fails. The message describes the cause of the failure and a possible remedy. Please be sure to note any error message(s), as they will be helpful with remedying the problem.

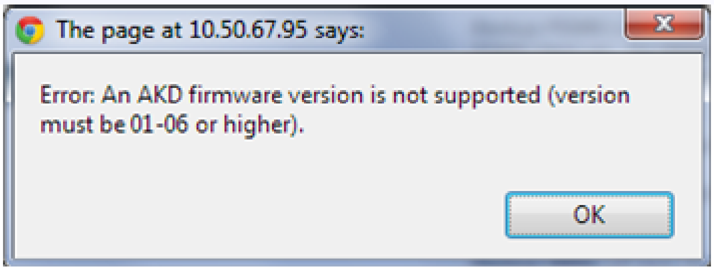
Described below are some common error messages and remedies. The message box format may appear differently depending on the web browser, but the message content is the same.

During AKD backup, EtherCAT fails to reach pre-op mode:



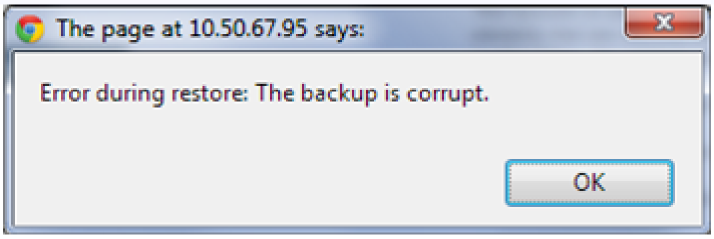
1. Using the IDE, open the EtherCAT view and scan the network.
2. Compare the nodes, their order, and types to the topology in your application.
3. After you identify the differences do one of the following:
   - Modify the application’s devices to match the physical network.
   - Correct the physical network by adding/moving/removing nodes.

During AKD backup, at least one AKD has an unsupported firmware error for backup:



1. Using the IDE, open the EtherCAT view.
2. Upgrade all of the AKDs drive firmware to at least version 01-06.

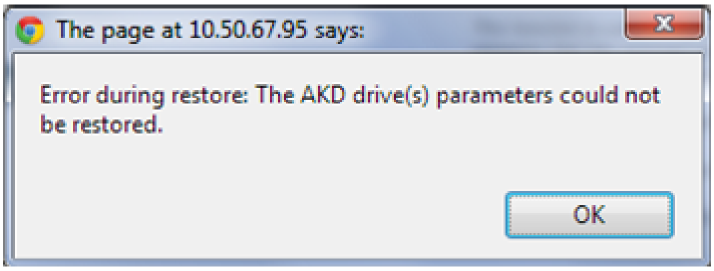
During AKD restore, the backup file is corrupt:



Before restoring AKD(s), the backup file must be valid. To correct a bad backup file on a controller, import a known good backup file from another source (local hard drive, network drive, USB flash stick, etc.). If you do not have a valid backup file, then you will need to manually configure the replacement AKDs by downloading firmware and modifying the drive's parameters using the AKD views in the IDE.

Not able to restore AKDs successfully.

The error message will describe at which step the restore failed. For example, failing to restore parameters:



1. If an AKD restore fails and you have already verified the controller has a valid backup and the network topology is correct, then retry the Restore.
2. If you still cannot restore successfully after two or three attempts, check your network cables and try a different replacement AKD drive(s). This test will isolate the problem to the specific drive(s) or the controller problem.
3. If you still cannot restore a replacement AKD, then you will need to manually configure the replacement AKDs by downloading firmware and modifying the drive's parameters using the AKD views in the IDE.

7.2.10 Configure AKD PDMM Onboard I/O

The procedure to define the local I/Os of the AKD PDMM drive is very similar to the one for I/O slices, with the following exceptions:

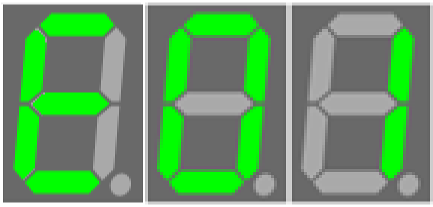
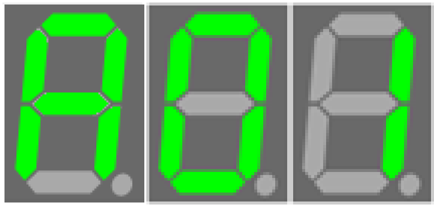
- AKD PDMM Onboard digital IO is updated synchronously with the EtherCAT update rate.
- AKD PDMM Onboard digital IO is limited to a 1kHz update rate.

For more details, refer to "Step 11 of 15 - Map Input and Output to Variables" (see page 270)

7.2.11 About Errors and Alarms

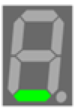
The AKD PDMM continuously displays any error or alarm codes after booting, and not in recovery mode.

Only one error or alarm code will be displayed at a time. Errors have a priority over Alarms and the code with the highest priority will be displayed until it is cleared.

Error code example	Alarm code example
	

7.2.12 Errors

When an Error or Alarm occurs, always check the controller log messages. The log messages will provide more details about the failure and the history of events leading up to the failure. From the log messages, you can determine the specifics about the cause of the failure to correct the underlying problem.

Code	Description	Cause	Remedy	Clear ‡
E01	Critical temperature exceeded. AKD PDMM operation is stopped after 20 seconds, CPU will be put to sleep. 	CPU temperature exceeded safe operating temperature limit.	Power-off. Check airflow and operating environment are within hardware specifications. Allow unit to cool before power-on.	HW
E02	Out of memory. KAS runtime is stopping.	Memory leak, memory corrupted, or hardware memory failure.	Power-off/on. If problem is recurrent, check release notes for firmware updates or return hardware for repair.	HW
E03	Fan failure.	CPU cooling fan was not able to operate properly.	Check temperature and monitor for High temp alarm (see A01). Return hardware for fan replacement.	HW

Code	Description	Cause	Remedy	Clear ‡
E10	Firmware is corrupted.	Flash memory corrupted during firmware download or flash hardware failure.	Re-download firmware or boot into recovery mode, download firmware, and power-off/on. If problem persists, return hardware for repair.	SW
E11	Flash is corrupted, no filesystem is available.	At startup the filesystem could not be mounted on the flash.	Reset to factory defaults. If problem persists, return hardware for repair.	SW
E12	Not enough flash memory available.	Flash memory is full, unable to write to flash.	Clean-up the flash memory by removing log files, application programs, recipes, or other data files.	SW
E13	Out of NVRAM space for retained variables.	NVRAM is full.	Change application to reduce the amount of retained variables.	SW
E14	Reset to Factory Defaults failed.	Flash memory could not be formatted during a Reset to Factory Defaults procedure.	Try reset to factory defaults again from power-on. If problem persists, return hardware for repair.	SW
E15	Cannot read/write files from/to a SD card	SD card is not plugged in or the file system is corrupt and cannot be mounted. PLC function failures will not cause this error.	Insert a valid SD card or reformat the SD card using Settings > SD Card > Format button.	SW
E16	Not enough space available on the SD card	SD card is full, unable to write to the SD card. PLC function failures.	Clean-up the SD card space by deleting files or re-format the card using Settings > SD Card > Format button.	SW
E20	Runtime plug-in, process, thread or application failed to start.	KAS runtime or application code failed to autostart at boot.	Power-off/on. Reset to factory defaults. If problem is recurrent, check release notes for firmware updates or download firmware.	HW
E21	Runtime process, thread, or driver failed to respond during operation.	KAS runtime code failed during normal operation.	Power-off/on. If problem is recurrent, check release notes for firmware updates.	HW
E22	Fatal error in PLC program, application stopped.	Virtual machine failed to execute an instruction.	Re-compile application, download, and re-start. Check the IDE and controller firmware versions are compatible.	SW
E23	CPU is overloaded. See "CPU Overload (E23)" (see page 374).	Either the motion engine did not complete or the PLC program did not complete within the timeout period due to excessive CPU load.	Stop the application or power-off/on. Reduce the sample rate, simplify the application, or reduce the application cycles and restart the application.	SW

Code	Description	Cause	Remedy	Clear ‡
E24	PLC application cannot be started	PLC application cannot be started, due to an existing condition. Possible reasons: 1. Maintenance operation is in progress. 2. Controller is in online config mode. 3. AKD Restore failed. 4. The IDE version of the compiled PLC code and controller runtime version do not match. 5. Previous download failed.	Check the following: 1. Controller web-server home page for any maintenance operation in-progress. Wait for the operation to finish. 2. Connect to the controller with the IDE and disable online config mode. 3. EtherCAT network topology by using the Scan network button in the web-server's Restore tab. Correct the physical topology and re-execute an AKD restore. 4. IDE version (only major.minor.micro) should match with runtime version. To correct, install the correct version of IDE or Runtime. 5. Connect IDE and download application.	SW
E30	EtherCAT communication failure during operational mode.	EtherCAT network operation failed due to a network communciation error.	Check the EtherCAT network wiring and devices state. Re-start the application.	SW
E31	EtherCAT communication failure during preop mode.	EtherCAT network operation failed due to a network communciation error.	Check the EtherCAT network wiring and devices state. Re-start the application.	SW
E32	EtherCAT communication failure during bootstrap mode.	EtherCAT network operation failed due to a network communciation error.	Check the EtherCAT network wiring and devices state. Re-start the application.	SW
E33	EtherCAT failed to initialize into operational mode.	EtherCAT network initialization failed due to a network communciation error.	Check the EtherCAT network wiring and devices state. Re-start the application.	SW
E34	EtherCAT failed to initialize into preop mode.	EtherCAT network initialization failed due to a network communciation error.	Check the EtherCAT network wiring and devices state. Re-start the application.	SW
E35	EtherCAT failed to initialize into bootstrap mode.	EtherCAT network initialization failed due to a network communciation error.	Check the EtherCAT network wiring and devices state. Re-start the application.	SW
E36	EtherCAT failed to discover the expected devices.	EtherCAT network discovery failed due to a mismatch between the discovered and expected devices.	Check the EtherCAT devices and wiring order. Correct the device order wiring or re-scan the network, re-compile, and download the updated application. Re-start the application.	SW

Code	Description	Cause	Remedy	Clear ‡
E37	EtherCAT failed to return to init state.	EtherCAT network initialization failed due to a network communciation error.	Check the EtherCAT network wiring and devices state. Re-start the application.	SW
E50	Backup to SD card failed	An unrecoverable error occurred during the backup operation.	Repeat the backup to SD card operation. If it fails again, replace the SD card.	SW
E51	Restore from SD card failed	An unrecoverable error occurred during the restore operation.	**Do not reboot the PDMM!** Repeat the restore operation. If it fails again, reset the PDMM to factory defaults. If the problem persists, return hardware for repair.	SW
E52	SD Backup files are missing or corrupt	The restore operation failed due to missing, incomplete, or corrupt files on the SD card.	Perform a backup operation before the restore or use and SD card with valid backup files.	SW

‡ Items labeled "SW" can be cleared from the web server. Items labeled "HW" require a reboot to be cleared.

7.2.13 Alarms

Code	Description	Cause	Remedy	Clear ‡
A01	High temperature exceeded	CPU temperature near the safe operating temperature limit.	Check airflow and operating environment are within hardware specifications.	SW
A02	Low on memory.	Memory leak or corruption.	Power-off/on. If problem is recurrent, check release notes for firmware updates or return hardware for repair.	SW
A04	Low input voltage	+24 volt input power is +19 volts or less.	Check power supply voltage and connection to the AKD PDMM.	SW
A12	Flash memory is low on free space.	Flash memory is almost full.	Clean-up the flash memory by removing log files, application programs, recipes, or other data files. Reset to factory defaults.	SW
A21	Recoverable process or thread failed to respond during operation.	KAS non-runtime code failed during normal operation and was automatically restarted.	If problem is recurrent, power-off/on. Check release notes for firmware updates.	SW
A23	CPU is heavily loaded	CPU usage is too high for 5 (or more) seconds.	Reduce the sample rate, simplify the application, or reduce the application cycles.	SW
A30	EtherCAT missed a send frame during operation mode.	EtherCAT master was unable to send a frame for one or more cycles.	Reduce the controller CPU load, so it has enough Real-Time margin to send EtherCAT frames every cycle.	SW
A38	EtherCAT missed a receive frame during operation mode.	EtherCAT master did not receive, or received too late, a frame for one or more cycles.	Check the EtherCAT network wiring and devices, or decrease the EtherCAT cycle rate.	SW

Code	Description	Cause	Remedy	Clear ‡
A40	Local digital IO missed a cyclic update	Local digital IO was not updated during a cycle or the updates are no longer synchronous.	Reduce the sample rate, simplify the application, or reduce the application cycles.	SW

‡ Items labeled "SW" can be cleared from the web server. Items labeled "HW" require a reboot to be cleared.

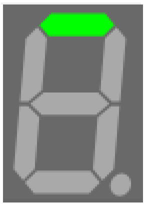
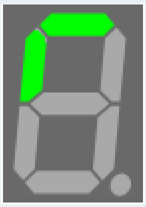
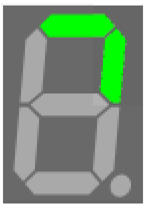
7.2.13.1 CPU Overload (E23)

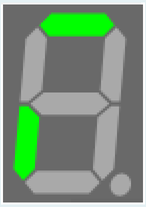
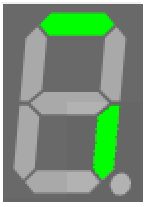
If the Motion Engine or PLC program execution (VM) do not complete a full cycle within their respective timeout periods, an E23 error will be flashed on the 7-segment display.

Process	Timeout
Motion Engine	200 milliseconds
PLC Program (VM)	10 seconds

The Real-Time operation for EtherCAT and the Motion Engine have the highest priority in the controller. The PLC Program (VM) has the second highest priority in the controller. These processes will continue to execute, even if their timeout values are exceeded.

If the CPU overload is severe, there may not be enough CPU time to execute the background operations. The background operations include the 7-Segment display update, monitoring push-buttons, web-server, Modbus, and communications with the KAS IDE. The 7-Segment will indicate a CPU overload or frozen software task by displaying one of the following patterns:

Display	Meaning
	CPU overload is extreme.
	Motion thread is not longer able to execute.
	EtherCAT Rx thread is no longer able to execute.

Display	Meaning
	PLC thread is no longer able to execute.
	Non-realtime threads are no longer able to execute.

To recover from an E23, stop the application from the IDE or web-browser (KAS Application view). If the CPU overload is severe, the controller may not have enough CPU time to respond to the IDE or web-browser. In this case, you will need to power-off/on the controller. If the PDMM is configured for Auto-start, press and hold the B3 menu button at boot-time to prevent the application from automatically re-starting. Then, you will be able to connect to the PDMM with the IDE.

7.3 Using the Web Server

7.3.1 Using the KAS Web Server

Kollmorgen Automation Suite™ comes with a web server that allows you to perform the following operations:

- Read information about the controller (model type, firmware version, version of your KAS application)
- Interact with your application (Start and Stop your KAS application)
- View real and simulated axes
- See all the log messages
- Upgrade the controller firmware
- Change the IP address
- View system diagnostics including storage space, memory and CPU temperature
- Reset the controller to factory settings

The web server may be accessed two ways:

1. Open a web browser and enter the controller's IP address.
2. From the Controller node in the Project tree in the KAS IDE.
   - Double-click the Controller node
   - Select "Access Webserver" from the right-mouse menu.

NOTE If you do not know the IP address assigned to the AKD PDMM, press and briefly hold B2, the 7-segment display will show the IP.

The web server consists of the home page, and the KAS Application, Settings, Backup & Restore, Diagnostics and Help tabs. The Help tab is a link which opens the KAS PDMM Web Server manual.

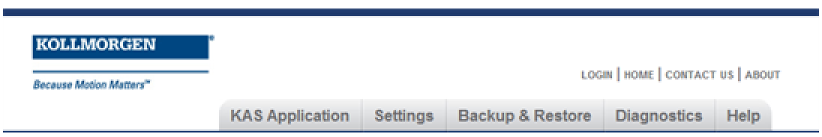
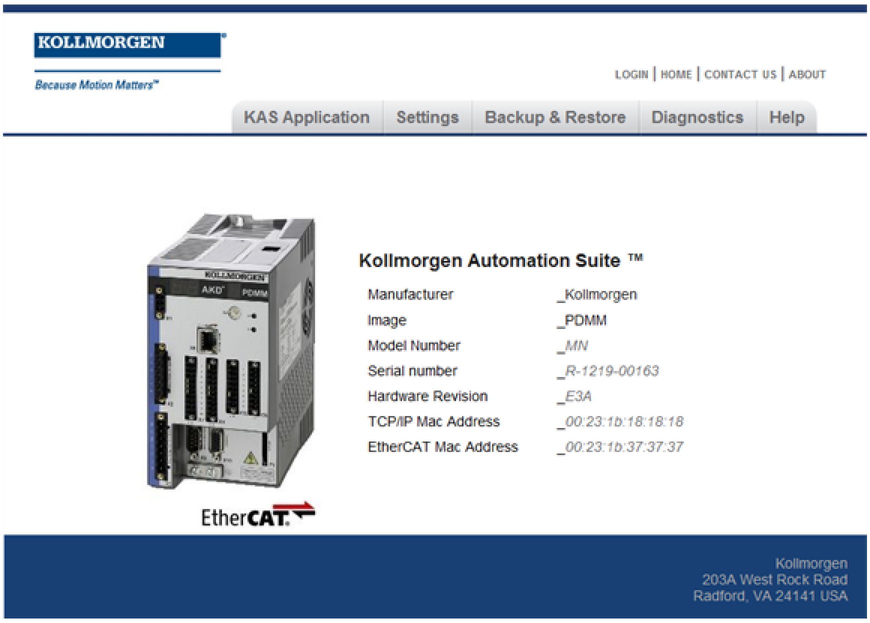


Figure 6-2: The Webserver Tabs as seen on an AKD PDMM webserver.

> **TIP**
> **Browser Requirements:** We recommend using Firefox 11, Google Chrome, or Internet Explorer 9 or later for accessing the web server.

7.3.1.1 Web Server Home Page

To access the KAS web server home page, enter the controller's IP address.



This page provides an overview of the device including:

- Manufacturer
- Image
- Model Number
- Serial Number
- Hardware Revision #
- TCP/IP MAC Address — a unique value associated with the TCP/IP network adapter that uniquely identifies the adapter on a LAN.
- EtherCAT MAC Address — a unique value associated with the EtherCAT network adapter that uniquely identifies the adapter on an EtherCAT network.

Security

Some parts of the web server are locked in order to protect critical operations from unauthorized users. Simply log into the web server to enable access to the locked functions. See "User Authentication" (see page 377) for more information.

> **NOTE**
> Functions will not be locked if you access the web server through the IDE. Doing so automatically grants administrator access.

The functions which are locked are:

- KAS Application Tab
  - Start/stop/Cold Start an application
  - "Clear User Data"
  - "Clear all Errors"
  - Configure 'Auto-start'

- Settings tab
  - Firmware upgrade -->'Choose File' &'upgrade' ‡
  - Reboot ‡
  - "Reset to Factory Settings"
  - SD card Format ‡
  - Change password
  - Change the network settings (IP address) ‡
- Backup & Restore
  - Backup PDMM
  - Backup & Restore AKDs
  - Restore PDMM
  - Scan network
  - Export Backup
  - Choose File & Replace Backup
- Diagnostics tab ‡
  - Reboot the PDMM controller
  - Clear Errors and alarms
  - Clear Crash dump

‡ AKD PDMM only

Timeout After Inactivity

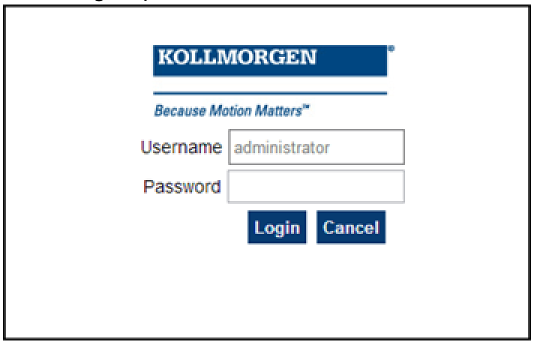
To prevent misuse, if the webserver has been idle (no keyboard activity or mouse clicks) for 20 minutes, the user account will be automatically logged out. A dialog box will open to alert you that the session has timed out.

NOTE The idle logout only occurs if you've logged into the webserver with a web browser. Logging in through the KAS IDE will never time out.

User Authentication

Logging In

Logging into the web server is required to prevent unauthorized access or changes. This is accessed from the **LOGIN** link at the top of all web server pages. Clicking the link brings up a form to enter user credentials.



Enter the password to log in. The factory default password is `administrator`. This can be changed after logging in.

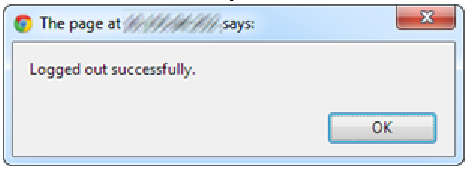
NOTE

- As there is only one user Administrator, "administrator" is the default user name.
- You must re-enter the password each time you log in; the entered password is not stored in the login form.
- Accessing the webserver from the IDE automatically logs you in as *administrator*.

When you are successfully logged in, the user name will appear in the top-right corner of all web server screens.

Logging Out

After successfully logging in, the menu in the top right corner of the web server contains a link to **LOGOUT**. Clicking this link will immediately log you out of the web server, and informs you of this.



Changing the Password

The user password is managed from the User Account section of the Settings tab. See "User Account" (see page 387) for more information.

7.3.1.2 KAS Application

This tab allows you to:

- Display general information about your project that is currently loaded on the controller (PAC or AKD PDMM)
- Start and stop the motion
- Display the Axes run by the controller from the "Axis" (see page 379) tab
- Manage log messages from the "Log Configuration" (see page 380) and "Log Data" (see page 381) tabs
- Display User Data present on the controller from the "User Data" (see page 383) tab

Item	Description
Version of KAS **App**	This label provides information about the name and version number of the application that is in the controller. The format is <project_name>:<version>. The application's source code may be downloaded to the local computer if it is present on the controller. This is accomplished by clicking on the download icon ( ). This icon is found by the name and version information and is only present when source code is available.
Status of KAS **App**	The state of the application, *Started* or *Stopped*.
Start	Default mode (warm start) where the "Retain Variables" (see page 79) are loaded at the application startup. They are **Not** re-initialized; whereas other variables are started with their initial values.
Cold Start	Use retain variables with their default values. Such starts occurs from time to time but are few.
Stop	Stop the application
Auto-start	Select this option to automatically start the KAS application when the PDMM is powered up. The application will start using retained variables (a "warm start") after the controller has booted up. To change this setting, click the Auto-start checkbox to either activate or deactivate this option and click the Apply button. The control will use the new setting at the next power-up. **NOTE** You can choose to start the application manually when debugging with the Simulator. Whereas the Auto-start mode is recommended when the system is in production, in order to prevent from doing inappropriate actions.
Clear all errors	Clicking this button will clear the error log for all axes.

Axis

You can view a visual representation of the motors from the Axis tab. The axis wheels are visible after your application is started. The following can be monitored from the display:

- Real and Simulated axes
- Actual position with solid line and actual position value
- Command position with the dotted line and (command position value) in parentheses
- Axis State: Powered-off , Powered-On, or Error as well as Simulated Powered Off and ON
- Identify the axes from the label, as defined by the axis name in your application
- Axis status or positions snapshot

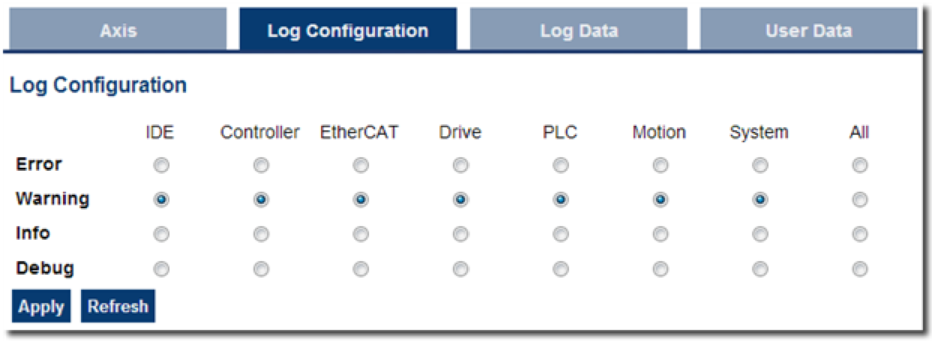
Information available by clicking on the axis

Title	Image (PN axis)	Image (PLCopen axis)	Image (digitizing axis)
AXIS1	AXIS1	PLCOpenAxis2	Axis3
Axis status snapshot	Axis positions snapshot	Axis positions snapshot	Axis positions snapshot
Initialised true	Actual 174.220505	Actual 3141.856728	Actual 0.000000
Power ON false	Current 0.000000	Command 3141.856728	
Enable true	Feedback 174.220505	Normal 3141.856728	
Found true	Generator 0.000000	Phase 0.000000	
Configured true	Pipe offset 0.000000	Super imposed 0.000000	
Running false	Pipe 0.000000		
Error false	Power ON Delta Offset 0.000000		
Simulated false	Reference 0.000000		
Connected false	Zero offset 0.000000		
Warning false			
Stopping false			
Stopped false			

Additionally, if an axis is in error, the error can be cleared by clicking the text below the axis title.

Log Configuration

You can configure the log to filter the messages that are displayed. Each source can be set with its own level.



Each message has one of the following levels, with importance in descending order: Error > Warning > Info > Debug

TIP **How to Choose the Appropriate Level?**

TIP When a level is set for a source, only messages with the same or higher importance are recorded. For example, if a source is set to WARNING, then all messages with levels WARNING, ERROR and CRITICAL are recorded (DEBUG and INFO messages are discarded).

Therefore, DEBUG is the most verbose and ERROR is the least verbose level. Filtering is quicker with less verbose levels, due to the number of messages.

NOTE Critical messages are always recorded. Therefore, the Critical level is not visible.

Source

Source	Apply to...
IDE	Win32 applications: the KAS IDE and the KAS Runtime Server (also called the KAS Runtime Front-end)
Controller	For the KAS Runtime items: Drivers, IOEngine, SinopEngine...
EtherCAT	For all kinds of EtherCAT items: Motion bus, I/Os
Drive	Messages from the drive (AKD or AKD PDMM)
PLC	For application engineers to create custom log within the PLC programs (similar to printf)
Motion	Messages coming from the Motion engines: PLCopen, Pipe network or VM
System	For common API and libraries. Also includes messages issued from the operating system.

Level

Level	Icon	Description
DEBUG		Any information logged for development purpose. You may safely ignore this log.
INFO		Information status of the current process. You may safely ignore this log.
WARNING		System is stable but the KAS IDE warns that an unexpected event can occur. You can ignore this log.
ERROR		The application does not behave as expected but the processes remain stable.
CRITICAL		Application crashes or becomes unstable. Data is corrupted. At this point the application behavior can be unpredictable.

Log Data

KAS log files may be viewed from the Log Data tab. These messages can help describe the current state of the system and to help identify any operation errors encountered when developing your system. An AKD PDMM will display as many as 10 files.

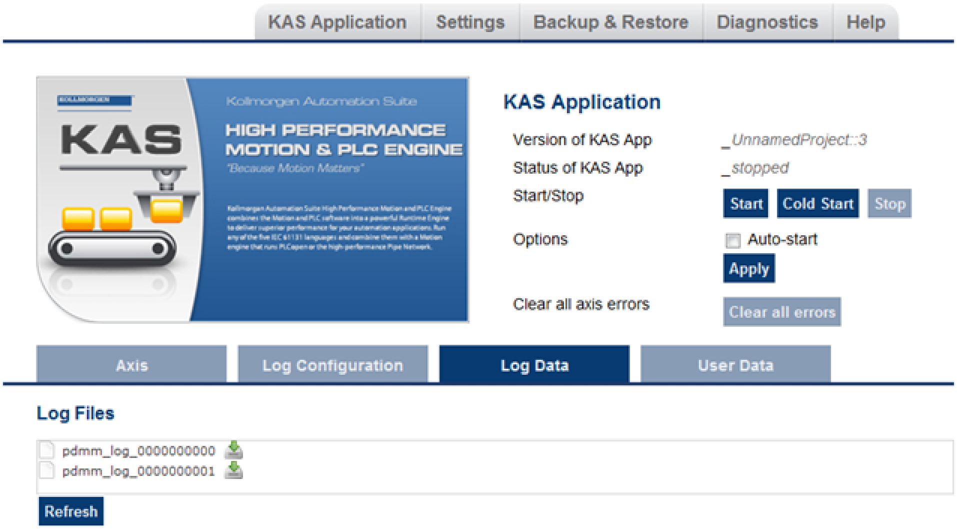


Figure 7-1: Example of log files displayed from an AKD PDMM webserver.

Clicking on a listed log file will open it in your web browser. The log file may be downloaded by clicking on the green download icon next to the log entry. The default name is the same as the file's name. If you try to open a file that no longer exists, the message "*/logfiles/<selected file name> not found.*" Refresh your browser window and try again.

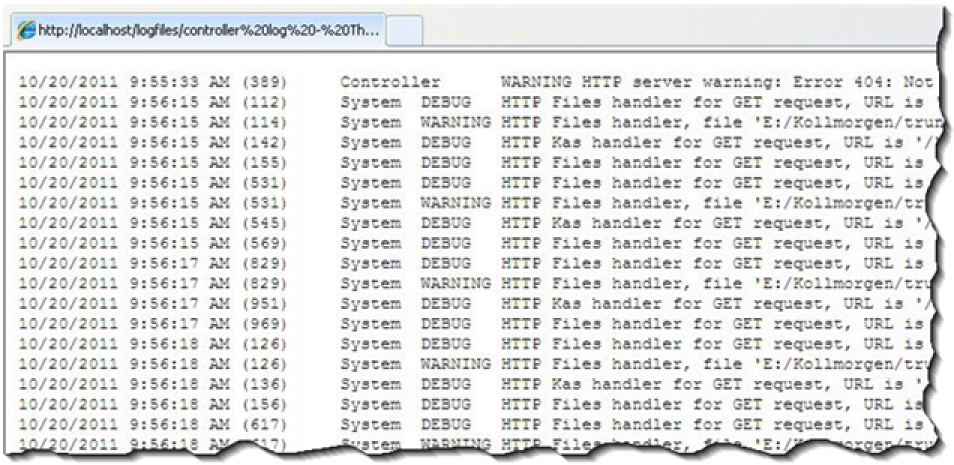


Figure 7-2: Example of a log file's content, displayed in a browser.

NOTE Log data is collected and updated every 15 seconds on a AKD PDMM and a new log file will be created when the current file is full. You may need to wait for up to 15 seconds for a log to show up in the list.

Log Message Content

Every log message in the table has the following information:

Field	Description
Time	Time when the log was recorded with the format: DD-MMMM-YY hh:mm:ss (millisecond)
Source	Identifies a software or hardware component issuing the messages. Each source is configured with a specific Level.

Field	Description
Level	Each message has one of the following levels with importance in ascending order: DEBUG > INFO > WARNING > ERROR > CRITICAL
Message	Text of the message issued from the source

Table 7-1: Log Messages - List of Field

TIP Log messages is an important source of information when you are troubleshooting your project.
When reporting an issue to Support, copy/paste the logs in your report.

AKD PDMM Log Files

Logs generated on a AKD PDMM are stored in flash memory at `/mount/flash/log`. The files are stored in a rotating pool consisting of a maximum of 10 files. The files have a maximum size of 200 kilobytes each; the most amount of space the log files will consume is 2 MB. Once an "eleventh" file is created the earliest file is flushed to make room for the new file.

The AKD PDMM generated log levels can be controlled form the KAS IDE and Web Server. From the IDE, the log levels can be filtered in the configuration window in the *Logs and Information* tab.

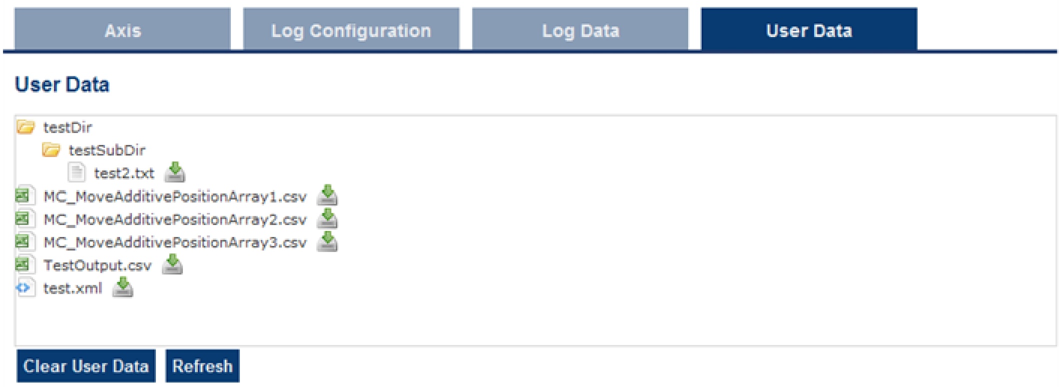
Log File Naming Convention

The logs have the naming format `pdmm_logs_n` where *n* is a value ranging from 0000000000 to 4294967295, which is the maximum value a 32-bit location can store.

As an example, when the files are first created they will be named `pdmm_logs_0000000000`, `pdmm_logs_0000000001`, `pdmm_logs_0000000002` and so on. The file that will be created after `pdmm_logs_4294967295` is `pdmm_logs_0000000000`. The naming gets reset and continues.

User Data

This tab lists any user-generated files or folders found on the flash drive. Clicking a folder will display the folders contents. Clicking on the green download icon will immediately download the file.



The **Clear User Data** button will erase all of the files in the user data folder.

This page intentionally left blank.

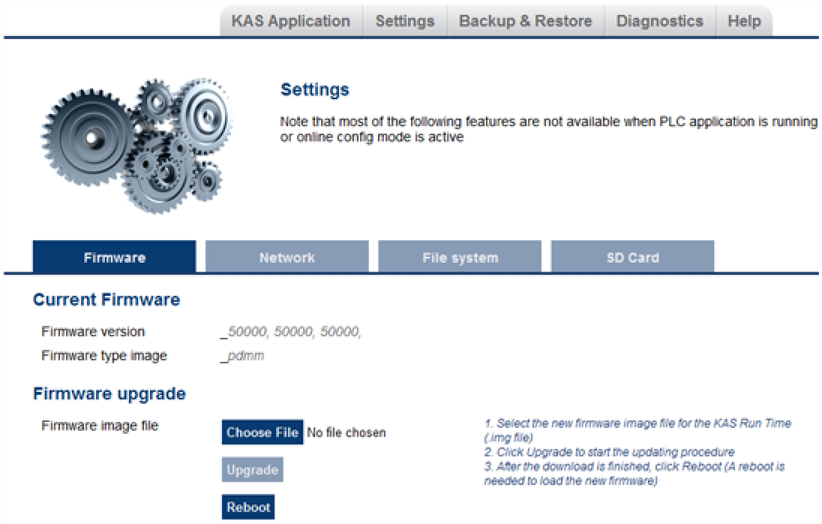
7.3.1.3 Settings

This section allows you to:

- Display and update the firmware for the KAS Runtime
- Display the network settings and modify the IP address
- Reset the control to factory settings
- Access the SD Card Actions
- Access the "User Account" (see page 387) to change the password.

Firmware Tab

This tab displays the current firmware version and type. Additionally, you may upgrade the firmware from this tab.



Upgrading the Firmware

You can upgrade the firmware of the AKD PDMM by using the web server as follows:

1. Open AKD PDMM web server in your Internet browser by entering its IP address.
2. Select the **Settings** tabbed-page
3. In the **Firmware** pane, click the **Choose File** button to select the new firmware image file for the KAS Runtime.
   The firmware files are IMG files that start with `KAS-PDMM`, followed by the software version; for example, `KAS-PDMM-2.5.0.29020.img`.
4. Click **Upgrade** to start the updating procedure
   At this point the 7-segment display shows a chasing lights animation.
5. After the animation is finished, click **Reboot** (for more details on the boot sequence, refer to Booting the AKD PDMM)

This operation downloads the KAS Runtime and its version number to the on-board flash memory in the AKD PDMM.

IMPORTANT Do not try to refresh the web page until firmware upgrade is done.

Recovery Mode

If the AKD PDMM detects a problem in the firmware, it displays an "r" on the 7-segment display and will automatically enter Recovery Mode. Recovery Mode provides the ability to select a firmware image file to build a new KAS Runtime image on the AKD PDMM. In the rare case when Recovery Mode cannot be automatically accessed, pressing and holding B2 at boot will force the AKD PDMM to boot into Recovery Mode.

Network Tab

The contents of this tab display the current rotary switch position of the AKD PDMM and its MAC address. Additionally, you may manually change the AKD PDMM's IP address.

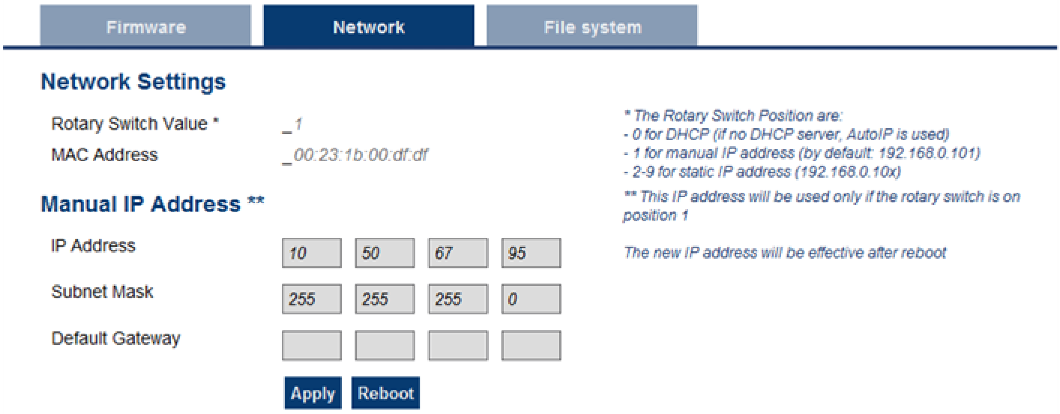


Figure 8-1: Example of an AKD PDMM with a manually defined IP address

About the Rotary Switch

The rotary switch on the AKD PDMM can be set on a position from 0 to 9.

Position 0	The drive tries to get an IP address from a DHCP server. If the DHCP fails, then the PDMM uses AutoIP to get a usable IP address.
Position 1	The default custom static IP address, 192.168.0.101 or a custom IP address.
Positions 2-9	The drive is pre-configured with static IP addresses ranging from 192.168.0.102 (Position 2) to 192.168.0.109 (Position 9).

TIP If a DHCP server is not present, the drive will assume an Automatic Private IP Address of the form 169.254.x.x

Change the IP Address

To connect and use your AKD PDMM within your computer network, you may configure its IP address by using the web server as follows:

1. Open AKD PDMM web server in your Internet browser
2. Select the **Settings** tabbed-page
3. In the **Network** pane, set static IP address according to the position defined via the rotary switch
   - If the rotary switch is set to Position 1 you may use the default custom address or set a value in the Manual IP Address fields.
4. Configure the Manual IP Address
5. Configure the subnet mask (default is 255.255.255.0)
6. (Optional) Configure the gateway address if the AKD PDMM is outside your local network
7. Click **Apply**
8. Click **Reboot**

File System Tab

This section contains a button which allows you to reset the control to the factory settings. The steps to reset the control vary slightly based on the platform.

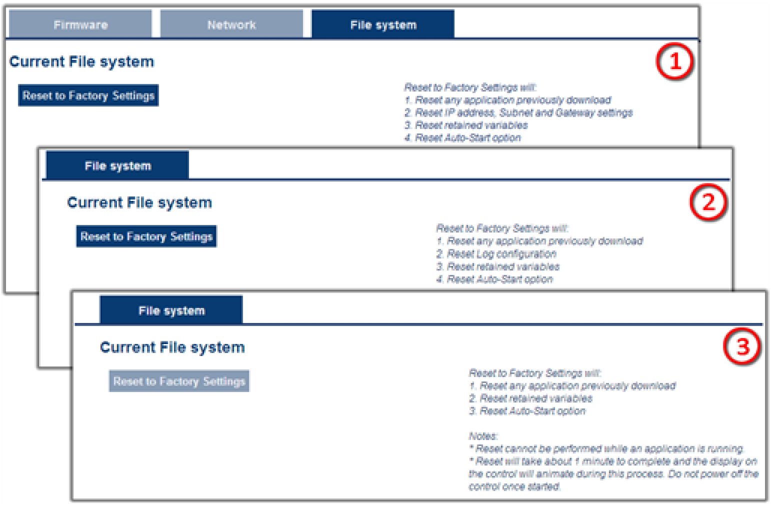


Figure 8-2: File System tab on an AKD PDMM web server, PAC web server, and when using Simulator.

Reset to Factory Settings

When this button is pressed, the control will be reset to factory default settings. The user is prompted to confirm this action before the function is performed.

The following changes occur during factory reset:

- Reset any application previously downloaded
- Reset the IP address, Subnet and Gateway settings
- Reset any retained variables
- Reset the Auto-Start option

Notes about the reset:

- The factory reset cannot be performed while an application is running. The "Reset to Factory Settings" button is disabled while an application is running.
- The factory reset will take 4-5 minutes to complete and the 7-segment display on the control will animate during this process. The control should not be turned off during this procedure.
- After the factory reset is complete, the control will be powered down and restarted automatically.
- The controls webpage will not update during the reset procedure and can be closed.
- After the control is restarted, the IP address of the control may change based on the controls rotary switch. If the rotary switch is at position 0, the same IP address as before should be assigned to the control. If the rotary switch is set to 1-9, a pre-configured IP address will be defined and must be taken into account when trying to reconnect to the controls webpage using a web browser.

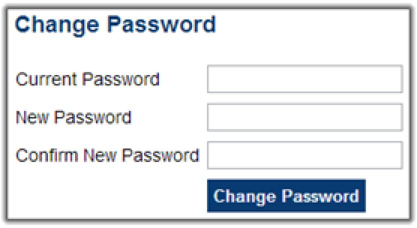
SD Card Tab

SD Card Actions

The *Format* function formats the SD card as FAT32, erasing all data from the card. This function cannot be performed while an application is running.

User Account

To change the password you must enter the current password and the new password twice.



The new password must meet the following conditions:

- It must be 6-20 characters long
- It may not contain semicolons (`;`), ampersands (`&`), spaces, quotes (`'` and `"`), slashes (`/` and `\`), or the number sign(`#`).

<u>I forgot my password</u>

Should this happen, you can set a new password from the IDE.

1. Open the webserver from the IDE.
2. Click on the Settings tab.
3. Click on the User Account tab.
4. Enter and confirm the new password.

This allows you to create a new password without entering the current one.

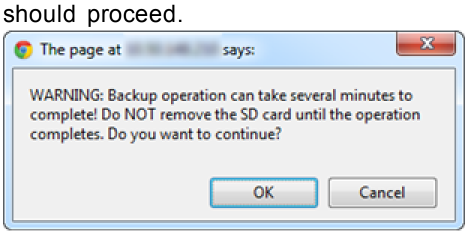
7.3.1.4 Backup & Restore

These functions are used to replicate a PDMM (*Backup* and then *Restore*). The elements that are backed up or restored are the firmware, the network configuration, the "Retain Variables" (see page 79), and the PLC application.

- These functions cannot be performed while an application is running.
- *Restore* and *Backup* take several minutes to complete. Do not power off the control once started.
- The PDMM is rebooted after a *Restore*.

TIP This section provides an overview of the backup and restore processes. For a deeper discussion, see "EtherCAT Devices Backup and Restore" (see page 363).

Backup Tab

Backup Controller	This function is used to replicate the controller part of the PDMM. Clicking the button will save the data to the SD card. An alert will be presented to confirm the backup should proceed. 

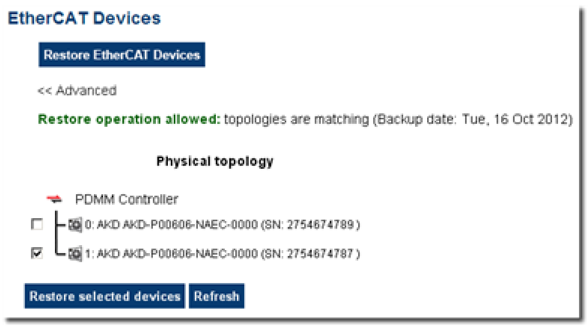
<table>
<tr>
<td>Backup EtherCAT Devices</td>
<td>This function replicates the network topology as well as the drives' firmware and data. To accomplish the backup, a copy of the firmware package is required. There are several possible scenarios upon clicking this button.

If an archived copy of the same firmware package is on the controller as is used on the drives then clicking the button will start the backup.
If a copy of the firmware package cannot be found, you will be prompted to browser for one. A link to the Kollmorgen website is provided; all firmware packages can be found on the site.


If an archive is found on the controller but it does not match the network configuration then you will be prompted to browse for one or download one from the website.

TIP Files on the website are saved in ZIP format. You must unzip the download to access the TGZ file.</td>
</tr>
</table>

Restore Tab

<table>
<tr>
<td>Restore Controller</td>
<td>This function restores a PDMM's firmware, network configuration, retained variables and PLC application from the SD card.</td>
</tr>
<tr>
<td>Restore EtherCAT Devices</td>
<td>This function automatically restores the data and firmware of any replaced drives.</td>
</tr>
<tr>
<td>Advanced</td>
<td>Clicking this button creates a map of the network and it's components. If a valid firmware file is found and the current topology matches the backup file you may select the device(s) that need to be restored. Clicking Restore selected devices will restore the drive's firmware.

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

Import/Export

These functions allow you to save a copy of the backup file to a computer and Import the backup file so it may be used for restore functions.

Export Backup	This button allows you to save the backup file to a computer.
Choose File	This button allows you to select a previously exported backup file to be imported.
Replace Backup	This button imports the selected backup file, replacing any existing backup.

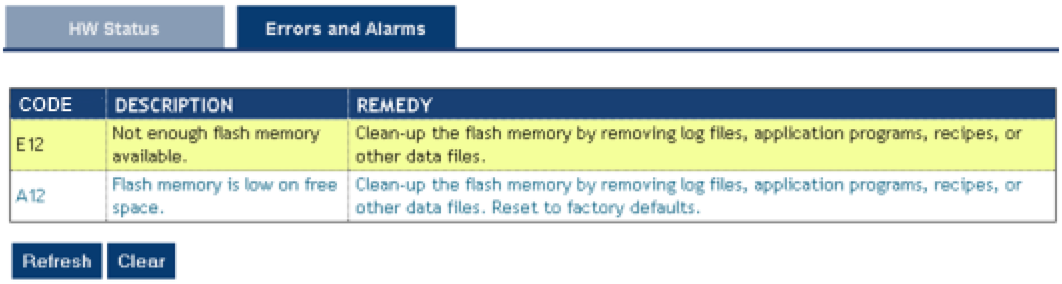
7.3.1.5 Diagnostic

This page displays information about the hardware status (storage space, memory and CPU temperature) and errors and alarms.

Errors and Alarms

Any controller errors or alarms generated by the system will be shown here and on the 7-segment display. A common error or alarm is due to the flash memory being full. This is often caused by heavy use of the PLC Advanced File function blocks.

The **Refresh** button updates the list. The **Clear** button will remove the contents of this tab. Please note that some errors or alarms are only cleared by powering off and restarting the AKD PDMM.



See Errors and Alarms for a complete list of codes.

ⓘ TIP	Axis errors can be seen in the KAS Application *Axis* tab.

Hardware Status

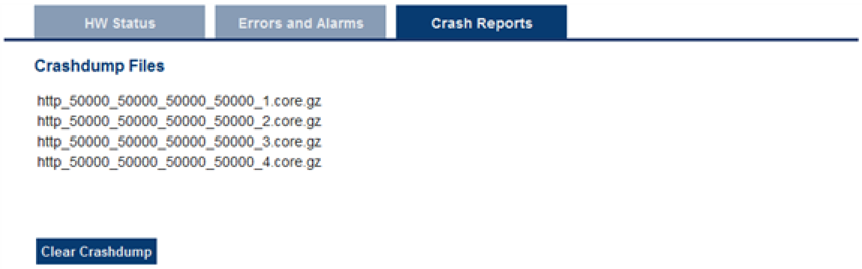
Storage Space	The diagnostic displays both the used and total available amount of storage space in megabytes (MB). **Used** is the amount of file space currently being used by all files in flash memory. **Total** is the total amount of file space available for files in flash memory.
Available Memory	This field displays the amount of RAM memory available on the AKD PDMM.
CPU usage	This field displays the current load on the CPU. If the load goes over 90%, the field turns red.

CPU Temp	This field displays the temperature of the CPU in Celsius. If the CPU temperature is greater then the CPU warning limit, the temperature background color will be changed to yellow. If the CPU temperature is greater than the CPU critical temperature, the temperature background color will be changed to red. The normal operating range is 0-125°C.
CPU Fan Present	This field is either **True** or **False**, depending upon if there is a CPU fan present in the controller.
Refresh	Clicking this button will refresh the Hardware Status information.
Reboot	Clicking this button will reboot the web server.

IMPORTANT Do not try to refresh the web page until the server has rebooted.

Crash Reports

The files shown on this tab are reports of the process that failed if there is a crash. These files (GZ archives) may be sent to Kollmorgen for analysis.



This page intentionally left blank.

8 Tools

8.1 Pipe Network Editor

8.1.1 Overview

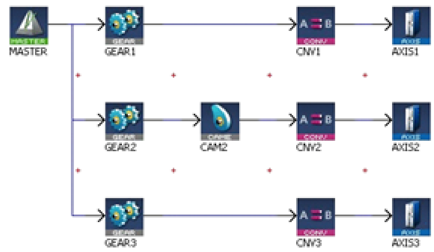


Figure 9-1: Pipe Network Structure

The Pipe Network Editor is a graphical tool dedicated to the description of the motion part of the application (See also "Pipe Network Concept" on page 95).

Functions of the Pipe Network Editor are accessed via context sensitive menus.

When the Pipe Network Editor is used, an ST file containing all the calls to the Motion Library is automatically generated during compilation, and based on the graphical description of the Pipe Network.

Pipe Network Editor is optional

Although strongly recommended, the Pipe Network Editor is optional: you can use it to graphically create a Pipe Network or you can decide to manually instantiate Pipe and Pipe Blocks by calling the appropriate functions in the Pipe Library directly from the IEC 61131-3 editors (SFC, FBD, ST, IL, FFLD).

Grid

The layout of the editor is grid oriented, which means that items (except the comments) are placed in the middle of a rectangular area called a grid unit.

NOTE Comments are not centered in the grid unit but merely placed at the cursor position.

8.1.2 Insert Pipe Blocks or Comments

To insert Pipe Blocks or comments, right-click on a free grid unit and choose the corresponding command in the contextual menu.

8.1.3 Insert Connections

Connections are simply inserted by clicking on an adequate[1] point and dragging the mouse to another adequate point. For more details, refer to paragraph "Step 12 of 15 - Adding Motion" on page 279.

Two kinds of connection can be inserted.

8.1.3.1 Connect Two Pipe Blocks

Connections are drawn between an input and an output port of two different Pipe Blocks. Connections can be drawn from input to output ports or vice-versa.

[1]As explained below, an adequate point depends on the type of the connection

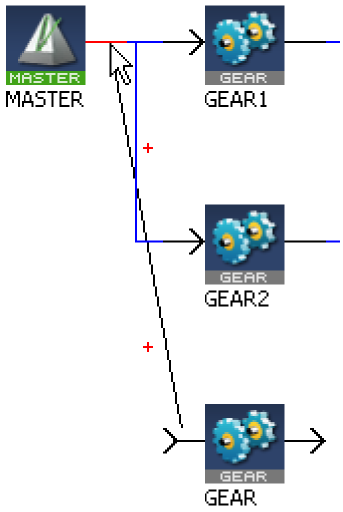


Figure 9-2: Pipe Network - Create a Link

When you try to connect two Pipe Blocks, the editor highlights the target port in red when the connection is allowed.

Relation type for output-input is 1-n

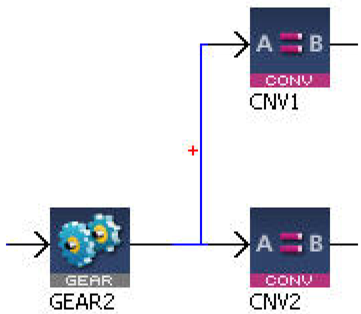


Figure 9-3: Pipe Block - Relation Type for Output-Input

One output can be connected to several inputs, but one input can only be connected to one output.

8.1.3.2 Connect Comment to Pipe Block

Connections are drawn between the text area of the comment (title bar is reserved for moving the comment) and the Pipe Block icon.

NOTE The connection cannot be drawn from the Pipe Block to the comment. Allowed target is not highlighted.

8.1.4 Edit Pipe Blocks or Comments

To edit Pipe Blocks or comments, double-click an item to open its **Property** dialog box

NOTE You can also access the property dialog box of an item through its contextual menu.

8.1.5 Move Comments

You can drag-and-drop a comment by selecting its title bar.

8.1.6 Move Pipe Blocks

Pipe Blocks are moved by dragging their center. When dragging a Pipe Block, a colored shadow is shown under the Pipe Block indicating where the Pipe Block is dropped. When the shadow fills out a complete grid unit, the Pipe Block is placed in this grid unit.

8.1.6.1 Insert rows and columns

When the shadow does not fill out a whole grid unit, but is squeezed between two grid units, a row or column is inserted before placing the Pipe Block in the newly created grid unit. When the Pipe Block is dropped on the crossing point of four grid units, a row and a column are inserted simultaneously.

NOTE You cannot drop a Pipe Block into a grid unit which is already occupied by a Pipe Block or a comment.

8.1.6.2 Remove Rows and Columns

It is not yet possible to remove rows or columns. If a row or column has been inserted by error, click the **UNDO** icon in the toolbar (**Ctrl+Z**).

8.1.7 Move Connections

You can move an end-point of a connection from one item to another. To do this, select the connection and drag an end-point to a new target.

8.1.8 Remove Pipe Blocks, Comments and Connections

Select one or several items (Pipe Blocks, comments or connections) and choose **Delete Selection** in the menu.

NOTE You can select several items by clicking on them while pressing either the Ctrl or Shift keys.

8.1.9 Plug/Unplug Channels

Right-click on a Pipe Block to plug/unplug a channel of the Softscope. For more details, refer to paragraph "How to Plug Motion Variables" on page 333.

8.2 Cam Profile Editor

8.2.1 About the Cam Profile Editor

To open the cam profile editor in a new tab of the workspace, you have to double-click on the profile in the Project Explorer.

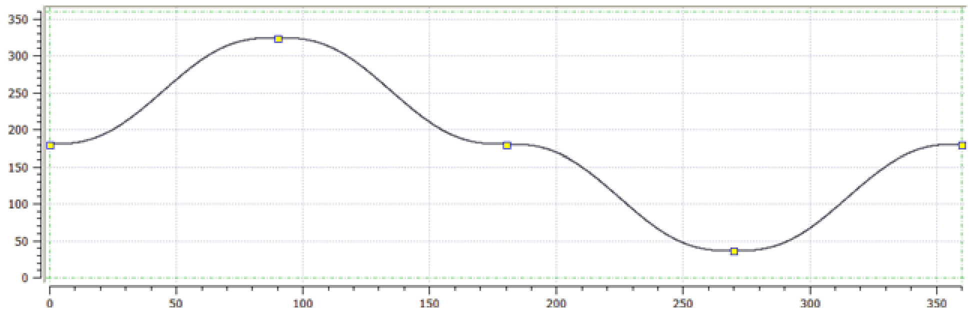


Figure 9-4: Cam Profile

The cam profile editor enables you to create and/or modify a profile definition that describes the position evolution of the cam. This evolution is displayed in a 2D graphical format.

You can add, delete, or modify cam elements which consist of points and lines. Based on those elements and some constraints, the KAS IDE calculates a complete cam shape.

Master/Input (X-Axis) and **Slave/Output** (Y-axis) coordinates can be specified to define the position.

In addition to the position, it is also possible to visualize the velocity, acceleration, and jerk diagrams.

8.2.1.1 Windows Overview

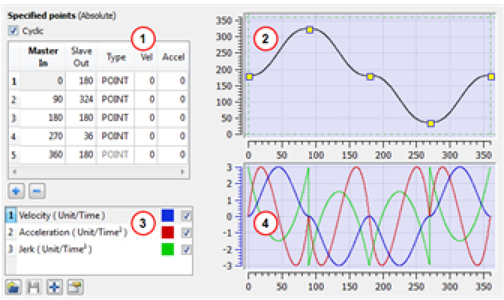


Figure 9-5: Cam Profile Editor Main Window

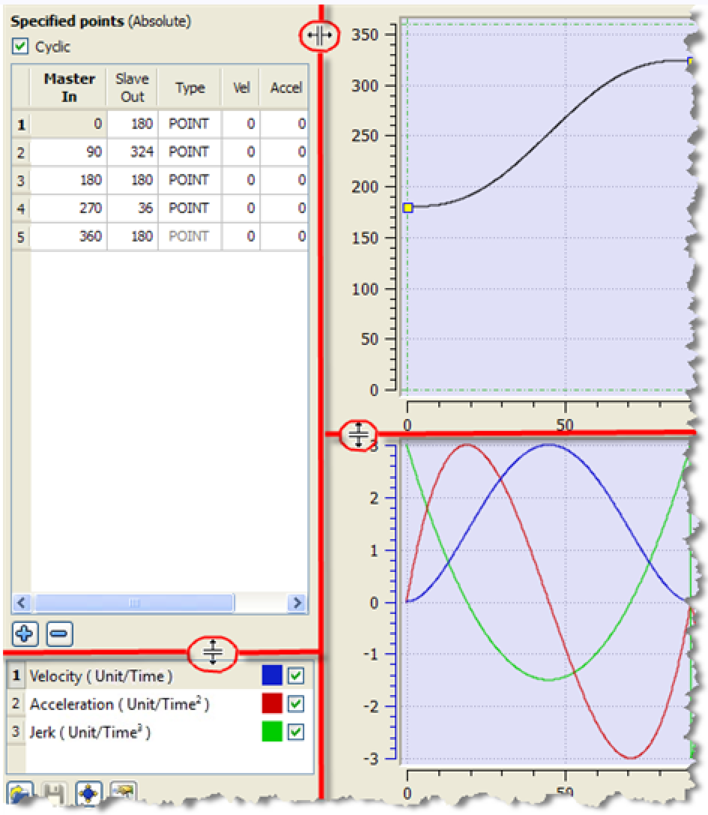
The cam profile editor contains four distinct parts separated by splitters:

1. The cam table (see call out 1) displays each element and allows editing of the cam
2. The Graphical Area for the cam profile 2
   The upper graph displays a graphical representation of the cam elements
3. The Curve Selection and Color Table 3 allows you to select which plots (velocity, acceleration and jerk) are displayed
4. The Graphical Area for Curves 4
   The lower graph displays a graphical representation of the velocity, acceleration and jerk plots

TIP Undo (Ctrl+Z) and Redo (Ctrl+Y) operations are available for any changes you make to the cam profile.

Splitters allow you to resize each part.

Improve your display with the splitters

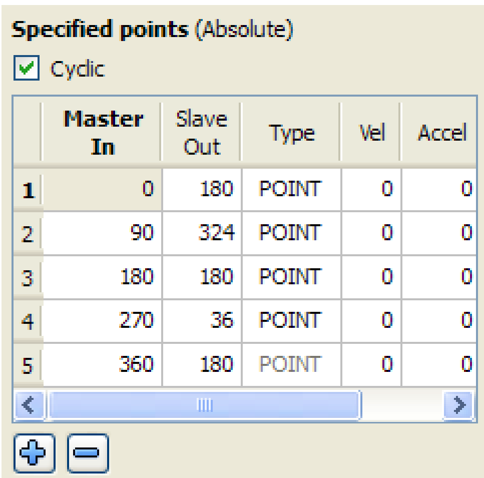


NOTE The tables and the graphs are separated by a vertical splitter so that you can completely hide the tables to increase the graphical area.

Additionally, there are four icons at the bottom, providing the following functions.

For more information on cam profiles see "Step 13 of 15 - Adding Cam Profiles" (see page 305) and "Profiles" (see page 598).

8.2.2 Cam Table



Specified points (Absolute)

Cyclic

	Master In	Slave Out	Type	Vel	Accel
1	0	180	POINT	0	0
2	90	324	POINT	0	0
3	180	180	POINT	0	0
4	270	36	POINT	0	0
5	360	180	POINT	0	0

Figure 9-6: Cam Table

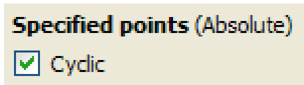
When a new profile is created, the cam profile contains five points by default.

NOTE These points could be different from those in the figure above, depending on the offsets and amplitudes specified in the cam profile Properties dialog box.

Column	Description
Master/In	The time is located in the **Master/In** column. It is the X-axis of the cam profile graph
Slave/Out	The position is located in the **Slave/Out** column. It is the Y-axis of the cam profile graph
Type	The **Type** column defines whether this element is a point or a line. If the element is a line, In/Out specify the start point of the line. The next element in the table defines the end of the line **NOTE** The last element type in the table cannot be changed, since a line cannot exist as the last element
Vel	The **Velocity** of the current element (first derivative)
Accel	The **Acceleration** of the current element (second derivative)

Table 9-1: Cam Editor - Table Parameters

About Cyclic Cam Element



If the *Cyclic* check box is selected, the cam profile is executed cyclically. This means that, when the axis attached to this cam runs continuously, the same profile is executed again. In this case, the first and last element must have the same **Vel** and **Accel** values. Therefore, changing the **Vel** or **Accel** value of the first or last elements automatically changes the other elements' value.

NOTE When *Cyclic* is first turned on, the Vel/Accel values will automatically be copied from the first element to the last element when they do not match. A warning dialog is displayed to inform you that this change has happened. This alert can be suppressed until KAS is closed.

There are some combinations of points and lines where *Cyclic* will automatically be turned off. If this occurs, the cyclic checkbox label will be changed to *Cyclic (automatically turned off)*. The following changes to the profile will automatically turn off cyclic:

1. The first element has been changed from a point to a line. If needed, cyclic can manually be turned back on which will affect the velocity of the last element.
2. The next to last element has been changed from a point to a line and now both first and next to last elements are lines. Cyclic will be disabled and will only be re-enabled when the first and next to last elements are not lines.
3. The first element is a line and the first element is moved. If needed, cyclic can manually be turned back on which will affect the velocity of the last element.
4. The first element is a line and the second element is moved. If needed, cyclic can manually be turned back on which will affect the velocity of the last element.
5. The first element is a line and the last elements velocity (or slope line) has changed. If needed, cyclic can manually be turned back on which will change the velocity setting just made.

8.2.2.1 Modifying an Element using the Cam Table

You can modify a cam element by clicking in the **Master/Input**, **Slave/Output**, **Vel**, or **Accel** column and typing in a new value. For **Type**, refer to the relevant paragraph.

The graphs are updated automatically when an element changes.

Some rules apply to the value entered:

- The **Master/Input** value must lie between adjacent **Master/Input** points
- The **Master/Input** value of the first and last point cannot change. These values are determined by the profile properties X offset and X amplitude
- The **Slave/Output** value must lie between the Y offset and Y amplitude set in the profile properties

If an entered value is invalid (due to the interpolation calculation), it is superseded with the original value without any error message.

About interpolation

The section between two consecutive cam elements is automatically calculated by a fifth order polynomial algorithm.

Modification of one cam element only affects the two adjacent segments.

8.2.2.2 Modifying the Type of a Cam Element

The type of element can either be a point or a line. The element type can be modified by double-clicking in the **Type** column of an element and then clicking on the down arrow. A list of choices is displayed as shown in "Figure 9-7: Modifying an Element Type " on page 400. Select the type of element from the list.

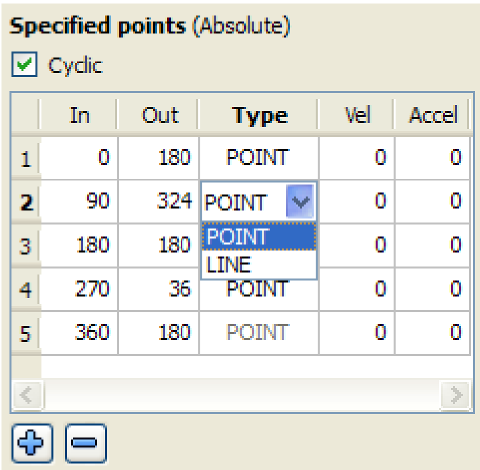


Figure 9-7: Modifying an Element Type

8.2.2.3 Cam Table Contextual Menu

Right-clicking on an entry in the cam table displays a contextual menu.

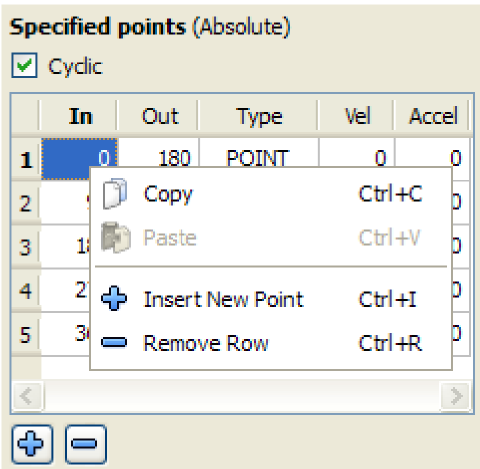


Figure 9-8: Cam Table Contextual Menu

Command	Shorcut	Description
Copy	Ctrl+C	Copy data from the selected cell in the clipboard
Paste	Ctrl+V	Paste the data from the clipboard into the selected cell
Insert New Point	Ctrl+I	Inserts a new row in the cam table above the highlighted entry. This command is described in paragraph "Adding a Point" on page 401
Remove Row	Ctrl+R	Deletes the row that contains the highlighted entry. This command is described in paragraph "Removing a Point" on page 402

8.2.2.4 Adding a Point

You can add a point to the cam table using one of the following methods:

- Use the menu in the cam table (shown in "Figure 9-8: Cam Table Contextual Menu " on page 401)
- Click the button  located below the cam table
- Use the menu in the cam profile graph

All of these methods displays the **Add New Point** dialog box:

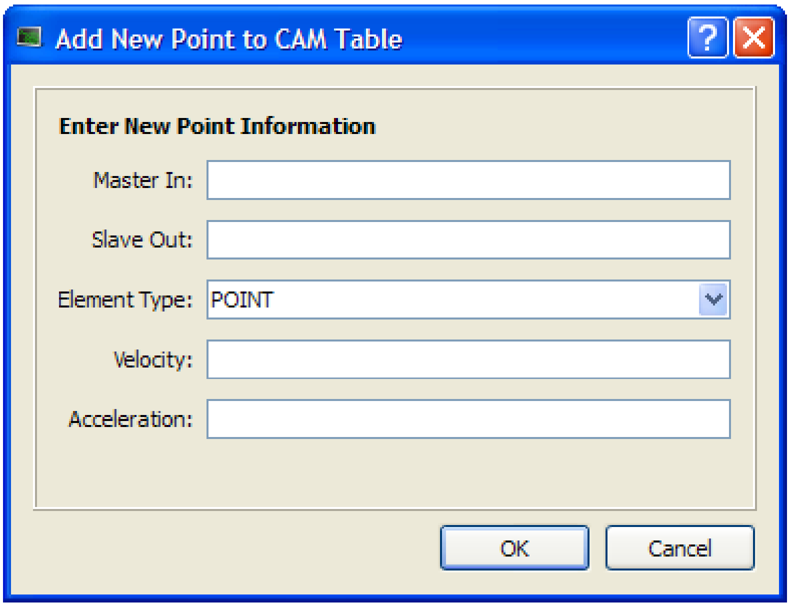


Figure 9-9: Add New Point

Field/Command	Description
Master In	The X value of the new point
Slave Out	The Y value of the new point
Element Type	POINT or LINE
Velocity	The velocity of the new point (first derivative)
Acceleration	The acceleration of the new point (second derivative)
OK	Accept the entry and verify if the point can be added.
Cancel	Cancel the dialog box – no point is added.

Table 9-2: Cam Editor - New Point Parameters

When you click OK, a check is performed to see if the point can be added to the cam profile. If not, an error dialog box is displayed.

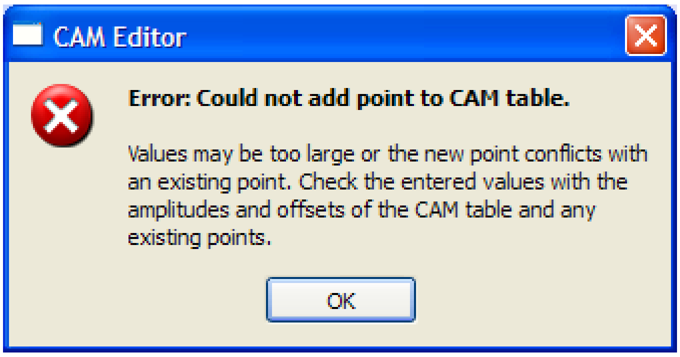


Figure 9-10: Cam Table Contextual Menu

If no problem is found, the point is added to the cam table and the graphical plots are updated.

NOTE A new point cannot be inserted above the first element in the cam table.

8.2.2.5 Removing a Point

You can remove a point from the cam table with one of the following methods:

- Use the menu in the cam table (shown in "Figure 9-8: Cam Table Contextual Menu " on page 401)
- Click the button  located below the cam table
- Use the menu in the cam profile graph

The selected point is removed without prompting.

NOTE The first and last points cannot be removed.

8.2.3 Cam Profile Graph

The upper graph displays the points and lines specified in the cam table along with the calculated curve. It also allows you to add, delete or modify a cam element.

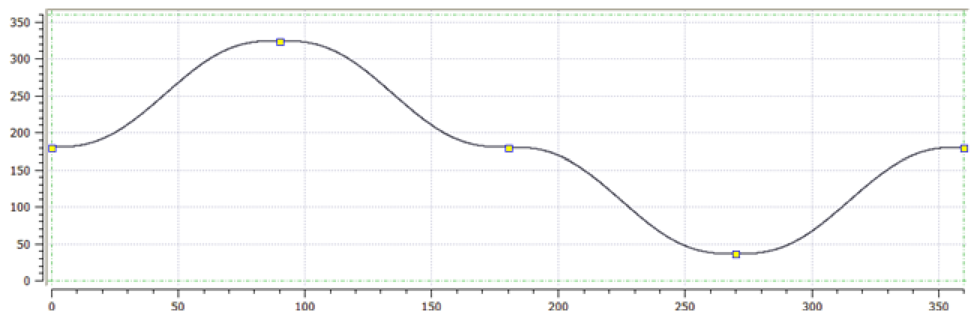


Figure 9-11: Cam Profile Graph

Points and endpoints of lines are displayed as yellow squares ( ) in the graph. The profile offset and amplitude specified in the properties are displayed with a green dashed rectangle. The yellow squares are always contained within the green dashed rectangle (although calculated points can extend outside it).

8.2.3.1 Modifying an Element

You can modify the profile by moving point with the mouse as follows:

1. Move the mouse over a yellow square (the cursor becomes ✥ indicating that the point can be selected)
2. Click to select the point and hold down the mouse button (left-click). When you move the mouse, the point follows the cursor (note that graphical curves and In/Out values are dynamically updated)

In addition, when a point is selected, a slope line is drawn over the point. This line is dashed purple with two additional grips (◆) attached to it. The slope line can be used to change the velocity of the selected point.

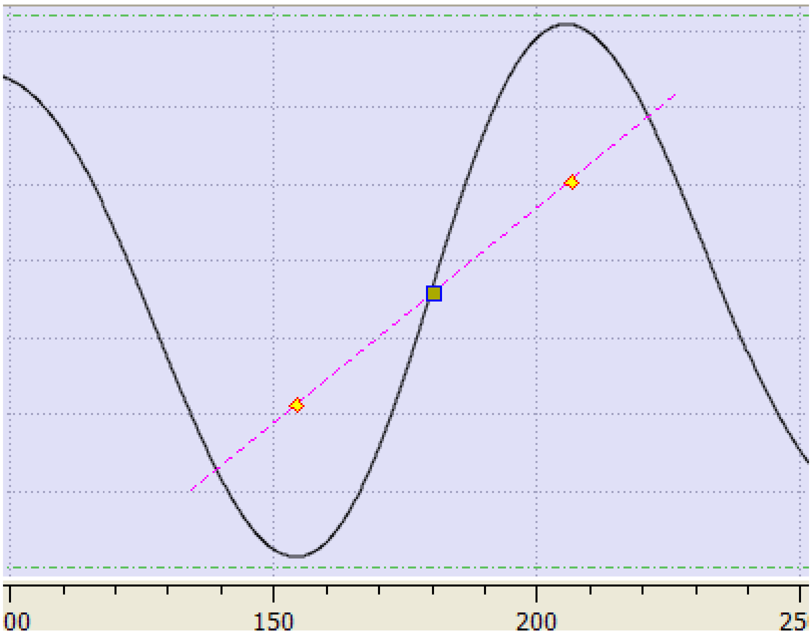


Figure 9-12: Cam Profile Graph - Slope Line

You can change the velocity of the selected point as follows:

1. Move the mouse over a slope grip (◆). The cursor changes to an open hand
2. Click to select the grip and hold down the mouse. The cursor changes to a closed hand
3. When you move the mouse, the slope line follows the cursor, rotating about the selected point and causing the velocity of the selected point to change. (Note that graphical curves and Vel value are dynamically updated)

8.2.3.2 Cam Profile Graph Contextual Menu

A right-click on the cam profile graph displays a contextual menu.

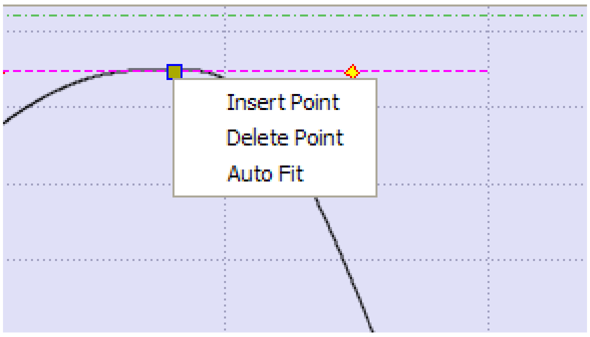


Figure 9-13: Cam Profile Graph - Contextual Menu

Command	Description
Insert Point	Inserts a new point at the X-Y location of the cursor
Delete Point	Deletes the highlighted point If the mouse is not near enough to a point, no point is highlighted and this command remains grayed-out
Auto Fit	Adjusts the zoom and pan settings so that the entire graph is displayed in the graphical area

8.2.3.3 Zoom In and Out

In the cam profile graph, you can zoom in or out as follows:

1. Move the cursor in the graphical area
2. Turn the mouse wheel forward or backward

The current cursor becomes the center point of the zoom function and the area under the cursor remains stationary on the graph.

8.2.3.4 Panning

In the cam profile graph, you can also pan (or move) in any direction as follows:

1. Click on any part of the graph (but not on a yellow square) and hold down the mouse button (left-click)
2. Move the mouse to move the graph accordingly

8.2.3.5 Restoring Zoom and Pan

To restore the zoom and pan settings, so the entire curve is displayed in the graphical area, click on the Auto Fit button  or select the Auto Fit command in the cam profile graph menu.

8.2.4 Curve Selection and Color Table

Velocity (first derivative), acceleration (second derivative) and jerk (third derivative) plots are displayed in the lower graph. If the element is a line, the velocity is constant and acceleration is 0.

With the check boxes in the Curve selection table shown in figure below, you can select or clear each individual curve to be displayed.

When a curve is selected (see blue highlighted row in figure below), the Y-scale of the Curves graph is adjusted to display the Y-scale of the selected curve. Also, the color of the 'tick' line of the scale is changed to match the color code of the selected curve.

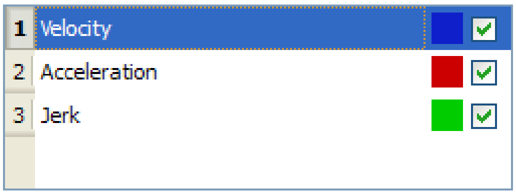


Figure 9-14: Curve Selection Table

8.2.4.1 How to change color

You can change the color of a plot as follows:

1. Double-click on a colored square shown in the Curve Selection Table to open the color selection dialog box

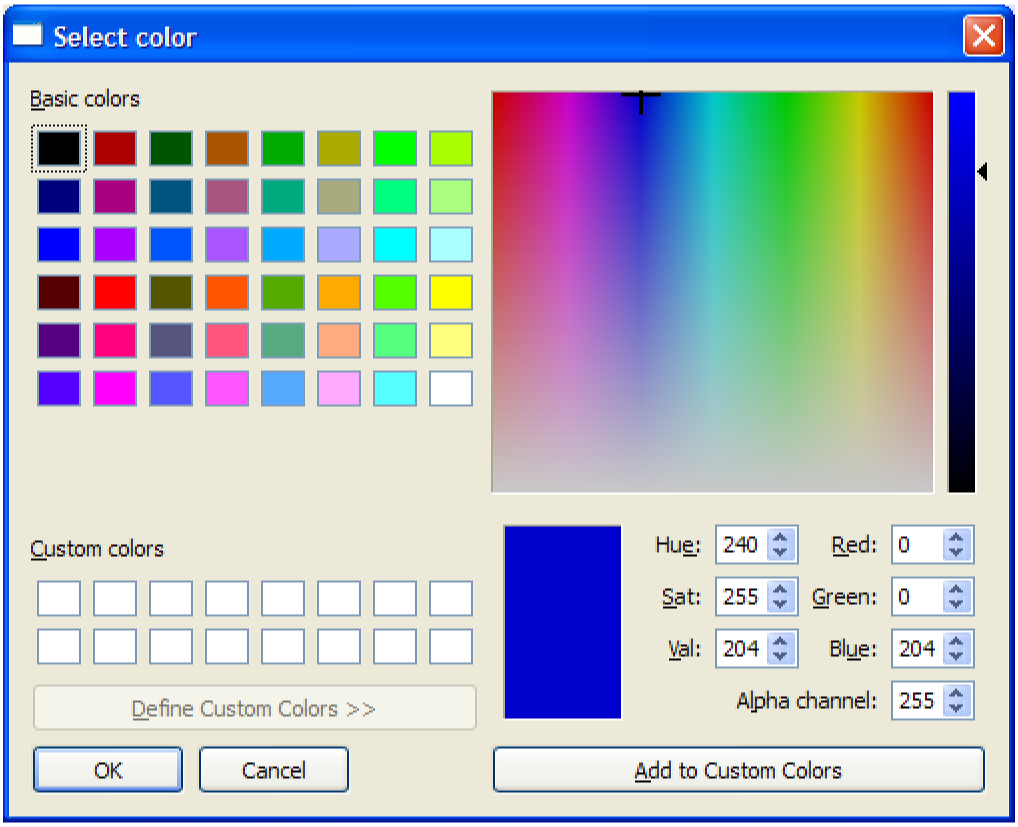


Figure 9-15: Standard Color Selection

2. Click on an existing color square to select it, or specify the numerical values for a color. (You can also move the black indicator on the right side until the desired color appears in the large colored rectangle)

8.2.5 Curves Graph

Velocity (the first derivative), acceleration (the second derivative) and jerk (the third derivative) curves are displayed in the lower graph. All plots are displayed by default.

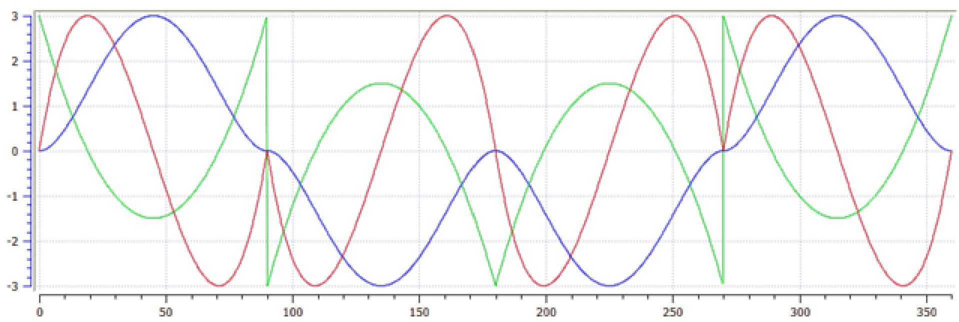


Figure 9-16: Curves Graph

With the check boxes in the Curve selection table shown in "Figure 9-14: Curve Selection Table " on page 405, you can select or clear each individual curve that you want to be displayed.

The Y-scale of the Curves graph is adjusted to display the Y-scale of the selected curve in the Curve Selection Table. The color of the Y-axis scale "tick" lines is also changed to match the color code of the selected curve.

Y axis	Unit	Description
Vel	Units/Time	Being the rate of change of position, the velocity is the ratio between the slave and master derivatives
Accel	Units/Time^2	Rate of change of velocity with time
Jerk	Units/Time^3	Rate of change of acceleration; more precisely, the derivative of acceleration with respect to time

In general the numbers relate to how the Y-axis positions (Cam Output) change with respect to the X-axis positions (CAMinput).

The zoom and pan functions, when performed on the cam profile graph, are duplicated in the Curves graph.
Zoom and pan functions are not available when the cursor is in the curves graph.

8.2.6 Reload, Save, Auto Fit, and Properties Buttons

The following buttons are provided:

Icon	Description
Reload	Reload the saved profile. If unsaved changes have been made to the profile, a dialog box asks you to confirm that you want to discard the changes.
Save	Save a modified profile.
Autofit	Adjusts the zoom and pan settings so that the entire graph is displayed in the graphical area.
Properties	Open the Cam Profile Properties dialog box to modify the Master-/Input and Slave/Output Offset and Scale values

Table 9-3: Cam Editor - List of Icons

8.2.7 Import Cam Profile

The KAS IDE can import legacy cam profiles that follow the CSV format described below:

Row	Syntax
1	`CYCLIC;YES;`
2	`TABLE_BEGIN;;`
3	`0;0;SPLINE`
4	`X;Y;SPLINE`
:	`X;Y;SPLINE`
N	`1000;1000;SPLINE`
N+1	`TABLE_END;;`

Each row from 4 to N specifies the successive points that are part of the cam profile. The X and Y coordinates can be specified as floating-point values with sufficient digits after the decimal point (example: 995.2514255). To be valid, a CSV file must have **at least 4 spline segments** in it.

When a CSV file is imported the X, Y values are normalized with respect to maximum X, Y values present in the CSV file. The normalized X, Y values are scaled with respect to Master/Input scale and Slave/Output scale. They are added with Master/Input Offset and Slave /Output Offset respectively and will be displayed in the Specified points (Absolute) section of the cam profile.

Example:

CSV file X,Y Values:

```
0;0;SPLINE
100;111;SPLINE
200;222;SPLINE
300;333;SPLINE
```

Max Value in CSV is

```
300;333;SPLINE
```

Normalized values:

```
0; 0;
0.333333333333333; 0.333333333333333;
0.666666666666667; 0.666666666666667;
1;1;
```

Offset:

```
10      20
```

Scale:

```
300     360
```

Value displayed in profile:

```
10;20;
109.9999999999999; 139.99999999999989;
210.00000000000009; 260.00000000000011;
310;380;
```

8.2.7.1 About the Import

A quick validation is performed when the profile is first imported.

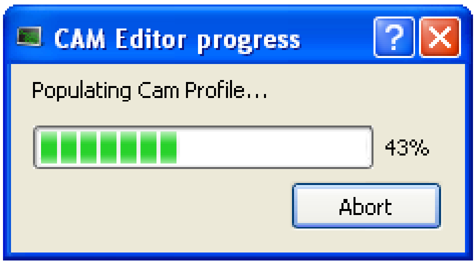
- The profile name is checked and if it is in use you are promoted to provide a new name.

- The data format is validated and we check to see if the profile can be compiled without error.

If cyclic is on and the Vel/Accel values of the first element do not match the Vel/Accel of the last element, the first elements Vel/Accel will be copied to the last elements. A warning message will be posted to the log if this change takes place.

8.2.7.2 When Displaying the Imported Cam Profile

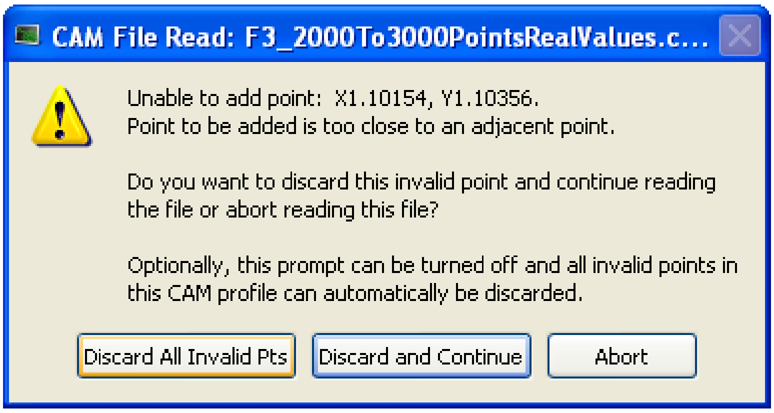
When you display an imported profile a dialog box indicates the progression of the import process.



Click the **Abort** button to abort the process, then a default cam profile is created.

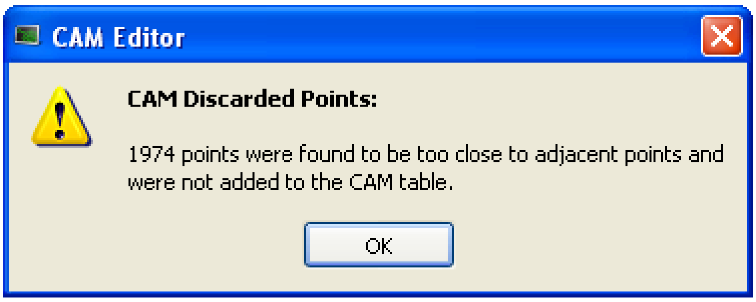
8.2.7.3 About Invalid Data

When you display a CAM profile where two points are too close, a dialog box indicates the error.



Click the **Discard All Invalid Pts** button to discard all additional invalid points found in this cam profile.

A summary is displayed when the process is finished.



8.3 Softscope

The soft oscilloscope (commonly known as softscope or scope) is a tool that allows you to view, in a two-dimensional graph, one or more variables' evolution (vertical axis) across the time (horizontal axis).

As shown on the figure below, the scope has a set of channels where each can acquire the evolution of a value. A value can be the feedback position of an axis, the speed of a machine, or anything else that can be measured with the softscope probes (for more details on how to attach a variable, see page 417).

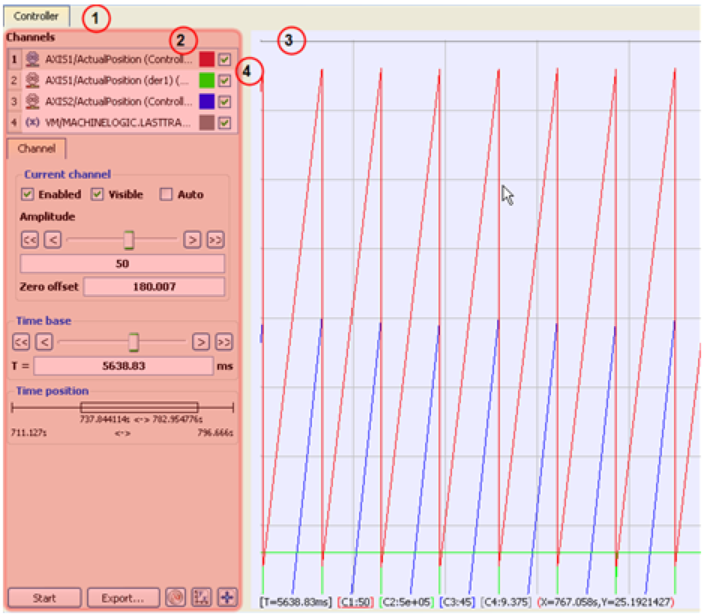


Figure 9-17: Scope View

The soft oscilloscope is a window where the tab's name is the controller's name (see call out 1). This scope view has two visually distinct parts:

- The Control Panel 2 enables you to change the settings of the soft oscilloscope (including those of the channels)
- The Graphical Area 3 shows the traces acquired by the channels

The control panel and the graph are separated by a splitter 4

TIP You can hide the Control Panel for the best user experience with a drag-and-drop operation.

How to access the softscope view?

In order to access the softscope view, select the **Oscilloscope** command from the **Tools** menu.

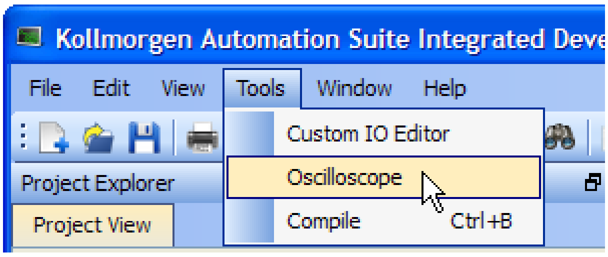


Figure 9-18: Accessing the Scope

About OpenGL

For the Graphical Area, the scope uses **OpenGL** for performance reasons. It does not work under **Windows XP Embedded** (which has no OpenGL libraries installed by default). On other systems, if you encounter problems in the quality of drawings, we suggest that you consider the following points before contacting our support desk:

Check that your graphical card driver is up-to-date.

Newer drivers often fix the rendering bugs of OpenGL.

Disable some optimizations on the Display hardware acceleration

Open **Display** Properties[1]. In the **Settings** tab, click the **Advanced** button, then select the **Troubleshoot** tab. If **Hardware acceleration** is set to full, try to disable some optimizations. This procedure has proven to be useful in particular with cursor drawing problems that appear when the user performs high-zooming operations (the cursor can indicate a value which is out of the trace).

Change the settings of your graphic card

Open the manufacturer-specific settings of your graphic card. If there are some settings related to **Performance and quality**, try to set them to **quality** (but not high quality) instead of performance, at least for the specific program: **KAS IDE.exe**. This solves many drawing problems that occur when zooming a lot in the graph.

Ignore line width and line style properties of channels

For the moment, line width and line style properties of channels are not supported. Please do not try to change them. Changing them causes drawing problems and consumes system resources.

Display a given amount of samples, according to the refresh rate

If your channels have acquired a large number of samples, and the refreshing of the graph does not occur frequently enough, do not display all samples at the same time either by:

- Hiding some less useful channels (use the visibility property)
- Reducing the time-base and/or restricting the time-frame in the time position.
  In any cases, this action does not stop acquisition or lose your acquired samples.

IMPORTANT Disabling most or all OpenGL accelerations is compensated by an increase in CPU consumption. It can lead to a point where the soft oscilloscope is not very usable when limited hardware is trying to display loads of samples.

8.3.1 The Control Panel

[1] The Properties command is accessible in the contextual menu on your desktop (you can also access the Display from the Windows Control Panel)

As shown on "Figure 9-19: Scope Control Panel " on page 411, the control panel consists of the following items:

- The **Channels** list (see call out 1)
- The **Current channel** property 2
- The **Time-base** 3
- The **Time position** 4
- Five buttons 5

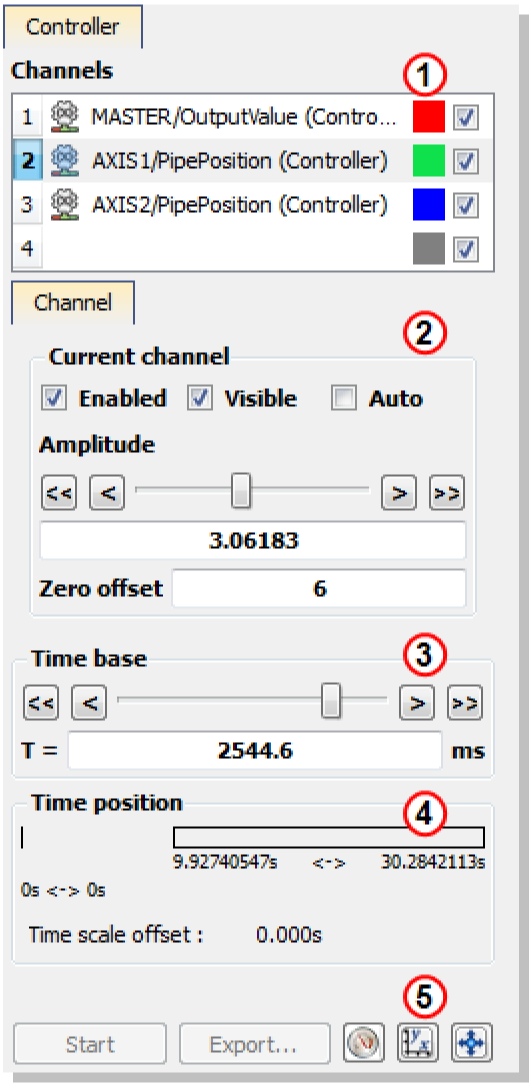


Figure 9-19: Scope Control Panel

The Channels item

It lists all the available channels.

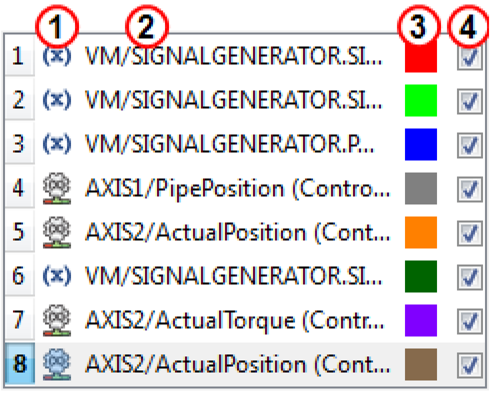


Figure 9-20: Scope Control Panel - Channels

For each channel, it shows:

- The **type** of the associated variable (IEC 61131-3 or Pipe Block) with a symbolic icon 1
- The **name** of the associated variable 2
- The **color** of the associated curve in the graph with a color icon 3
- The **visibility** of the associated curve with a check box 4

TIP You can change the color of a curve by double-clicking on its color icon, and its visibility by clicking on its check box.

NOTE Double-click on any channel in the list to open the Edit all channels dialog box.

When selecting a channel in the channels list, it is superimposed on the existing traces, and some related information are displayed on the left and lower sides of the graph.

The Current Channel item

It is a tab widget that holds properties related to the channel selected in the list. On some special devices, some more tabs that are specific to extra configurations appear in this widget. For example, S300 device provides trigger functionalities, so an additional tab is displayed for the trigger configuration.

The current channel properties are:

Properties	Description
Enabled	A channel has to be enabled to acquire the samples sent by its associated probe
Visible	A channel has to be visible to be drawn on the graph Even if not visible, it continues to acquire the samples sent by its associated probe

Properties	Description
Auto	A channel in auto mode automatically adapts its amplitude (unit/division[1]) and zero offset in order to be able to display all its samples. Setting the auto mode disables the possibility of changing the Amplitude and the Zero offset (see paragraph "Setting Scale" on page 421 for more details about scaling)
Amplitude	Allows you to control the amplitude (unit/division) of the channel. The buttons and slider change the amplitude according to a logarithmic scale. The dialog box allows a more precise definition of the value
Zero offset	The curve is vertically shifted so that this value is located halfway through the graph height

Table 9-4: Scope - Current Channel Properties

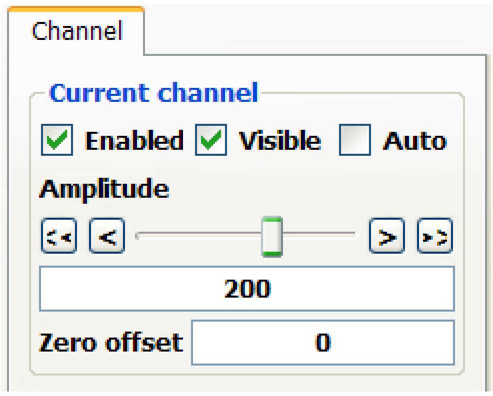


Figure 9-21: Scope Control Panel - Current Channel

The time-base item

This enables you to set the speed at which all the lines for each channel are drawn, and is calibrated in milliseconds per division.

Its usage is similar to the Amplitude property described in the above section. The time-base can always be changed, even during sampling (see also paragraph "Time Scale" on page 421).

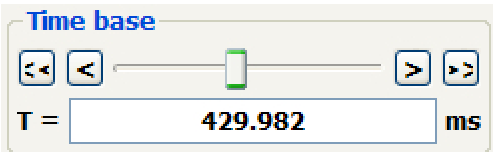


Figure 9-22: Scope Control Panel - Time-base

To setup the time-base properly, the total measurement duration and the required time resolution have to be taken in account.

The time position item

This enables you to change the time-frame of the acquired samples shown on the graph. It is composed of:

[1] The term refers to the time-base value for the X-axis and to the amplitude value for the Y-axis. For example, if the user sets a time-base of 10ms and an amplitude of 1, each division in the soft oscilloscope grid corresponds to a time of 10ms for the X-axis and an amplitude of 1 for the Y-axis.

- A single **horizontal line** representing all the acquired samples with start and stop timings
- A **rectangle** representing only the time slot of the acquired samples, which is displayed in the graphical area (the time-frame) with timings:

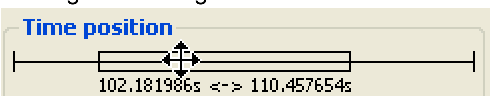
Figure 9-23: Scope Control Panel - Time Position

NOTE The acquisition of samples is limited to 100'000 cycles (ie. 100 s when cycle time is set to 1000 µs, and 25 s when cycle time is set to 250 µs). When you reach this limit:

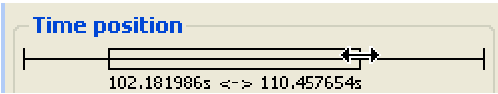
- The first data that are added to the queue are the first data to be removed (FIFO queue)
- The start timing increases

You can change the time slot with the mouse by:

- moving the rectangle



- changing the size



The **Time Scale Offset** is the time value of the first sample the graph when plotting is started. Using this as an offset, the time axis is always started at 0 seconds. To get the actual time value of any sample, add the time scale offset to the Time axis value.

```
Actual Sample Time Value = Time Scale Offset + Time axis value
```

How to set the time-frame?

When clicking anywhere on the horizontal line, the time-frame is centered on the clicked point. It is also possible to move the time-frame by clicking on its rectangle part and dragging.

You can resize the time-frame in a user-friendly manner by clicking on its left or right ends and dragging.

NOTE During acquisition the time position item is disabled and displays the progression of acquisition.

Five buttons

At the bottom of the controls are five buttons:

1. The **Start/Stop** button 

allows you to start or stop the acquisition of samples. When starting acquisition, all previous samples are lost.
2. The **Export...** button 

allows you to save the acquisition data in a CSV file. For more details, see page 416.

3. The **TraceTimes** button allows you to display the four following channels
    - Channel **1**: Cycle Jitter (in µs)

      When the motion is started, the current cycle time remains constant on an average of several cycles, and equal to the EtherCAT cycle time which is a constant value (1000, 500 or 250 µs). The CycleJitter is due to EtherCAT transmissions that can vary in a particular cycle (see call out 1).

      The channel 1 of the scope monitors the time difference between the expected Cycle Time and the actual Cycle Time. (see figure below).

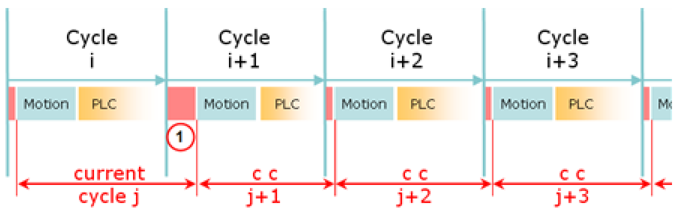


Figure 9-24: Cycle Time Calculation

- Channel **2**: Motion execution time (microseconds)
- Channel **3**: PLC execution time (microseconds)
- Channel **4**: Real Time Margin (microseconds) This channel monitors the available execution time (Cycle Time Period - EtherCAT network execution time - MotionExecTime - PLCProgExecTime) in each cycle period.

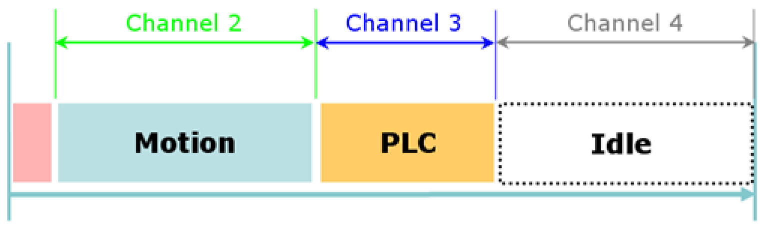


Figure 9-25: Motion, PLC and Real Time Margin Time Calculations

NOTE This feature is **not** relevant with the KAS Simulator. The MotionExecTime and PLCProgExecTime traces will be visible with simulated values. The CycleJitter and RealTimeMargin will always remain at zero with the simulator.

For more explanations, refer to "Tasking Model / Scheduling" (see page 177)

4. The **graduations** button displays or removes the axis graduations of the graphical area.

5. The **autofit** button changes the time-frame of the graph and amplitudes and zero offsets of channels so that they all fit entirely into the graphical area.

8.3.2 The Graphical Area

The graph displays a subset of the collected data: the **time-frame**. To better view and analyze the data, the graph has the following features:

- Graduations are displayed on the left and lower sides of the graph
- Information concerning the time-frame of the graph and the amplitude of channels also appears at the bottom of the graph. The current channel amplitude is underlined and the coordinates of the nearest collected sample are displayed
- It is possible to **zoom** in the graph using various methods (for more details, see paragraph "Trace Zoom Feature" on page 422)
- It is possible to **move** the contents of the graph within the time-base (for more details, see page 422)

NOTE Moving the contents is possible **only** when the acquisition is stopped.

How to Export the Collected Data?

To copy the trace data into a CSV file:

1. Display the softscope
2. Ensure the channels you want to export are Enabled and Visible
3. **Start** the data collection
4. Wait for the probe data you want to save to be collected
5. **Stop** the data collection
6. Click the **Export...** button
7. Select where you want to save the CSV file
8. Click the **Save** button

NOTE A warning is displayed if you try to save the file in an invalid location, or to overwrite a file that is currently in use.

You can now import the data into Microsoft Excel.

NOTE The Export operation is possible even when acquisition of samples is in progress. But in that case, the latest exported data are the data collected when you have defined the CSV file.

NOTE The acquisition of samples is limited to 100 s when the cycle time is set to 1000 µs (respectively 50 s with 500 µs, and 25 s with 250 µs)

About the CSV file format

Each channel takes 2 columns: one for the **time** and the other for the **value**. This allows exporting channels with different time-base.

The **List separator** and the **Decimal symbol** are hard-coded (they are not bind to the regional settings)

- List separator is comma (**,**)
- Decimal symbol is dot (**.**)

TIP If your regional settings are different, then you have to specify explicitly those two characters in Microsoft Excel to correctly import the CSV file

8.3.3 Traces

The trace is the resulting graph of the variable's evolution against time, with the more distant past on the left and the more recent past on the right.

NOTE The acquisition of samples is limited to 100'000 cycles (ie. 100 s when cycle time is set to 1000 µs, and 25 s when cycle time is set to 250 µs). When you reach this limit:

- The first data that are added to the queue are the first data to be removed (FIFO queue)
- The start timing increases

8.3.4 Plugging Probes

A probe is a virtual measurement point that can be connected to a variable.

Three types of variables can be plugged:

1. Pipe Block variable which is a Pipe Block related variable.
2. IEC 61131-3 variable which is any other variable.
3. PLCOpen axis values.

NOTE Your application **must be connected and running** to let you plug a channel to a variable

You can connect a probe to a variable in one of the following ways:

- from the Softscope
- from the Dictionary
- from the Pipe Network

8.3.4.1 Plugging a probe from the softscope

In order to directly plug a probe from the softscope:

1. Double-click on any channel in the channels list to open the **Edit all channels** dialog box

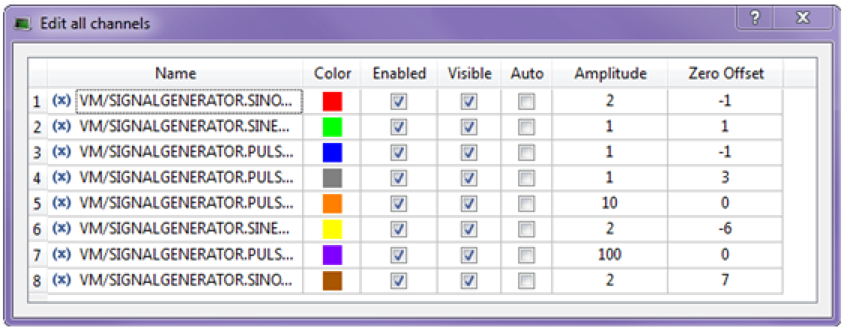


Figure 9-26: Edit all Channels

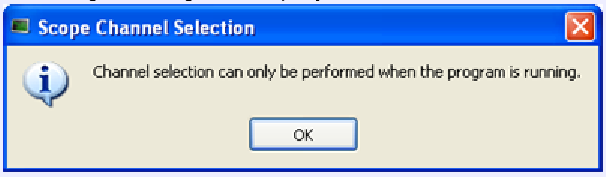
This dialog enables you to manage all the channels in the same view. For each channel, the following information is displayed:

Field	Description
Name	Name of the variable plugged on this channel
Color	Color assigned to this channel's trace. Performing a double-click on the color allows you to change the color
Enabled	Controls the channel's enabled state
Visible	Controls the channel's visible state
Auto	Sets the channel's scale as automatic if enabled
Amplitude	Unit per division scale value for this channel
Zero offset	Zero offset value of this channel

Table 9-5: Scope - Channels Properties

2. Double-click on a channel's name to open the **Variable Selector**

NOTE If your application is not connected and running, the following message is displayed.



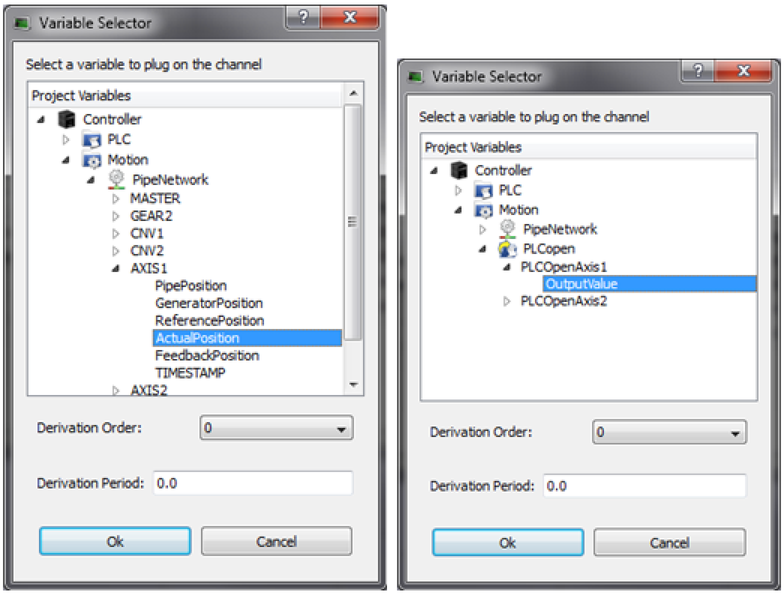


Figure 9-27: Scope - Variable Selector for **Pipe Network** and **PLCopen**

3. Navigate through the available variables and select the one you want to connect to the channel

NOTE The Variable Selector contains only the PLC variables that are eligible for the softscope (i.e. BOOL, INT, SINT, DINT, LINT, UINT, USINT, UDINT, ULINT, BYTE, WORD, DWORD, LWORD, TIME and LREAL, as long as they are not in a UDFB instance).
In addition, in simulated mode, only a subset of variables are displayed (e.g. ActualVelocity [1] is not visible).

[1]The measured value is the instant velocity of the axis in RPM*1000. Note that you can see some oscillations because it is an instant velocity, not an average velocity.

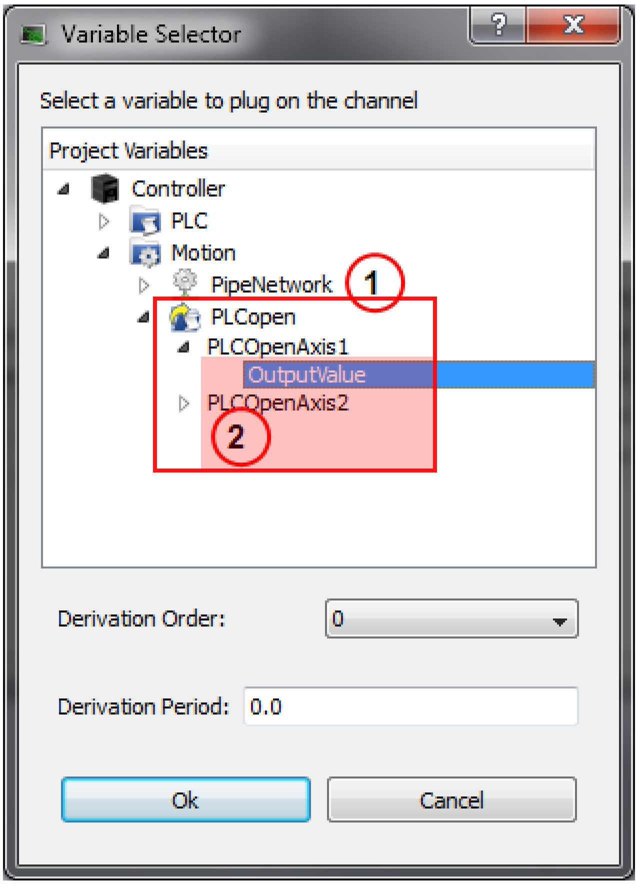


Figure 9-28: Scope - Variable Selector of an item in a **array** (see call out 1) which is part of a **structure** 2

For more details on:

- Axis pipe block positions, see page 101
- PLCopen Axis positions, see page 145

4. (Optional) Set the Derivation Order.
5. (Optional) Set the Derivation Period. The value entered should be either `0.0` (no modulo) or the Modulo Period, e.g. 360.0.
   If the selected Derivation Order is greater than zero, the Derivation Period of the selected signal can be used to remove rollover spikes in the derivative value if the variable is of a periodic nature as the result of "modulo" behavior.

You can also disconnect a probe as follows:

Unplugging a probe

In order to unplug a probe:

1. Double-click on any channel in the channels list to open the **Edit all channels** dialog box
2. Right-click on the corresponding channel(s)
   ⓘ TIP Multiple channels selection is allowed for this action.
3. Select the **Unplug probe** command in the menu to disconnect the probes on the selected channel(s)

8.3.4.2 Plugging a probe from the Dictionary

1. In the **dictionary** toolbox, right-click on the variable
2. In the menu, select the **Plug on channel...** command

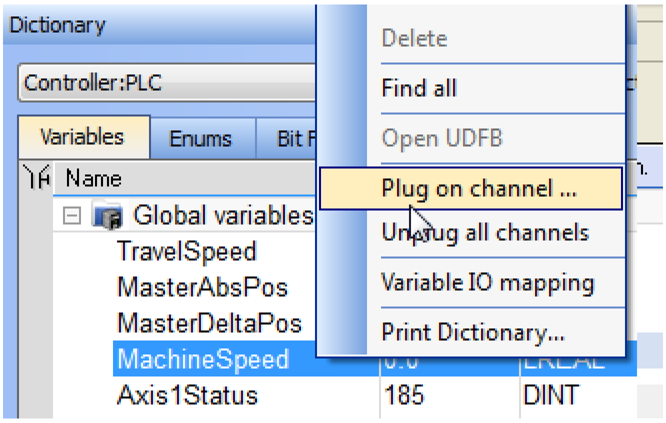


Figure 9-29: Plugging a Probe from the Dictionary

NOTE This command is enabled if the type of variable is eligible for the softscope (i.e. BOOL, INT, SINT, DINT, LINT, UINT, USINT, UDINT, ULINT, BYTE, WORD, DWORD, LWORD, TIME and LREAL, as long as they are not in a UDFB instance).
When you want to plug a probe to a variable in an array or a structure, you have to navigate with the **Variable Selector** (see more details here).

3. Define the probe parameters

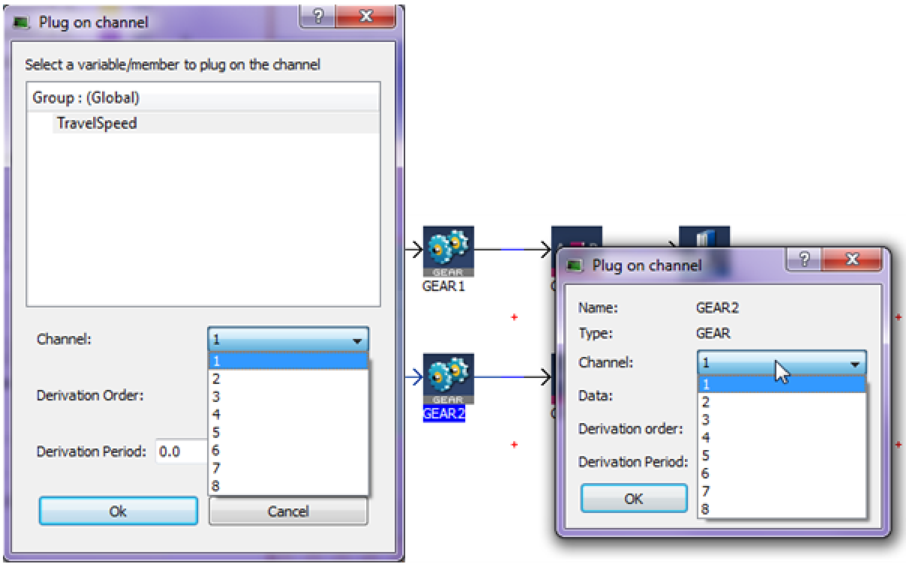


Figure 9-30: Methods for associating a Variable to a Channel

Field	Description
Name	Variable's name
Type	Variable's type
Channel	Channel's number where the variable has to be plugged
Data	Desired variable information to show (the list depends on the type of Pipe Block.)

Field	Description
Derivation order	Performs a derivation of the measurement of the selected variable. If this value is different from 0, the derived value of the selected order is shown on the selected channel
Derivation Period	Specifies the modulo period for a periodic variable to remove spikes in the display of derivative orders greater than zero. The value entered should either be 0.0 (No Modulo) or the Modulo Period (eg. 360.0).

Table 9-6: Scope - Probe Parameters

NOTE In order to enable the **Plug on channel...** dialog box, the KAS IDE must be connected to the device first!

8.3.4.3 Plugging a probe from the Pipe Network

In order to plug a probe from the Pipe Network:

1. Right-click on a Pipe Block
2. Select **Plug on channel...** in the menu

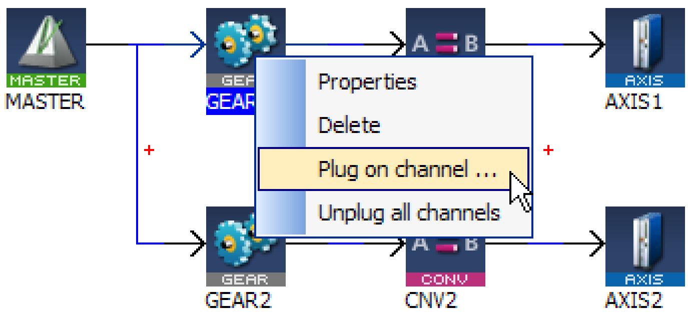


Figure 9-31: Plugging a Probe from the Pipe Network

3. Define the probe parameters (this step shows the same dialog box used in the paragraph above)

8.3.5 Setting Scale

The soft oscilloscope graph is divided into 8 units for the horizontal time scale (X-axis) and 8 units for the amplitude (Y-axis). These units can be user-defined by using the configuration panels described below.

NOTE **Unit per division**: the term refers to the time-base value for the X-axis and to the amplitude value for the Y-axis. For example, if the user sets a time-base of 10ms and an amplitude of 1, each division in the soft oscilloscope grid corresponds to a time of 10ms for the X-axis and an amplitude of 1 for the Y-axis.

Time Scale

The time scale can be configured with the **Time-base** configuration panel. The default value is 100ms/unit with the limits being 0.1ms to 25,000ms. The new value can be entered by hand directly in the text field or by using the buttons:

Buttons	Description
	Used to divide / multiply the time-base by 2 (performing a **division** corresponds to a **zoom in** while performing a **multiply** corresponds to a **zoom out**)
	Used to divide / multiply the time-base by 10

The base time unit is 1 ms.

TIP You can also modify the time scale by scrolling the mouse wheel with the cursor located in the graphical area.

Variable Scale

Variable scaling is done by modifying the amplitude and offset value of a channel.

The variable scale can be configured in different places:

- The Current channel control panel.
- The Edit all channels dialog.

TIP You can also modify the variable scale by pressing down the Ctrl key while scrolling the mouse wheel with the cursor located in the graphical area.

NOTE The changes affect only the selected channel.

8.3.6 Trace Zoom Feature

The zoom feature is used to magnify or reduce a portion of a trace. Two zoom modes are available:

Time zoom	Used to expand/collapse the time-base in order to have a better view of the signal evolution through time. This zoom operation updates the time-base value.
Amplitude zoom	Used to have a better view of a part of a signal. This zoom operation updates the amplitude & zero offset value

The zoom operations can be done:

- By modifying the corresponding values by hand
- By using the mouse wheel

For more details on setting the amplitude, zero offset and time-base values, refer to paragraph "Setting Scale" on page 421.

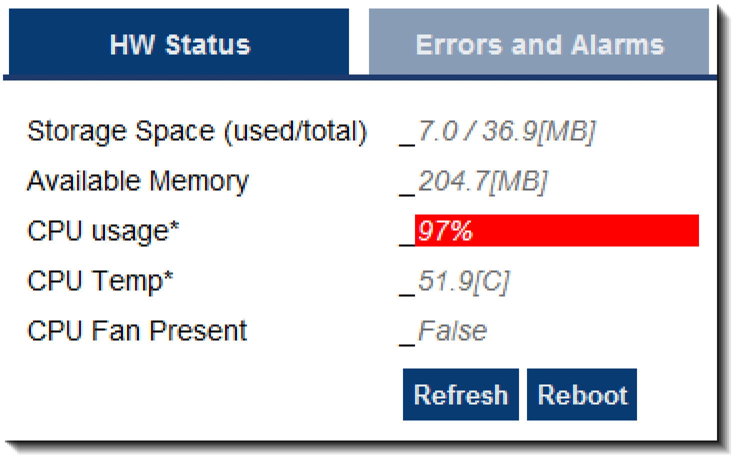
Mouse Shortcuts

Action	Result
Scrolling up the mouse wheel	Expands the time-base value
Scrolling down the mouse wheel	Collapses the time-base value
Pressing the Ctrl key while scrolling up	Makes the amplitude value greater
Pressing the Ctrl key while scrolling down	Makes the amplitude value smaller

NOTE When performing an amplitude zoom, the zero offset is automatically set by the cursor position.

8.3.7 Practical Application: Using Trace Time To Measure CPU Load

To determine the overall controller CPU usage, look at the HW Status tab on the Diagnostics page of PAC or PDMM web server. If the **CPU usage** is less than 90% then the CPU load (both Real Time and Non-Real Time) is okay. If the **CPU usage** is 90% or higher then the CPU is too heavily loaded and should be reduced by simplifying the application or reducing the CycleTime update rate.



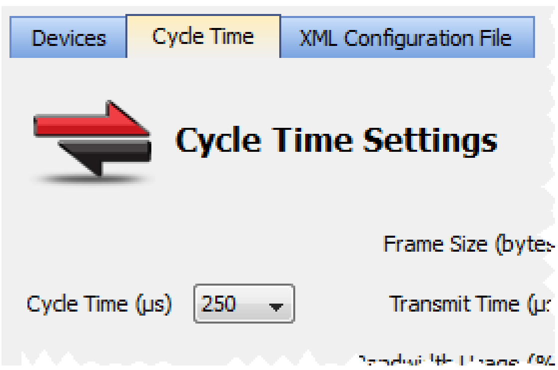
The IDE Oscilloscope trace times can be used to analyze the application performance on a controller or programmable drive. This section describes some techniques you can use to interpret the trace times to examine the real-time performance.

There are two major parts to consider when evaluating total performance:

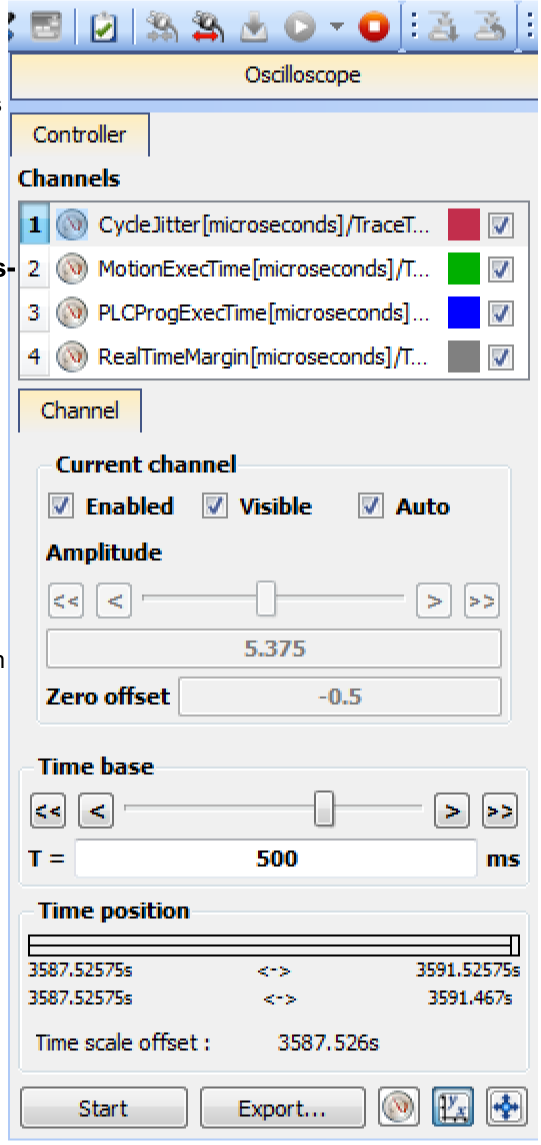
Real Time	EtherCAT + Motion Engine + PLC program
Non-Real Time	everything else (the background tasks)

The Oscilloscope trace times provide a very good tool to examine the Real Time response. Although it doesn't provide the complete system picture, it is a good place to start. It can provide some indication about the Non-Real Time load, but the best indicator is the overall **CPU usage** and the Controller Log messages.

First, you will want to know the Cycle Time for your system. From the **Project View**, select the **EtherCAT** view and the **"EtherCAT Master Settings" (see page 214)** tab. The update period for the system in this example is set to 250 microseconds.



The "Trace Times" traces are enabled by pressing the **Plug Trace Times channels** button in the Oscilloscope view when your application program is running. This button automatically configures the Channels, as seen here.



8.3.7.1 Collect some data by pressing the “Start” button

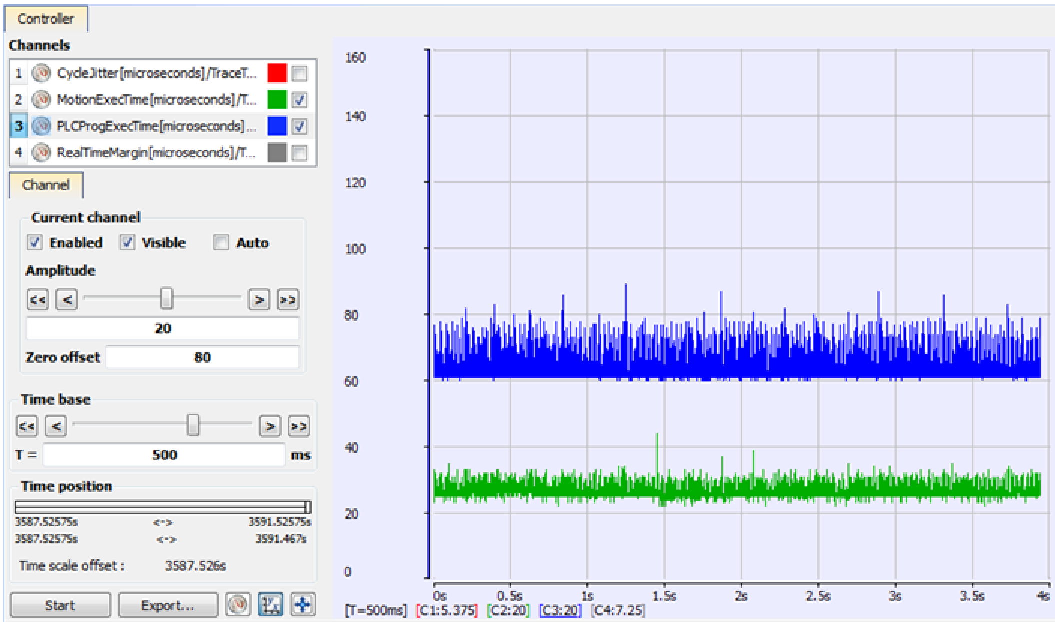
The first thing to do is to collect data during the normal application operation, particularly once the system has reached a steady state. Press **Start** and let the data collect for a few seconds and then press the **Stop** button.

The first traces to examine are the “MotionExecTime” and “PLCProgExecTime”. Configure the **Amplitude** and **Zero offset** so you can see both traces easily. Below are some recommended values based on several Cycle Time values.

Cycle Time	Amplitude	Zero Offset
250ms	20	80
500ms	40	160
1000ms	80	320

ⓘ *TIP* Unchecking the "CycleJitter" and "RealTimeMargin" traces is useful so they don't clutter the view.

The following example has a Cycle Time of 250 microseconds. The "MotionExecTime" average is about 27 microseconds and the "PLCProgExecTime" average is about 68 microseconds.



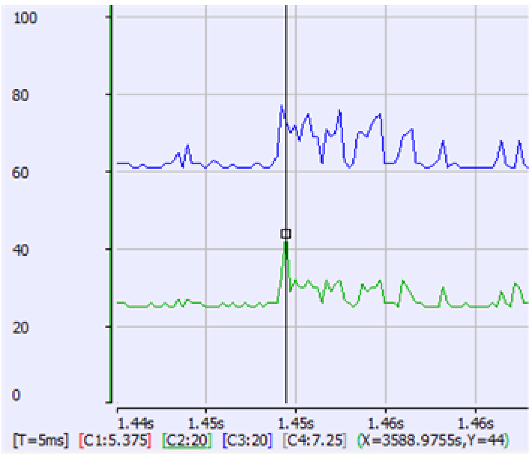
The average time for the MotionExecTime + PLCProgExecTime is 95 (27 + 68 = 95), which is about 38% of the cycle (95 / 250). This is a good value.

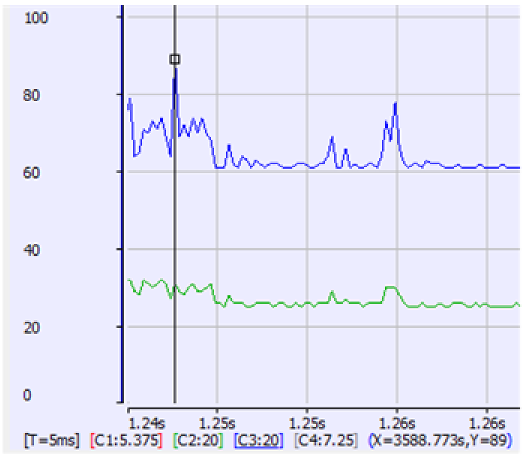
8.3.7.2 Check the peak times

The next step is to examine the spikes. We will examine the "MotionExecTime", "PLCProgExecTime", "RealTimeMargin" and "CycleJitter" traces.

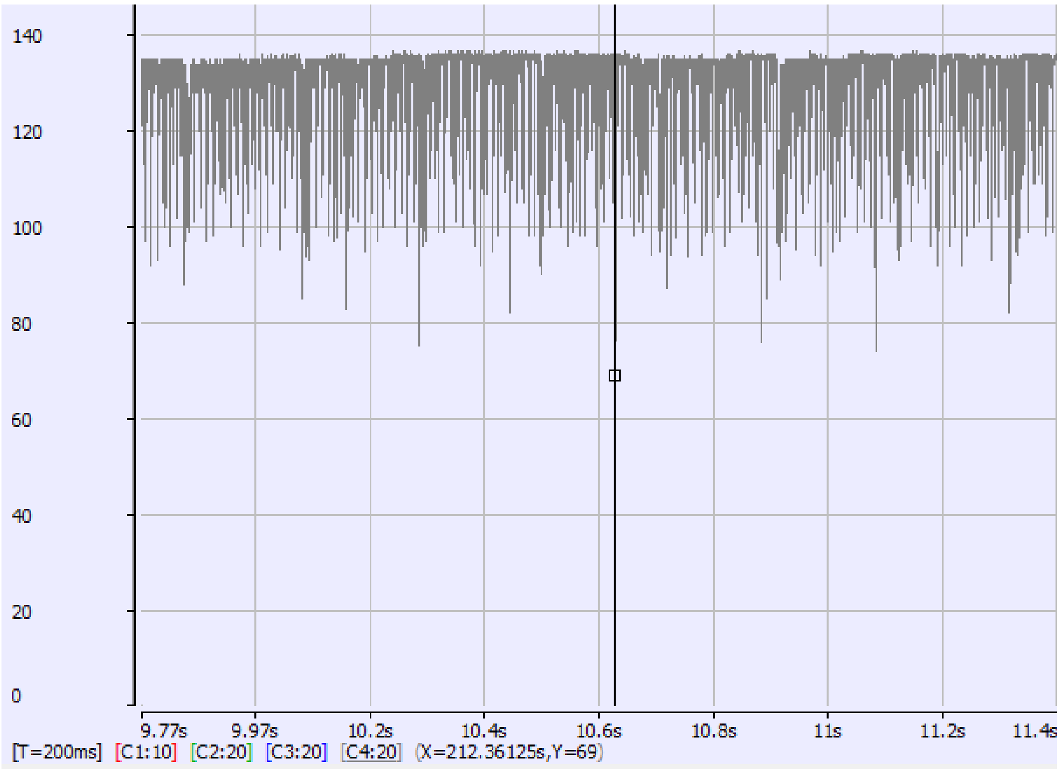
1. Reduce the **Time** base and move the traces left or right with the mouse while holding the left mouse button.
2. Position the cursor to measure the peak.

In this example the "MotionExecTime" peak is 44 and the "PLCProgExecTime" peak is 89. This is reasonable.

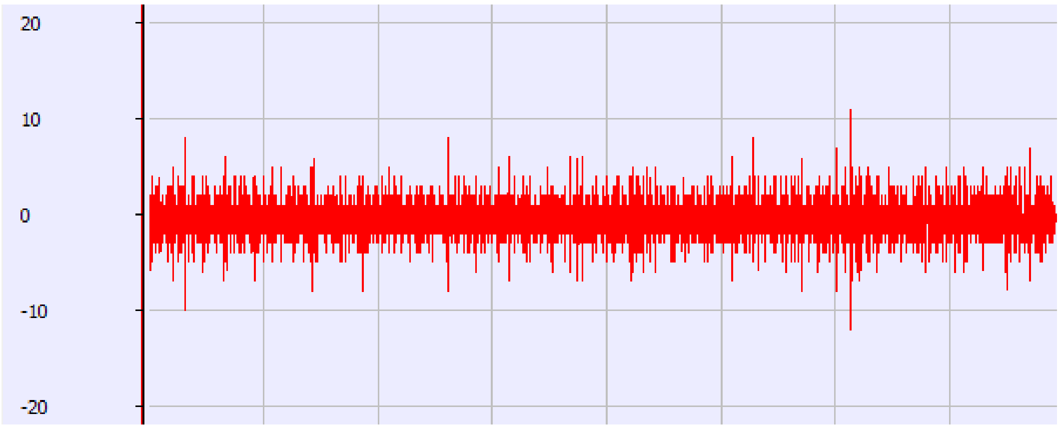




For the "RealTimeMargin" peaks configure the **Amplitude** and **Zero offset** so you can see the trace near zero. In this example the minimum peak (closest to zero) is 69 microseconds. This provides a 28% (69 / 250) Real Time margin which is good.

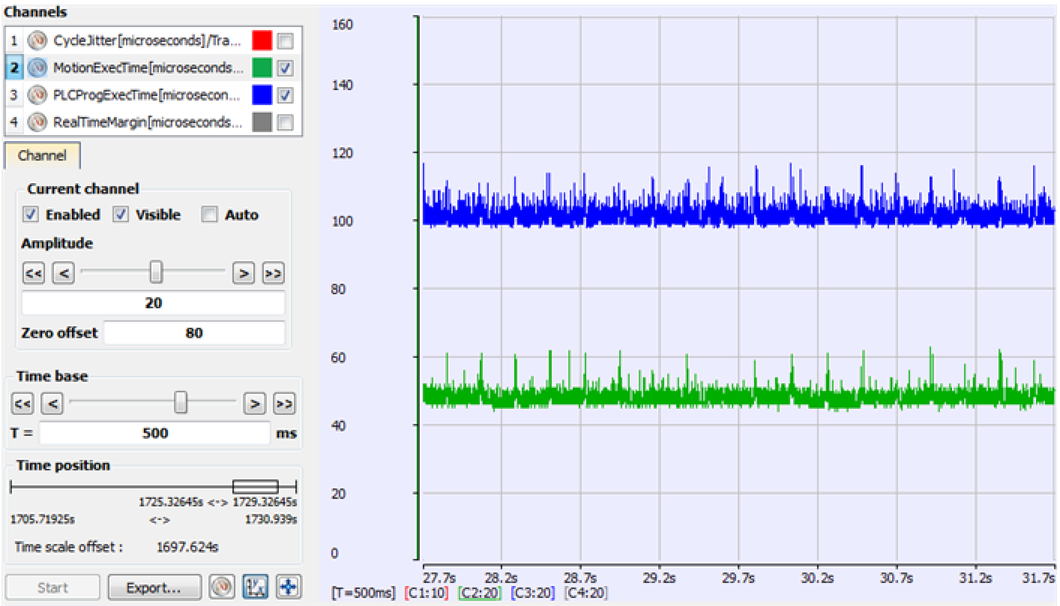


For the "CycleJitter" trace configure the **Amplitude** and **Zero offset** so you can see the trace *centered* at zero. This trace is not too interesting unless a system is misbehaving. A jitter of +/-15 microseconds is acceptable.



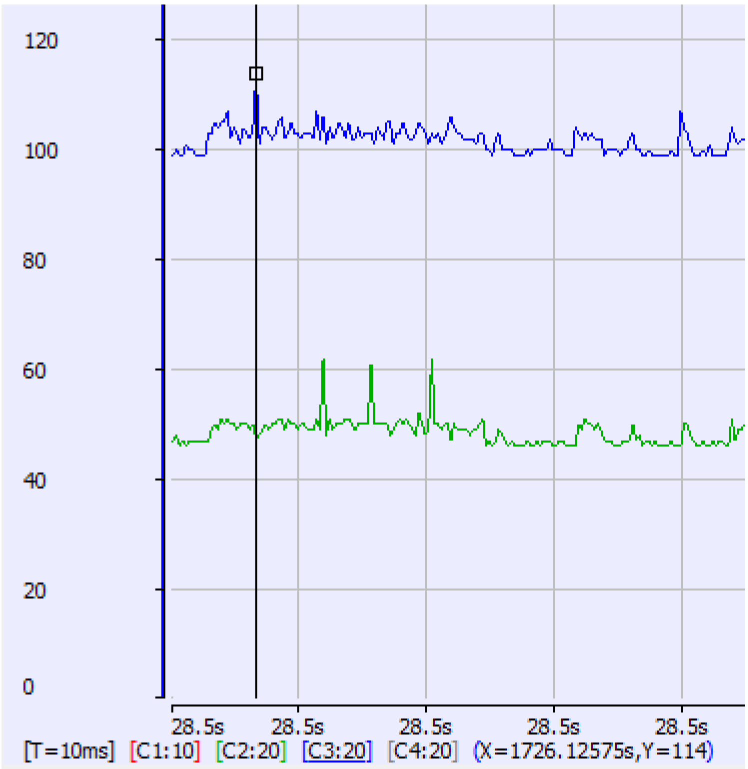
8.3.7.3 Heavily Loaded CPU Example

Here is an example of an application that is heavily loading a PDMM with the EtherCAT Cycle Time = 250 microseconds. Using the techniques described in "Practical Application: Using Trace Time To Measure CPU Load" (see page 423), examine the "MotionExec" and "PLCProgExec" times first:



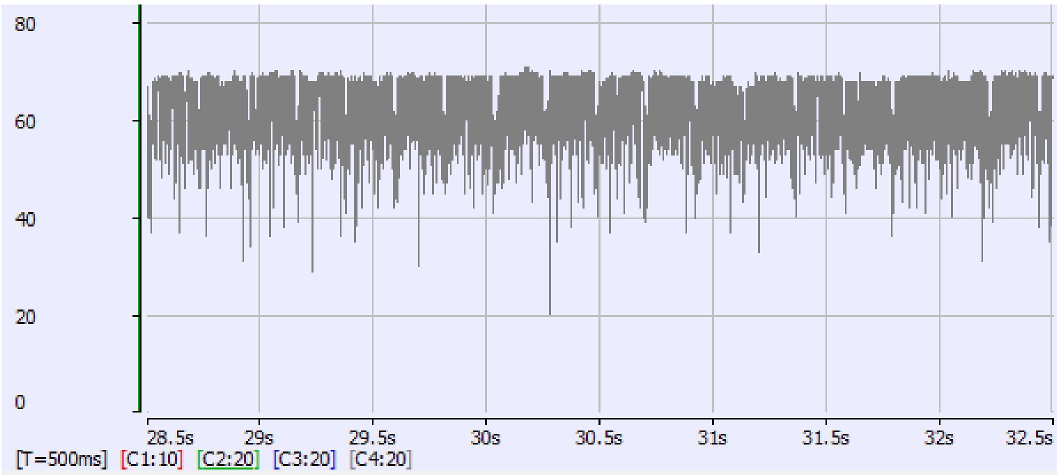
The average MotionExec + PLCProgExec = 50 + 105 = 155 microseconds. This is about 62% (155 / 250) of the cycle time.

Take a look at the peaks:



This shows the MotionExec at 62 microsec and the PLCProgExec at 114; there is not much time left over.

Check the "RealTimeMargin":

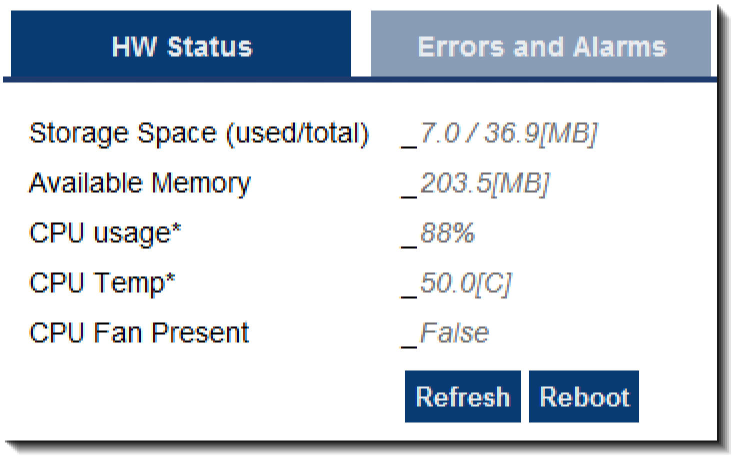


Notice the minimum time is 20 microseconds or 8% Real-Time margin (20 / 250). This is not a comfortable margin for deterministic Real-Time performance.

Checking the Controller log we see that the Virtual Machine (PLCProgExec) is missing a cycle occasionally:

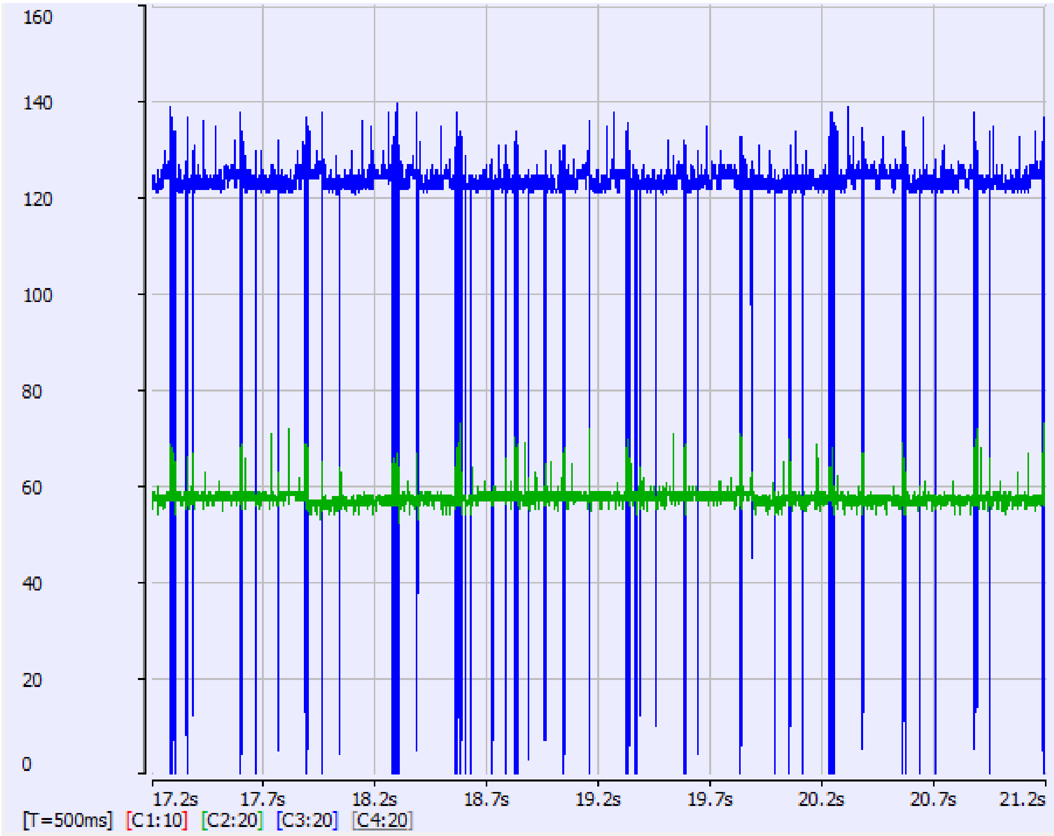
744	4/10/2012 10:37:21 AM (654)	Motion	WARNING	The Virtual Machine missed 1 cycle(s) of PLC execution.
745	4/10/2012 10:37:22 AM (154)	Motion	WARNING	The Virtual Machine missed 1 cycle(s) of PLC execution.
746	4/10/2012 10:37:22 AM (654)	Motion	WARNING	The Virtual Machine missed 1 cycle(s) of PLC execution.
747	4/10/2012 10:37:23 AM (154)	Motion	WARNING	The Virtual Machine missed 1 cycle(s) of PLC execution.
748	4/10/2012 10:37:23 AM (654)	Motion	WARNING	The Virtual Machine missed 1 cycle(s) of PLC execution.
749	4/10/2012 10:37:24 AM (154)	Motion	WARNING	The Virtual Machine missed 1 cycle(s) of PLC execution.
750	4/10/2012 10:37:24 AM (583)	Motion	WARNING	The Virtual Machine missed 1 cycle(s) of PLC execution.
751	4/10/2012 10:37:25 AM (083)	Motion	WARNING	The Virtual Machine missed 1 cycle(s) of PLC execution.
752	4/10/2012 10:37:25 AM (583)	Motion	WARNING	The Virtual Machine missed 1 cycle(s) of PLC execution.
753	4/10/2012 10:37:26 AM (083)	Motion	WARNING	The Virtual Machine missed 1 cycle(s) of PLC execution.

Lastly, take a look at the overall **CPU load**. At 88% usage there's not much CPU bandwidth available.



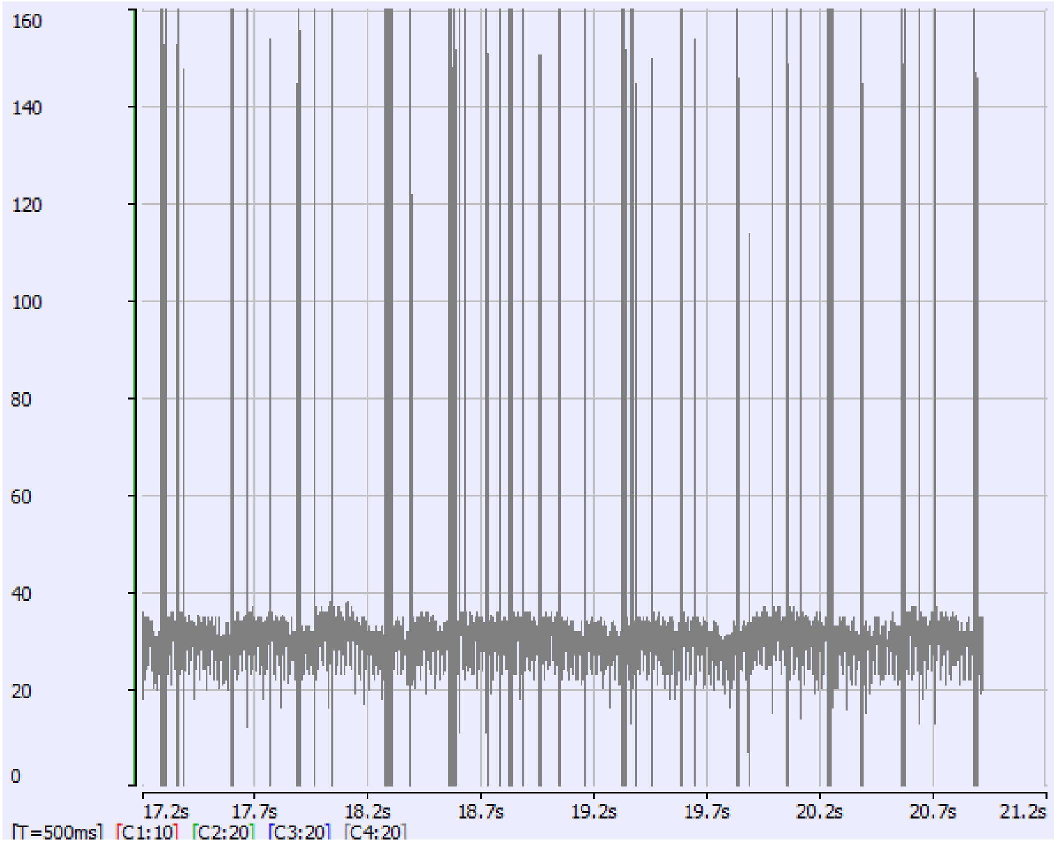
8.3.7.4 Over Loaded CPU Example

Now, let's take a look at an example of an application that is overloading a PDMM with the EtherCAT Cycle Time = 250 microseconds. Using the techniques described above, examine the "MotionExec" and "PLCProgExec" times first:



The average MotionExec and PLCProgExec times are 57 + 125 = 182 or 73% (182 / 250) of the Cycle Time. Notice the big spikes on the PLCProgExec?

Next, look at the "RealTimeMargin":

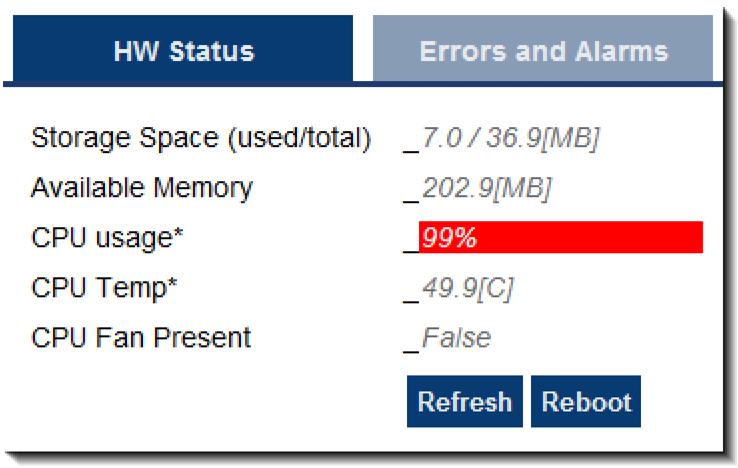


There are many cycles with zero real-time margin. Notice the big spikes? This is a degraded case.

The Controller log confirms the missing VM cycles and an A23 alarm:

992	⚠	4/10/2012 10:52:26 AM (876)	Motion	WARNING	The Virtual Machine missed 68 cycle(s) of PLC execution.
993	⚠	4/10/2012 10:52:27 AM (376)	Motion	WARNING	The Virtual Machine missed 40 cycle(s) of PLC execution.
994	⚠	4/10/2012 10:52:27 AM (876)	Motion	WARNING	The Virtual Machine missed 104 cycle(s) of PLC execution.
995	⚠	4/10/2012 10:52:28 AM (376)	Motion	WARNING	The Virtual Machine missed 64 cycle(s) of PLC execution.
996	⚠	4/10/2012 10:52:28 AM (876)	Motion	WARNING	The Virtual Machine missed 70 cycle(s) of PLC execution.
997	⚠	4/10/2012 10:52:29 AM (376)	Motion	WARNING	The Virtual Machine missed 30 cycle(s) of PLC execution.
998	⚠	4/10/2012 10:52:29 AM (620)	Controller	WARNING	UserInfo : Alarm A23 : CPU is heavily loaded
999	⚠	4/10/2012 10:52:29 AM (876)	Motion	WARNING	The Virtual Machine missed 54 cycle(s) of PLC execution.
1000	⚠	4/10/2012 10:52:30 AM (376)	Motion	WARNING	The Virtual Machine missed 47 cycle(s) of PLC execution.

Lastly, the overall **CPU load** is 99%. Clearly this application is overloading the CPU:



8.4 Human-Machine Interface Editor

This chapter covers the tools you can use to design your HMI panels

- The Kollmorgen Visualization Builder to control your application
- The internal Control Panel editor to debug your application with the KAS Simulator

8.4.1 Using Kollmorgen Visualization Builder

8.4.1.1

To work with Kollmorgen Visualization Builder, do as follows:

- Tag the PLC variables you want to export and map with the HMI (for more details, refer to paragraph "Map Variables to HMI" on page 316)
- Compile your project to generate the Modbus mapping file
- Create a KVB project [1] within the KAS IDE, and open it
- Design your HMI with KVB
- Save and close KVB

  NOTE **Important!** Be sure to use "Save" and not "Save As". The KVB is self-contained within the KAS archive and the Save As function moves the KVB out of the archive.

- Save your KAS project

NOTE When you create the KVB panel with the KAS IDE, all the creation and mapping procedure is done automatically after compiling your project. So you can directly go to the paragraph "Design the Panel" on page 436.

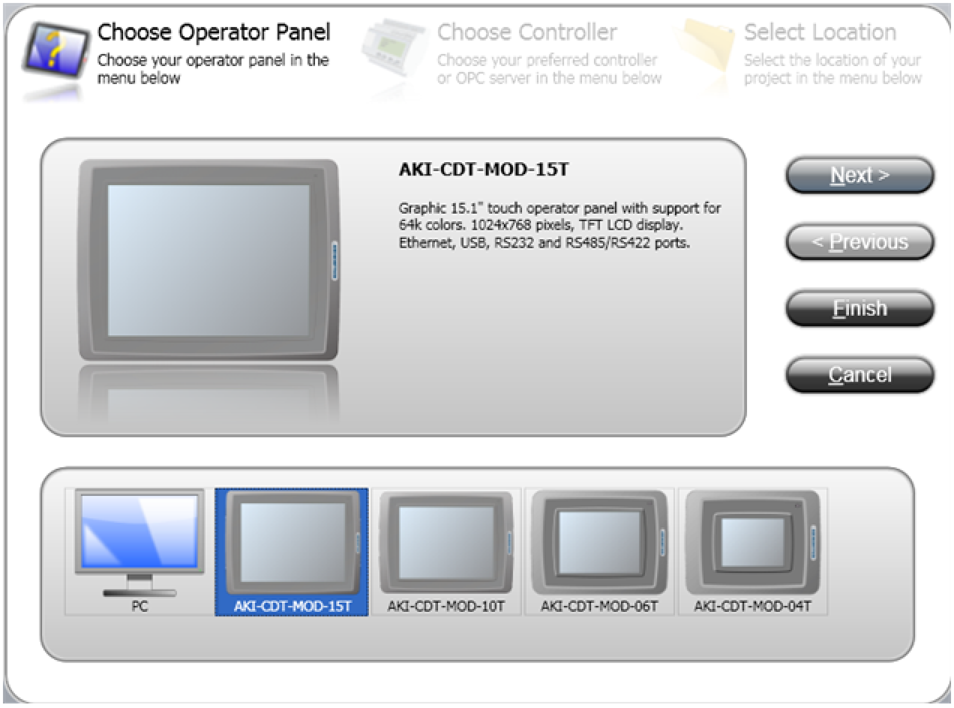
IMPORTANT Be aware that as soon as you change the PLC variables exported for the HMI, the mapping file must be re-imported in Kollmorgen Visualization Builder to have an up-to-date version.

[1] There is no built-in feature to import/export KVB projects

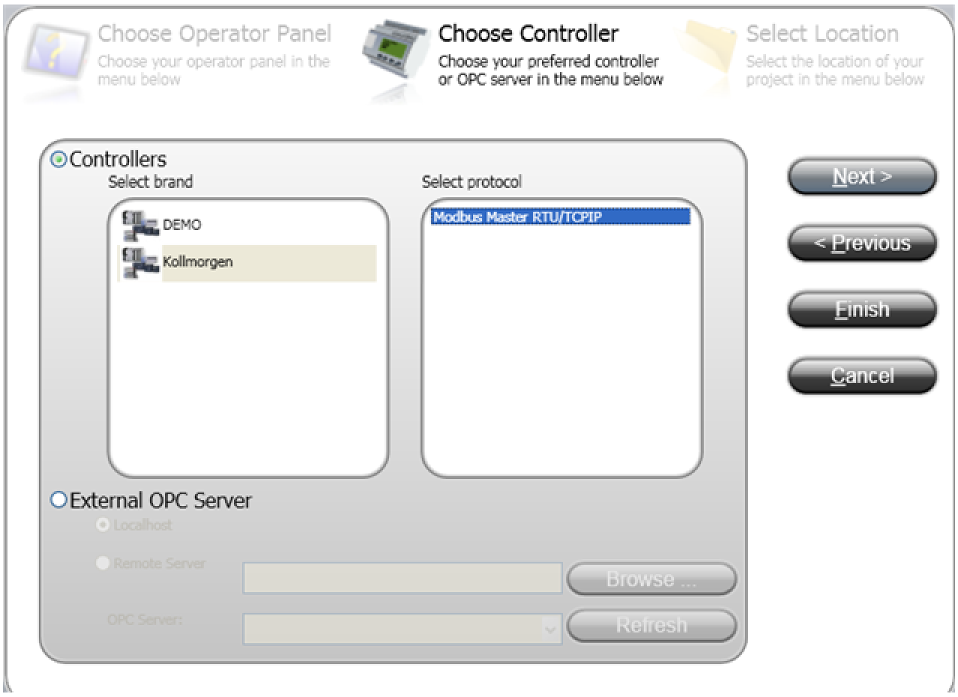
8.4.1.2 Create a new controller

This procedure is applicable when you use Kollmorgen Visualization Builder externally.

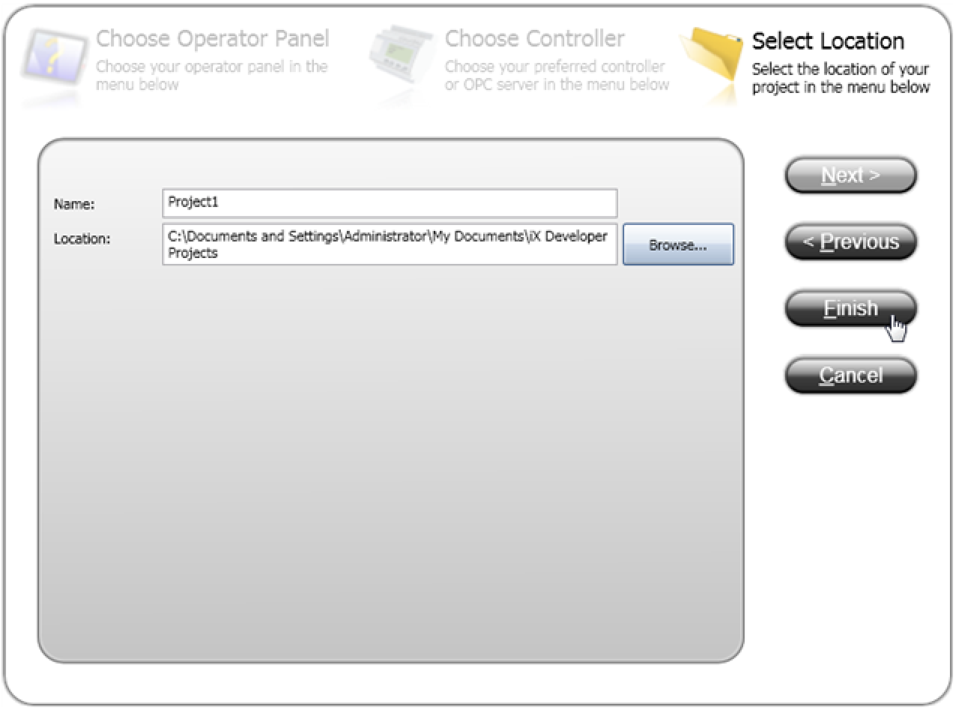
- After choosing to create a new project, select the type of operator panel to be used



- On the next dialog, select the **Kollmorgen** controller with the Modbus protocol, then click the **Next** button



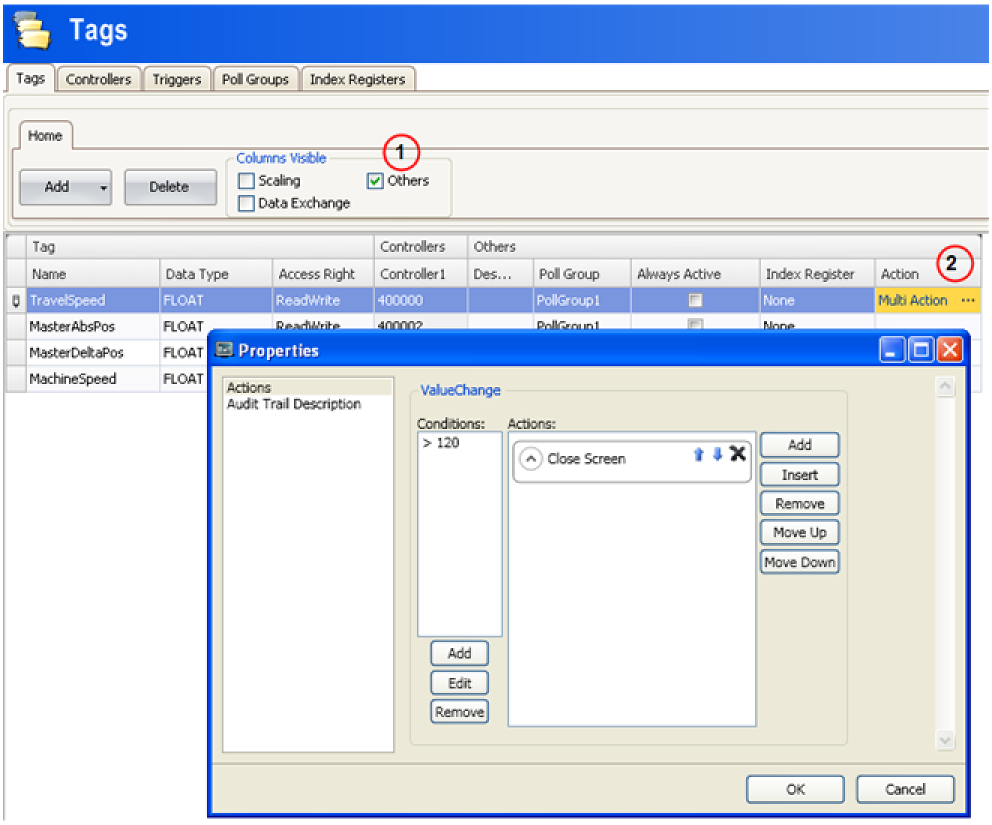
- Enter the name of the project and where you want to create the project. Then click the **Finish** button



8.4.1.3 Import variables into the project

When you open the Kollmorgen Visualization Builder with your KVB panel (by double-clicking the KVB panel from the project explorer) all the variables tagged into the Dictionary are automatically imported into Kollmorgen Visualization Builder. Once the file is imported, all PLC variables are available for use within Kollmorgen Visualization Builder.

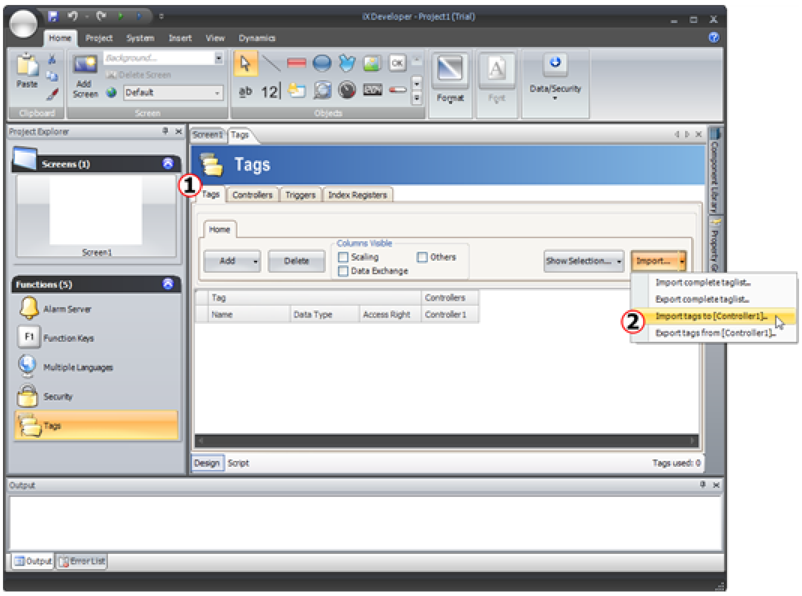
1. Select **Others** to display the Action column
2. You can edit the tag actions



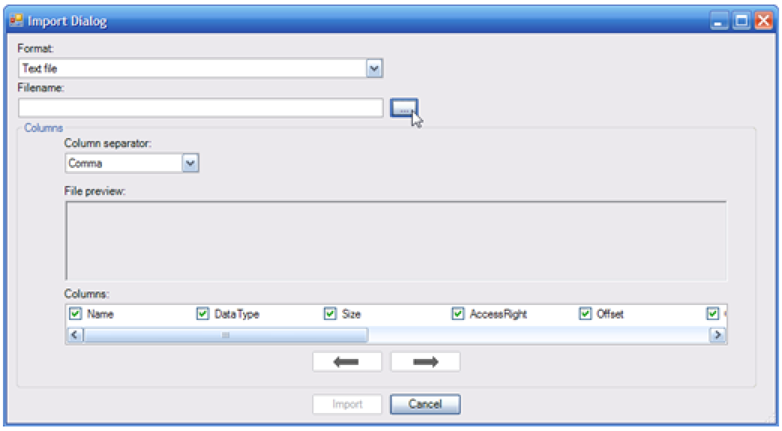
To sort out the limitation stated above, you need to manually export/import the variables (tags) of your project.

The import procedure is as follows:

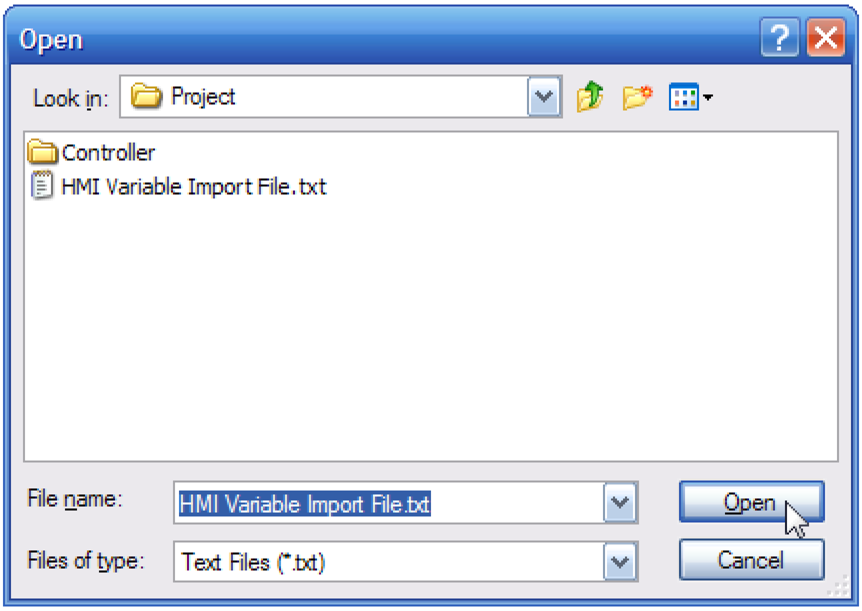
- Select the **Tags** tab
- Click the arrow of the **Import** button, then select **Import tags to [Controller1]…** in the drop-down menu



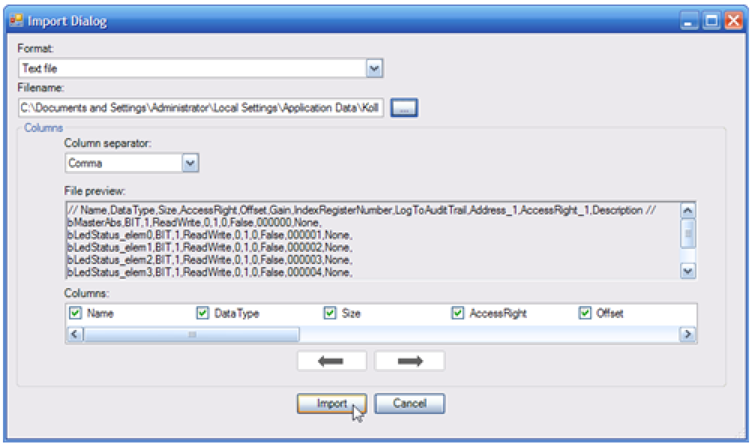
- In the import dialog, specify the filename by clicking the **...** button



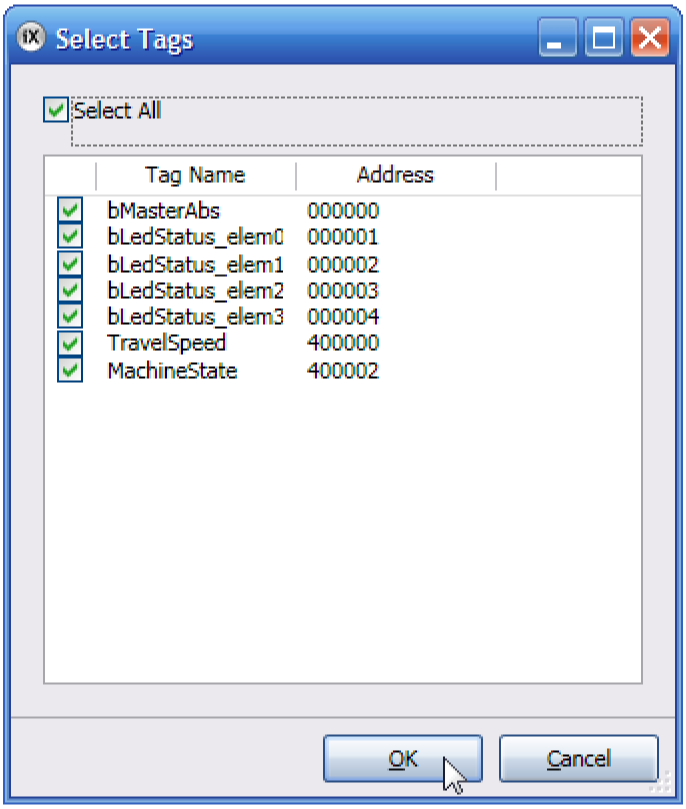
- Then use the open file dialog to find the **.txt** file
- Once the file is specified, click the **OK** button



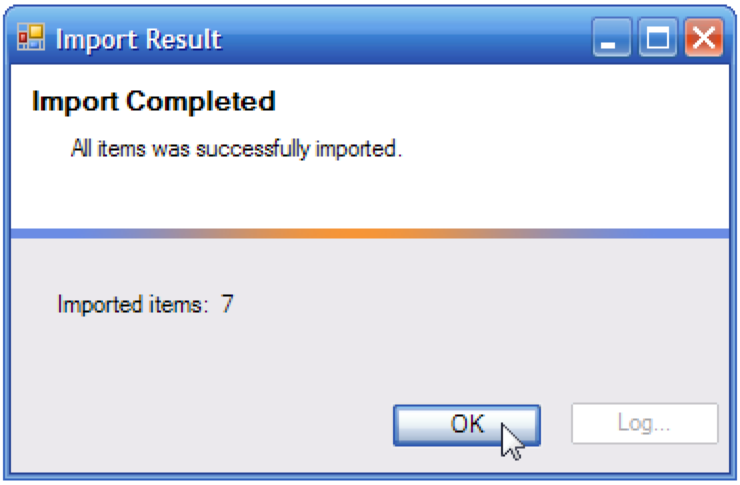
- Back in the import dialog, make sure the **Column separator** is set for Comma, and leave all options selected
- Then click the **Import** button



- Specify which tags (variables) you want to import. To select all tags, click the **Select All** option
- When you have finished selecting the tags, click the **OK** button



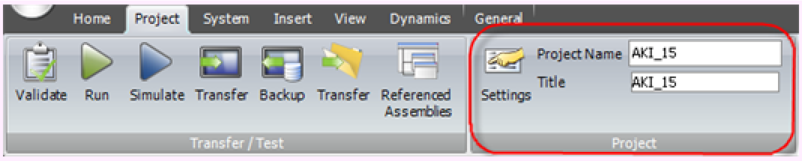
- You are now notified how many items are successfully imported. Click the **OK** button to return to the project



8.4.1.4 Design the Panel

IMPORTANT Do not modify **Project Name** and **Title** to keep consistency between Kollmorgen Visualization Builder and the KAS IDE.

IMPORTANT



Add Object

You can drag-and-drop predefined objects from the library to the screen. The library is located in the **Home** tab of Kollmorgen Visualization Builder.

Customize Object

Select an object and click the **General** tab to customize:

- its settings in the Settings section
- its style to a different template in the Style section

Map Variable to the Object

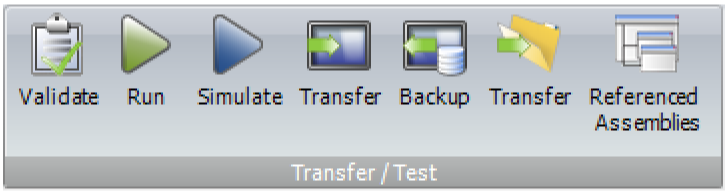
In the **General** tab, you can set the Variable or Tag that maps to the current object in the Tag/Security section.

NOTE Click the **F1** key to open the Kollmorgen Visualization Builder online help (or use the Help button in the ribbon tab heading)

IMPORTANT Be aware that as soon as you change the PLC variables exported for the HMI, the mapping file must be re-imported in Kollmorgen Visualization Builder to have an up-to-date version.

8.4.1.5 Download the Panel

To download your panel you have to use the Project ribbon in Kollmorgen Visualization Builder that contains the **Transfer** command .

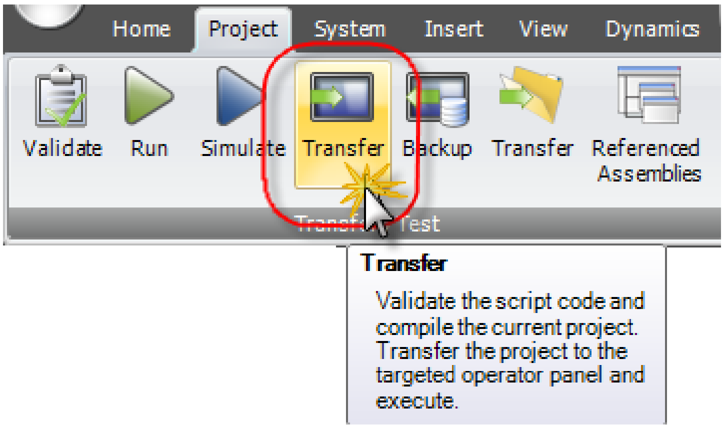


Command	Description
Validate	Compiles the project to check for errors
Run	Validates the program, runs it on the development computer and communicates with the PAC
Simulate	Validates the program, runs it on the development computer, but does not communicate with the PAC
Transfer (1st icon)	Is for projects with a dedicated HMI device (AKI) Validates the current project and sends it to the selected hardware
Transfer (2nd icon)	Validates the project and saves it to a folder with an executable program that can be run on a PAC with the HMI runtime (Visualizer RT) installed or a dedicated HMI panel

NOTE For more details, refer to the online help in Kollmorgen Visualization Builder

How to download on the HMI device (AKI)

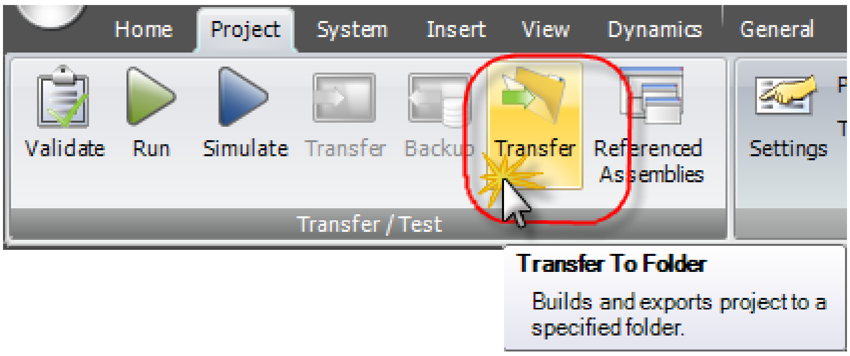
To download you must be connected directly to the HMI panel through an Ethernet cable. As the IP address is already defined (for more details, see page 188), nothing special has to be done before transferring your panel to the graphic operator terminal.



NOTE If you transfer your project on a USB stick, place it in the USB port of the AKI panel while it is booting up.

How to download on the PAC (AKC)

- Click the **Transfer** button (as shown below) to validate your HMI project.



- Select a desired location to save the project (a folder is created with all of the necessary files along with one executable program)
- Place this folder on a USB stick
- Copy this folder anywhere on the PAC hard disk
- Ensure you have Visualizer RT installed on the PAC (with a USB stick containing a valid license key)

TIP For an easy access, you can add a shortcut to the executable program on the desktop, or to the windows startup folder so it launches automatically when the PAC boots up.

8.4.1.6 Related Documents

For further information on Kollmorgen Visualization Builder, refer to the following manual:

KVB Guide	Description
Kollmorgen Visualization Builder™ Quick Start Guide	Quick Start that covers the most important points to install and use Kollmorgen Visualization Builder, in order to configure HMI Panels and PC operated control applications, including applications for PACs
Kollmorgen Visualization Builder™ User Manual	Contains all the content to help you with Kollmorgen Visualization Builder

8.4.2 Design the Control Panel with the Internal Control Panel Editor

This section details the Controls and Properties used to define the Control Panel when you need to debug your application, as well as the procedure to map variables to Control Panel controls.

8.4.2.1 Create Control Panel

Control Panel are managed in the Project Explorer and can be created as follows:

1. In the Project Explorer, right-click the Controller item to open the menu
2. Select the **New Control Panel** command
3. Right-click on the newly created item and select the **Rename** command to change its name
4. Double-click the new Control Panel to open it in the graphical editor

8.4.2.2 Use the Control Panel control library

- Select a control in the Libraries toolbox (Controls tab) and drag-and-drop it in your Control Panel.

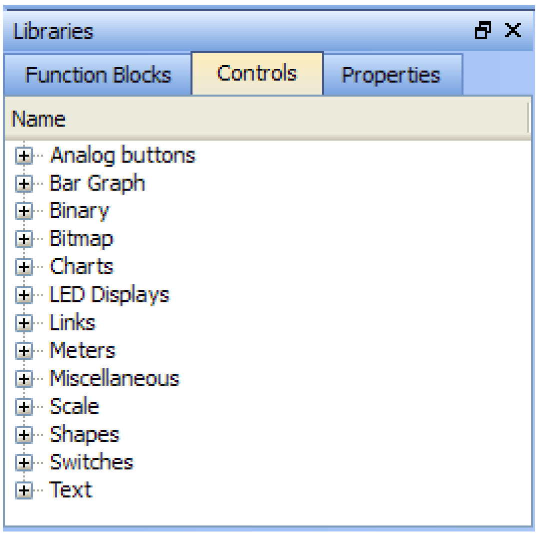


Figure 9-32: Control Panel Control Library

For an exhaustive list of controls, refer to "Graphic Objects" (see page 442).

8.4.2.3 Edit the Control panel

- When a control is selected, you can change its properties (displayed in the Libraries toolbox) by double-clicking the **Value**

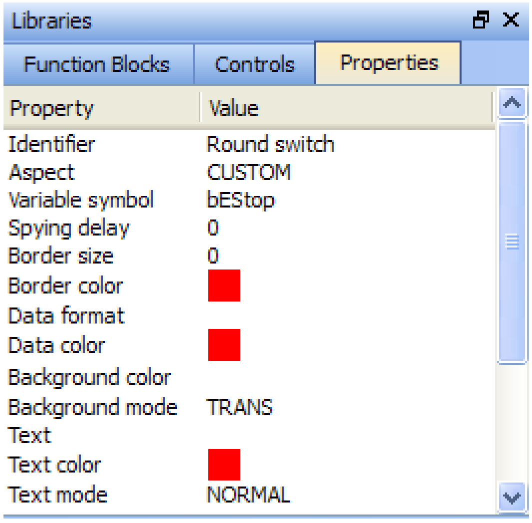


Figure 9-33: Control Panel Control Properties

For an exhaustive list of properties, refer to paragraph "Graphic Objects Properties" on page 449

NOTE

- You can perform multi-selection with the mouse (all the controls that are even partly inside the selection area are selected)
- You can add controls to your selection either with the **Ctrl** or **Shift** keys
- You can use Arrow keys to move the Control Panel page Up, Down and sideways.
- You can use Shift + Arrow keys to move the selected Control up-down and sideways

TIP To duplicate all the selection, hold down **Ctrl** and click the right mouse button while performing your move operation (do not forget to release the mouse button first, before the **Ctrl** key).

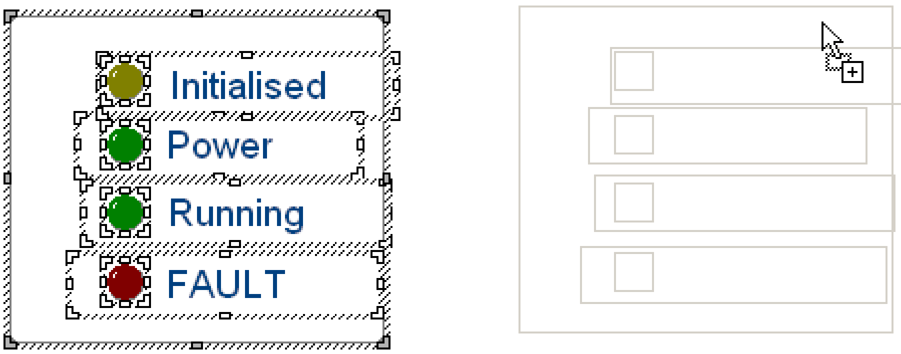
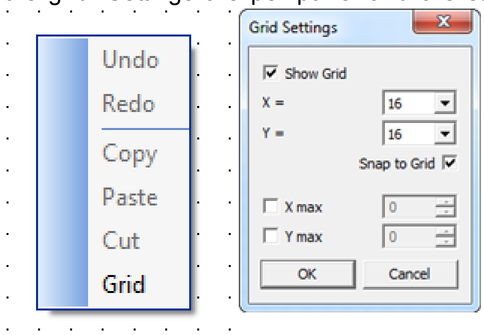


Figure 9-34: Control Panel - Selection of Controls

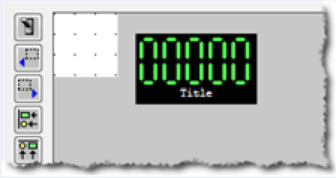
HMI Grid Settings

Right-clicking in the Control Panel's graphical editor provides access to the Grid Settings by selecting Grid. This may also accessed by pressing Ctrl-G. The settings allow you to control the appearance of the grid as well as forcing objects to "snap" to the grid. Settings are per panel and are saved with the project.



Show Grid	Display or hide the grid in the Control Panel
X= / Y=	Space between each grid dot in X/Y directions
Snap to Grid	Controls will be forcibly aligned with the grid when they are placed/moved
X max / Y max	Set a maximum area in which the grid is shown. The editor is grayed out beyond these dimensions.

NOTE Objects may still be placed within the grayed out area as it is merely a visual aid.



8.4.2.4 Mapping Variables to the Control Panel

How do I define a variable for PLC programs?

To link your Control panel with the PLC programs, some controls contain a property called **Variable symbol**

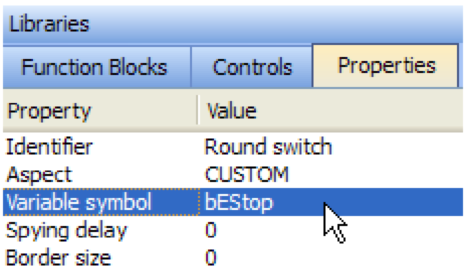


Figure 9-35: Map variables to a Control Panel control

To map the variable:

1. Select the variable in the Dictionary toolbox

2. Move it to the control to be linked in the Control panel editor using drag-and-drop

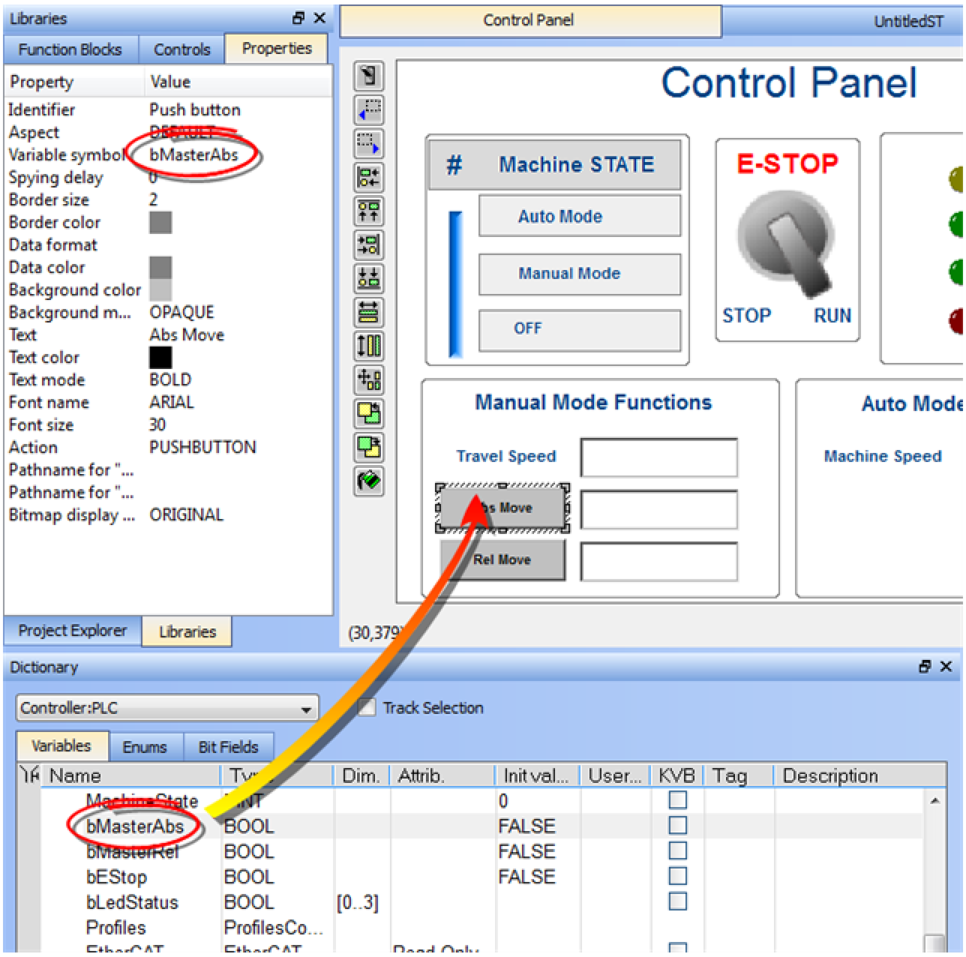


Figure 9-36: Map Variables to a Control Panel Control in the Graphical Editor

The Variable symbol is automatically updated in the **Properties** tab.

> **NOTE**
> A warning will be generated when the program is compiled if the control is mapped to a variable which is not in the Dictionary. Double-clicking the warning will highlight the control object and open the editor so the variable can be defined.

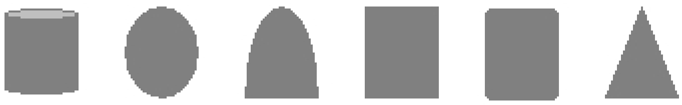
8.4.2.5 Graphic Objects

Below are available basic objects you can insert in your graphics:

Analog buttons
Bar Graphs
Bitmaps
Charts
Connection status
LED displays
Links

Meters...	Analog Meters Digital Meters
Scales	
Shapes	
Sliders	
Switches	
Text	

Basic shapes



A collection of basic drawings is available. Each object can be either static, or linked to a variable used to enable its visibility (show/hide).

Properties:

Identifier
Aspect
Variable symbol
Spying delay
Border size
Border color
Data format
Color when not connected
Background color
Background mode
Text
Text color
Text mode
Font name
Font size
TRUE color
FALSE color
Direction

Bitmaps

Bitmap file (BMP, GIF, JPG) can be inserted in the graphic area.

Properties:

Identifier
Border size
Border color
Border style
Background color
Background mode
Text
Text color
Text mode
Font name
Font size
Pathname
Bitmap display mode

NOTE Large bitmaps are time-consuming during animation and can lead to poor performance, mainly if they have the "STRETCH" display mode or the "TRANS" (transparent) background mode.

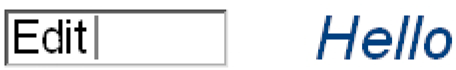
Scales

Scales are static drawings representing an X or Y axis, generally used to document other objects such as trend charts or bargraphs.

Properties:

Identifier
Border size
Border color
Border style
Background color
Background mode
Text
Text color
Text mode
Font name
Font size
Minimum value
Maximum value
Direction
Placement
Nb divisions (main)
Nb divisions (small)
Scale color

Text boxes

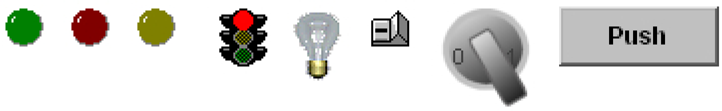


Static, animated or edit text boxes are available for displaying / forcing variables. For edit boxes at runtime, double-click on the object to enter the value and then hit ENTER to validate the input.

Properties:

Identifier
Variable symbol
Spying delay
Border size
Border color
Border style
Data format
Background color
Background mode
Text
Text color
Text mode
Font name
Font size
Action

Switches and 2-state displays

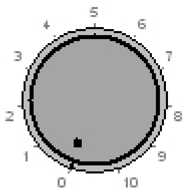


Buttons, switches and 2-state displays are used for control or display of a boolean variable.

Properties:

Identifier
Aspect
Variable symbol
Spying delay
Border size
Border color
Data format
Data color
Background color
Background mode
Text
Text color
Text mode
Font name
Font size
Action
Pathname for "TRUE" state
Pathname for "FALSE" state
Bitmap display mode

Analog buttons



Analog buttons are used for setting the value of an integer or real variable. The mouse is used for setting the value.

Properties:

Identifier
Variable symbol
Spying delay
Border size
Border color
Border style
Data format
Data color
Background color
Background mode
Text
Text color
Text mode
Font name
Font size
Minimum value
Maximum value
Scale color

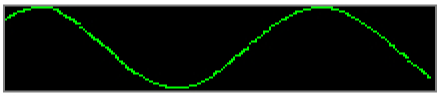
Bargraphs

Bargraphs are rectangles filled according to the value of an analog variable. Bargraphs can be horizontal or vertical.

Properties:

Identifier
Variable symbol
Spying delay
Border size
Border color
Border style
Data format
Data color
Background color
Background mode
Text
Text color
Text mode
Font name
Font size
Minimum value
Maximum value
Direction

Charts

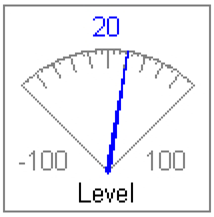


Charts enable the tracing of a variable as with an oscilloscope.

Properties:

Identifier
Aspect
Variable symbol
Spying delay
Border size
Border color
Border style
Data format
Data color
Background color
Background mode
Text
Text color
Text mode
Font name
Font size
Minimum value
Maximum value
Nb of points

Analog meters

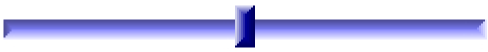


Analog meters provide a graphical display of an analog value.

Properties:

Identifier
Variable symbol
Spying delay
Border size
Border color
Border style
Data format
Data color
Background color
Background mode
Text
Text color
Text mode
Font name
Font size
Minimum value
Maximum value
Scale color
Nb divisions (main)
Nb divisions (small)

Sliders

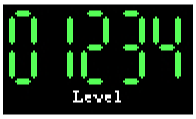


Sliders are used for entering an analog value with a horizontal or vertical mouse driven cursor.

Properties:

Identifier
Variable symbol
Spying delay
Border size
Border color
Border style
Data format
Data color
Background color
Background mode
Text
Text color
Text mode
Font name
Font size
Minimum value
Maximum value
Scale color
Direction

Digital meters



Digital meters (digits) display the value of a variable with the same aspect as a digital clock.

Properties:

Identifier
Aspect
Variable symbol
Spying delay
Border size
Border color
Border style
Data format
Data color
Background color
Background mode
Text
Text color
Text mode
Font name
Font size
Minimum value
Maximum value

Links

Back to main page

Links are mouse-driven hyperlinks that are used as shortcuts to open another graphic document. Using links enables the design of multi-page animated applications.

Properties:

Identifier
Border size
Border color
Background color
Background mode
Text
Text color
Text mode
Font name
Font size
Link

Connection status

Connection status is a box actuated with the current status of the connection and the connected run-time application. It is mainly dedicated to diagnostic.

Properties:

Identifier
Spying delay
Border size
Border color
Border style
Data format

Data color
Background color
Background mode
Text
Text color
Text mode
Font name
Font size

Binary

BitsField allows you to display a Real value into a binary form.

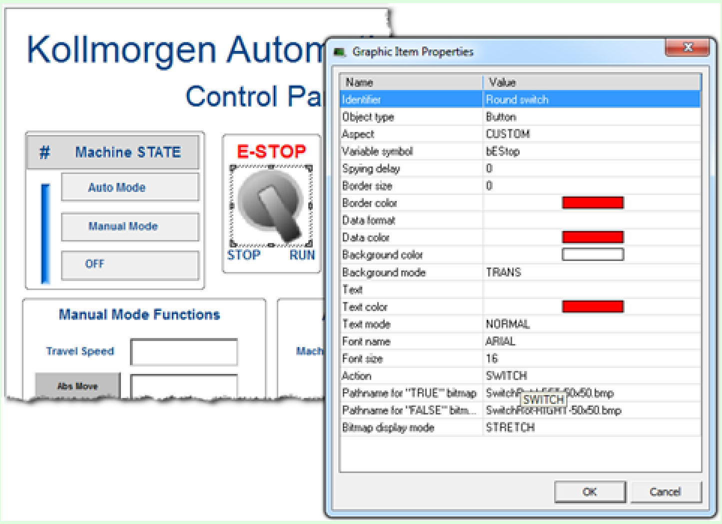
The main properties are:

- the associated variable (an integer)
- **SETNBBYTE** that indicates the number of bytes to display. If that number is less than the real size of the associated variable, then the LSB (Least Significant Bytes) are displayed.

8.4.2.6 Graphic Objects Properties

This page details all possible properties for graphic objects. Refer to the list of available objects for further information on which property is used for which object.

TIP Double-clicking an object will open the Graphic Item Properties dialog box, which lists the properties particular to that object.



Identifier	You can freely attach a text identifier to each graphic object inserted in a document. Identifiers are useful for arranging overlapped objects as they appear in the "Z-order" list.
Variable symbol	It is the full name of the application variable connected to the graphic object. In case of a local variable, its symbol must be prefixed with the parent program name, separated with "/". Example: "MyProg/MyVar".
Spying delay	It is the minimum period for actuating the value of the connected variable, expressed as a number of milliseconds. If the delay is not specified or equal to 0, refresh is done as fast as possible.

Border size	This property indicates the width of the border drawn around the object, expressed as a number of pixels. If this property is 0, then no border is drawn.
Border color	This property indicates the color of the border drawn around the object.
Border style	This property indicates the possible 3D effect used for drawing the border around the object. Possible values are: FLAT = no 3D effect 3DUP = depressed 3D effect 3DDOWN = pressed 3D effect 3D = default 3D effect
Text color	This property indicates the color used for inserting texts in the graphic object.
Text mode	This property indicates the font effect used for drawing texts in the graphic object. Possible values are: HIDE = text is not displayed NORMAL = normal font BOLD = bold text ITALIC = italic text UNDERLINE = underlined text
Font name	This property indicates the name of the character font used for drawing texts in the graphic object.
Font size	This property indicates the size of the character font used for drawing texts in the graphic object. The size is expressed as a percentage of the actual height of the object. Maximum possible value is 100. This ensures that the ratio is kept when the object is resized.
Background color	This property indicates the color used for filling the background of the object. In case of a bitmap, it specifies the color that must not be drawn if the TRANS (transparent) background mode is specified.
Background mode	This property indicates whether the background of the object must be filled or not. If this property is OPAQUE, then the background is filled with the specified background color. If this property is TRANS (transparent) then the background is not filled. Transparent drawing mode can be useful in the case of overlapping objects. **IMPORTANT** Specifying the TRANS (transparent) mode for large bitmaps is time-consuming and will affect the real-time performances of graphic updates.

Data format	If defined, this property indicates that the value of the connected variable must be displayed on the graphic object. You must specify for this property a format string that indicates how the data will be formatted. **IMPORTANT** The "**text**" property is ignored when a data format is specified. Format string has the same format as the famous "printf" function of "C" language. It can include static characters together with one of the following possible pragmas that specify the value: %s = default formatting according to IEC syntax %d = integer (decimal) %X = hexadecimal %g = floating point %.*n*f = decimal real (*n* is the number of displayed decimal digits) Below are some examples: **Format** / **Value** / **Displayed string** %d / 12.3 / 12 Var = %g meters / 1.2 / Var = 1.2 meters %.2f / 1.12345 / 1.12 **NOTE** Only one % pragma can be used in a string.
Text	If defined, this property indicates the text to be displayed on the graphic object. **IMPORTANT** This property is ignored when a **data format** is specified.
Bitmap display mode	For bitmap-based objects, this property indicates whether the attached bitmap must keep its original aspect or be stretched to the actual size of the object. Possible values are: ORIGINAL = keep the original aspect of the bitmap (cut if too large) STRETCH = stretch or shrink the bitmap for fitting the actual size of the graphic object **IMPORTANT** Large bitmaps with "STRETCH" display mode are time-consuming during animation and can lead to poor performance.
Minimum value	For analog animated objects (meters, bargraphs or trends) this property indicates the minimum possible value that can be displayed. For static scales, it indicates the value of the lowest mark.
Maximum value	For analog animated objects (meters, bargraphs or trends) this property indicates the maximum possible value that can be displayed. For static scales, it indicates the value of the highest mark.
Data color	This property indicates the color used to represent the value of a connected variable within the object (for example the filled part of a bargraph).
Nb divisions (main)	For objects including a graphic scale, this property indicates the number of main division marks to be drawn in the scale.

Nb divisions (small)	For objects including a graphic scale, this property indicates the number of small division marks to be drawn in the scale, between each main division mark.
Scale color	For objects including a graphic scale, this property indicates the color used for drawing the axis, the division marks and corresponding values of the scale.
Bitmap pathname	For bitmaps, this property specifies the pathname of the bitmap to be displayed. BMP, GIF and JPG formats are supported. If no directory is specified, the specified file name is searched: • in the project folder • in the "\BITMAP" folder of the KAS IDE
Bitmap for "TRUE" state	For two-state objects having the "CUSTOM" aspect, this property specifies the pathname of the bitmap to be displayed when the value of the attached variable is TRUE (or not zero for analogs). BMP, GIF and JPG formats are supported. If no directory is specified, the specified file name is searched: • in the project folder • in the "\BITMAP" folder of the KAS IDE
Bitmap for "FALSE" state	For two-state objects having the "CUSTOM" aspect, this property specifies the pathname of the bitmap to be displayed when the value of the attached variable is FALSE (or zero for analogs). BMP, GIF and JPG formats are supported. If no directory is specified, the specified file name is searched: • in the project folder • in the "\BITMAP" folder of the KAS IDE
Color when not connected	For shapes, this property indicates the color used for filling shapes when no variable is attached to the graphic object.
TRUE color	For shapes, this property indicates the color used for filling shapes when the attached variable has the TRUE state, or non zero for analogs.
FALSE color	For shapes, this property indicates the color used for filling shapes when the attached variable has the FALSE state, or zero for analogs.
Direction (basic shapes)	For oriented shapes such as triangles, half ellipses or cylinders, this property indicates the direction of the drawing; to the left, to the right, to the top or to the bottom.
Direction (scale)	For scales, this property indicates the direction of the axis. If LEFT, the minimum value is on the left side. If RIGHT, the minimum value is on the right side.
Placement (scale)	For scales, this property indicates the location of the scale within the object rectangle: on the left, on the right, on the top or at the bottom.
Action (text)	Indicates the possible mouse actions for text boxes. The following values are possible: STATIC = no mouse action EDIT = double-click opens an edit box for entering the variable value

Action (switch)	Indicates the possible mouse action for switches. The following values are possible: STATIC = no mouse action PUSHBUTTON = the variable is forced to TRUE when pressed and to FALSE when released SWITCH = the status of the variable is inverted when the button is pressed ONESHOTBUTTON = same as switch, but the display continues to appear released
Direction (bargraph)	For bargraphs, this property indicates the growing direction: to the left, to the right, to the top or to the bottom.
Nb of points (trends)	For trend charts, this property indicates the maximum number of stored points. If the width of the object (in pixels) is less than this number, then oldest points are not visible.
Direction (slider)	For slider, this property indicates whether the slider is horizontal (RIGHT) or vertical (TOP).
Link	This property indicates the name of the target .GRA animated document for shortcuts. If no directory is specified in the link, then the file is searched in the project folder.
Aspect (shapes)	This property indicates the type of basic shape to be drawn. Possible aspects are: CYLINDER = a 3D like cylinder ELLIPSE = an ellipse HALFELLIPSE = one half of an ellipse GATE = a simple vector drawing for a valve RECTANGLE = a rectangle ROUNDRECT = a rectangle with rounded corners TRIANGLE = a triangle
Aspect (switches)	This property indicates the type of switch to be drawn. Possible aspects are: DEFAULT = a standard Windows-like push button CUSTOM = a button with TRUE and FALSE drawings defined with bitmaps
Aspect (trend charts)	This property indicates the type of drawing for a trend chart. Possible aspects are: POINT = only relevant dots are drawn LINE = lines are drawn from point to point HISTO = histogram style
Aspect (digits)	This property indicates the type of drawing for a digital meter. Possible aspects are: DEFAULT = plain drawing BEZEL = all segments have a 3D effect

8.4.2.7 Operate the Control Panel

The Example program has a default control panel built-in to make it easy to start an application.

Perform the following steps to operate the control panel:

1. Double-click on **Control Panel** in the Project Explorer to open the form

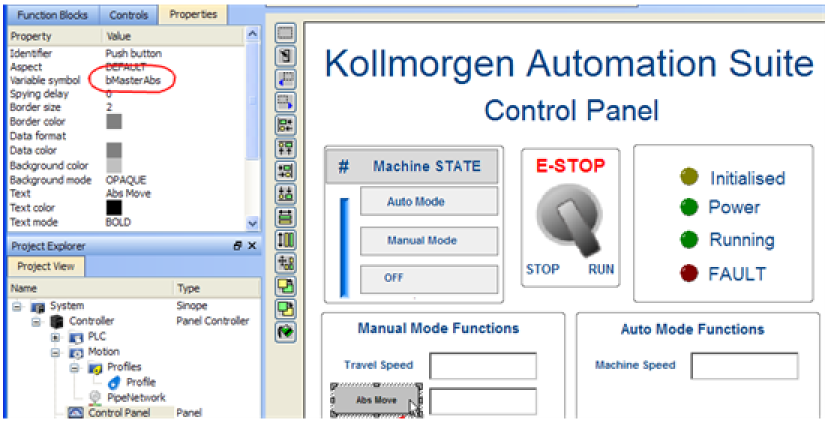


Figure 9-37: Control Panel

2. Start by moving the vertical slider bar to select the Machine STATE as **Manual Mode**
3. In the Manual Mode Functions area, double-click the text box for the Travel Speed
4. Enter the numeric value for the Travel Speed and press Enter

About KAS Simulator Display

The KAS Simulator displays the status and position of the axes.
It also displays the log messages.

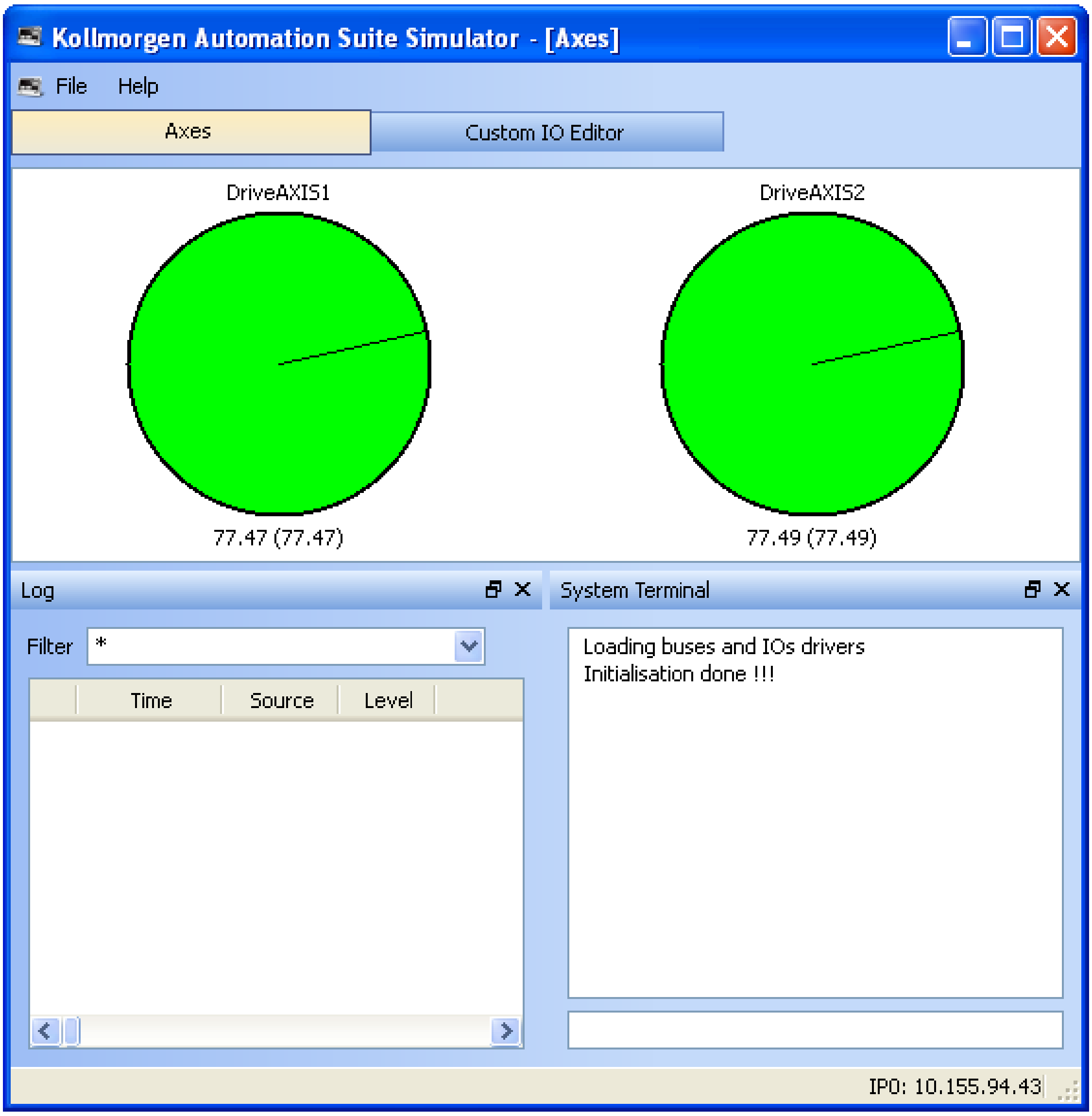


Figure 9-38: Display of KAS Simulator

You can continue to use the Control Panel to:

- Experiment with the controls and observe the simulated output
- Perform an absolute move by entering a position in the text box
- Perform a relative move

8.4.2.8 Exiting Simulation Mode

To exit Simulation mode, do as follows:

1. Click the Stop Device button 
2. Click the Disconnect Device button 

This concludes the 30 minutes to motion tutorial.

NOTE For additional information about Kollmorgen Automation Suite, see the following documentation:

- Getting Started
- User Manual

NOTE

- Technical Reference - PLC Library
- Technical Reference - Motion Library
- Online Help

8.5 Custom Input/Output Editor

NOTE This tool is reserved for Profibus fieldbuses only.

The Input/Output Editor (hereafter I/O Editor) is a tool used to declare and set up I/O devices, and establish the link between the application variables and physical equipment.

It shows a list of the currently defined I/Os.

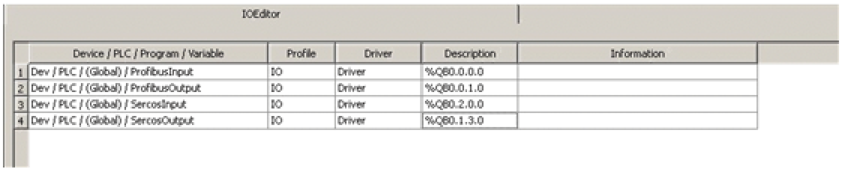


Figure 9-39: Input/Output Editor

For the **Description** field, see format explanations.

8.5.1 Add Input/Output

To add an I/O, simply drag-and-drop a variable from the dictionary to the I/O editor, then modify it.

8.5.2 Modify Input/Output

To modify an I/O:

1. Double-click the cell you want to edit

TIP You can also use the arrow keys to select the cell and press the **F2** key to start edition.

2. Set its driver name to the one of your choice, for example: **CIFDriver** (column 3)
3. Set its description to the corresponding driver address (column 4)

The description field has the following format...

• It begins with a "**%**" character	
• Followed by the type of I/O	**I**: input **Q**: Output
• Followed by the size of I/O	**X**: Boolean (1 bit) **B**: byte (8 bits) **W**: word (16 bits) **D**: double word (32 bits) **L**: long word (64 bits)
• Followed by its address on the selected bus	The address has the following format: "**deviceId.slaveId.moduleId.bitOffset**", where deviceId, slaveId, moduleId and bitOffset are integers ranging from 0 to 65535 NOTE: set **deviceId** to 0 set **slaveId** to the id of the I/O node set **moduleId** to the id of the slice **bitOffset** must always be 0 for non-Boolean I/Os

NOTE

- The sizes of the variable and the I/O must be the same.
- The bitOffset must always be 0 for non-Boolean I/Os.

Example:

%IX0.1.2.4 is an input Boolean located on deviceId=0, slaveId=1, moduleId=2 at bitOffset=4

%QB0.1.2.0 is an output byte located on deviceId=0, slaveId=1, moduleId=2

NOTE

If you enter an invalid text, the table cell becomes red, and an explanation is also displayed in the **information** column.

See also "Step 11 of 15 - Map Input and Output to Variables" on page 270

8.5.3 Delete Input/Output

To delete an I/O:

1. Click somewhere on the I/O's row (or go to the row with the up/down arrow keys)
2. Press the delete key
3. Confirm the deletion.

9 Advanced Topics

9.1 Coordinated Motion

Coordinated motion in KAS IDE is discussed in several locations and manners.

- "Overview" (see page 459) — this section helps you to understand the concepts behind Coordinated Motion and the terminology associated with Coordinated Motion.
- "How-To: Coordinated Motion" (see page 462) — this section helps you to get started quickly with coordinated motion by walking you through the steps of setting up a project that uses coordinated motion.
- Functions and Function Blocks — this is the reference section for function block parameters.

9.1.1 Overview

This section provides an overview of Coordinated Motion, including general concepts you will need to understand to use Coordinated Motion.

- "Coordinated Motion Terminology" (see page 459)
- "Group State Diagrams" (see page 461)
- "Coordinate Systems" (see page 461)

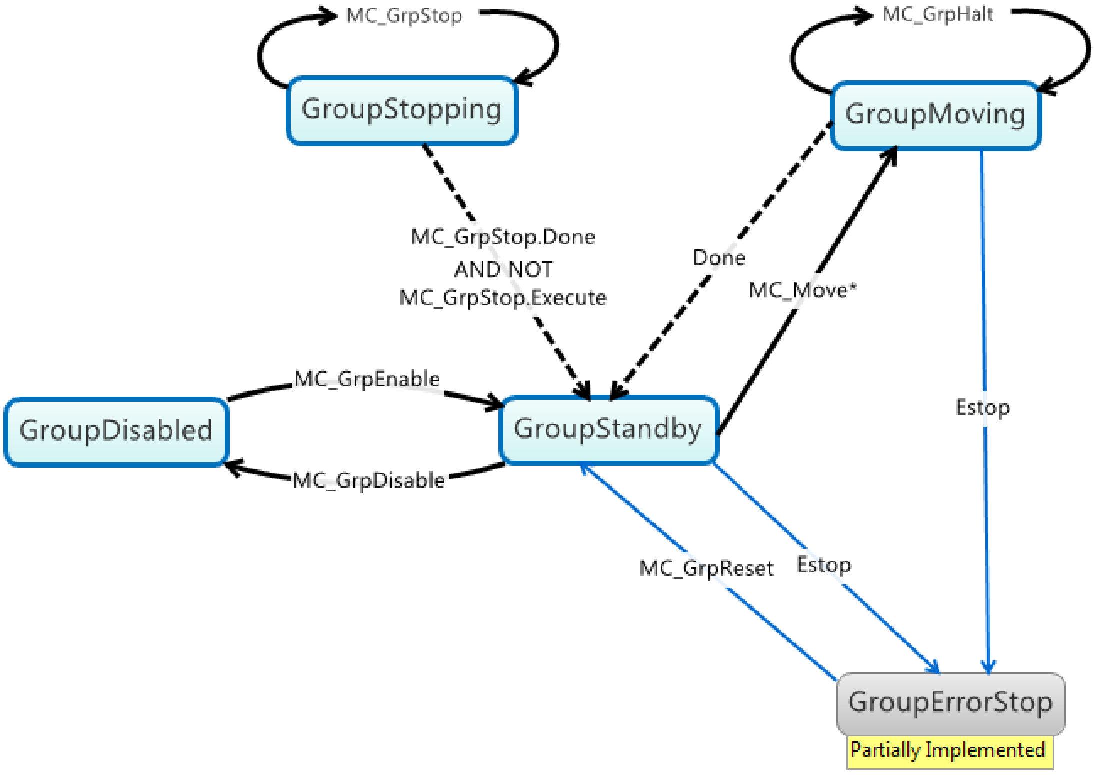
See also "Create a Linear or Circular Coordinated Motion Application" on page 462.

9.1.1.1 Coordinated Motion Terminology

Term	Definition
ACS	Axes Coordinate System. The system of coordinates related to the physical motors and the single movements caused by the single drives.
Blending	A way that consecutive function blocks cooperate in the transition from the first to the next.
Contour Curve	Inserted curve that modifies the original path. It is the resulting curve after blending
Coordinate system	The reference system in which a coordinate or path is described
Corner deviation	The shortest distance between the programmed corner point and the contour curve
Corner distance	Distance of the start point of the contour curve to the programmed target point.
Direction	The orientation components of a vector in space. (Note: this is different from the MC_Direction input as used in part 1).
Drive	A unit controlling a motor via the current and timing in its coils
Group-FB	The set of function blocks that can work on a group of axes
MCS	Machine Coordinate System. The system of coordinates that is related to the machine. A Cartesian coordinate system with the origin in a fixed position relative to the machine (the origin is defined during the machine setup). Sometimes called “World Coordinate System” or “Base Coordinate System”. (Note: with Cartesian build machines, MCS is a Cartesian Coordinate system and may be identical to ACS, or mapped via a trivial transformation). The coordinate system from the physical multiple axes ACS is linked to the MCS via a kinematic transformation (forward and backward conversion). The MCS represents an imaginable space with up to 6 dimensions.

Term	Definition
Motor	An actuator focused to a movement, converting electrical energy in a force or torque.
Orientation	The rotational components of a vector in space.
Path	Set of continuous positions and orientation information in multi-dimensional space Geometrical description of a space curve that the TCP of an axesgroup moves along.
PathData	Description of a path which can include additional information like velocity and acceleration.
PCS	Product Coordinate System or Program Coordinate System. The PCS is based on the MCS typically by shifting and maybe rotating the MCS. The Zero point of the PCS is related to the product and can be changed during runtime by the program. The real work piece can have a rotation or shift to the MCS coordinate system or even might be moving relative to the MCS coordinate system. By specifying a trajectory in PCS one is able to describe the trajectory independent from the machine situation. To map these two worlds (MCS to PCS and vice versa), a Cartesian or cylindrical transformation is normally done.
Position	Position means a point in space which is described by different coordinates. Depending on the used system and transformation it can consist of up to 6 dimensions (coordinates) meaning 3 Cartesian coordinates in space and 3 coordinates for the orientation. In ACS there can be even more than 6 coordinates. If the same position is described in different coordinate systems the values of the coordinates are different.
Scara	A special kinematic for robot or handling applications.
Speed	Speed is the absolute value of the velocity without direction.
Synchronization	Combines an axis or axes group (as slave) with an axis as master in order that the slave executes its path with synchronization to the progress of the master, meaning linked to a one dimension source for synchronization.
TCP	Tool Center point, the point in the machine that is commanded to move, typically the center or the head of the tool. It can be described in different coordinate systems.
Tracking	Is characterized by an axis group that follows with its movement the movement of another axis group.
Trajectory	Time dependent description of the path the TCP of an axes group moves along. Additionally to the geometrical description of the space curve, time dependent state variables like velocity, acceleration, jerk, forces etc. are specified.
Velocity	For a group of axes this means: • in ACS the velocities of the different axes • in MCS and PCS it provides the velocity of the TCP

9.1.1.2 Group State Diagrams



9.1.1.3 Coordinate Systems

There are three different coordinate system (CS) types:

- Machine (MCS)
- Axes (ACS)
- Product/Program (PCS)

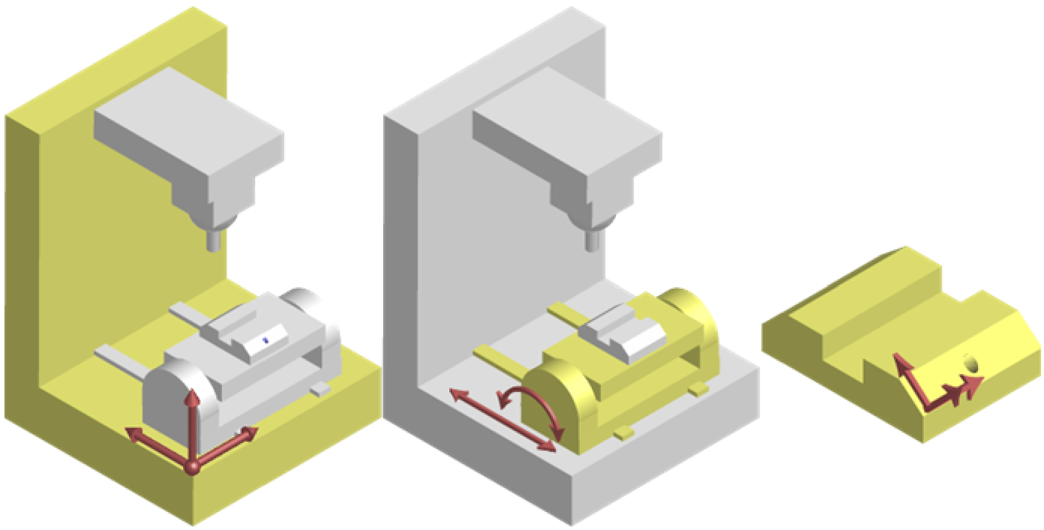
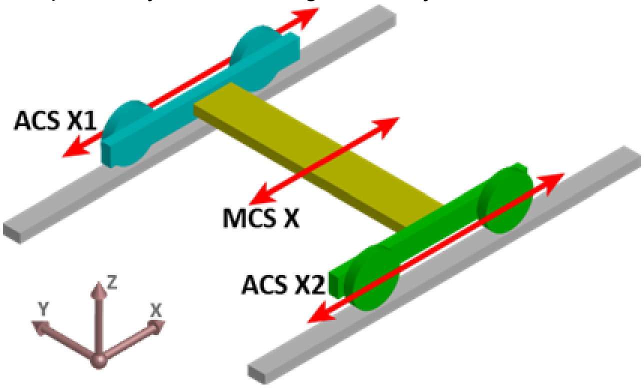


Figure 10-1: Examples of CS types on a machine and part.

Many coordinated moves may be done in a choice of coordinate systems. The differences between the types are offsets and possibly transformational algorithms to convert between the different systems, which ultimately control the actual axes on a piece of machinery.

For example, the X-axis of a Machine CS is meant to command a pair of Axes CS axes (X1 and X2) which together form a gantry. The relative movement of the MCS X axis would be added to both ACS axes. The two ACS axes can also be commanded independently for minor alignment adjustments.



TIP The Product Coordinate System is often rotated and/or offset from the Machine Coordinate System.

9.1.2 How-To: Coordinated Motion

This section discusses how to create a coordinated motion application, including adding coordinated motion to existing applications.

For more information on Coordinate Motion an the associated functions and function blocks see:

- "Coordinated Motion" (see page 459) in the Advanced Topics section
- Coordinated Motion Function Blocks

9.1.2.1 Create a Linear or Circular Coordinated Motion Application

A Coordinated Motion application can be created in one of two ways:

- Use a Coordinated Motion template to create a new application. Two Coordinated Motion templates are currently available.
  - The first template controls two PLCopen axes in coordinated motion.
  - The second template controls two PLCopen axes in coordinated motion plus a third independent Pipe Network axis.
- Modify an existing application to included coordinated motion functions. When modifying an existing application, axes need to be grouped to define the axes that will be active when performing coordinated motion on that group. More information about Axes Groups can be found in the section "What are Axes Groups?" (see page 465).

NOTE Coordinated motion can only be performed with PLCopen axes. Pipe Network axes do not support this feature, although Pipe Network axes can be moved independently from coordinated motion groups. Any synchronization between coordinated motion and Pipe Network axes must be performed by the PLC application.

Related axes are "grouped" in an axes group. Coordinated motion is then performed on an axes group. For more information see "What are Axes Groups?" (see page 465).

Typically, the following set of function blocks should be called before executing coordinated motion.

1. Call MLMotionInit (BasePeriod) to initialize the motion engine. Base period is specified in microseconds.

```
MLMotionInit(1000.0);    // 1000 µSec -> 1 mSec
```

2. Call MC_CreateAxesGrp (Enable, GroupName, UpdateRate, MaxNumberOfAxes, AxesGroupRef) to create a Coordinated Motion Axes Group

> **NOTE** MC_CreateAxesGrp needs to be called between MLMotionInit() and MLMotionStart().

```
Inst_MC_CreateAxesGrp(TRUE, 'GROUP1', 6, 2, Group1_
ref);
```

In the example above, the axes group name is 'GROUP1', the update rate is 1 mSec (specified by '6') and the maximum number of axes that can be added to the group is 2. The group reference variable 'Group1_ref' will be used in future coordinated motion function block calls to reference this newly created group.

3. Call MC_InitAxesGrp (Enable, AxesGroup, VelLimit, AccLimit, DecLimit, JerkLimit) to initialize the path limits for velocity, acceleration, deceleration, and jerk.

```
Inst_MC_InitAxesGrp(TRUE, Group1_ref, 100.0, 300.0,
300.0, 1000.0);
```

In the example above, the kinematic limits for axes group 'Group1_ref' will be set. The velocity limit will be set to 100.0 user units/second, acceleration and deceleration limits will be set to 300.0 user units/second^2 and jerk will be set to 1000.0 user units per second^3 (Jerk will be supported in a future release).

4. Call MC_CreateAxis (AxisName, BusInterface, BusAddress, AxisNumber, AxisType, UserUnits, FeedbackUnits, Rollover, UpdateRate) to create a Coordinated Motion Axis. This function needs to be called for each Coordinated Motion Axis wanted in the application.

> **NOTE** MC_CreateAxis needs to be called between MLMotionInit() and MLMotionStart().

```
Inst_MC_CreateAxis(TRUE, 'CoordAxis1', 'Ether-
CATDriver', 1001, CoordAxis1_AxisNum, 0, 360, 1048576,
0, 6);
Inst_MC_CreateAxis(TRUE, 'CoordAxis2', 'Ether-
CATDriver', 1002, CoordAxis2_AxisNum, 0, 360, 1048576,
0, 6);
```

In the example above:

- Two axes are created and are named 'CoordAxis1' and 'CoordAxis2'.
- The bus interface for both is 'EtherCATDriver'.
- The address of the drive on the bus is 1001 and 1002.
- The axis numbers are set with variables CoordAxis1_AxisNum and CoordAxis2_AxisNum which is set to an integer value between 1 and 256. Each axis number is unique.
- The axis type for both, '0', indicates a servo axis.
- The user units are 360, which is the 'user unit' portion of the 'user unit/feedback' ratio.

- The feedback units are 1048576, which is the 'feedback' portion of the 'user unit/feedback' ratio.
- The rollover position for both, '0' indicates no rollover.
- The update rate for both, '6', indicates a 1mSec update rate.

5. Call MLMotionStart () to start the Motion and the motion bus driver. This also initializes the EtherCAT network to operational mode.

```
MLMotionStart();
```

6. Call MC_AddAxisToGrp (Execute, AxesGroup, Axis, IdentInGroup) for each axis to be added to the group.

```
Inst_MC_AddAxisToGrp(TRUE, Group1_ref, CoordAxis1_ref,
0);
Inst_MC_AddAxisToGrp(TRUE, Group1_ref, CoordAxis2_ref,
1);
```

In the example above, we are adding two axes, CoordAxis1 and CoordAxis2, to the group referenced by 'Group1_ref'. The axes are stored in the IdentInGroup positions 0 and 1. Note that when the group was created, it was specified that no more than 2 axes will be part of this group. Therefore, valid IdentInGroup locations are 0 and 1.

7. Call MC_Power (Enable, Axis, EnablePositive, EnableNegative, BufferMode) for each Coordinated Motion Axis to enable the drive and close the servo loop.

```
Inst_MC_Power1(TRUE, CoordAxis1_ref, TRUE, TRUE, 0);
Inst_MC_Power2(TRUE, CoordAxis2_ref, TRUE, TRUE, 0);
```

In the example above, drives CoordAxis1_ref and CoordAxis2_ref will be enabled and the position loop will be closed. Note that parameters 'TRUE, TRUE, 0' are place holders for future use and are not currently used.

8. Call MC_GrpEnable (Execute, AxesGroup) to change the state of the Coordinated Motion Axis Group from GroupDisabled to GroupStandby and allow motion to be performed on the group.

```
Inst_MC_GrpEnable(TRUE, Group1_ref);
```

In the example above, 'Group1_ref' state will be changed from GroupDisabled to GroupStandby. The group must be in GroupStandby in order to perform motion.

9. For the examples that follow, we want to set the current location of the axes in the group to 0, 0. This can be done by calling MC_GrpSetPos (Execute, AxesGroup, Position[], Relative, CoordSystem, BufferMode)

```
PosAbs[1]:= 0;
PosAbs[2]:= 0;
Inst_MC_GrpSetPos(TRUE, Group1_ref, PosAbs, 0, MC_
COORDINATE_SYSTEM_ACS, 0);
```

In the example above, the axis positions of 'Group1_ref' will be set to 0, 0. 'PosAbs' specifies the position for each axis in the group. 'Relative' input, '0', uses 'PosAbs' to set the absolute position. The coordinate system is set to ACS . The buffer mode, '0', is a placeholder for future use and is not currently used.

> **NOTE** No motion will be performed when this function block is executed.

After the above function calls have been made, we can start coordinated motion moves.

"Performing a Linear Move" (see page 465)

"Performing a Circular Move" (see page 467)

What are Axes Groups?

Related axes are grouped in an AxesGroup to support interpolation. AxesGroups are accessed via the type AXES_GROUP_REF. The following image shows the relationships between the different CSs and groups.

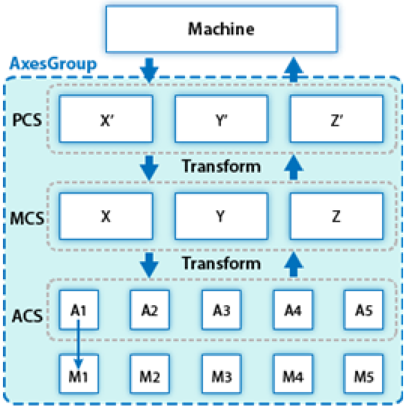


Figure 10-2: Overview of AxesGroup

The AxesGroup, shown in blue above, provides the interface to the user of the group of axes. To access the relevant coordinate system, the relevant function blocks have an input CoordSystem which supports the three levels ACS, MCS, and PCS.

Parameters in the AxesGroupRef can include remaining time and remaining distance before target position (or velocity or equal) is reached.

Performing a Linear Move

Linear moves can be programmed using absolute or relative positions using the following function blocks:

- MC_MoveLinAbs which commands interpolated linear movement on an axes group to the specified absolute positions.
- MC_MoveLinRel which commands interpolated linear movement on an axes group to the specified relative positions.

Prior to performing any coordinated moves, some setup is needed (see "Create a Linear or Circular Coordinated Motion Application" on page 462). Once these steps have been performed, a linear move can be performed.

In the following examples, two linear moves will be performed. The first move is an absolute linear move that goes from (0, 0) to (100, 200). The second move is a relative linear move that goes a distance of (-75, 50) from the end of the first move.The BufferMode input is set to 'Buffered', meaning this move will wait for the first move to complete before it begins executing.

- **To Perform an Absolute Linear Move**

  Call MC_MoveLinAbs (Execute, AxesGroup, PositionArray, Velocity, Acceleration, Deceleration, Jerk, CoordSystem, BufferMode, TransitionMode, TransitionParameter). PositionArray is an array of absolute end positions containing one position for each axis in the group. The inputs velocity, acceleration, deceleration, and jerk establish the maximum values for the move.

  In this example, PosArrayAbs[0] represent the x-axis and PosArrayAbs[1] represent the y-axis.

```
PosArrayAbs[0] := 100;
PosArrayAbs[1] := 200;
TransParam[0]  :=   0;
TransParam[1]  :=   0;

Inst_MC_MoveLinRel(TRUE, Group1_ref, PosArrayAbs,
MaxVel, MaxAcc, MaxDec, 0, MC_COORDINATE_SYSTEM_ACS, 1,
0, TransParam);
```

  In the example a linear move will be performed on axis group 'Group1_ref'.

  - PosArrayAbs contains the absolute end points of the axes in the group. The axis stored in position 0 (IdentInGroup) of the group will be moved to 100.0. The axis stored in postiion 1 of the group will be moved to 200.0.
  - The maximum velocity is specified by variable MaxVel and is specified in 'user units/sec'.
  - The maximum acceleration and deceleration are specified by variables MaxAcc and MaxDec and are specified in 'user units/sec2'.
  - The maximum jerk is currently not supported and can be set to a value of 0.
  - The coordinate system is ACS
  - The BufferMode is set to 1, indicating the move is buffered. For more information about buffer modes, see the "Buffer Modes" (see page 136) overview.
  - The TransitionMode is set to 0, indicating no transition mode will be used. For more information about transition modes, see the "Transition Between Moves" (see page 477) section.
  - The TransParam array is required and the contents can be set to 0 since the transition mode is not being used. There has to be one array entry for each axis in the group.

- **To Perform a Relative Linear Move**

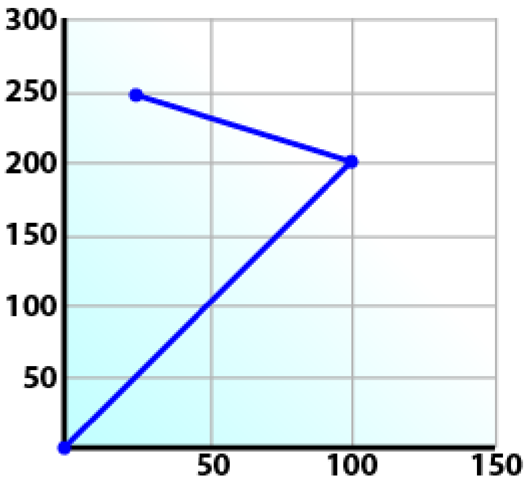
  Call MC_MoveLinRel (Execute, AxesGroup, Distance, Velocity, Acceleration, Deceleration, Jerk, CoordSystem, BufferMode, TransitionMode, TransitionParameter). The Distance input is an array of distances, one distance for each axis in the group. The inputs velocity, acceleration, deceleration, and jerk establish the maximum values for the move.

  In this example, DistArrayRel[0] represent the x-axis and DistArrayRel[1] represent the y-axis.

```
DistArrayRel[0] := -75.0;   // Start pt 100 - rel 75 ->
 25 absolute end pt
DistArrayRel[1] :=  50.0;   // Start pt 200 + rel 50 ->
250 absolute end pt
TransParam[0]   :=   0;
TransParam[1]   :=   0;

Inst_MC_MoveLinRel(TRUE, Group1_ref, DistArrayRel,
```

```
MaxVel, MaxAcc, MaxDec, 0, MC_COORDINATE_SYSTEM_ACS, 1,
0, TransParam);
```

In the example above, all the variables have the same meaning as the absolute linear example except DistArrayRel. DistArrayRel contains the relative distance to move for each axis in the group. The axis stored in position 0 (IdentInGroup) of the group will be moved a distance of -75.0. The axis stored in postiion 1 of the group will be moved a distance of 50.0.



Performing a Circular Move

Circular moves can be programmed using absolute or relative positions using the following function blocks:

- MC_MoveCircAbs which commands interpolated circular movement on an axes group to the specified absolute positions.
- MC_MoveCircRel which commands interpolated circular movement on an axes group to the specified relative positions.

Prior to performing any coordinated moves, some setup is needed (see "Create a Linear or Circular Coordinated Motion Application" on page 462). Once these steps have been performed, a circular move can be performed.

In the following examples, two circular moves will be performed. The first move is an absolute circular move that goes from (0, 0) to (90, 90). CircMode specifies that the aux point (0, 180) will be crossed during the paths start to end. The second move is a relative circular move whose end point is (90, 90) from the end of the first move. In this move, CircMode specifies that the aux point (0, 90) is the relative center of the circle. The BufferMode input is set to 'Buffered', meaning this move will wait for the first move to complete before it begins executing.

- **To perform an Absolute Circular Move:**

  Call MC_MoveCircAbs (Execute, AxesGroup, CircMode, AuxPoint[], EndPoint[], PathChoice, Velocity, Acceleration, Deceleration, Jerk, CoordSystem, BufferMode, TransitionMode, TransitionParameter).

  In this example, PosCircAuxAbs[0] and PosCircEndAbs[0] represent the x-axis. PosCircAuxAbs[1] and PosCircEndAbs[1] represent the y-axis.

```
PosCircAuxAbs[0] :=   0;   // A point on the circle
that is crossed on the
PosCircAuxAbs[1] := 180;   // path from start to end
```

```
point.
PosCircEndAbs[0] :=  90;   // Absolute end point.
PosCircEndAbs[1] :=  90;

Inst_MC_MoveCircAbs(TRUE, Group1_ref, MC_CIRC_MODE_
BORDER, PosCircAuxAbs, PosCircEndAbs, MC_CIRC_
PATHCHOICE_CLOCKWISE, MaxVel, MaxAcc, MaxDec, 0, MC_
COORDINATE_SYSTEM_ACS, MC_BUFFER_MODE_BUFFERED, MC_
TRANSITION_MODE_NONE, TransParam);
```

In the example a circular move will be performed on axis group 'Group1_ref'.

- CircMode is defined as MC_CIRC_MODE_BORDER. This mode indicates that the AuxPoint array input will indicate a point on the circle which is crossed on the path from the starting point to the end point. See "Circular Moves Diagrams" (see page 469) for more information on CircMode movement options.
- The AuxPoint array, 'PosCircAuxAbs', defines an absolute point on the circle which is crossed on the path from the starting point to the end point. The contents of this array are determined by the CircMode variable, MC_CIRC_MODE_BORDER.
- The EndPoint array, 'PosCircEndAbs', contains the absolute end point for each axis in the group. The absolute end point of the axis stored in position 0 (IdentInGroup) of the group will be 90.0. The absolute end point of the axis stored in position 1 of the group will be 90.0.
- PathChoice is only relevant when CircMode is set to MC_CIRC_MODE_CENTER. In this case, this parameter is not used.
- The maximum velocity is specified by variable MaxVel and is specified in 'user units/sec'.
- The maximum acceleration and deceleration are specified by variables MaxAcc and MaxDec and are specified in 'user units/sec^2'.
- The maximum jerk is currently not supported and can be set to a value of 0.
- The coordinate system is ACS
- The BufferMode is set to MC_BUFFER_MODE_BUFFERED, indicating the move is buffered. For more information about buffer modes, see the "Buffer Modes" (see page 136) overview.
- The TransitionMode is set to MC_TRANSITION_MODE_NONE, indicating no transition mode will be used. For more information about transition modes, see the "Transition Between Moves" (see page 477) section.
- The TransParam array is required. The TransParam array is a 2-element array containing the corner distance and velocity for the transition. Transitions are not used in this example and therefore the contents can be set to 0.

- **To perform a Relative Circular Move:**

Call MC_MoveCircRel (Execute, AxesGroup, CircMode, AuxPoint[], EndPoint[], PathChoice, Velocity, Acceleration, Deceleration, Jerk, CoordSystem, BufferMode, TransitionMode, TransitionParameter).

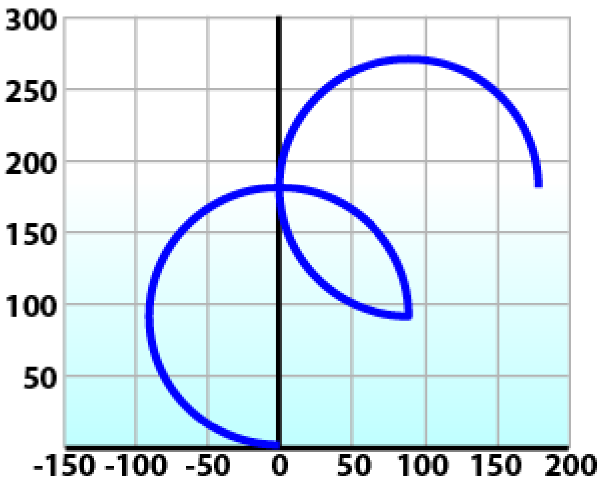
In this example, PosCircAuxRel[0] and PosCircEndRel[0] represent the x-axis. PosCircAuxRel[1] and PosCircEndRel[1] represent the y-axis.

```
PosCircAuxRel[0] :=  0;   // Relative center of the
circle.
PosCircAuxRel[1] := 90;
PosCircEndRel[0] := 90;   // Relative end point.
PosCircEndRel[1] := 90;   // Start pt 90,90 + rel 90,90
```

```
-> 180,180 absolute end pt

Inst_MC_MoveCircRel(TRUE, Group1_ref, MC_CIRC_MODE_
CENTER, PosCircAuxRel, PosCircEndRel, MC_CIRC_
PATHCHOICE_CLOCKWISE, MaxVel, MaxAcc, MaxDec, 0, MC_
COORDINATE_SYSTEM_ACS, MC_BUFFER_MODE_BUFFERED, MC_
TRANSITION_MODE_NONE, TransParam);
```

In the example all the variables have the same meaning as the circular absolute example except:

- CircMode is defined as MC_CIRC_MODE_CENTER. This mode indicates that the AuxPoint array input will indicate the center point of the circle. See "Circular Moves Diagrams" (see page 469) for more information on CircMode movement options.
- The AuxPoint array, 'PosCircAuxRel', defines the relative center point of the circle. The contents of this array are determined by the CircMode variable, MC_CIRC_MODE_CENTER.
- The EndPoint array, 'PosCircEndRel', contains the relative end point for each axis in the group. The relative end point of the axis stored in position 0 (IdentInGroup) of the group will be 90.0. The relative end point of the axis stored in postiion 1 of the group will be 90.0.
- PathChoice is relevant when CircMode is set to MC_CIRC_MODE_CENTER. In this case, PathChoice is MC_CIRC_PATHCHOICE_CLOCKWISE which specifies the direction of the path.

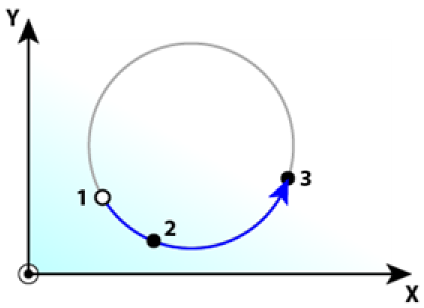


Circular Moves Diagrams

9.1.2.2 CircMode = BORDER

The user defines the end point and a border point (= input 'AuxPoint') on the sector of the circle which the machine will traverse. For Relative mode, both points are defined relative to the starting point.

Advantages	• The border point can usually be reached by the machine, i.e. it can be taught.
Disadvantages	• Restricted to angles < 360° in one single command.

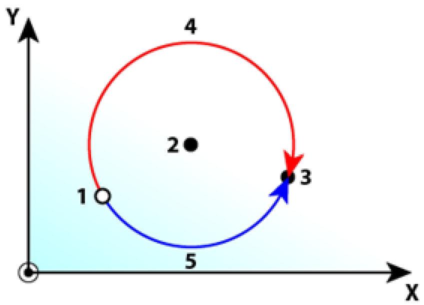


1. Starting point
2. Border point
3. End point

9.1.2.3 CircMode = CENTER

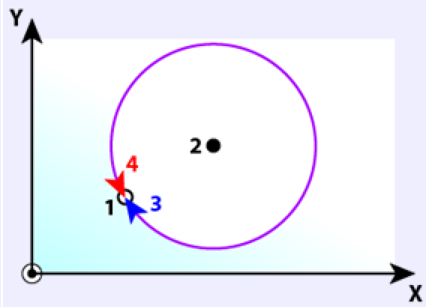
The user defines the end point and center point (= input 'AuxPoint') of the circle. The input 'PathChoice' defines clockwise or counter-clockwise motion. For Relative mode, both points are defined relative to the starting point.

Advantages	• Full 360° moves are possible.
Disadvantages	• Cannot perform zero-distance moves. • Over-determination of the circle equation.



1. Starting point
2. Center point
3. End point
4. Clockwise move
5. Counter-clockwise move

NOTE A 360° move will be performed if the end point is the same as the start point.



1. Starting & Ending point
2. Center point
3. Clockwise move
4. Counter-clockwise move

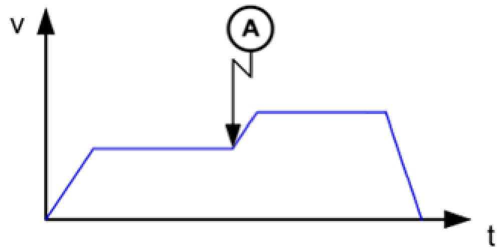
9.1.2.4 Blending Between Moves

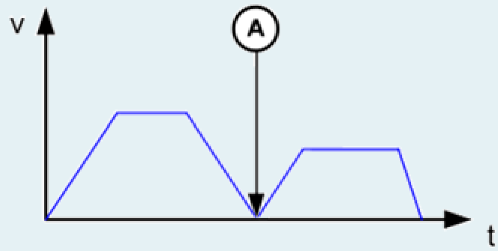
Some Coordinated Motion Function Blocks have a `BufferMode` input parameter. Possible buffer modes include:

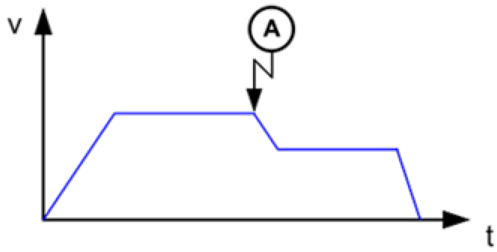
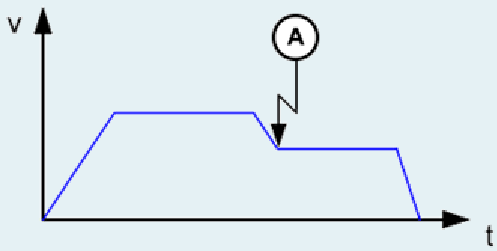
- MC_BUFFER_MODE_ABORTING = 0
- MC_BUFFER_MODE_BUFFERED = 1
- MC_BUFFER_MODE_BLENDING_PREVIOUS = 2
- MC_BUFFER_MODE_BLENDING_NEXT = 3
- MC_BUFFER_MODE_BLENDING_LOW = 4
- MC_BUFFER_MODE_BLENDING_HIGH = 5

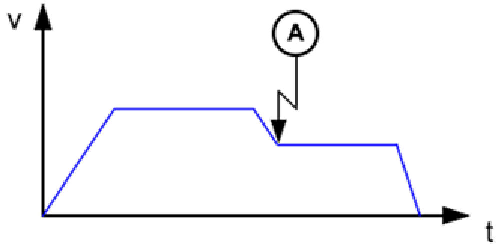
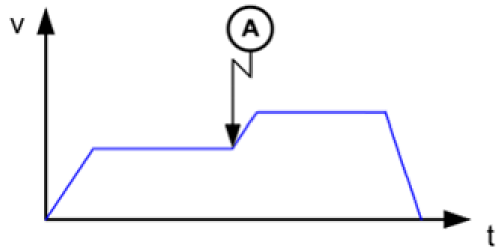
When the current and next motion function blocks are blended (2 through 5 above), the axes group will not stop between motions. The velocity will be blended according to the specified blending mode.

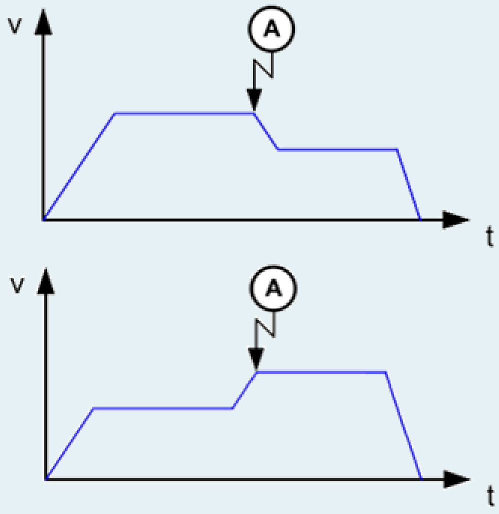
In addition, the Function Block `TransitionMode` parameter can be set to provide a smooth circular arc between moves. Transition modes are described in "Transition Between Moves" (see page 477).

Blend Type	Profile	Description
MC_BUFFER_MODE_ABORTING		The current move stops execution immediately. The next move becomes active and begins to accelerate or decelerate from the current velocity to its programmed velocity. In the image to the left the second function block starts execution at point "A".

Blend Type	Profile	Description
MC_BUFFER_MODE_BUFFERED		The current move completes as was originally specified, coming to a stop. When the current move reaches its endpoint, the next move becomes active and begins to accelerate to its programmed velocity. In the image to the left the second function block starts execution at point "A".

Blend Type	Profile	Description
MC_BUFFER_MODE_BLENDING_PREVIOUS		The current move stays at its programmed velocity to its endpoint. When the current move reaches its endpoint, the next move becomes active and begins to accelerate or decelerate to its programmed velocity. In the image to the left the second function block starts execution at point "A".
MC_BUFFER_MODE_BLENDING_NEXT		The current move will begin to accelerate or decelerate so that it reaches the next move's programmed velocity at the time the current move reaches its endpoint. In the image to the left the second function block starts execution at point "A".

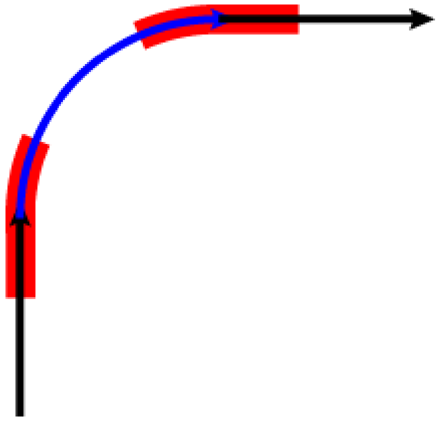
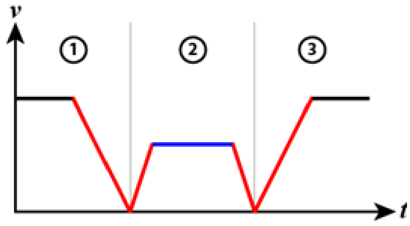
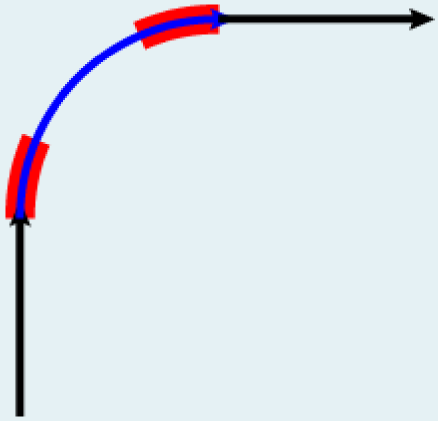
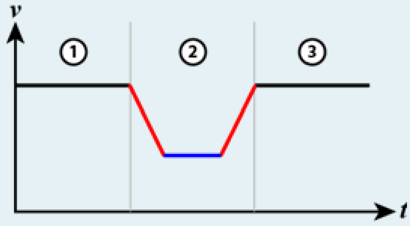
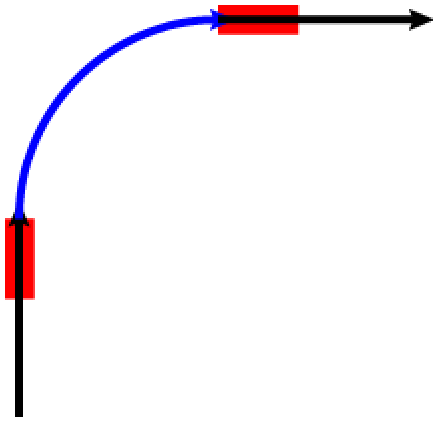
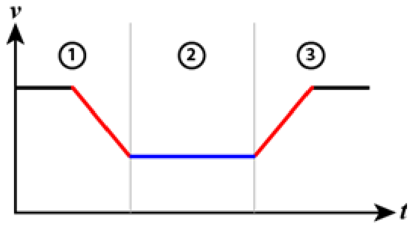
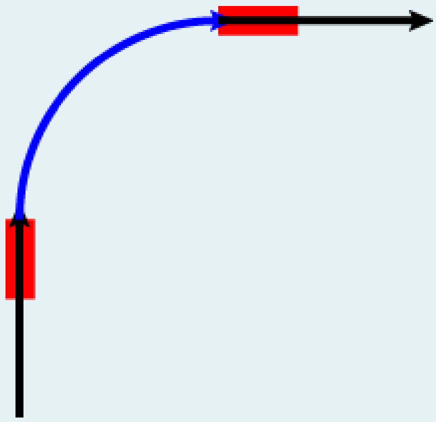
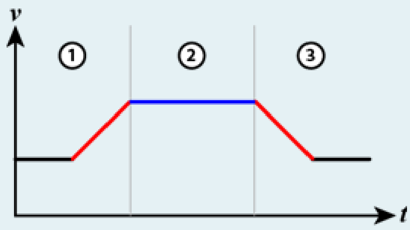
Blend Type	Profile	Description
MC_BUFFER_MODE_BLENDING_LOW	 	The velocity is blended to the lowest velocity of the two moves. • If the current move's velocity is lower than the next move's velocity, **Blending Previous** will be used. • If the next move's velocity is lower than the current move's velocity, **Blending Next** will be used. In the images to the left the second function block starts execution at point "A".

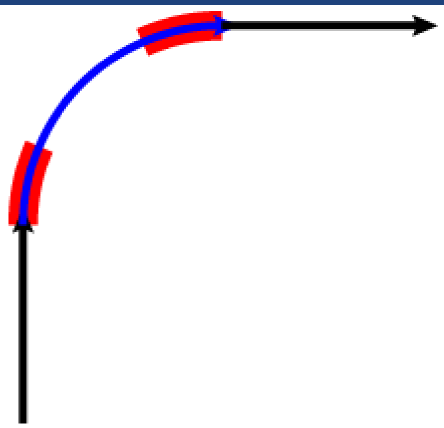
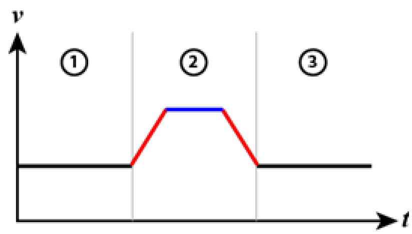
Blend Type	Profile	Description
MC_BUFFER_MODE_BLENDING_HIGH		The velocity is blended to the highest velocity of the two moves. • If the current move's velocity is higher than the next move's velocity, **Blending Previous** will be used. • If the next move's velocity is higher than the current move's velocity, **Blending Next** will be used. In the images to the left the second function block starts execution at point "A".

9.1.2.5 Blending with Transitions

When blending is specified when a transition is specified, then the blending mode is used to blend velocities of path segments when the path changes from the incoming segment to the transition segment and from the transition segment to the outgoing segment.

The most common blending mode choices for applications are listed below showing the velocity profile and where acceleration occurs on the path. In the examples, blue denotes the transition arc, red denotes where the acceleration occurs, 1 indicates the incoming move, 2 indicates the transition, and 3 indicates the outgoing move.

Example	Profile
MC_BUFFER_MODE_BLENDING_BUFFERED 	
MC_BUFFER_MODE_BLENDING_HIGH When the transition velocity is smaller than the incoming and outgoing move velocities. 	
MC_BUFFER_MODE_BLENDING_LOW When the transition velocity is smaller than the incoming and outgoing move velocities. 	
MC_BUFFER_MODE_BLENDING_HIGH When the transition velocity is larger than the incoming and outgoing move velocities. 	

Example	Profile
MC_BUFFER_MODE_BLENDING_LOW When the transition velocity is larger than the incoming and outgoing move velocities. 	

9.1.2.6 Transition Between Moves

A transition mode must be specified when a new move is appended to a move that is already in progress. Different transition parameters may be required, depending on the transition mode. This characterizes the contour of the transition segment.

The supported transition modes are:

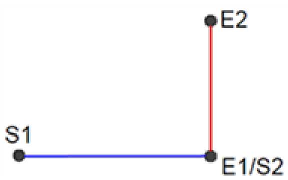
- "No Transition (“TMNone”)" (see page 477)
- "Corner Distance (“TMCornerDistance”)" (see page 477)

Transition Mode	Number of Transition Parameter Array Elements	Transition Parameter Index	Transition Parameter Name	Transition Parameter Description	Units
TMNone	0				
TMCornerDistance	2	0	Corner Distance	Distance to the corner of the deviation and the return point from the original contour	User units
		1	Velocity	The velocity value fo the transition segment	User units per second

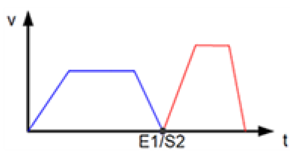
Table 10-1: Transition Mode Parameters

No Transition (“TMNone”)

"*Insert no transition contour segment.*"



Motion Path



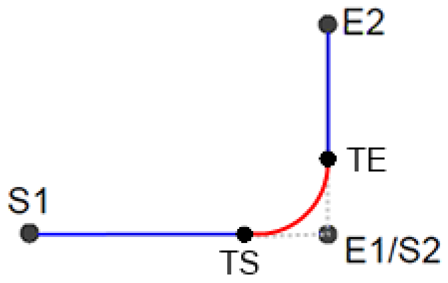
Motion Velocity Profile

The motion blocks are not modified and no transition curve is inserted using this mode. This is the only possible transition mode for the “Buffered” buffer mode.

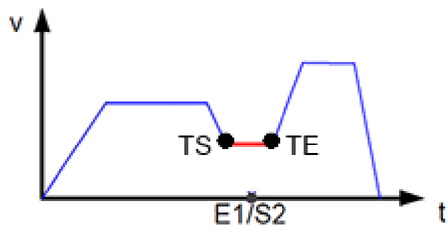
No transition parameters are used for this transition mode.

Corner Distance (“TMCornerDistance”)

"*Transition with given corner distance.*"



Motion Path



Motion Velocity Profile

The corner distance transition mode is specified using `TMCornerDistance` as the transition mode for a motion block.

TransitionParameter Index	Name	Description
0	Corner Distance	Distance to the corner of the deviation and the return point from the original contour.
1	Velocity	The velocity value of the transition segment.

Corner Distance transitions are handled differently, depending upon whether the connecting moves are lines or arcs, and all of the possible combinations, (line-line, arc-arc, line-arc, arc-line).

- Line-to-Line transitions will shorten the next move by the corner distance. See "Line to Line Transitions" (see page 478) for more information.
- Line-to-Arc and Arc-to-Line transitions shorten the linear move. See "Line-to-Arc and Arc-to-Line Transitions" (see page 480) for more information.
- Arc-to-Arc transitions will shorten the arc with the larger radius by the corner distance. See "Arc-to-Arc Transitions" (see page 481) for more information.

Related Functions

MC_MoveCircAbs (Function Block)

MC_MoveCircRel (Function Block)

MC_MoveLinAbs (Function Block)

MC_MoveLinRel (Function Block)

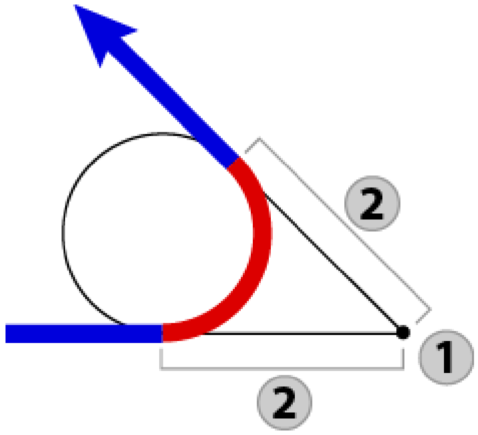
Line to Line Transitions

When both moves are linear the current and next moves are shortened using this transition mode. The amount is specified by the corner distance. A circular arc connects the two moves (except as noted below), allowing for a smooth transition (see "Figure 10-3: n-Degree Transition " on page 479 below). The circular arc that connects the two moves derives its velocity from the transition parameter and the acceleration and deceleration values are derived from the next move.

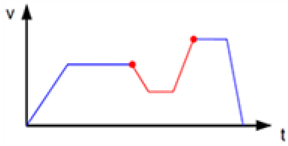
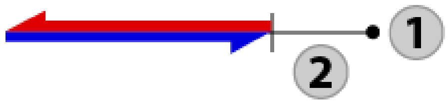
NOTE There are three special cases for line to line transitions:

- Zero-degree transitions:
  No transition move will occur when the current and next moves are in the same direction. The next move will act as if the TMNone transition mode was specified.
- 180-degree transitions:
  The current move will be shortened by the corner distance when the next move travels in the opposite direction.
  If the buffer mode is "Buffered" then the path velocity will go to zero. ***All other buffer modes may cause a large acceleration and jerk.***
- Zero-distance transitions:
  A zero distance transition move will be inserted, which will only affect

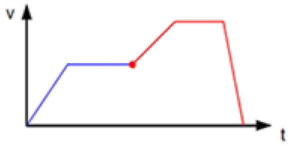
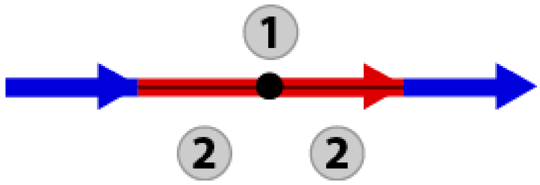
> **NOTE**
> blending.
> *If the buffer mode is "Buffered" then the path velocity will go to zero.* ***All other buffer modes may cause a large acceleration and jerk.***

<table>
<tr><th>Motion Path - Line to Line</th><th>Key</th><th>Motion Velocity Profile</th></tr>
<tr><td></td><td>Blue: Incoming, outgoing linear motion

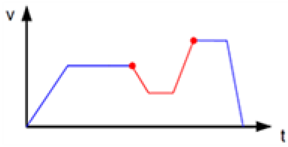
Red: Transition arc

1. Original end-point of 1st move
2. Corner distance</td><td></td></tr>
<tr><td colspan="3">Figure 10-3: n-Degree Transition</td></tr>
<tr><td></td><td>Blue: Incoming linear motion

Red: Outgoing linear motion

1. Original end-point of 1st move
2. Corner distance</td><td></td></tr>
<tr><td colspan="3">Figure 10-4: 180-Degree Transition: New move is in the opposite direction as old move.</td></tr>
<tr><td></td><td>Blue: Incoming, outgoing linear motion

Red: Transition line segment

1. Original end-point of 1st move
2. Corner distance</td><td></td></tr>
<tr><td colspan="3">Figure 10-5: 0-Degree Transition: New move continues in same direction as old move — continuous behavior</td></tr>
</table>

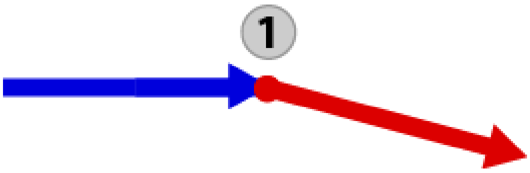
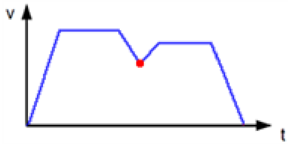
Motion Path - Line to Line	Key	Motion Velocity Profile
	Blue: Incoming, outgoing linear motion Red: Zero distance transition move 1. Original end-point of 1st move	

Figure 10-6: 0-Distance Transition: Special behavior for 0-distance transitions.

Line-to-Arc and Arc-to-Line Transitions

With this transition mode, the linear move is shortened when one move is linear and the other is circular. The amount is specified by the corner distance. A circular arc connects the two moves (except as noted below), allowing for a smooth transition (see "Figure 10-7: n-Degree Transition " on page 480 below). The circular arc that connects the two moves derives its velocity from the transition parameter and the acceleration and deceleration values are derived from the next move.

NOTE There are three special cases for Line to Arc Arc and Arc to Line transitions:

- Tangent transitions:
  The linear move will be shortened by the corner distance and a linear transition move will be inserted to cover the distance that was removed.
- Intersection transitions:
  This will only when the arc intersects the line corner distance away from the point where the line and the arc meet. The transition move will be zero distance.
  If the buffer mode is "Buffered" then the path velocity will go to zero. ***All other buffer modes may cause a large acceleration and jerk.***
- Zero-distance transitions:
  A zero distance transition move will be inserted, which will only affect blending.
  If the buffer mode is "Buffered" then the path velocity will go to zero. ***All other buffer modes may cause a large acceleration and jerk.***

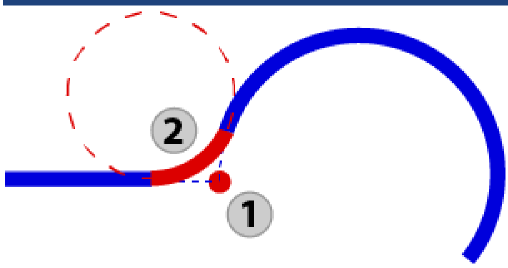
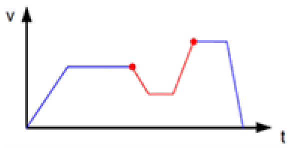
Motion Path - Line to Arc / Arc to Line	Key	Motion Velocity Profile
	Blue: Incoming, outgoing linear and circular motion Red: Transition arc 1. Original start/endpoint of linear move 2. Corner distance	

Figure 10-7: *n*-Degree Transition

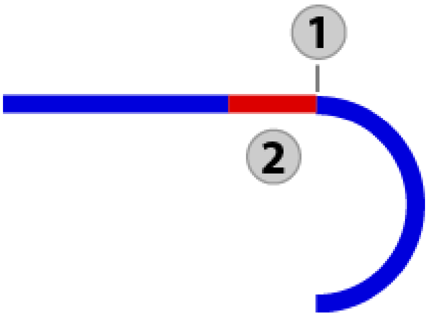
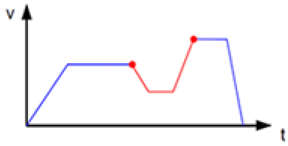
Motion Path - Line to Arc / Arc to Line	Key	Motion Velocity Profile
	Blue: Incoming, outgoing linear and circular motion Red: Transition line segment 1. Original start/endpoint of linear move 2. Corner distance	

Figure 10-8: Tangent Transition: Line is tangent to the arc

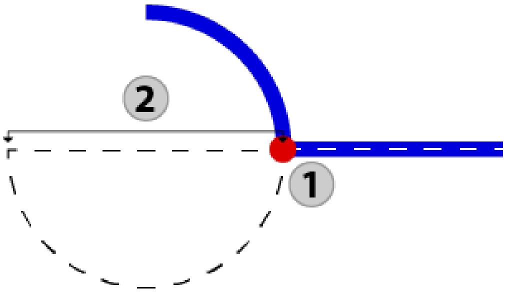
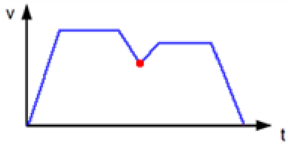
	Dashed: Original shape Blue: Incoming, outgoing linear and circular motion Red: Zero distance transition 1. Original endpoint of 1st move 2. Corner distance	

Figure 10-9: Intersection Transition: The line intersects the arc a "corner distance" away from the beginning of the new move.

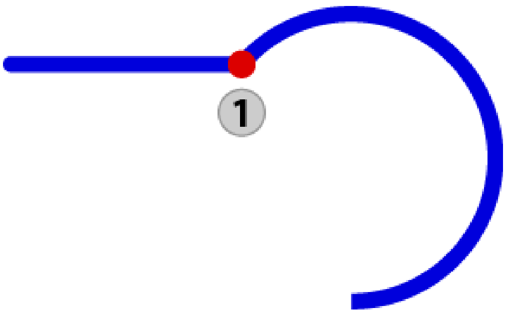
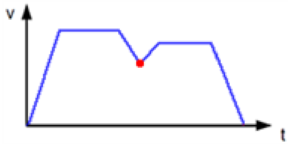
	Blue: Incoming, outgoing linear and circular motion Red: Zero distance transition 1. Original endpoint of 1st move	

Figure 10-10: 0-Distance Transition: Special behavior for 0-distance transitions.

Arc-to-Arc Transitions

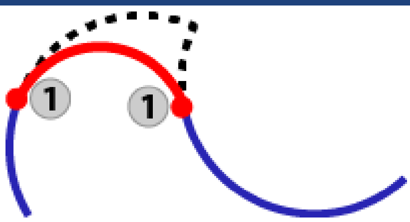
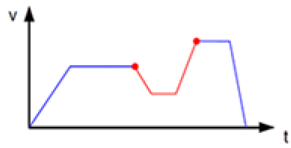
In arc-to-arc transition mode the transitions are handled as follows.

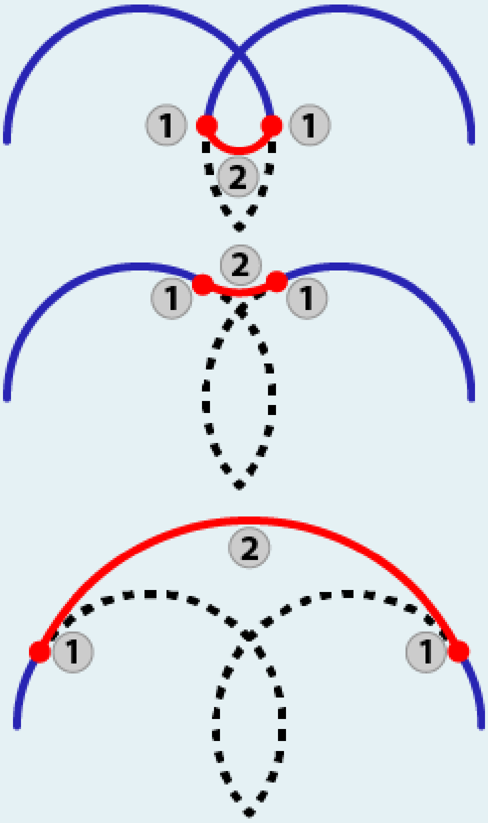
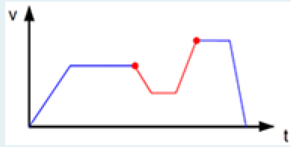
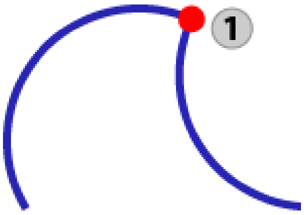
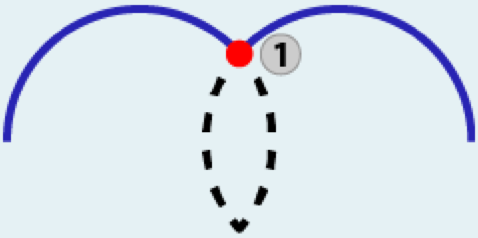
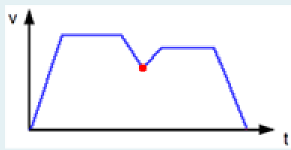
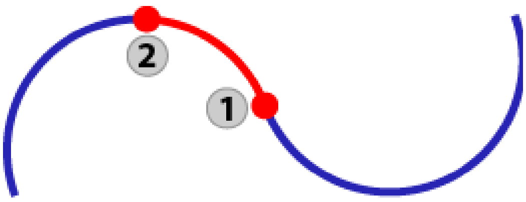
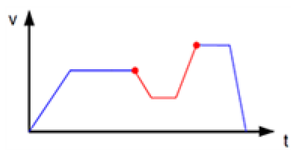
1. The arc with the larger radius is shortened by an arc length equal to the corner distance.
2. Using this point, a transition move is calculated such that it is tangent to both arcs.
3. The arc with the smaller radius is shortened to the point where it is tangent with the transition move.

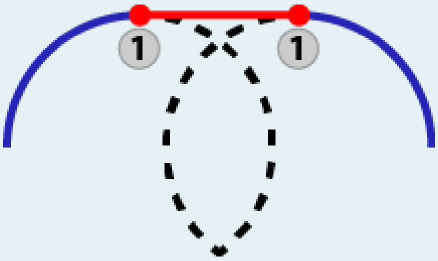
The transition move that connects the two original arcs derives its velocity from the velocity transition parameter and the acceleration and deceleration values are derived from the next move.

There are six special cases for Arc-to-Arc transitions.

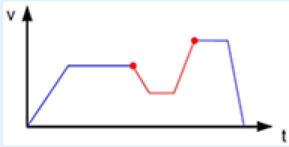
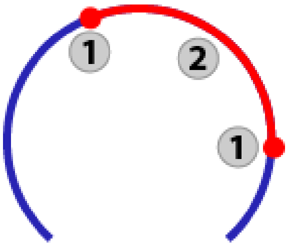
Transition Type	Description
Zero-distance Transitions	A zero-distance transition move is inserted when the corner distance is zero, which will only affect blending. The path velocity will go to zero if the buffer mode is "Buffered". All other buffer modes may cause a large acceleration and jerk.
Intersection Transitions	This will occur when the arcs intersect at two locations and the corner distance is equal to the arc length between the two intersections on the larger arc. The transition move will be zero distance. The path velocity will go to zero if the buffer mode is "Buffered". All other buffer modes may cause a large acceleration and jerk.
Tangent Transitions with Opposite Direction	A portion of the arc with the larger radius will be replaced by an arc whose length is equal to the corner distance. This will not affect the path, but will affect blending. If the arcs have the same radius, the incoming arc will be treated as having a larger radius.
Line Segment Transitions	For certain values of corner distance, the transition arc has an infinite radius and a line segment is used instead.
Same Circle, Same Direction Transitions	A transition arc with a length of twice the corner distance will be added if both arcs lie on the same circle and are in the same direction. This will not affect the path, but will affect blending.
Same Circle, Opposite Direction Transitions	The arcs will be shorted by an arc length equal to the corner distance and a zero distance transition will be inserted if both arcs lie on the same circle and are in the opposite direction. The path velocity will go to zero if the buffer mode is "Buffered". All other buffer modes may cause a large acceleration and jerk.

Case	Motion Path - Line-toArc / Arc-to-Line	Key	Motion Velocity Profile
***n*-degree, arcs in opposite direction**		Blue: Incoming and outgoing circular motion Black: "Trimmed" path Red: Transition arc 1: Start / endpoint of arc move	

Case	Motion Path - Line-toArc / Arc-to-Line	Key	Motion Velocity Profile
***n*-degree, arcs in same direction**		Blue: Incoming and outgoing circular motion Black: "Trimme d" path Red: Transition arc 1: Start / endpoint of arc move 2: Corner distance	
Zero-dis- tance		Blue: Incoming and outgoing circular motion 1: Original Start / Endpoint of arc move	user units/sec
Inter- section		Blue: Incoming and outgoing circular motion Black: "Trimme d" path 1: Intersection of the two arcs	
Tangent arcs, opposite direction		Blue: Incoming and outgoing circular motion Red: Transition arc 1: Original start / end point of moves 2: Start / Endpoint of arc move	

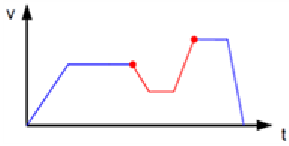
<table>
<tr><th>Case</th><th>Motion Path - Line-toArc / Arc-to-Line</th><th>Key</th><th>Motion Velocity Profile</th></tr>
<tr><td>Line segment</td><td></td><td>Blue: Incoming and outgoing circular motion

Red: Line Segment

1: Start / Endpoint of transition move</td><td></td></tr>
<tr><td>Same circle, same direction</td><td></td><td>Blue: Incoming and outgoing circular motion

Red: Transition arc

1: Start / Endpoint of arc move

2: Original start/end point</td><td></td></tr>
<tr><td>Same circle, opposite direction</td><td></td><td>Blue: Incoming and outgoing circular motion

1: Original Start / Endpoint of arc move

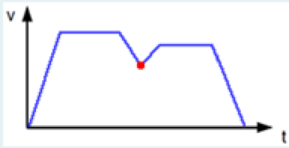
2: Start / Endpoint of arc move</td><td></td></tr>
</table>

Figure 10-11: Examples of Arc-to-Arc Transitions

9.1.2.7 What Does MC_GrpHalt Do?

MC_GrpHalt (Execute, AxesGroup, Deceleration, Jerk) performs a controlled motion stop of all axes in a coordinated motion group. See also "Differences between MC_GrpHalt and MC_GrpStop" on page 489.

MC_GrpHalt Application Example

The following example demonstrates a linear coordinated move with a starting point of (0,0) and an ending point of (200, 0). The first and second MC_GrpHalt commands are called while the linear coordinated move is still moving to the endpoint. As noted in the oscilloscope, the second call to MC_GrpHalt aborts the first MC_GrpHalt as seen by the new deceleration rate. The path velocity reaches zero (approximately (60, 0)) before the linear coordinated move reaches its end point value.

NOTE For demonstration purposes it is assumed that the axes and group have been properly setup and configured. Example steps to setup coordinated motion are listed in the section "Create a Linear or Circular Coordinated Motion Application" (see page 462).

```
1: // Perform Linear ABSOLUTE move start (0,0) end (200,0)
PosAbs[0]:= 0;
PosAbs[1]:= 200;
```

```
PosAbs[2]:= 0;

pathVelocity := Inst_MC_GrpReadCmdVel2.PathVelocity;

Inst_MC_MoveLinAbs(  TRUE, Group1_ref, PosAbs, Velocity, Acceleration,
Deceleration, Jerk,
     MC_COORDINATE_SYSTEM_ACS, MC_BUFFER_MODE_ABORTING, MC_TRANSITION_
MODE_NONE, TransParam);
Inst_TON3( true, t#600ms );    //Allow for the move to reach path velocity
before calling MC_GrpStop
Inst_TON3.Q  THEN
     MC_MoveCounter := MC_MoveCounter + 1;
     Inst_TON3(false, t#100ms);
END_IF;

2: //Perform a halt on the group
halt_deceleration := 50.0;

pathVelocity := Inst_MC_GrpReadCmdVel2.PathVelocity;

Inst_MC_GrpHalt(TRUE, Group1_ref, halt_deceleration, default_jerk);

Inst_TON3( true, t#200ms );   //Allow for first halt deceleration rate to
be captured on the scope
IF Inst_TON3.Q THEN
     MC_MoveCounter := MC_MoveCounter + 1;
     Inst_TON3(false, t#100ms);
END_IF;

3: // Perform a second halt increasing the deceleration value.
// The second call to MC_GrpHalt will abort the first MC_GrpHalt
halt_deceleration := 200.0;

pathVelocity := Inst_MC_GrpReadCmdVel2.PathVelocity;

Inst_MC_GrpHalt1(TRUE, Group1_ref, halt_deceleration, default_jerk);

Inst_TON3( true, t#200ms );
IF ((Inst_TON3.Q) and (Inst_MC_GrpHalt1.Done Or Inst_MC_GrpHalt1.Error))
THEN
     Inst_TON3(false, t#100ms);
     Inst_MC_MoveLinAbs(  FALSE, Group1_ref, PosAbs, default_velocity,
default_acceleration, default_deceleration, default_jerk,
          MC_COORDINATE_SYSTEM_ACS, MC_BUFFER_MODE_ABORTING, MC_
TRANSITION_MODE_NONE, TransParam);
     Inst_MC_GrpHalt1(FALSE, Group1_ref, default_deceleration, default_
jerk);
END_IF;
```

When MC_GrpHalt is called in the example above, the current move will be aborted and a controlled motion stop will be applied to axes group 'Group1_ref'. The deceleration value, 'halt _deceleration' is set to 50.0 user units/sec^2 on the first call and 200.0 user units/second^2 on the second call. Jerk is currently not supported.

When MC_GrpHalt is called, the deceleration value from the function block is applied to the path velocity until it reaches zero when the MC_GrpHalt command is issued during a coordinated motion move. The group state is "GroupMoving" while the coordinated move is decelerating. The group state goes to “GroupStandBy” once the path velocity reaches zero. Any coordinated moves in the buffer are flushed, and new coordinated moves can be queued up upon completion of the MC_GrpHalt command.

> **NOTE** This behavior is different than the MC_GrpStop command. For differences between MC_GrpStop and MC_GrpHalt, see "Differences between MC_GrpHalt and MC_GrpStop" on page 489.

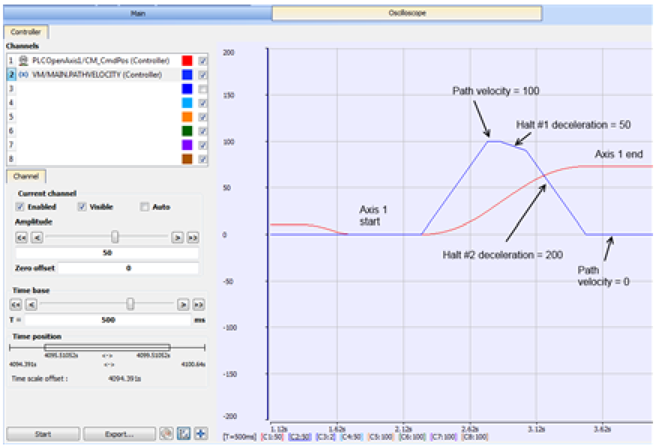


Figure 10-12: Oscilloscope Representation of linear coordinated move with a MC_GrpHalt command called twice

Exceptions:

- The deceleration rate from the MC_GrpHalt function block is only applied to the absolute and relative coordinated motion moves. Direct coordinated motion moves use the default deceleration value as defined by the AxisRef.
- A MC_GrpHalt cannot occur if a group is not enabled.
- A MC_GrpHalt command may be aborted by another MC_GrpHalt command, a MC_GrpStop command or a MC_Move* command.
- When there are two coordinated motion moves (*Active* and *Next*) in the queue, and the path velocity does not reach zero before the end of the *Active* move, the path velocity will continue to reach zero during the *Next* move.
- The deceleration rate will be increased if there is only one *Active* coordinated move and the path velocity cannot reach zero before the endpoint. This will prevent overshooting the endpoint.
- A MC_GrpHalt command does not prevent single axis motion from being performed.

9.1.2.8 What Does MC_GrpStop Do?

MC_GrpStop (Execute, AxesGroup, Deceleration, Jerk) performs a controlled motion stop of all axes in a coordinated motion group. See also "Differences between MC_GrpHalt and MC_GrpStop" on page 489.

MC_GrpStop Application Example

The following example demonstrates a linear coordinated move starting point of [0,0] and ending at point of [200, 0]. MC_GrpStop is called while the linear coordinated move is still moving to the endpoint. As noted in the oscilloscope, the path velocity reaches zero (approximately [100,0]) before the linear coordinated move reaches its end point value.

Linear Move Parameters	MC_GroupStop Parameters
Velocity = 100	stop_velocity = 75
Acceleration = 200	
Deceleration = 200	
Jerk = 0	

NOTE For demonstration purposes it is assumed that the axes and group have been properly setup and configured. Example steps to setup coordinated motion are listed in the section "Create a Linear or Circular Coordinated Motion Application" (see page 462).

```
Inst_MC_GrpReadCmdPos( TRUE, Group1_ref, MC_COORDINATE_SYSTEM_ACS, CmdPos-
itionArray );
Inst_MC_GrpReadCmdVel2( TRUE, Group1_ref, MC_COORDINATE_SYSTEM_ACS, Velo-
cityArray);
CASE MC_MoveCounter OF

0: // Enable the group
Inst_MC_GrpEnable(TRUE, Group1_ref);
Inst_TON3( true, t#1500ms );    // Allow for turning on the scope
IF ((Inst_TON3.Q) and (Inst_MC_GrpEnable.Done OR Inst_MC_GrpEn-
able.Error)) THEN

     Inst_TON3(false, t#1s);
     IF (Inst_MC_GrpEnable.Error) THEN
          PrintMessage( LEVEL_INFO (*DINT*),  'MC_GrpEnable failed -
ErrorID: ' + any_to_string(Inst_MC_GrpEnable.ErrorID));
     END_IF;

     Inst_MC_GrpEnable(FALSE, Group1_ref);
     MC_MoveCounter := MC_MoveCounter + 1;
END_IF;

1: // Perform Linear ABSOLUTE move start (0,0) end (200,0)
PosAbs[0]:= 0;
PosAbs[1]:= 200;
PosAbs[2]:= 0;

pathVelocity := Inst_MC_GrpReadCmdVel2.PathVelocity;

Inst_MC_MoveLinAbs(  TRUE, Group1_ref, PosAbs,Velocity, Acceleration,
Deceleration, Jerk,
     MC_COORDINATE_SYSTEM_ACS, MC_BUFFER_MODE_ABORTING, MC_TRANSITION_
MODE_NONE, TransParam);

Inst_TON3( true, t#600ms );     //Allow for the move to reach path velocity
before calling MC_GrpStop
IF Inst_TON3.Q  THEN
     MC_MoveCounter := MC_MoveCounter + 1;
     Inst_TON3(false, t#100ms);
END_IF;

2: //Perform a stop on the group
stop_deceleration := 75.0;

pathVelocity := Inst_MC_GrpReadCmdVel2.PathVelocity;

Inst_MC_GrpStop(TRUE, Group1_ref, stop_deceleration, default_jerk );

Inst_TON3( true, t#200ms );
IF ((Inst_TON3.Q) AND (Inst_MC_GrpStop.Done Or Inst_MC_GrpStop.Error))
THEN
     IF Inst_MC_GrpStop.Error THEN
          PrintMessage( LEVEL_INFO (*DINT*), 'Step '+any_to_string(MC_
```

```
MoveCounter)+',MC_GrpStop ERROR. ErrorID('+any_to_string(Inst_MC_
GrpStop.ErrorID)+'), Description:'+MC_ErrorDescription(any_to_int(Inst_
MC_GrpStop.ErrorID)));
     END_IF;

     Inst_TON3(false, t#100ms);

     Inst_MC_MoveLinAbs(  FALSE, Group1_ref, PosAbs, default_velocity,
default_acceleration, default_deceleration, default_jerk,
     MC_COORDINATE_SYSTEM_ACS, MC_BUFFER_MODE_ABORTING, MC_TRANSITION_
MODE_NONE, TransParam);

     Inst_MC_GrpStop(FALSE, Group1_ref, stop_deceleration, default_jerk
);
END_IF;
```

When MC_GrpStop is called in the example above, the current move will be aborted and a controlled motion stop will be applied to axes group 'Group1_ref'. The deceleration value, 'stop_deceleration', is set to 75.0 user units/sec^2 and is applied to the path velocity until it reaches zero. Jerk is currently not supported.

The group state is “GroupStopping” when the MC_GrpStop function block becomes active. While the axes group is in the GroupStopping state, no other function blocks can perform any motion on the same axes group. Once the path velocity reaches zero the **Done** output is TRUE. The **Execute** input must be set to FALSE before the group state can go to “GroupStandBy”. Any coordinated moves in the buffer are flushed, and new coordinated moves can be queued up upon completion of the Stop command

NOTE This behavior is different than the MC_GrpHalt command. For differences between MC_GrpHalt and MC_GrpStop, see "Differences between MC_GrpHalt and MC_GrpStop" on page 489.

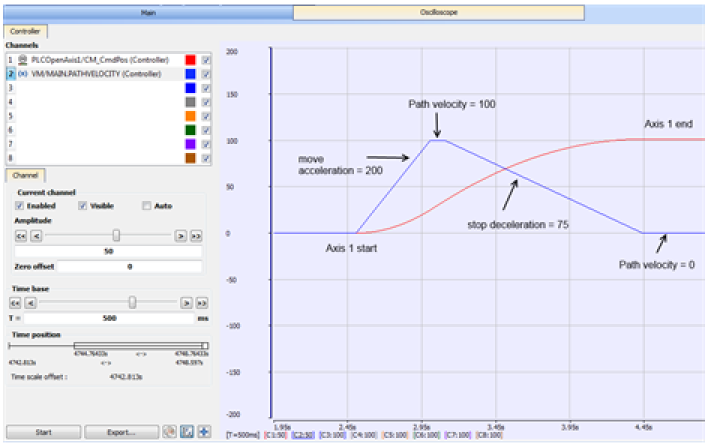


Figure 10-13: Oscilloscope Representation of linear coordinated move with a MC_GrpStop

Exceptions:

- The deceleration rate from the MC_GrpStop function block is only applied to the absolute and relative moves. Direct moves use the default deceleration value as defined by the AxisRef.
- MC_GrpStop cannot occur if a group is not enabled.
- A MC_GrpStop command *cannot* be aborted by any other commands (while MC_GrpHalt can be).

- When there are two coordinated motion moves (*Active* and *Next*) in the queue, and the path velocity does not reach zero before the end of the *Active* move, the path velocity will continue to reach zero during the *Next* move.
- The deceleration rate will be increased if there is only one *Active* coordinated move and the path velocity cannot reach zero before the endpoint. This will prevent overshooting the endpoint.
- A MC_GrpStop command does not prevent single axis motion from being performed.

9.1.2.9 Differences between MC_GrpHalt and MC_GrpStop

While MC_GrpHalt and MC_GrpStop both perform a controlled motion stop of all axes in an axes group, some differences exist between the operation of the function blocks.

- MC_GrpStop can not be aborted by any other command. MC_GrpHalt can be aborted by another MC_GrpHalt command, a MC_GrpStop command or a MC_Move command such as MC_MoveLinAbs, MC_MoveCircRel, etc.
- While MC_GrpStop is active, the group state (see "Group State Diagrams" (see page 461)) is 'GroupStopping'. While MC_GrpHalt is active, the group state is 'GroupMoving'.
- When MC_GrpHalt is complete, the group state goes to 'GroupStandBy'. When MC_GrpStop is complete, the DONE output will be true. The EXECUTE input must be set to false before the group state can go to 'GroupStandBy'.

9.1.2.10 Handling Axis Errors

Coordinated Motion Error handling is configurable on a per axis group basis. When a PLCopen axis error occurs the "Default Behavior" (see page 489) is for all axes in the group stop. This means motion interpolation stops, active and next queues are cleared, and the group state "GroupErrorStop" is enabled. Additionally, the position loop on the drives in the group are opened and the drives are disabled.

The "Optional Behavior" (see page 490) when a PLCopen axis error occurs is for only the PLCopen axis that caused the error to have its position loop opened and that drive will be disabled. The application is then expected to control the motion for the remaining axes in the group. The optional feature may be used when stopping all axes is worse than just having one axis stop.

To configure the optional behavior, use the function block MC_GrpWriteBoolPar, setting the parameter 'IGNORE_AXIS_ESTOP' (1000) to TRUE. To check if the optional behavior has been set, call the function block MC_GrpReadBoolPar, reading the state of parameter 'IGNORE_AXIS_ESTOP' (1000). If the result is true, the optional behavior is enabled.

The types of PLCopen Axis Errors that affect coordinated motion include:

- Drive errors
- Drive communication errors
- User causes E-stop using MC_EStop on an axis in a group

Default Behavior

In the default motion error handling case, a user can detect errors using any of the following group function blocks:

- MC_GrpReadStatus — Output GroupErrorStop is true if a group error has occurred.
- MC_GrpReadError — Error output will be true and Error ID output will be set to 12 if the error is due to E-stop.
- The originating Coordinated Motion command (MC_MoveLinAbs, MC_MoveCircAbs, etc.) will return Error ID 12

To determine which PLCopen axis has generated the error, call MC_ReadAxisErr for each axis in a group. The output ErrorID will be set to 12. In addition, MC_ReadStatus for each axis in the group can be called where the output ErrorStop will be true if in a E-stop condition

Once an error has been detected, the error must first be resolved. MC_GrpReset can be called to reset all PLCopen axis errors. This also resets the group status from GroupErrorStop to GroupStandby. The Done output of MC_GrpReset will be TRUE if all axis errors have been reset. This function block may take up to 3 seconds to reset some error conditions.

Optional Behavior

TIP This is configurable on a per group basis.

In the optional motion error handling case, the group status GroupErrorStop will not be set when an axis is in error. **The application is responsible for monitoring and handling error conditions**. The remaining axes in the group will continue moving.

Errors can be deteced using the following group function block:

- MC_GrpReadBoolPar — Read the result of parameter ID 'AXIS_ESTOP_ACTIVE' (1001). If the result is TRUE, an axis error exists. Note that the group function blocks used for the default case of detecting errors (MC_GrpReadStatus and MC_GrpReadError) will not return an error.

To determine which PLCopen axis has generated the error, call MC_ReadAxisErr for each axis in a group. The output ErrorID will be set to 12. In addition, MC_ReadStatus for each axis in the group can be called where the output ErrorStop will be TRUE if in a E-stop condition.

Once an error has been detected, the error must first be resolved. MC_ResetError can then be called on each axis in error. This function only sends a request to the drive to clear any error. The error will not yet be reset when this function returns. MC_ReadStatus will still have to be called to verify that the drive error has been resolved.

Recovery of the System State After an Axis Error

Recovery from axis errors is more complex with the addition of Coordinated Axes Groups (PLCopen Part 4) to KAS Runtime, This is due to additions in the operation of PLCopen Motion and Coordinated motion. This includes:

- Addition of the Coordinated motion Boolean variable that allows the Runtime to IGNORE_AXIS_ESTOP(ID number 1000).
- Addition of the Coordinated motion Boolean variable that allows the Application to read the state of the Runtime Axis errors: AXIS_ESTOP_ACTIVE (ID number 1001).
- Addition of MC_GrpReadBoolPar
- Addition of MC_GrpWriteBoolPar

When the application needs to take control of the default behavior where all axes in a group stop when any of them detect an Estop condition, and the Estop condition does not in itself stop all axes (Loss of EtherCAT communication), the application can set the Group Boolean parameter IGNORE_AXIS_ESTOP (ID := 1000) using MC_GrpWriteBoolPar(). When set to true, the runtime will try to keep axes in the same group as a faulting axis still able to be commanded. The intent is to not give up control of the commanded motion, but allow the application to substitute error handling motion that allows a group of axes to stop in a controlled manner. For example: command the remaining axes to a Home position, or a relative move away from the position of the faulted axis.

Once the IGNORE_AXIS ESTOP parameter is set TRUE, the Group of axes will not enter the ERRORSTOP state when an axis encounters an ESTOP condition but remain in STANDBY or MOVING state. Instead, the AXIS_ESTOP_ACTIVE (ID :=

1001) parameter can be monitored to trigger a response that will abort the current operation and take control of the axes that can still be controlled. To determine which axis has faulted, the MC_ReadStatus() FB can be used to monitor each axis and select the appropriate single axis or Coordinated motion commands to execute to take control of the motion and recover from the error.

When the error recovery is completed and the faulting axis restored to normal operation, in order to restart Coordinated Motion, an MC_GrpSetPos() command must be issued to tell the Coordinated Motion Engine where the axes in the faulted Axes Group are then at, following restoration of the single axis fault(s). This command must be issued in order to reset the faulted axis.

Alternatively, if the MC_GrpWriteBoolPar() function is issued to set the IGNORE_ AXIS_ESTOP parameter to FALSE, the Group will enter the ERRORSTOP state if an Axis error has not been reset, and the default behavior and default usage of MC_ GrpResetError() function can be used to clear faults instead of the MC_GrpSetPos() function.

9.2 Motion Techniques

This chapter explains advanced concepts and procedures related to **motion techniques** that are possible with the KAS IDE.

9.2.1 PLC Online Change

IMPORTANT You have to save and compile your project before doing an online change.

This section provides a detailed description of the PLC Online Change functionality. See "Using PLC Online Change" (see page 497) for an overview of using this functionality.

9.2.1.1 What is Online Change

Online Change enables you to update your PLC application on the fly, while it is running on the controller. You do not need to stop the controller, download the new code and start again. You only need to modify, recompile and download the new code as shown in the figure below; and then ask the controller to switch the execution to the new application.

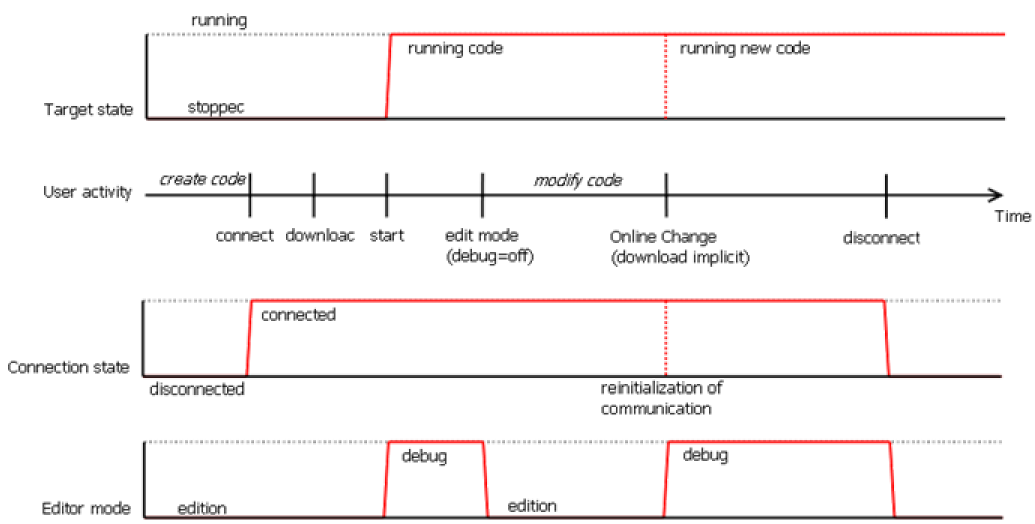


Figure 10-14: Online Change - Process Diagram

TIP This capability applies only to PLC code. This is not supported in the

TIP PipeNetwork editor, the PLCOpen axis, or any other part of the system.

NOTE Depending on the P-code size, the time to perform the Online Change operation can take more than one cycle. In that case, you can miss one PLC cycle before the changeover becomes effective.
This duration is also displayed in the Log window with an **INFO** level message as follows: **Online Change done in X µs**. For more details, click here The INFO measurement corresponds to the duration for the code hotswap. The download and loading of new code in memory is not taken into account in this measurement because they occur when the previous code is still running.

This feature is used in the following situations:

- **Development phase**: you can modify the application and apply these modifications incrementally without stopping the controller
- **Update in production**: you can update the running motion application (for instance with a bug fix release) without stopping the whole production chain

When Online Change is enabled, you can perform the following kinds of changes on the fly:

- Rename a program
- Change the code of a program
- Change the condition of an SFC transition or the actions of an SFC step
- Create, rename or delete global and local variables
- Create, rename or delete global and local function block instances
- Rename "Retain Variables" (see page 79)

The following **are not allowed**:

- Create or delete a program
- Change SFC charts: you cannot add or remove steps in the First Level of an SFC chart (but you can modify existing steps)
- Change the local parameters and variables of a UDFB
- Change the type or dimension (or string length) of a variable or function block instance
- Add or remove variables in a Structure
- Create a new Structure or a new UDFB
- Change the set of Input/Output or any modification that leads to an update in the EtherCAT Motion Bus configuration
- Create or delete Retain variables (their position in the runtime cannot be re-allocated)
- Being part of the motion engine, Pipe Network as well as Cam profile modifications are not taken into account
- Pulse (P or N) contacts and coils (edge detection)

  NOTE Using Pulse contacts in FFLD does not give any error, but the behavior of the contact during the switch is not always safe (for more details, as well as workaround, see page 496).

- The WAIT and WAIT_TIME instructions must not be used

  NOTE **Important!** The Online Change and Revert functions will fail while executing a WAIT.

- Loops in FBD with no declared variable linked. In this case, you need to explicitly insert a variable in the loop.

IMPORTANT When Online Change is active and custom libraries are being used, some

IMPORTANT errors can occur during the compilation. This happens if you open your project on another PC, or under a different user account in Windows. To fix this limitation:

1. Deactivate the Online Change
2. Save and then reopen the project
3. Turn the Online Change back on if desired

NOTE Your new application can contain more variables than the previous one. A memory with sufficient pre-allocated space is defined for the eventual new variables. If you exceed this limit, a warning message is displayed.

For limitation about breakpoint with Online Change, see page 329.

About the states and transitions

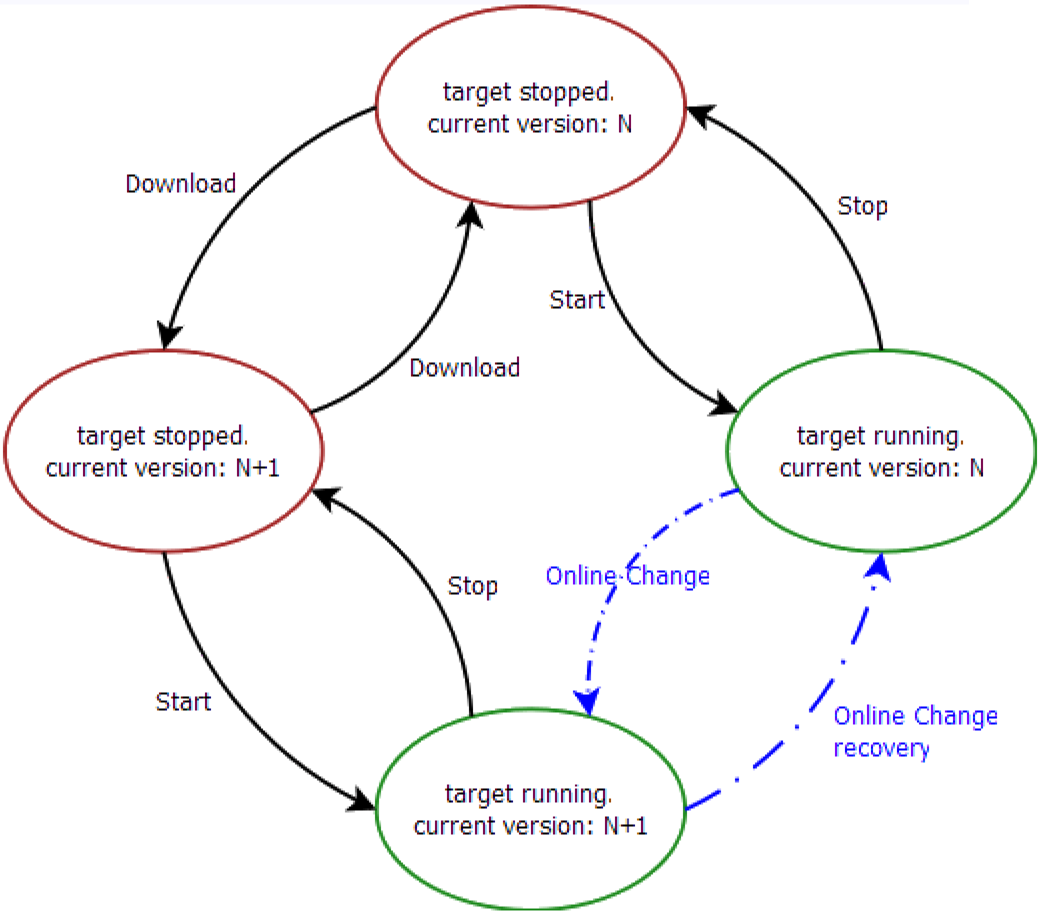


Figure 10-15: Online Change - States and Transitions

9.2.1.2 How to Activate Online Change

To allow Online Change, you need to open the PLC options and set the relevant parameters.

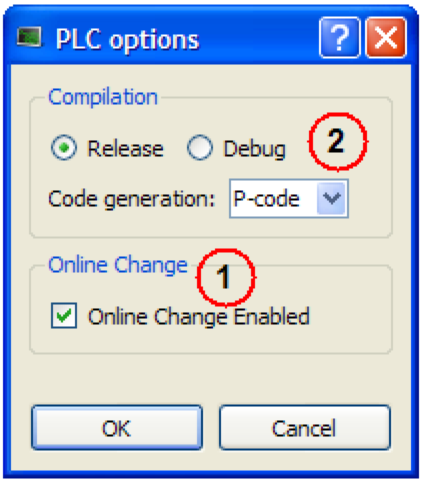


Figure 10-16: PLC Options - Online Change Enable

Set the parameters

This box allows you to enable or disable the **Online Change** feature (see call out 1).

IMPORTANT If you deactivate the Online Change, the next PLC application generated is no more compatible for an online change, even if you re-activated the online change before the compilation.

As a result, you can only apply an Online Change to a running application under the two following conditions:

- The Online Change was already activated
- You have never deactivated the Online Change between the compilation of the running application and the compilation of the new application

Note: Check the Controller Log window for any errors that occur.

You also need to ensure that you have selected P-code 2 as Online Change is not possible with native code (machine code). Note that when native code is selected, then Online Change is always deactivated.

Then you can compile your application, which now allows future changes on the fly.

Switch to Edit mode

When you start the application, the Debug mode is automatically activated: you can see the values changing in the editors and the Dictionary , showing what is happening on the controller. In this mode the editor is read-only, so you are not able to modify the code.

To edit your code, go out of the Debug mode and enter the Edit mode by clicking the button in the main toolbar.

Perform the Online Change

When your new code has compiled correctly, you can perform the Online Change. To do so, click the  button. When you click this button, the KAS IDE opens a window showing the execution of current actions (download, activation of new code).

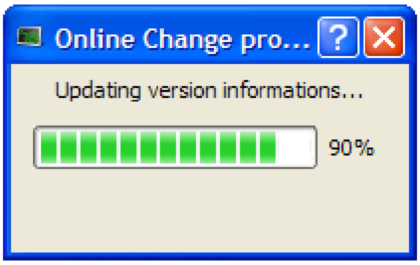


Figure 10-17: Online Change - Updating Controller Version

Once the Online Change is applied, the result is displayed in the window and you can click OK to acknowledge the operation and do a Warm start.

Dictionary behavior

When the Online Change is enabled, the dictionary shows:

- new variables in blue
- deleted variables in red

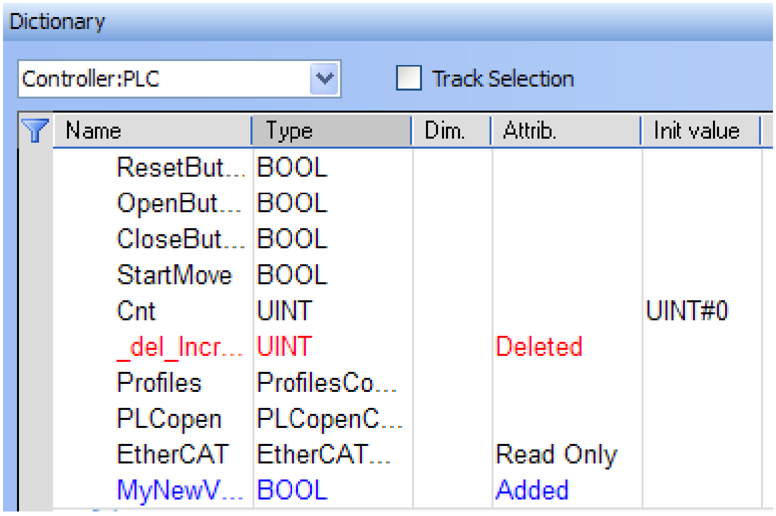


Figure 10-18: Online Change - Dictionary

TIP The deleted variables can be for new variables.

9.2.1.3 What is the Revert button

The **Revert** button is for security purposes. It allows you, after an Online Change, to revert your change quickly and go back to the previous application. That means switching the execution of the controller to the P-code that was running before the last Online Change (note that the source code in the KAS IDE is not replaced). The WAIT and WAIT_TIME instructions can not be used with Revert.

After the Revert, the KAS IDE automatically goes back to Edit mode.

NOTE You can go back to the previous version only when the Online Change feature is activated and while the controller is not stopped.

IMPORTANT After a revert operation, the Online Change feature is **deactivated**.

The Revert button is active when you are connected and the controller is running.

Revert is not possible:

- if you did not perform an Online Change
- if the controller has been restarted since the previous Online Change
- after another Revert
- during a WAIT

9.2.1.4 Difference between Local and Controller versions

When you restore a project with the Revert feature after an Online Change, KAS provides a tool to show the differences between two versions of the project. This tool can help you in checking all modifications before the next Online Change. It is also a useful tool when you want to compare your code with the last version after a Revert.

For more details, refer to "Compare PLC Programs" (see page 336).

9.2.1.5 Pulse Limitations with Online Change

At the first cycle, the pulse evaluation is ignored, and the memory is updated. This memory enables the pulse evaluation from the second cycle.

When we apply the Online Change between t0 ant t1, the cases where this method is not correct are the two following:

- When we want to detect a **falling** edge:

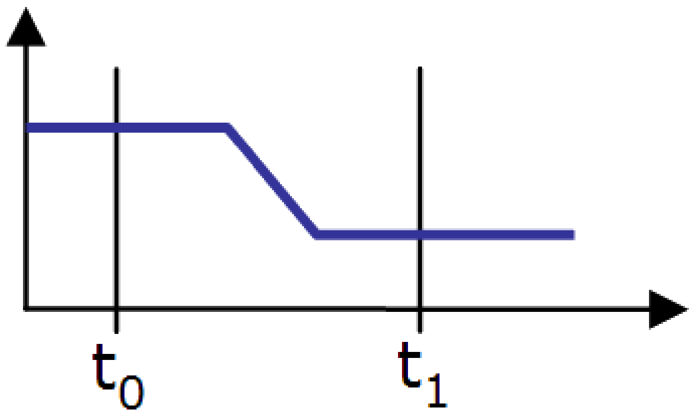


Figure 10-19: Pulse Limitations with Falling Edge

- When we want to detect a **rising** edge:

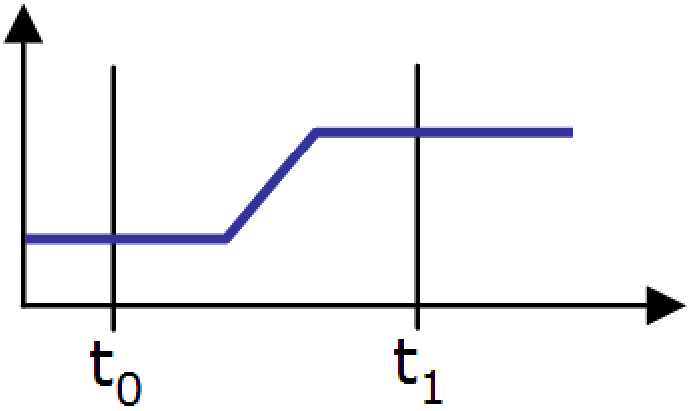


Figure 10-20: Pulse Limitations with Rising Edge

TIP	If you want to avoid this limitation, you must use declared instances of R_TRIG and F_TRIG function blocks.

NOTE	This limitation is temporary and is going to be fixed in a future release.

9.2.2 Using PLC Online Change

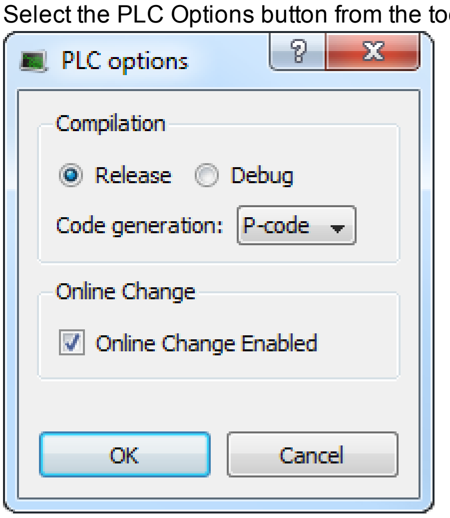
This section provides an overview of how to use Online Change. See "PLC Online Change" (see page 491) for descriptions of the functionality.

9.2.2.1 Set up an application

1. Create a new PLC application.
2. Connect to a controller and scan for EtherCAT devices.
3. Add logic and function blocks to the application.
4. Compile the project , connect  and download  the application to the device.

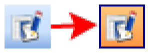
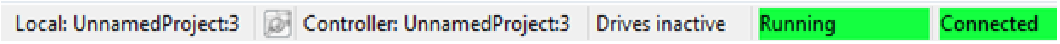
9.2.2.2 Enable Online Change Mode

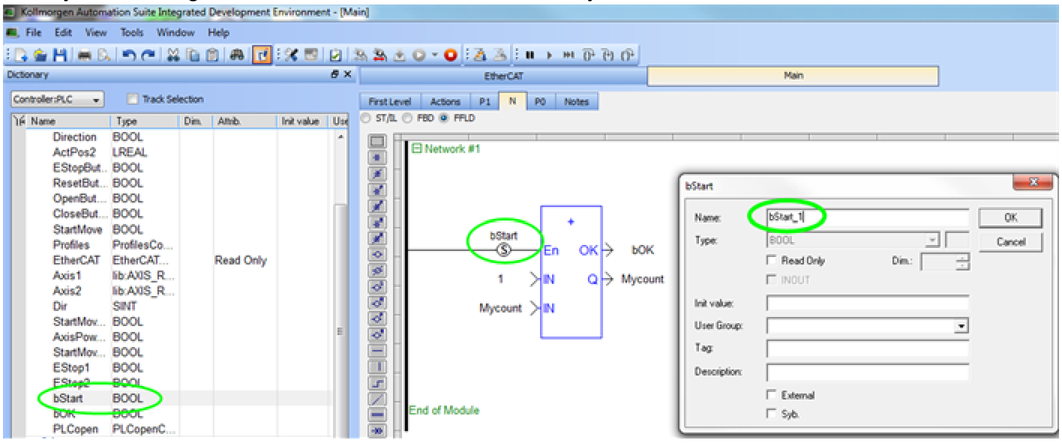
1. Select the PLC Options button from the tool bar.



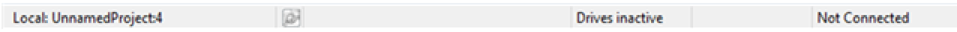
2. Enable Online Change.
3. Start executing the application. 

9.2.2.3 Using Online Change

1. Enable the Toggle Edit/Debug mode button in the tool bar. 
   Note that any application variables will not be updated, even though the application is running.
   

2. Modify a local or global variable name in the Dictionary.
   
3. Compile  and click the PLC Online Change download button  from the tool bar to download the changes to the controller.
   Note that the status bar has changed fro "Running" to "Paused" during the download, and back to "Running" once the download has finished.
   The applications variables should be updating.

9.2.2.4 Revert Online Change

1. Click the PLC Online Change Revert button  from the tool bar.
2. A message should be displayed stating that revert was successful.
3. Note the reverted state of the application in the IDE.
   

4. Connect to the device. 
5. Note the state of the application and the device.
   

6. From this state, you can choose to modify the application in the IDE using either Online Change or by:
   1. Stopping the application
   2. Making changes
   3. Recompile
   4. Download the application to the controller

9.2.3 Fast Inputs with Pipe Network

This section describes the Fast Input concept with Pipe Network motion engine, as well as how they can be used in your applications.

NOTE For PLCopen, refer to MC_TouchProbe

What are Fast Inputs?

Fast inputs allow a high-speed application to get position information about the occurrence of an external event at a higher resolution than the cycle time. Thanks to the precise timing of external events, an application can improve its control algorithm, resulting in higher operating performance. Fast (or high-speed) inputs are digital inputs of a drive that are configured to latch the time at which they are triggered.

The time capture can be triggered either by the positive (rising) edge or by the negative (falling) edge of the digital input. Note that it is also possible to configure a Fast Input to latch the motor position instead of latching the time (see "AKD Drive" on page 175). However, when working with KAS, time latching is more useful, because the positions of all the drives in the application can then be interpolated by means of the trigger block with the MLTrigReadPos function block. As a consequence, we assume in the procedures described below that Fast Inputs are configured to latch the time.

NOTE Only digital inputs 1 and 2 can be used as Fast Inputs.

TIP When using both fast inputs on one axis, a custom .XML file is required. Contact Kollmorgen.

TIP When using S300 or S700, Fast input has to be enabled by setting the drive keywords IN1MODE to 26 and IN2MODE to 26. This can be achieve using DriveGUI configuration tool.

Distributed Clock

When the input is triggered, the timestamp is latched. With EtherCAT, the timestamp sent to the KAS IDE via the MLAxisTimeStamp or MLTrigReadTime function blocks is based on the distributed clock that manages the reference clock (for more details on this concept, see page 159). The KAS IDE converts this timestamp into a relative offset inside the cycle.

9.2.3.1 Drive Configuration

The AKD drive has two capture engines which can be freely linked to any input. These high speed inputs can be used in application which, when triggered, caused a drive position to be captured and reported back to the controller.

However, KAS requires that the parameters MLFI_FIRST and MLFI_SECOND correspond to the physical Fast Inputs 1 and 2. Therefore, the AKD must be configured in order to link the fast input 1 with the engine 0 and the fast input 2 with the engine 1. The configuration is achieved by setting the drive parameters with the AKD GUI View (See also "AKD Drive" on page 175 for more details), or by using SDO write FB in the application program.

have to be doing the following:

CAP0.Trigger = 0

CAP1.Trigger = 1

This configuration must be done via SDO and can be done via initCommands of the master XML file.

9.2.3.2 How to Use Fast Inputs in PLC Programs

Once the drives are ready, you can use the trigger block or call the motion library functions that work with Fast Inputs from your PLC programs.

List of function blocks related to the Fast Input

- MLAxisCfgFastIn(write in the Latch Control Word the configuration for arming the

Fast Inputs on falling or rising edge)

- MLAxisIsTrigged or MLTrigIsTrigged (the Last Status Word is read to check if the Fast Input is triggered)
- MLAxisRstFastIn(write in the Latch Control Word to reset the Fast Input)
- MLAxisTimeStamp or MLTrigReadTime(read the absolute distributed clocks timestamp, and convert it to a relative offset inside a cycle)

Code Example

```
CASE StepCounter OF

0:

MLAxisRstFastIn(PipeNetwork.Feeder,MLFI_FIRST);

MLAxisMoveVel(PipeNetwork.Feeder,250.0); //Jog Feeder Axis to
search for sensor input

StepCounter := 1;

1:

IF MLAxisIsTrigged(PipeNetwork.Feeder,MLFI_FIRST,MLFI_RISING_
EDGE) THEN

MLAxisAbs(PipeNetwork.Feeder,MLAxisCmdPos
(PipeNetwork.Feeder)); //Stop motion when sensor is reached

StepCounter := 2;

END_IF;

2:

IF MLAxisGenIsRdy(PipeNetwork.Feeder) THEN

MLAxisWritePos(PipeNetwork.Feeder,0); //Set Feeder Axis
position to zero

StepCounter := 3;

END_IF;
```

Configuration of the Trigger Block

The trigger block is configured using its Properties dialog.

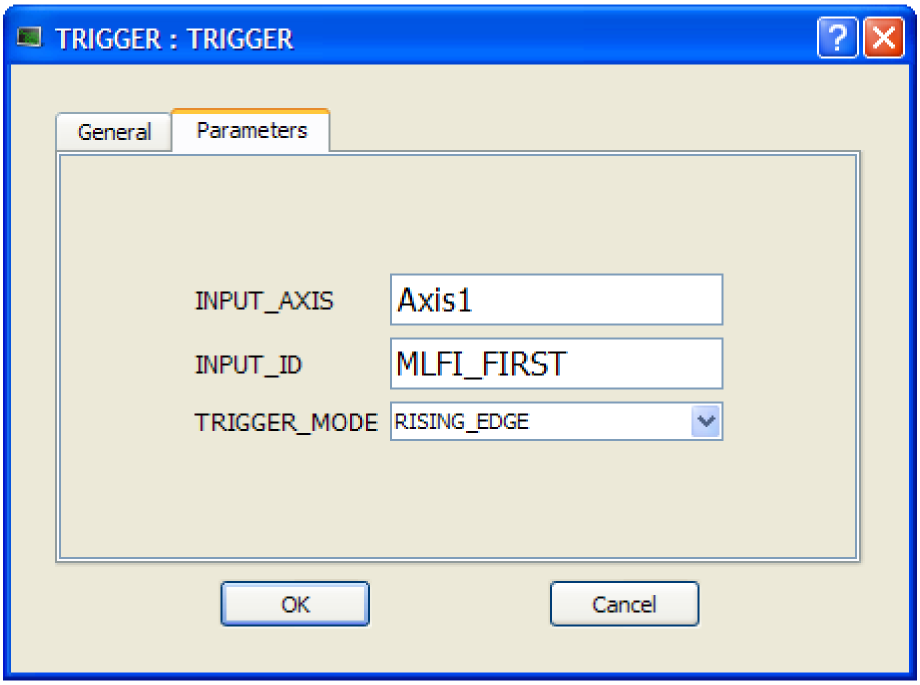


Figure 10-21: Configuration of the Trigger Block

Function	Description
INPUT_AXIS	Defines the axis whose Fast Input is used. This name is the same given to the corresponding axis block in the Pipe Network
INPUT_ID	Indicates which one of the two available Fast Inputs in that particular axis is used. The value can be MLFI_FIRST or MLFI_SECOND for the trigger block to be triggered on the arrival of the first or the second input respectively. Specify one of the following constants: MLFI_FIRST or MLFI_SECOND for the trigger block to be triggered on the arrival of the first or the second input respectively.
TRIGGER_MODE	Indicates if the trigger block responds to the rising edge or the falling edge of the Fast Input Specify one of the following constants: MLFI_RISING_EDGE or MLFI_FALLING_EDGE

Fast inputs with the Axis pipe block

This use case explains how to use the motion library functions of the axes when you want to detect the positive edge of the first Fast Input in the drive, and read its associated timestamp.

The sequence of calls is as follows:

1. MLTrigSetEdge(PipeNetwork.TRIGGER1,MLFI_FIRST,MLFI_RISING_EDGE)
   - This function reconfigures the edge of a trigger block.
   - This function only needs to be called if the desired edge is different than the edge specified in configuration of the trigger block or if the edge is different than the previous capture.
2. MLAxisCfgFastIn(PipeNetwork.AXIS1, MLFI_FIRST, MLFI_RISING_EDGE)

- Configure Fast Input 0 of AXIS1 to be triggered on the positive edge
- The first argument indicates the Axis pipe block in the Pipe Network that represents the drive to be configured
- The second argument identifies which of the two Fast Inputs of the drive is configured (can be 0 or 1)
- The third argument can indicate detection of positive edge when set to 1 and detection of negative edge when set to 2
  Note that if set to 0, Fast Input is disabled

3. MLAxisIsTrigged(PipeNetwork.AXIS1, 0, 1)
   - This function returns true if Fast Input 0 of AXIS1 has been triggered on the positive edge.
   - The meaning of the arguments is the same as in MLAxisCfgFastIn
4. MLAxisTimeStamp(PipeNetwork.AXIS1, 0, 1)
   - This function returns the time in microseconds when the Fast Input was triggered on the positive edge
     This time is relative to the start of the drive cycle time and its value is explained here
   - The meaning of the arguments is the same as in MLAxisCfgFastIn
5. MLAxisRstFastIn(PipeNetwork.AXIS1, MLFI_FIRST)
   - This function resets the Fast Input 0 of AXIS1. The reset keeps the configuration of the Fast Input, but it rearms it so it can be triggered again
   - The meaning of the first two arguments is the same as in MLAxisCfgFastIn

Fast inputs with the Trigger pipe block

This use case explains how to use the motion library functions of the trigger block, which allows an application to get the position at any point in the Pipe Network when a Fast Input is triggered. It is done by using the timestamp received and interpolating the position of the Pipe Network at that precise time.

NOTE Since timestamps of a Fast Input are obtained with a delay of some cycles, the correction done to the Pipe Network position with the trigger block is then relative to the cycle when the Fast Input is issued.

After configuring the trigger block, the order of calls to its motion library functions is as follows:

1. MLTrigSetEdge(PipeNetwork.TRIGGER1,MLFI_FIRST,MLFI_RISING_EDGE)
   - This function reconfigures the edge of a trigger lock.
   - This function only needs to be called if the desired edge is different than the edge specified in configuration of the trigger block or if the edge is different than the previous capture.
2. MLAxisCfgFastIn(PipeNetwork.AXIS1, MLFI_FIRST, MLFI_RISING_EDGE)
   - This function call is necessary at least one time, even if the Trigger pipe block is configured properly
3. MLTrigIsTrigged(PipeNetwork.TRIGGER1)
   - This function returns TRUE if the Fast Input associated to the Trigger pipe block given as argument has been triggered
4. MLTrigReadPos(PipeNetwork.TRIGGER1)
   - This function returns the position of the Pipe Network at the time that the Fast Input associated with the Trigger pipe block was issued

NOTE You have to correct the position by taking into account the delay due to the number of cycles needed to read the timestamp of the Fast Input

5. MLTrigReadTime(PipeNetwork.TRIGGER1)

- This function returns the time associated with the Fast Input as explained here Note that this function is of lesser importance compared to the previous one.

6. MLTrigClearFlag(PipeNetwork.TRIGGER1)
   - This function rearms the Trigger pipe block
7. MLAxisRstFastIn(PipeNetwork.AXIS1, MLFI_FIRST)
   - This function rearms the Axis pipe block

Delay compensation

Sometimes the sensor which is linked to the Fast Input introduces a significant delay in the latched timestamp. In such cases, the trigger block has a configurable parameter: the **Delay compensation**. This parameter allows you to interpolate the position correctly, taking into account the delay of the sensor as follows:

Corrected timestamp: = Fast input timestamp - DelayCompensation

Two function blocks allow you to set and read the DelayCompensation parameter:

• MLTrigReadDelay(DINT TRIGGERID)

• MLTrigWriteDelay(DINT TRIGGERID, LREAL delay)

Where the time parameter is specified in microseconds.

How to interpret the timestamp?

The timestamp is based on the EtherCAT system time. For this value to make sense, distributed clock must be activated in the drive and in the EtherCAT master.

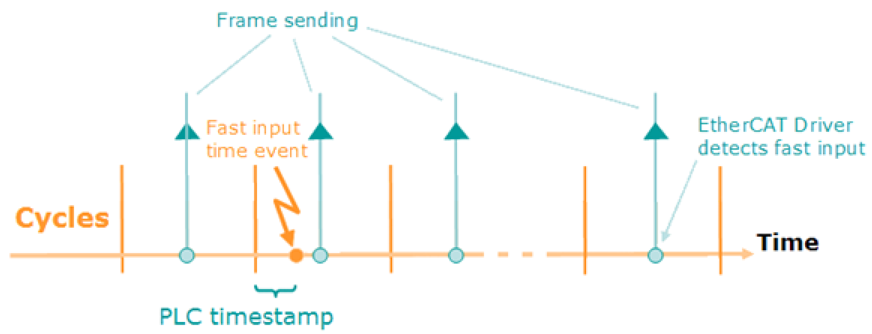


Figure 10-22: PLC Timestamp Related to Fast Input Event

The timestamp returned is relative to the beginning of the cycle in which the Fast Input is triggered. It is called PLC timestamp since it is the value that you can use in PLC programs.

When the EtherCAT driver realizes that a Fast Input has been triggered, it subtracts the m_BusRunningTime minus 2 times the cycle time (see diagram) from the 32-bit EtherCAT system time received in the frame. The difference is divided by 1000 to convert the timestamp value to microseconds.

Note: This diagram is currently valid for 1 ms cycle time or higher. For lower cycle times, more cycles are needed to detect the Fast Input, unless a new firmware corrects the situation. The latest firmware version tested with Fast Input support up to the writing of this document is 3.66 beta 5.

9.2.4 Torque Feed-forward

The torque feed-forward tells the controller what forces is required to move the axis in an arbitrary trajectory.

Here are the major features of torque feed-forward:

- Torque feed-forward results in virtually instantaneous response of the system.
- Feedback control loops (using PID loop or similar) take a finite amount of time before reacting.
- Torque feed-forward relies on an imperfect model of the system. This means that the feed-forwards need help from the feedback control loop in order to get accurate motion.
- Torque feed-forward can make the bulk of the move very quickly, while the feedback control loops correct the small errors that remain. As a result, a faster settling time can be achieved than if torque feed-forward was not used.
- There is a common misconception that torque feed-forward is similar to control loops and result in instability. Torque feed-forward is open loop, so it cannot suffer from closed loop instability.
- Torque feed-forward is typically less sensitive to being misadjusted than closed loop parameters.
- Feedback control systems can be excited into instability by grossly misadjusted torque feed-forward. However, the amount of misadjustment in the torque feed-forward necessary to cause such instability is very rare.

9.2.5 PLCopen Homing

9.2.5.1 PLCopen Homing Description

The homing features provided in PLCopen create tools for homing of PLCopen axes. Homing may be performed utilizing the MC_Reference function block, utilizing Custom Homing Library UDFB's or by writing your own homing cycles.

- Utilizing MC_Reference
  The application specifies a position for an axis to be assigned to a reference position, then invokes the MC_Reference function block to optionally generate motion to move the axis to the reference location. The AKD capture engine (previously set up by the application via SDO commands) captures the position of the reference location. Based on the desired reference position and the captured actual position, the coordinate system is shifted to correlate the desired reference position to this location.
- Writing your own homing cycles
  UDFBs can be written to provide specific "canned" homing cycles based on feedback type, and desired homing sequences such as homing off of limit switches, encoder markers, homing to “zero” or null positions etc. by proper configuration of the AKD capture engine, the MC_Reference and MC_Setposition function blocks.
- Utilizing Custom Libraries
  A library already contains a set of homing UDFB. Contact the Support for more information.
  To add the library to your project, refer to chapter "Step 10 of 15 - Create and Use Custom Libraries" on page 265

9.2.5.2 PLCopen Homing Methods

The following common homing methods (among others) can be performed in PLCopen. This section details the setting of the ADK parameters and the PLCopen function blocks to accomplish these methods.

PLCopen does not limit you to these methods, as the capture engine is very configurable.

Home using Current Position

Homing using the current position is simply accomplished using the MC_SetPosition function block. Using this function block, the current position can be set to any value.

Find Input

Homing using a drive input is accomplished by configuring the AKD capture engine, and then using the MC_Reference function block. The following capture engine parameters need to be configured, along with the following input parameters in the MC_Reference.

- Capture Event has to be set to ignore preconditions (0)
- Capture edge – capture edge is programmed in the MC_Referece block
- Capture Trigger must be set to the desired drive input (0-6)
- Capture mode must be set to capture position (0)
- Capture preselect is not used
- Capture Precondition edge is not used
- MC_Reference inputs:
  Trigger_Ref.InputID must be set to 0 or 1 to select which AKD capture engine to use
  Trigger_Ref.Direction must be set to Rising (1) or Falling (2) to select Capture Edge
  Trigger_Ref.Trigid is not required.
  Position input must be programmed to the desired position at the switch.
  Option input must be programmed to 0 for "use latched position".

Find Input then find Zero Angle

Homing using a drive input along with the zero angle is similar to "Find Input" except the position is defined at the zero angle of the feedback device, rather than the switch location. It is typically used for resolver feedback.

- Capture Event must be set to ignore preconditions (0)
- Capture edge – capture edge is programmed in the MC_Referece block
- Capture Trigger must be set to the desired drive input (0-6)
- Capture preselect is not used
- Capture Precondition edge is not used
- Capture mode must be set to capture position (0)
- MC_Reference inputs:
  Trigger_Ref.InputID must be set to 0 or 1 to select which AKD capture engine to use.
  Trigger_Ref.Direction must be set to Rising (1) or Falling (2) to select switch capture edge.
  Trigger_Ref.Trigid is not required.
  Position input must be programmed to the desired position at the null closest to the switch.
  Option input must be programmed to identify the number of poles the resolver has.

Find Input then find Index

Homing using a drive input along with the index is similar to "Find Input" except the position is defined at the index pulse of the feedback device, rather than the switch location. It is typically used for incremental encoder feedback. To accomplish this, a precondition is used in the capture engine. Specifically, the input is the precondition, and the index is the event. The reference method looks for the switch first, and then the index pulse.

- Capture Event must be set to the desired switch operation. Typically set to 1 to require the edge of the switch. Set to 2 or 3 if the state of the switch is required.
- Capture preselect must be set to the desired drive input (0-7)
- Capture edge – capture edge of index pulse is programmed in the MC_Referece block
- Capture Trigger must be set to the desired index input (10 = primary index,11 = tertiary index)
- Capture mode must be set to capture position (0)
- MC_Reference inputs:
  Trigger_Ref.InputID must be set to 0 or 1 to select which AKD capture engine to use
  Trigger_Ref.Direction must be set to Rising (1) or Falling (2) to select Capture Edge
  Trigger_Ref.Trigid is not required.
  Position input must be programmed to the desired position at the index pulse.
  Option input must be programmed to 0 for "use latched position".

Find Index

Homing using a drive index pulse is accomplished by configuring the AKD capture engine, and then using the MC_Reference function block. The following capture engine parameters need to be configured, along with the following input parameters in the MC_Reference.

- Capture Event must be set to ignore preconditions (0)
- Capture edge – capture edge is programmed in the MC_Referece block
- Capture Trigger must be set to the desired index input (10 = primary index,11 = tertiary index)
- Capture mode must be set to capture position (0)
- Capture preselect is not used
- Capture Precondition edge is not used
- MC_Reference inputs:
  Trigger_Ref.InputID must be set to 0 or 1 to select which AKD capture engine to use
  Trigger_Ref.Direction must be set to Rising (1) or Falling (2) to select Capture Edge
  Trigger_Ref.Trigid is not required.
  Position input must be programmed to the desired position at the index pulse.
  Option input must be programmed to 0 for use latched position.

9.2.5.3 AKD Capture Engine Configuration

The AKD capture engine provides a broad range of capabilities for configuration of the capture event(s). Furthermore, it is capable of configuring preconditions to allow the application programmer to specify sequential events or conditions that must be met before the capture event can be triggered. The capture Engine in the AKD is configured with SDO #0x3460 (subindexes 1 to 10). The AKD supports two capture engines (0 and 1); the application programmer must configure the desired engine.

Sub Index #	Function
1	Trigger for capture engine 0
2	Trigger for capture engine 1
3	Mode for capture engine 0
4	Mode for capture engine 1
5	Capture Event for capture engine 0
6	Capture Event for capture engine 1
7	Precondition edge for capture engine 0

Sub Index #	Function
8	Precondition edge for capture engine 1
9	Preselect for capture engine 0
10	Preselect for capture engine 1

The following section details the configuration parameters for the ADK capture engines.

- Capture event (SDO object #0x3460 subindex engine 0 = 5/engine 1 = 6)
  - 0 = ignore preconditions
  - 1 = trigger edge after the precondition edge
  - 2 = trigger edge while precondition = 1
  - 3 = trigger edge while precondition = 0
- Capture edge – capture edge is programmed in the MC_Reference function block.
- Capture Trigger (SDO object #0x3460 subindex 1/2)
  - 0 = general input 1
  - 1 = general input 2
  - …
  - 6 = general input 7
  - 7 = rs485 input 1
  - 8 = rs485 input 2
  - 9 = rs485 input 3
  - 10 = primary index
  - 11 = tertiary index

  For more details, refer to CAP0.PRESELECT, CAP1.PRESELECT section.
- Capture precondition edge (SDO object #0x3460, subindex 7/8)
  - 0 = reserved
  - 1 = precondition with rising edge
  - 2 = precondition with falling edge
  - 3 = precondition with rising and falling edges
- Capture preselect (SDO object #0x3460 subindex 9/10)
  - 0 = general input 1
  - 1 = general input 2
  - …
  - 6 = general input 7
  - 7 = rs485 input 1
  - 8 = rs485 input 2
  - 9 = rs485 input 3
  - 10 = primary index
  - 11 = tertiary index
- Capture mode (SDO object #0x3460 subindex 3/4)
  - 0 = capture position
  - 1 = capture internal time
  - 2 = capture EtherCAT distributed time (DCT)
  - 3 = capture zero angle position

9.2.6 Pipe Network Homing

UDFBs can be written to provide specific "canned" homing cycles based on feedback type. Contact the Support for more information.

9.2.7 Registration

Registration is a technique used to maintain the positional accuracy in repetitive processes. It uses a Fast Input switch, typically a photo eye, to measure product position and adjust the axis (or axes) to compensate for variations. There are two basic forms of registration: single-axis registration and master/slave registration.

9.2.7.1 Single-Axis Registration

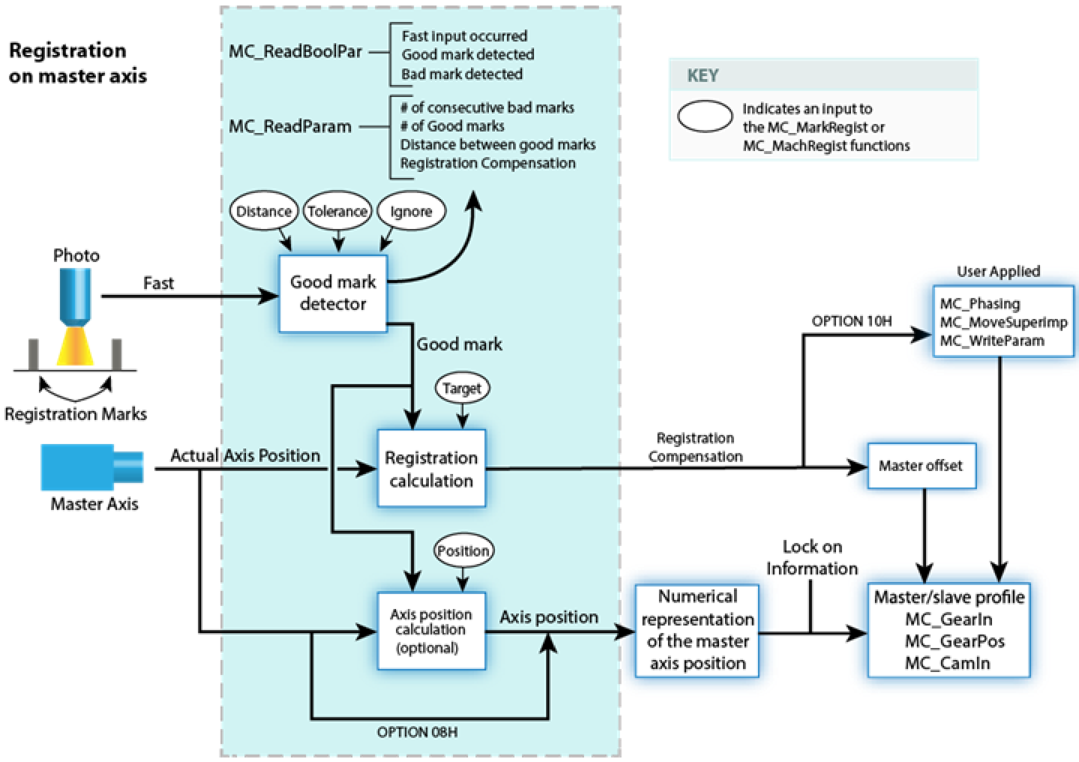
Single-axis registration is performed on an axis running a discrete move such as MC_MoveAbsolute or MC_MoveRelative. When the Fast Input latches the position of the product, the axis position is reset, typically to zero. This resets the axis's coordinates for each product to accommodate for variations in the distance between products and keep the process synchronized to the product over many repetitions.

9.2.7.2 Master/Slave Registration

Master/slave registration is performed on an axis running a master/slave move such as MC_GearIn or MC_CamIn. It can be performed by tracking the position of the master axis (Master Registration) or tracking the position of the slave axis (Slave Registration) or both. This type of registration adjusts the positional relationship between the master and slave axes to accommodate for variations in the distance between products and keep the process synchronized to the product over many repetitions.

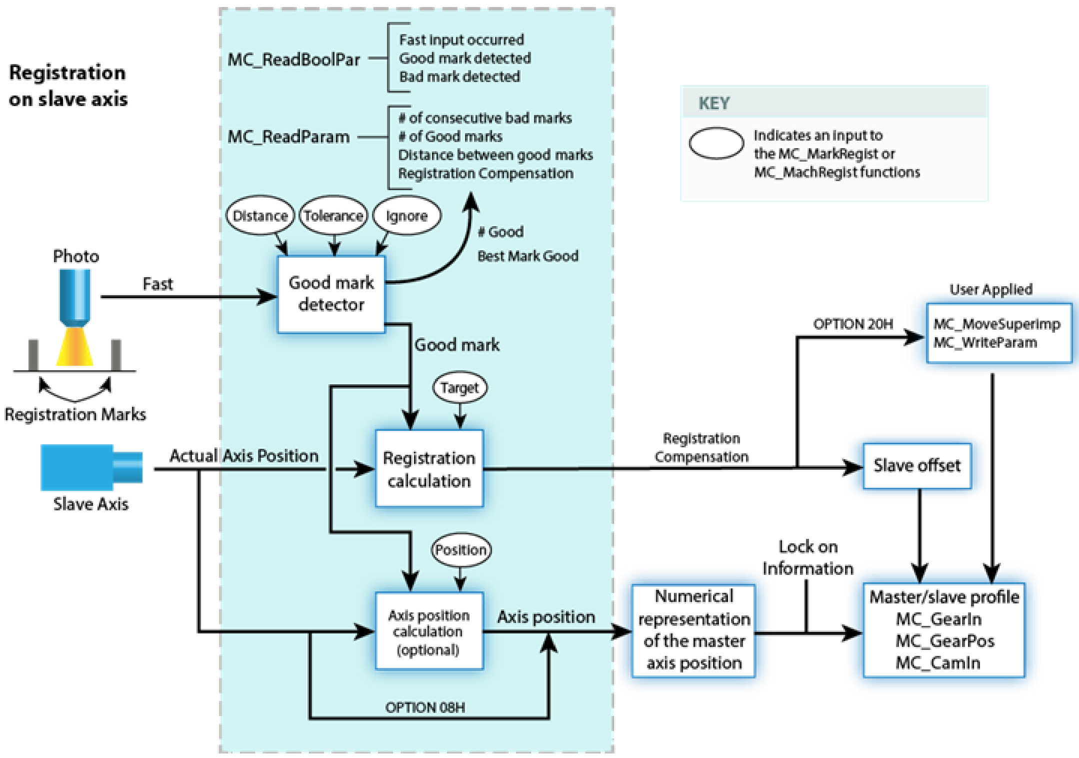
Master Registration

Master registration is performed by having the Fast Input switch trigger on a mark controlled by the master axis. When the Fast Input latches the position of the master axis at this mark, the distance between this position and the position of the previous mark is compared to an expected distance. This difference is added to the slave axis's master offset to adjust the position of the slave axis with respect to the position of the master.



Slave Registration

Slave registration is performed by having the Fast Input switch trigger on a mark controlled by the slave axis. When the Fast Input latches the position of the slave axis at this mark, the distance between this position and the position of the previous mark is compared to an expected distance. This difference is added to the slave axis's slave offset to adjust the position of the slave axis with respect to the position of the master.



The figure "Registration" (see page 510) below shows an example of a printing application using registration. The axis controlling the web is the master and the axis controlling the print head is the slave. When the photo eye detects a registration mark on the web, the master position is latched. The application calculates the amount of registration compensation required by comparing the actual distance between marks to the expected distance. Then, it writes that value to the slave axis's master offset delta. This adjusts the positional relationship between the web and the print head so that each print on the web are placed accurately.

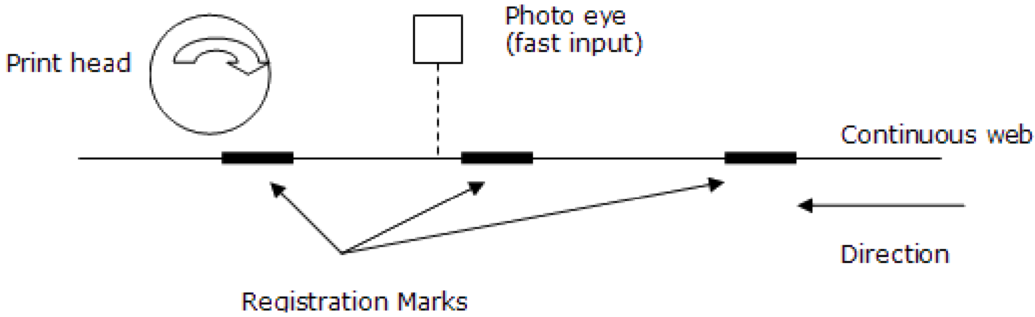


Figure 10-23: Registration

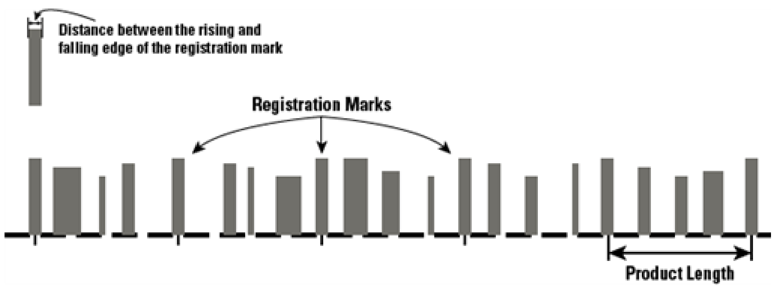
9.2.7.3 Registration Application Guide

In many closed-loop servo systems it is often necessary to maintain synchronization and accurate positioning repeatedly throughout a process. This can be difficult when the product or process itself is inconsistent. Using registration helps you to overcome this difficulty.

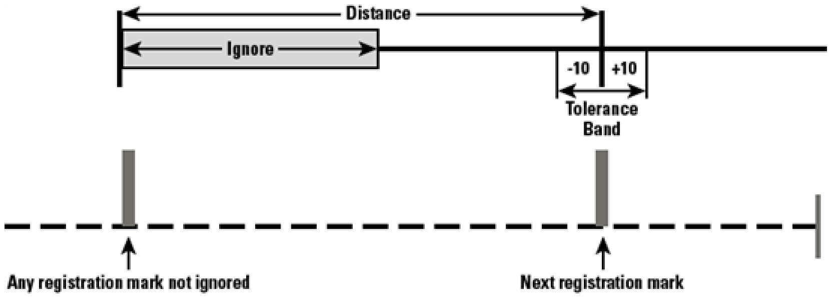
Many factors can contribute to inconsistency. Some examples are:

- Working with non-rigid material which may stretch or shrink during processing.
- Working with the mechanics of a system where the revolution of a feedback device may give you, for example, 5975 counts on one revolution and 5974 on the next.
- Unevenly spaced products on a web or belt.
- Materials which are sensitive to temperature, humidity, pressure, etc.

To overcome the various product and process inconsistencies registration capabilities may be required and can be applied on any servo or digitizing axis, and with any move type. Typically in these applications, sensors are used to detect the position of the product. With non-rigid materials, which may stretch or shrink or are unevenly spaced, a photo eye can detect registration marks on the material. With rigid products or processes a proximity sensor can detect leading or trailing product edges for material spacing.



With registration, the registration sensor is wired to the fast input on the servo drive providing the means for the registration trigger that will capture the axis position at the instance of the registration event. When the event occurs, the system will calculate a registration correction to compensate for the inconsistency and then can apply the correction to the registration axis and/or change the numerical representation of the registration axis position.



This is important in applications such as printing, packaging, and converting where the process must be precisely coordinated and any non-rigid material cannot be depended upon to retain dimensional relationships. These applications usually involve master/slave moves. Not only does registration provide correction but the fast input signals can also be used as repeatable references to which the master and all subsequent slaves continually synchronize. Two main types of registration are "Mark to Mark Registration" (see page 511) and "Mark to Machine Registration" (see page 512).

Mark to Mark Registration

Mark to Mark Registration is implemented with the function block MC_MarkRegist and is based on the desired distance between two registration marks. This distance is used to qualify a “good” mark and then calculate a registration correction. Examples of the different mark to mark registration variations include:

- Clear Lane Registration
- Print Registration
- Product Registration
- Rotary Registration

Clear Lane Registration is the most common type of registration used in industry. A dedicated lane on the material is reserved solely for registration purposes. Only the registration marks in the lane will trigger the sensor and fast input on the servo drive. The distance between one registration mark and the next mark is the basis for registration correction calculations.

Print Registration is less common than Clear Lane Registration. Print registration involves picking out a distinct distance between print features. Here too, this distance is the basis for registration correction calculations. Print registration is used when it is not practical to have a clear lane.

Product Registration uses cycle position where the relationship of the product position in the process cycle is important. Product registration can occur synchronously or asynchronously. In synchronous product registration, products typically flow continuously through the process. Registration will measure and calculate small adjustments required to move each product to the correct position. The distance between products is the basis for registration correction calculations. Asynchronous applications may require the axis to sit and wait for the product to pass the sensor and, when it does, initiate a move and also apply registration correction based on the position of the product.

Rotary Registration is used to maintain a fixed axis position reference point on a rotary axis that has non-integer feedback in one revolution. If the axis position is not adjusted, the actual axis position would walk away from the desired position every revolution because of the non-integer number of feedback units per revolution.

Mark to Machine Registration

Mark to Machine Registration is implemented with the function block MC_MachRegist and is based on a desired target position on a machine axis. Like mark to mark registration, mark to machine registration uses the desired distance between two registration marks to qualify a “good” mark (using Clear Lane, Print, and Product registration). But instead of using the mark to mark length, it uses an axis position as its basis for correction.

9.2.8 Error Management

When a non-fatal error occurs and motion must be stopped quickly, the following procedure can be taken:

For each axis:

Step	Example Application Code
Send Stop Command for each axis	MLAxisStop(PipeNetwork.AXI_A1_Axis, TRUE, DEF_A1_StopDec);
Stop the Axis Motion Generator	MLAxisMoveVel(PipeNetwork.AXI_A1_Axis, 0.0);
Wait for Axis to be stopped	AxisStatus := MLAxisStatus(PipeNetwork.AXI_A1_Axis); IF AxisStatus.11 THEN MLAxisStop(PipeNetwork.AXI_A1_Axis,FALSE,DEF_A1_StopDec);
Turn power off(disable) all the axes	MLAxisPower(PipeNetwork.AXI_A1_Axis,FALSE);
Disconnect Pipe Network from the axis	MLCNVDisconnect(PipeNetwork.CNV_A1);

For the machine:

Step	Example Application Code
Stop Command at the master block level	MLMstRun(PipeNetwork.MASTER, 0.0);
Wait for Master command to be stopped	IF A1_AckState = DEF_StateErrorStop AND A2_Ackstate = DEF_StateErrorStop AND MLBlkIsReady(PipeNetwork.MASTER) THEN PrintF('*** ErrorStop M1=%i ***', M1_StatusWord,0,0,0); M1_AckState := DEF_StateErrorStop;

This procedure for error management is based on the **Project Structure Guidelines** as described in paragraph "Application Software Structure - Implementation" on page 563

For information on **restarting the motion**, refer to paragraph "Restarting Motion" on page 512

9.2.9 Restarting Motion

An advantage of the Pipe Network is the ability to minimize machine downtime and reduce material waste when a non-fatal error occurs. After stopping the motion with MLAxisStop command, it can be restarting by using the MLAxisReAlign function block.

IMPORTANT MLAxisReAlign must be called after the MLAxisStop command, otherwise all motion commands are ignored

For each axis:

Step	Example Application Code
Check Axis Status	AxisStatus := MLAxisStatus(PipeNetwork.AXI_A1_Axis); IF AxisStatus.6 THEN StepCounter := 1; END_IF;
Turn axis back on (re-enable)	IF MLAxisPower(PipeNetwork.AXI_A1_Axis, PowerUp) THEN StepCounter := 2; END_IF;
Calculate position difference between the Reference and Actual Positions	DeltaPos := (MLAxisCmdPos(PipeNetwork.AXI_A1_Axis) - MLAxisReadActPos(PipeNetwork.AXI_A1_Axis));

Determine how far to move	IF DeltaPos > LREAL#0.5*DEF_A1_PosPeriod THEN
	DeltaPos := DeltaPos - DEF_A1_PosPeriod;
	ELSE
	IF DeltaPos < LREAL#-0.5*DEF_A1_PosPeriod THEN
	DeltaPos := DeltaPos + DEF_A1_PosPeriod;
	END_IF;
	END_IF;
	MLAxisReAlign(PipeNetwork.AXI_A1_Axis, 1000.0, 1000.0, 100.0, DeltaPos);
	StepCounter := 3;

Wait for move to be completed	IF MLAxisReAlgnRdy(PipeNetwork.AXI_A1_Axis) THEN StepCounter := 4; END_IF;

For the machine:

Step	Example Application Code
Execute multi-axis move	MLMstRun(PipeNetwork.MASTER, 500);

9.2.10 Superimposed Axes with PLCOpen

This feature allows the application program to superimpose the moves of multiple axes ("Superimposed Axes") on top of the move of another axis ("Receiving Axis"). This is performed internally by adding the command deltas of the Superimposed Axes to the command delta of the Receiving Axis. Up to four different Superimposed Axes can be superimposed upon a Receiving Axis.

9.2.11 Cam On The Fly

In addition to creating and modifying a Cam Profile using the IDE, an application programmer can also create or modify a new Cam Profile directly from their application. This allows new Cam Profiles to be defined while the application is still online, without stopping the machine to load a new application. An application programmer might use this feature to modify their application at runtime to adjust their Cam Profiles for varying product shapes and sizes.

The following steps will guide you through creating and building a new Cam Profile on the fly.

1. Before the motion engine is started, reserve memory for the profile that will be created on the fly. This is done using MLProfileCreate.
2. Define the profile properties and profile data points. This can be done at any time but must be done prior to calling MLProfileBuild.
3. Build the profile with MLProfileBuild. This takes a number of cycles and the profile will note be ready until the "Done" flag is set to TRUE.
4. The newly created profile can now be used exactly like a profile built in the IDE; it can be used in any number of camming relationships using any of the PipeNetwork and PLCOpen Cam Profile functions/function blocks.

Once you have created a Cam Profile using MLProfileBuild, if you need to modify that Cam Profile you must first release the existing Cam Profile. The following steps will release the Cam Profile.

1. Ensure the profile is not in use (Deactivate the existing profile). In Pipe Network perform a MLCamSwitch on an active Pipe to a different Profile or deactivate the pipe. In PLCOpen, perform an MC_CamOut on the profile or abort a profile move.
2. Call MLProfileRelease with the desired ProfileID as its input to release the profile so it can be modified.
3. When the "Done" flag is set, the ProfileID will be free. Modify the existing profiles data and then rebuild the profile MLProfileBuild.

An application is limited to 256 Cam Profile IDs. If additional profiles are desired, an existing profile can be released with MLProfileRelease. This frees its ProfileID so it can be used in the construction of a new profile. Releasing a profile is simple, and is performed with the following steps.

1. Ensure that the profile is not in use (Deactivate the existing profile).
2. Call MLProfileRelease with the desired ProfileID as its input to release the profile so it can be modified.
3. When the "Done" flag is set, the ProfileID will be free and the old profile can be used for a new profile, calling either MLProfileInit or MLProfileBuild.

9.3 Motion Bus and Fieldbuses

Depending on the fieldbus used in your project (EtherCAT, Profibus), you have to make use of the following configuration tools:

9.3.1 EtherCAT

- For configuration, see page 196
- For I/O mapping, see page 270
- For error management, see page 522

See the Beckhoff Web site for EtherCAT XML Device Description (http://www.beckhoff.se/english.asp?download/elconfg.htm).

9.3.2 Ethernet/IP

- Setting up " Ethernet/IP IO Client " (see page 525)
- Setting up " Ethernet/IP Server " (see page 526)
- Setting up " Ethernet/IP Tag Client " (see page 528)

9.3.3 Modbus

- Setting up "Modbus Slave" (see page 530)

9.3.4 Profibus

This fieldbus can be used to set the communication between a Profibus master (e.g. AKC with a PCI card) and Profibus slaves (e.g. Wago couplers and I/O terminals)

- For configuring the Profibus master, see page 515
- For I/O mapping, see page 516

9.3.5 Profinet

This fieldbus is Profibus over ethernet. We define one controller and have 1 or more devices.

- For configuring the controller, see " Profinet IO RT Controller Configuration " (see page 533).
- For configuring devices, see "Profinet IO RT Device Configuration" (see page 547).

9.3.6 Profibus Configuration

NOTE To configure Profibus, you first need to have INtime properly set up with the Profibus driver activated (for more details, refer to "Configuring INtime" in the Getting Started guide)

To configure the controller with **SyCon** when using Profibus slave, follow these instructions:

1. Install SyCon on both master and slave Profibus devices.
2. Start SyCon on the master device. You must have an empty configuration.
3. Add the master device to the configuration: click on the “Insert Master” icon, choose the **EC1-DEB-DPM** and change its station address if needed.
4. Add the slave device to the configuration: click on the “Insert Slave” icon, choose the **EC1-DEB-DPS** and change its station address if needed.
5. Right click on the slave representation and choose “Slave configuration...”
6. Insert a “blank space” module as the first module. It is to bypass a bug of the current slave firmware. Hilscher and Kontron are working on this and a fix will soon be available.
7. Insert I/O modules as you need. Please select modules with consistency “X byte(s) input/output con”. Selected module directions are from the master point of view: if you select an output module, it means an output for the master and an input for the slave.
8. Save the configuration into a *.pb (Profibus) file.
9. Copy the configuration file on the slave device.
10. Start SyCon on the slave device. Load the configuration file.
11. On the master device, in SyCon, select the master device representation (left click on it). Select the menu entry “Online > Download...”. If needed, select the “CIF Device Driver” and the board. Answer “Yes” to the question. The download then starts.

12. On the slave device, in SyCon, select the slave device representation (left click on it). Select the menu entry “Online > Download...”. If needed, select the “CIF Device Driver” and the board. Answer “Yes” to the question. The download then starts.
13. Ensure that the master and the slave are connected by a Profibus cable correctly setup (with termination).
14. On the master device, in SyCon, select the menu entry “Online > Start Debug Mode”. The bus representation must turn to green. If not, try to fix the problem. Select the menu entry “Online > Stop Debug Mode”.
15. On both devices, in SyCon, select the menu entry “Online > I/O Monitor...” and try to exchange some I/Os. If it does not work, try to fix the problem

After completing the configuration, you are ready to develop programs with the KAS IDE, declare some I/Os and launch the KAS Runtime. You have to launch the KAS Runtime on the PAC slave device before starting. If you do not, you can get a network error that can easily be fixed by unplugging the Profibus cable from the master and re-plugging it (this error will be better handled in a future release so that you do not need this manipulation).

For more details, refer to: SyCon® provided by Hilscher

Or open PDF file here: System Configurator PROFIBUS

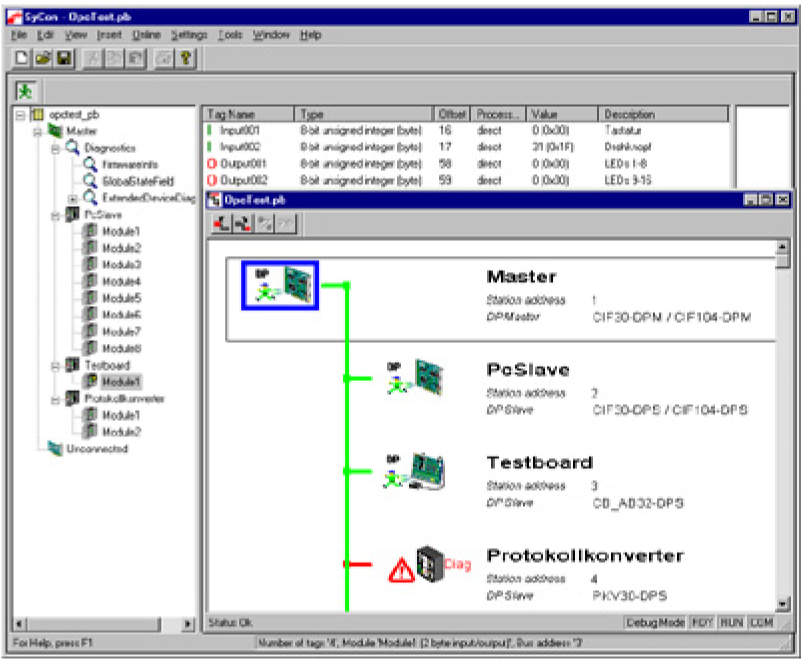


Figure 10-24: SyCon System Configuration

9.3.7 I/O Mapping (for Profibus Fieldbus)

This procedure describes how to map inputs and outputs to PLC variables on the Profibus fieldbus.

The mapping can be done from the Dictionary (as described below), but also with the **I/O Editor**.

NOTE For remote IOs on EtherCAT Motion Bus, refer to paragraph "Step 11 of 15 - Map Input and Output to Variables" on page 270

To map a variable from the Dictionary to a physical input or output:

1. Open the Variable list editor available in the **Dictionary** toolbox
2. Right-click on the variable to be mapped

3. Select the **Variable I/O Mapping** command in the menu to open the mapping dialog

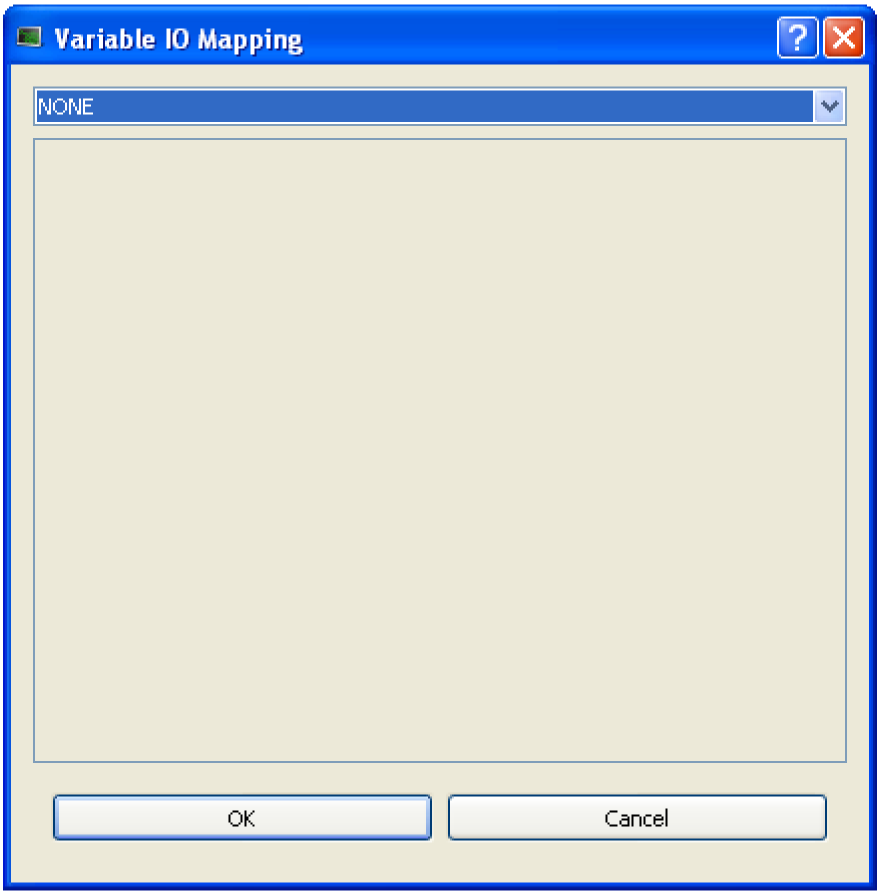


Figure 10-25: Mapping Dialog

By default the setting is NONE which means that the variable is a standard variable.

4. Select I/O (instead of NONE) and the I/O configuration panel appears:

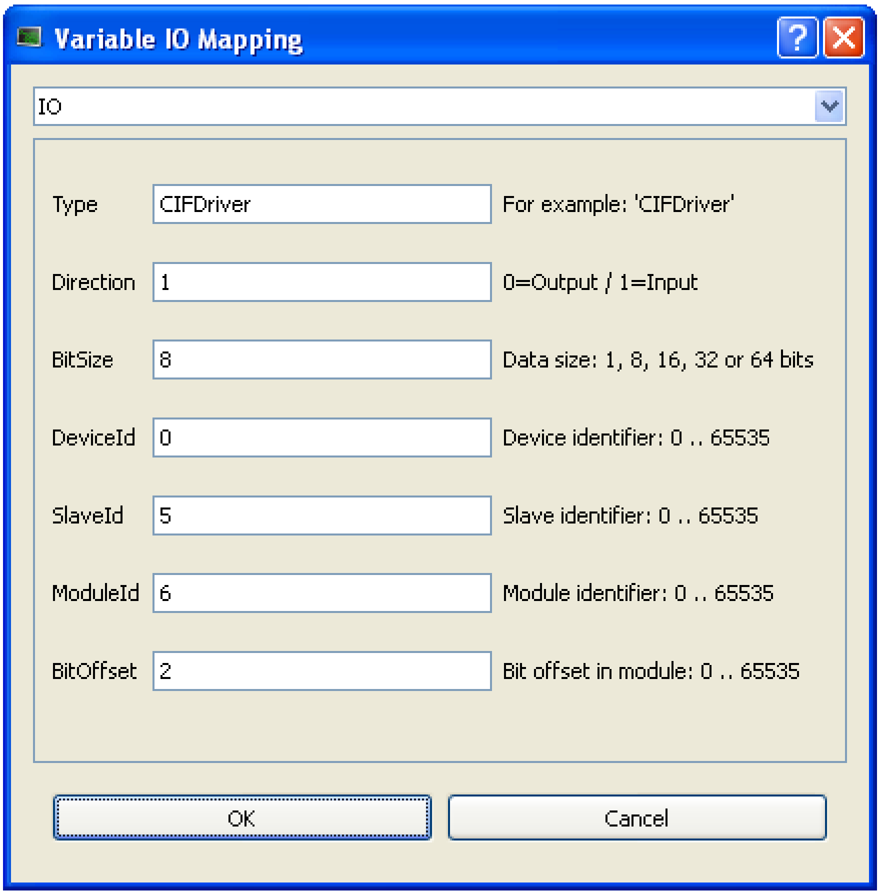


Figure 10-26: Variable I/O Mapping

This form allows you to configure the different types of I/Os supported by KAS by defining the following parameters:

Field	Description
Type	Defines the I/O type of fieldbus: CIFDriver for Profibus
Direction	Specifies if the variable is an Output or an Input
BitSize	Defines the length of the frame to be mapped (see length of data types here)
DeviceId	Defines the address of the I/O communication card located on to the target device (i.e. IPC)
SlaveId	Defines the address of the I/O node on the filedbus ring (See also "Communication and Fieldbus" on page 64) For EtherCAT, a fixed address is assigned to each slave node that follows the following convention: • first slave item on the network has address 1001 • second slave item has address 1002, and so on...
ModuleId	For the current variable, defines the address identifier (id) in the slice
BitOffset	Set to the first bit in the module of the slice which is mapped

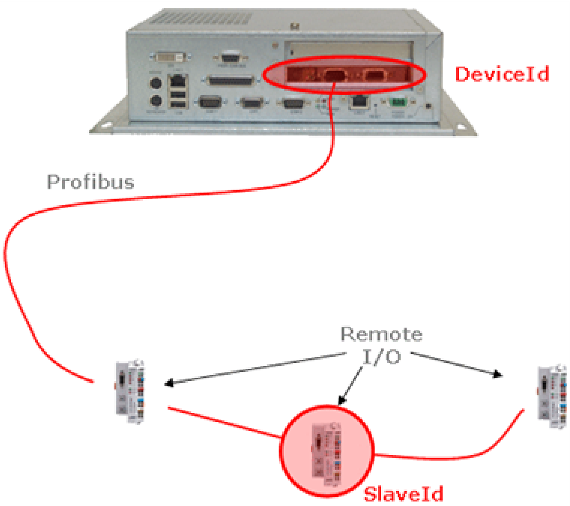


Figure 10-27: Variable I/O Mapping - Defining Addresses

To map a variable on Profibus, define the fields as follows:

Field	Definition
I/O type	Enter CIFDriver
DeviceId	Set to 0
SlaveId	Set to the id of the I/O node
ModuleId	Set to the id of the slice.
BitOffset	Set to the first bit of the slice which has to be mapped

Table 10-2: I/O Mapping on Profibus

NOTE For some drivers, you can also select CUSTOM.

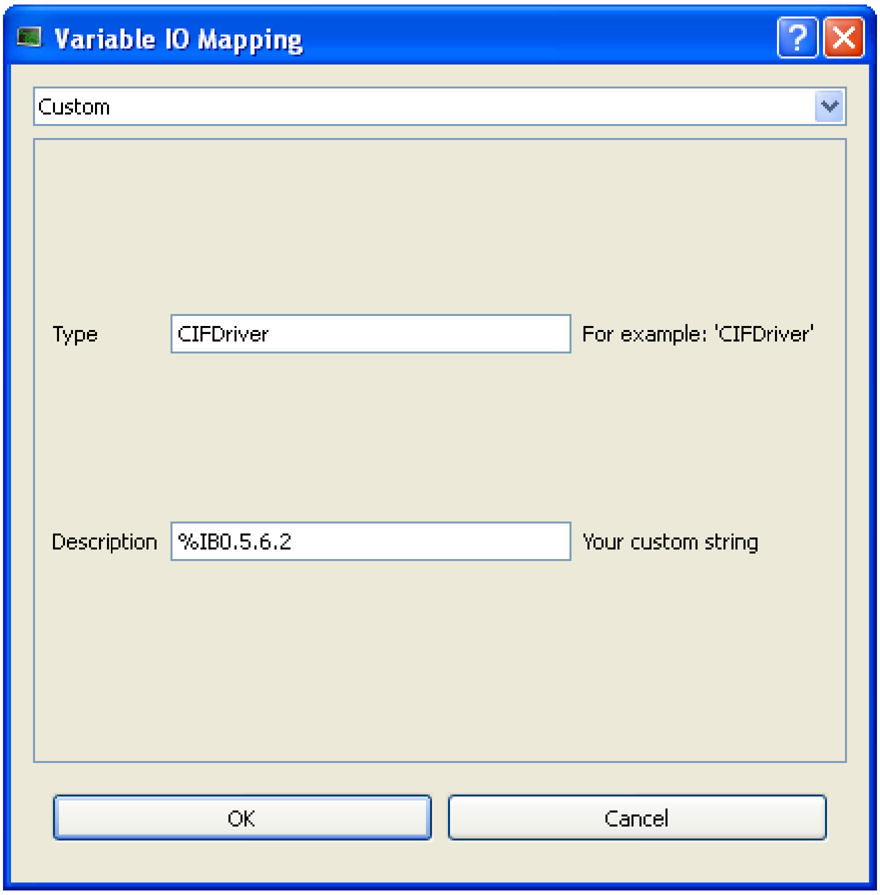


Figure 10-28: Variable I/O Mapping - Custom

For more details about the format of the **Description** field, see page 457.

9.3.8 Add Unsupported EtherCAT Device

NOTE This procedure is for **advanced** users only

When your project contains EtherCAT devices that are not supported by KAS, you have to create the configuration with an external tool, and perform the following steps:

1. Get the AKD device description ESI file from the official AKD distribution
2. Ensure all the device description ESI files are available for the external tool
3. Use the external tool and do all the configuration, including the following points:
   - Set the Cycle Time
   - Turn on the distributed clocks option for all slave drives in order to share a global system time through EtherCAT
   - Assign PDO to each drive (inputs and outputs)
   - Set the mode of operation of the drives into **position** mode
   - Insert variable names and do the mapping (see details below)
4. Use the external tool to export the ENI description file
5. In KAS, Import the ENI file describing all the EtherCAT devices included in your project

IMPORTANT Importing an external ENI file overrides all EtherCAT project device information and configuration settings in the IDE. The following views and configurations

IMPORTANT are *not* applicable when using an imported ENI file:

- Project View: All devices located under the EtherCAT node
- EtherCAT Device View tabs:
  - General Properties
  - PDO Selection/Mapping
  - Distributed Clock
  - CoE Init-Commands
- Slice I/O Properties
- Mapping PLC Variables to Slice I/O or PDO objects

Information displayed in the views may not match the imported ENI file.

9.3.8.1 How to modify the EtherCAT image in cyclic mode

In your application program, when integrating non-standard EtherCAT devices, use the following function blocks to update EtherCAT frame:

- ECATWriteData(Function)
- ECATReadData(Function)

9.3.8.2 How to configure EtherCAT device

You need to use the following Functions Blocks:

- ECATWriteSdo (Function Block)
- ECATReadSdo (Function Block)

9.3.8.3 How to map PLC variables

When you use an XML network description file generated with an external configurator, you need to add special tags to the PDO names to ensure the PLC variables can be mapped to IO channels. The tags must comply with the following convention:

```
@Scope.VariableName+StartBit-Size
```

Field	Description
@	prefix with character **@** the PLC variable names of each of the image attributes that must be mapped
Scope	Scope can be: • (Global) • (Retain) • ProgramName Note that even for the case of nested child SFC programs, the variables still belong to a unique well defined subprogram **IMPORTANT** Do not forget the parenthesis when the scope is Global or Retain
+StartBit	(Optional) Integer that defines the bit from which the data must be written or read from the PLC variable

Field	Description
-Size	(Optional) Integer that defines the number of consecutive bits in the image which must be copied to/from the PLC variable. **IMPORTANT** When present, this setting has precedence over the <BitSize> tag of the XML file

Examples:

```
(Global).MachineState
(Global).bLedStatus:0-1
(Global).bLedStatus:1-1
(Global).myINT:+4
main.variable:3+4-8
```

NOTE This convention is applicable for simple variables. KAS does not yet support mapping for **Structs and Arrays**.

9.3.9 EtherCAT Error Messages

This chapter covers the following error messages linked to the EtherCAT motion bus that are displayed in the Information and Logs window:

```
Abnormal response of slaves to cyclic commands. Please, check
number and state of slaves.
Link Error! Please, check IPC connection.
Slave <slave-name> is not responding. Please, check power
supply or connection.
Slave <slave-name> is not responding. Please, check power
supply or connection.
```

These messages can arise due to the following causes:

- Wrong/Missing Device
- Link loss/Device fault
- Frame loss
- Frame not processed
- Transmission Errors

9.3.9.1 Wrong/Missing Device

Case Description

The XML network configuration file contains the list of all EtherCAT devices present in the network.

At the EtherCAT initialization phase, the master checks that:

- Every physical device in the network corresponds to the configured devices (the master detects if the configuration does not match the physical devices)
- The configured **2nd address** matches the one in the physical device (this allows detection when two drives of the same kind have been swapped)

- The Standard I/O Couplers and I/O slices are correct by adding the proper commands in the network configuration file
  (this allows the detection of wrong or missing Standard I/O Coupler)

Results

An Error log is generated with the relevant information.

The EtherCAT startup is aborted, as well as the startup of the machine.

9.3.9.2 Link Loss/Device Fault

Case Description

This kind of error can appear anytime in the EtherCAT communication, typically when a cable is disconnected or cut or whenever an EtherCAT device is damaged.

The master has a mechanism that detects such situations.

Results

An Error log is generated with the relevant information.

The EtherCAT communication is aborted.

If the network is cut, the drives on the side of the network disconnected from the master are moved into an error state (**F29**). They are automatically stopped and powered off.

In addition, all still-reachable axes have to be stopped and powered off.

> **NOTE** It can be necessary to put the axes in a safe position before powering it off (this action is application dependent).

9.3.9.3 Frame Loss

Case Description

For security, all frames sent must be received in a given timeout period (at least before the next cycle is started).

The master detects this case by managing the appropriate timeout watchdogs.

Results

An Error log is generated with the relevant information.

The EtherCAT communication is aborted.

9.3.9.4 Frame Not Processed

Case Description

If a frame is not processed by a slave, a warning message is displayed. However, the network remains operational.

Results

A warning message is issued the first two times that working counters are not correct. After the third time, the system is stopped with an error message.

9.3.9.5 Transmission Errors

Case Description

Even if it is rather unlikely, there could be transmission errors at the Ethernet physical level. In this case the slaves are able to detect the error (based on the Ethernet

CRC of the Ethernet frame. The master detects this case by checking the CRC of the received frame.

Results

An Error log is generated with the relevant information.

The EtherCAT communication is aborted.

9.3.9.6 Other Messages Linked to EtherCAT

The following message is displayed when the IPC has an invalid Ethernet configuration:

```
Failed to open Ehternet NIC on the IPC. Verify that INtime
drivers are loaded.
```

See also "Communication and Fieldbus" on page 64

The following message is displayed if an error or inconsistency is discovered during the parsing of the XML file when the application is started:

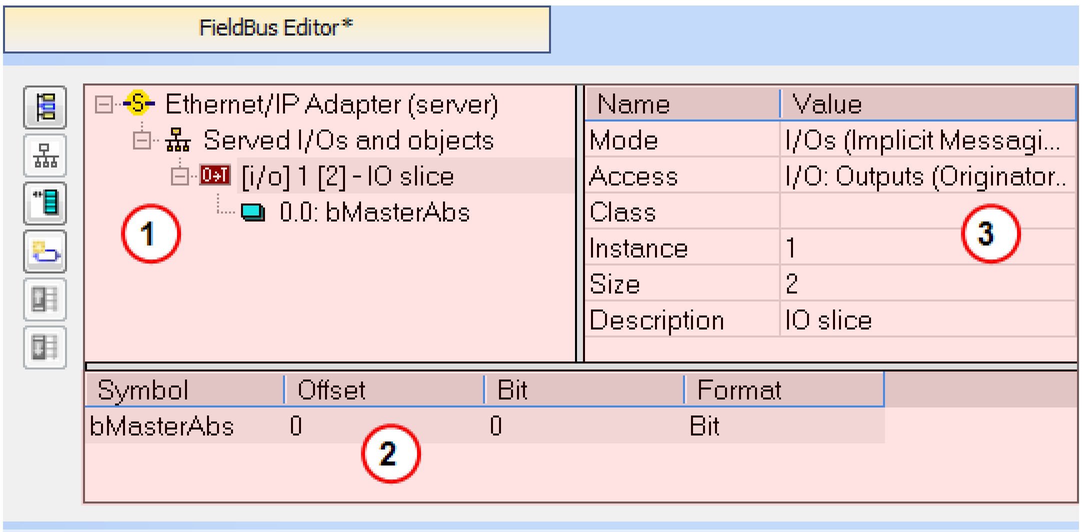
```
Unable to open EtherCAT config file <file-name>

<file-name>:<line>:<column>: <parsing error>
```

9.3.10 Fieldbus Editor

The KAS IDE includes an integrated Fieldbus Editor for various kinds of networked I/Os and protocols. This editor enables you to describe networks as configuration trees and to wire variables to the I/O channels of devices.

Icon	Description
	To open the Fieldbus Editor, double click on the icon or name in the Project Tree.

The Fieldbus Editor proposes the following workspace:



Call out#	Description
1	Fieldbus Configuration tree Each kind of fieldbus is shown as a top-level node in the Fieldbus Configuration tree. Click on the Insert Network icon ( ) to select a configuration to be added to the tree. Each configuration will be structured as a tree where the first level is the Fieldbus component type.
2	When an item is selected in the tree, all its children can be edited in the grid below
3	Selected node settings

Use the following icons in the toolbar for building the configuration tree:

Icon	Description
	Insert a new fieldbus component type (top level)
	Insert a new master/port node in the selected fieldbus
	Insert a new slave/data block node under the selected master
	Insert a new variable node under the selected slave
	Move up the selected slave device or data block
	Move down the selected slave device or data block

Table 10-3: Fieldbus Editor Toolbar - List of Icons

You can double-click an item in the tree to enter its properties in a dialog box. Use the View / Grid menu command to show or hide the grid area.

You can also drag a variable from the list of declared variables (on the right in the Dictionary) directly to a slave item in the configuration tree.

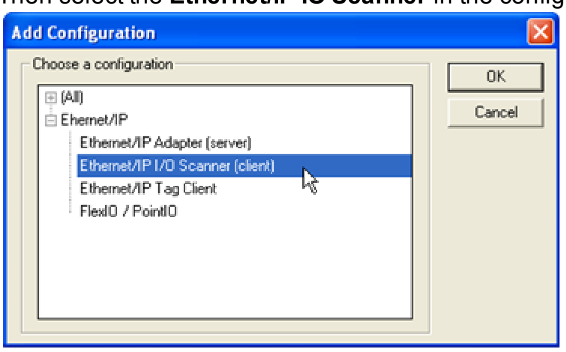
9.3.10.1 Ethernet/IP IO Client

The KAS Runtime includes a fully integrated Ethernet/IP client driver for exchanging CIP I/O assemblies as an Ethernet/IP scanner in your applications.

Data exchange - configuration

A dedicated configuration tool is integrated in the KAS IDE. Run it using the File / Open / Fieldbus Configuration menu command from the main window

1. Double-click the Fieldbus node in the project explorer to open it
2. Click the Insert Configuration icon to add the Fieldbus configuration
3. Then select the **Ethernet/IP IO Scanner** in the configuration selector



The configuration is represented as a tree:

- Ethernet/IP IO Scanner
  - Server (an Ethernet/IP adapter device) (*)
    - IO Assembly (Originator to Target)
      - Exchanged Variable (*)
    - IO Assembly (Target to Originator)
      - Exchanged Variable (*)

(*) The items with this mark can appear several times in the configuration.

Configuration

Click the Insert Master icon the Run the Edit / New master command to declare an server (slave device). Each server is identified by its IP address and an optional description text.

Then click the Insert Slaver icon run the Edit / Slave - Data Block command to declare a CIP I/O assembly. Each assembly is identified by:

Identifier	Meaning
Type	Direction of the I/O assembly. Can be one of: - Originator to Target (outputs) - Target to Originator (inputs)
Instance	Instance of the CIP assembly
Size	Data size in bytes
Connection type	Type of the CIP connection. Can be Point To Point or MultiCast
Priority	CIP priority: Low, High, Scheduled or Urgent
32 bit header	Check this option if a 32 bit header is to be sent on notifications
RPI(ms)	Minimum period for notification of changes, in milliseconds
Description	Optional description text

Then you can map IEC61131-3 variables on the data of the assembly, for each variable you must specify:

Identifier	Meaning
Symbol	The name of the IEC61131-3 variable
Offset	Offset in bytes in the assembly data
Bit	Bit offset in the selected byte if format is "Bit"
Format	Format of the data in the assembly
Mode	Kind of data exchanged through the variable: Data Exchange: a piece of input or output data in the assembly Server OK: indicates the status of the IP connection to the server I/O connection OK: indicates the status of the CIP I/O connection

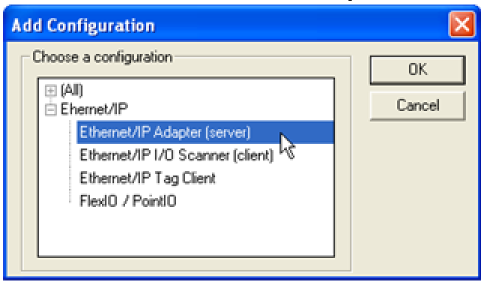
9.3.10.2 Ethernet/IP Server

The KAS Runtime includes fully integrated Ethernet/IP server driver for exchanging CIP I/O assemblies as an Ethernet/IP adapter in your applications.

Data exchange - configuration

A dedicated configuration tool is integrated in the KAS IDE. Run it using the File / Open / Fieldbus Configuration menu command from the main window

1. Double-click the Fieldbus node in the project explorer to open it
2. Click the Insert Configuration icon to add the Fieldbus configuration
3. Then select the **Ethernet/IP Adapter** in the configuration selector



The configuration is represented as a tree:

- Ethernet/IP IO Scanner
  - Served I/Os and objects
    - IO Assembly or Vendor Specific Object (*)
      - Exchanged Variable (*)

(*) The items with this mark can appear several times in the configuration.

Configuration

Click the Insert Master icon the Run the Edit / New master command to declare an server (slave device). Each server is identified by its IP address and an optional description text.

Select the **Served I/Os and objects**, then click the Insert Slave icon run the Edit / Slave - Data Block command to declare a CIP I/O assembly or a vendor specific object. Each assembly is identified by:

Identifier	Meaning
Mode	Kind of CIP object. Can be one of: • I/O assembly • Vendor specific object
Access	In case of a vendor specific object, this property defines the access rights: • Read/Write = free access • Read Only = the client (scanner) cannot write the object data
Class	CIP class in case of a vendor specific object. This field should be ignored in case of an I/O assembly.
Instance	Instance of the CIP assembly or object
Size	Data size in bytes
Description	Optional description text

When defining a vendor specific objects, the following attributes are available for scanners:

1 (get only) = size of the object data

3 (get/set) = object data

Then you can map IEC61131-3 variables on the data of the assembly, for each variable you must specify:

Identifier	Meaning
Symbol	The name of the IEC61131-3 variable
Offset	Offset in bytes in the assembly data
Bit	Bit offset in the selected byte if format is "Bit"
Format	Format of the data in the assembly

TIP You can drag a variable from the Dictionary directly to a slave item.

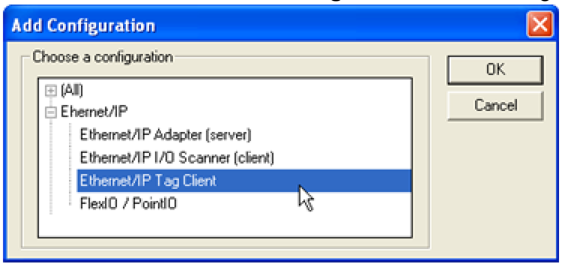
9.3.10.3 Ethernet/IP Tag Client

The KAS Runtime includes fully integrated Ethernet/IP client driver for exchanging tags with Ethernet/IP tag based devices such as PLCs.

Data exchange - configuration

A dedicated configuration tool is integrated in the KAS IDE. Run it using the File / Open / Fieldbus Configuration menu command from the main window

1. Double-click the Fieldbus node in the project explorer to open it
2. Click the Insert Configuration icon to add the Fieldbus configuration
3. Then select the **Ethernet/IP Tag Client** in the configuration selector



The configuration is represented as a tree:

- Ethernet/IP Tag Client
  - Server (an Ethernet/IP adapter device) (*)
    - Tag (generally an array) (*)
      - Exchanged variable (*)

(*) The items with this mark can appear several times in the configuration.

Driver and configurator are optimized for exchanging arrays (tags declared as arrays in the PLC). However it is also possible to exchange single tags.

Configuration

Click the Insert Master icon the Run the Edit / New master command to declare an server (slave device). Each server is identified by its IP address and an optional description text.

Then you need to configure tags such as declared in the PLC:

- The easiest way is to right-click on the server in the tree and select the **Add ARRAY Tag** command in the contextual menu. Then you enter the properties of the tag request and the symbol of the corresponding array to be used in your IEC61131-3 application. Configuration of the tag and mapping of all array items is performed automatically.
- Alternatively you can click the Insert Slaver icon run the Edit / Slave - Data Block command to declare the tag and map some variables later on.

A tag request is identified by:

Identifier	Meaning
Tag name	The name of the tag such as declared in the PLC
PLC Slot	PLC slot number
Mode	Read or Write (note that the same tag can be configured twice for both reading and writing)
Nb Elements	Number of array items to read or write
Offset	O-based index of the first item to read or write in the array
Tag data type	Data type of the tag such as declared in the PLC. Available Types are: ▪ BOOL (single boolean variable on 1 byte - 00=FALSE / FF=TRUE) ▪ SINT (8 bit signed integer) ▪ INT (16 bit signed integer) ▪ DINT (32 bit signed integer) ▪ DWORD (32 bit string) DWORD should be selected if the tag is declared in the PLC as an array of bits.
Period(ms)	You can specify in this parameter a period for continuously sending the request. Enter "0" for a request sent "on demand"
Timeout	Request timeout in milliseconds

IEC61131-3 variables are mapped on the data of the tag, for each variable you must specify:

Identifier	Meaning
Symbol	The name of the IEC61131-3 variable
Offset	Offset in bytes in the assembly data
Bit	Bit offset in the selected byte if format is "Bit"
Format	Format of the data in the assembly
Mode	Kind of data exchanged through the variable: Data Exchange: a piece of input or output data in the assembly Server OK: indicates the status of the IP connection to the server Send Request Now: will be used as a command for activating the request [transaction counter]: increased each time the request is sent [general status]: CIP error code (0 = OK) [extended status]: CIP extended error code (0 = OK)

The tag will be read or written:

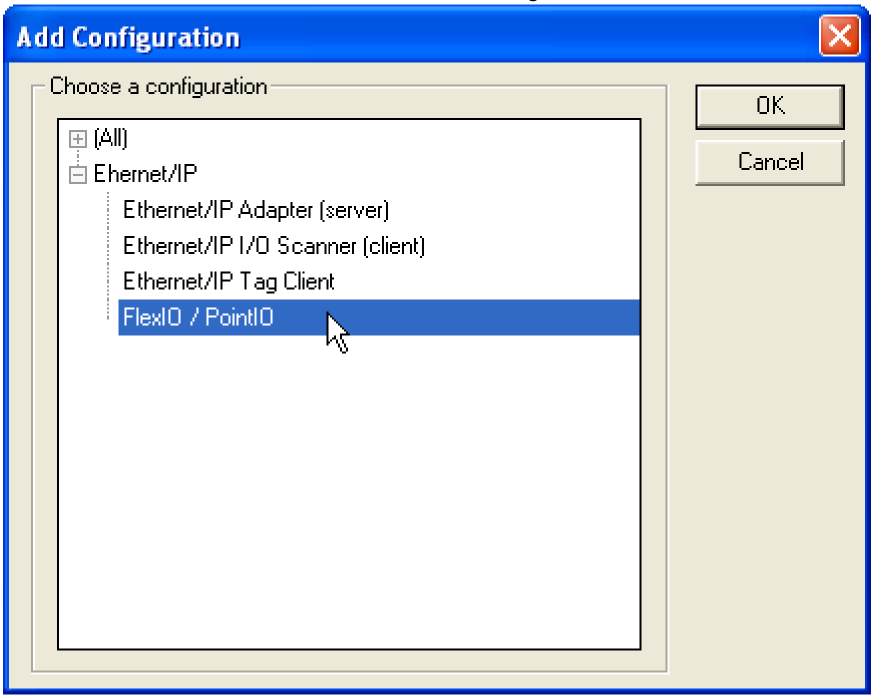
- periodically if a non zero period is specified in the tag configuration
- when a variable configured as "Send Request Now" becomes TRUE

In the case of a command variable, the variable is automatically reset to FALSE when the request is sent.

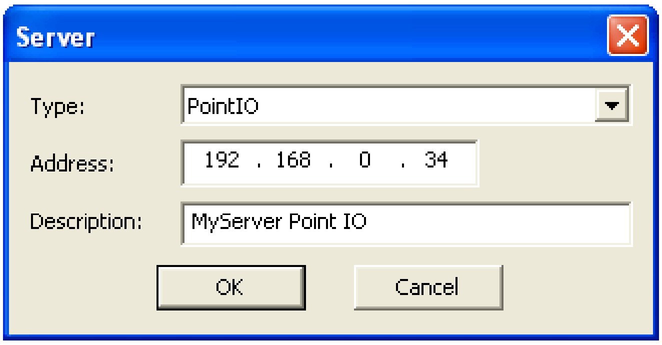
9.3.10.4 FlexIO / PointIO

Before establishing the connection to the POINT IO, these modules require configuration. This is done through the WEB interface of the POINT IO bus coupler.

1. Double-click the Fieldbus node in the project explorer to open it
2. Click the Insert Configuration icon to add the Fieldbus configuration
3. Then select the **FlexIO/PointIO** driver in the configuration selector



4. Click the Insert Master icon and select PointIO



5. Click the Insert Slave icon
   Only modules in the list are supported. When inserting, the module variables can be declared automatically by checking Declare variables and set a prefix.

NOTE Modules need to be inserted in the right order.

Configuration is ready and you can download the application to the KAS Runtime.

9.3.10.5 Modbus Slave

The KAS Runtime includes fully integrated slave functions for enabling Modbus communication on a serial link or Ethernet.

Protocol specification

The protocol supported is Open Modbus on Ethernet. The following Modbus function codes are supported:

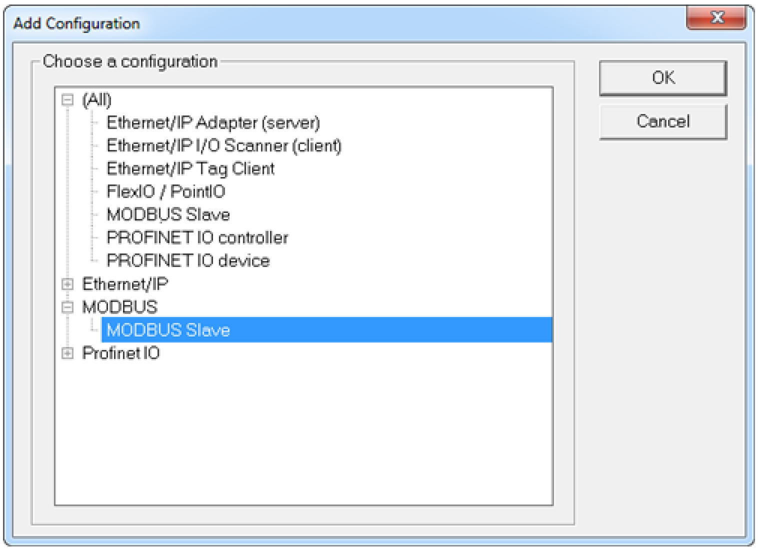
1	read coils
2	read bit inputs
3	read holding registers
4	read input registers
5	write 1 coil
6	write 1 register
15	write *n* coils
16	write *n* registers

As a default, the first valid address for each kind of data is 1. If you use Modbus devices with other addressing conventions, you can change the base offset for each kind of data using the **Tools** > **Addresses** menu command.

Data exchange - configuration

A dedicated configuration tool is integrated in the KAS IDE. To run it,

1. Double-click the **Fieldbus** node in the project explorer to open it
2. Click the Insert Configuration icon to add the Fieldbus configuration
3. Then select the **Modbus Slave** in the configuration selector



The Modbus Slave configuration is represented as a tree:

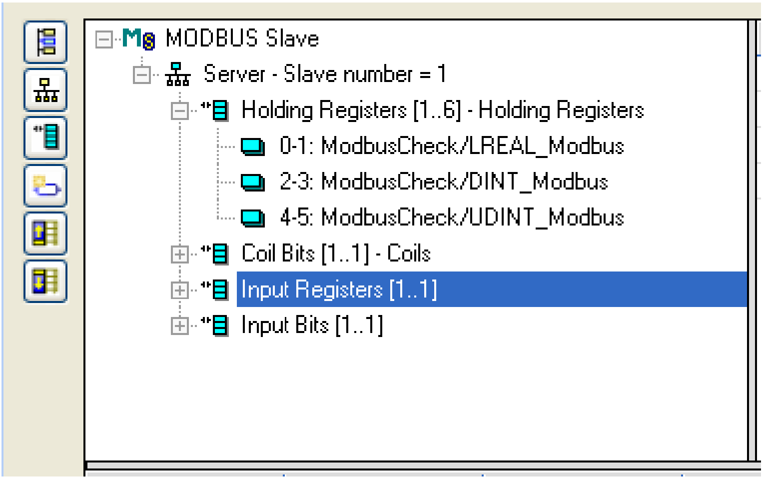
- Modbus Slave
  - Slave number (variables that can be accessed from external Modbus masters)
    - Input bits data block (read by masters)
      - Variable (*)
    - Input words data block (read by masters)
      - Variable (*)
    - Coil bits data block (forced by masters)
      - Variable (*)

- Holding bits data block (forced by masters)
    - Variable (*)

(*) The items with this mark can appear several times in the configuration.

Modbus Slave configuration

You need to configure the Modbus Slave in order to make variables visible from external Modbus masters such as SCADA systems. Below is a simple example of slave configuration:



Double click on the **Server** item to setup the Modbus slave number that will identify the runtime application. When the local server is selected, use the Insert Slave/Data Block menu command to insert Modbus data blocks. The following kinds of block are available:

- Input Bits: bits read by external masters (function 2).
- Coil Bits: bits forced by by external masters (function 5 or 15).
- Input Registers: words read by external masters (function 4).
- Holding Registers: words forced by external masters (function 6 or 16).

Each data block is identified by a Modbus base address and a number of items (bits or words).

NOTE Read and write requests sent by Modbus masters will be denied if the range specified in the request does not fit within a data block defined in the configuration. Requests overlapping two data blocks will be denied.

For example, if you configure a block of 16 words starting at address 1 and another block of 16 words starting at address 17, a request for read or write of 32 words starting at address 1 will be denied and an "address error" exception will be reported.

When a server data block is selected, use the ‘New symbol’ command to map a variable to an item of the data block. Each variable is identified by a valid symbol of a variable in the open project and an offset in the data block according to Modbus addressing.

- For exchanging Boolean variables through Modbus words, a hexadecimal mask is available in order to define to which bit of a word a variable is attached. For example, enter the mask "0001" to map a Boolean variable to the less significant bit of a word.
- For exchanging 32 bit variables (DINT, REAL...), you can select to map the variable on two consecutive words.

You can sort the variables of each data block according to their offset using the ‘Sort symbols’ menu command at any time.

Data types

You can freely map a variable of any data type to a Modbus item. The Runtime automatically converts the value to the type of the variable.

- For exchanging Boolean variables through Modbus words, a hexadecimal mask is available in order to define to which bit of a word a variable is attached. For example, enter the mask "0001" to map a Boolean variable to the less significant bit of a word.
- For exchanging 32 bit variables (DINT, REAL...), you can select to map the variable on two consecutive words.

NOTE
- 64 bit variable (LINT and LREAL) cannot be extracted directly without lost of accuracy or data.

9.3.10.6 Profinet IO RT Controller Configuration

The KAS IDE contains a fully integrated configurator for Profinet IO RT Controller.

IMPORTANT Referring to the Profinet standard, the units of a Profinet network are named as IO Controllers (Masters) and IO Devices (Slaves).

TIP To use some of the additional features of the Profinet controller field bus editor, such as “Browse Network for Slaves”, you need to install the "CD Prot" driver. The installer can be found in `<Kollmorgen Automation Suite installation directory>\Bin\CDProtDriver`.

Please note that your computer will need to be rebooted after installing the driver.

NOTE Profinet is only supported on AKD PDMMs.

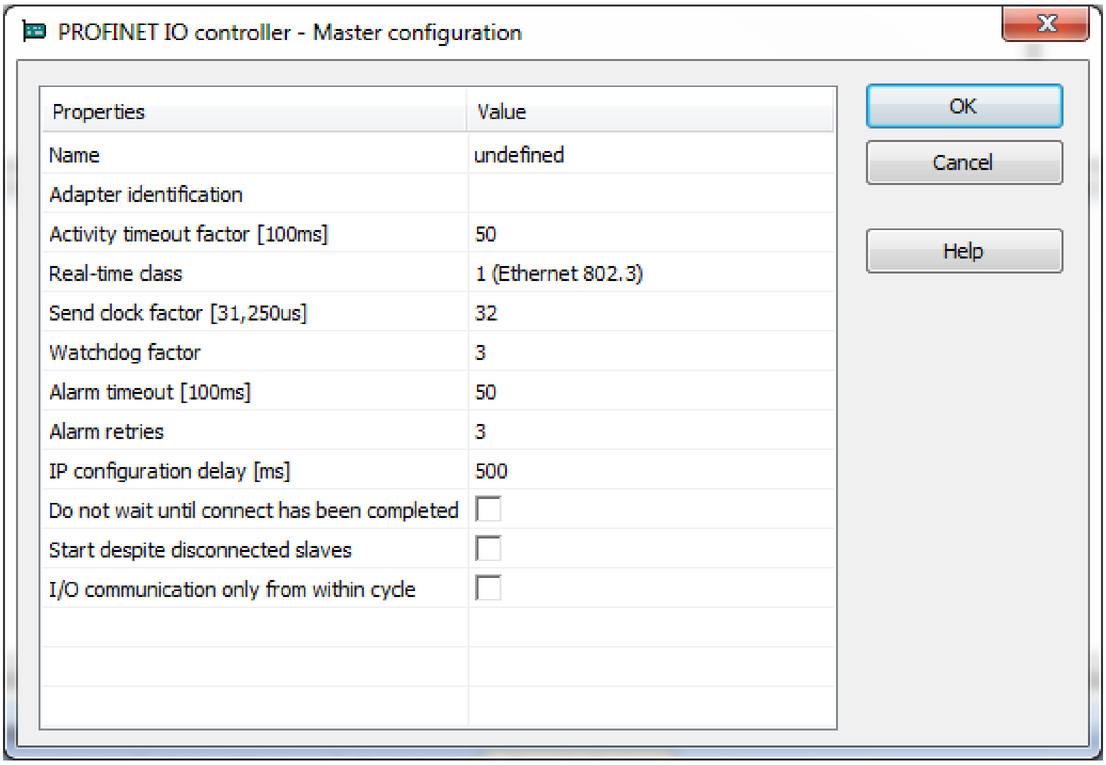
NOTE Unlike an EtherCAT fieldbus which has a dedicated Ethernet interface, the Profinet fieldbus communicates through the same Ethernet interface as Modbus and other non-fieldbus communication. *There is no prioritization of Profinet over the other communication protocols*, so frequent HTTP or IDE Oscilloscope communication can cause delay in the Profinet communications, and ultimately Profinet frame loss.

Configuration

The I/Os of the Profinet network must be connected to the variables via a Profinet IO controller. Start the declaration of a Profinet controller with:

1. Insert > Insert Master/Port.

   The following window opens.

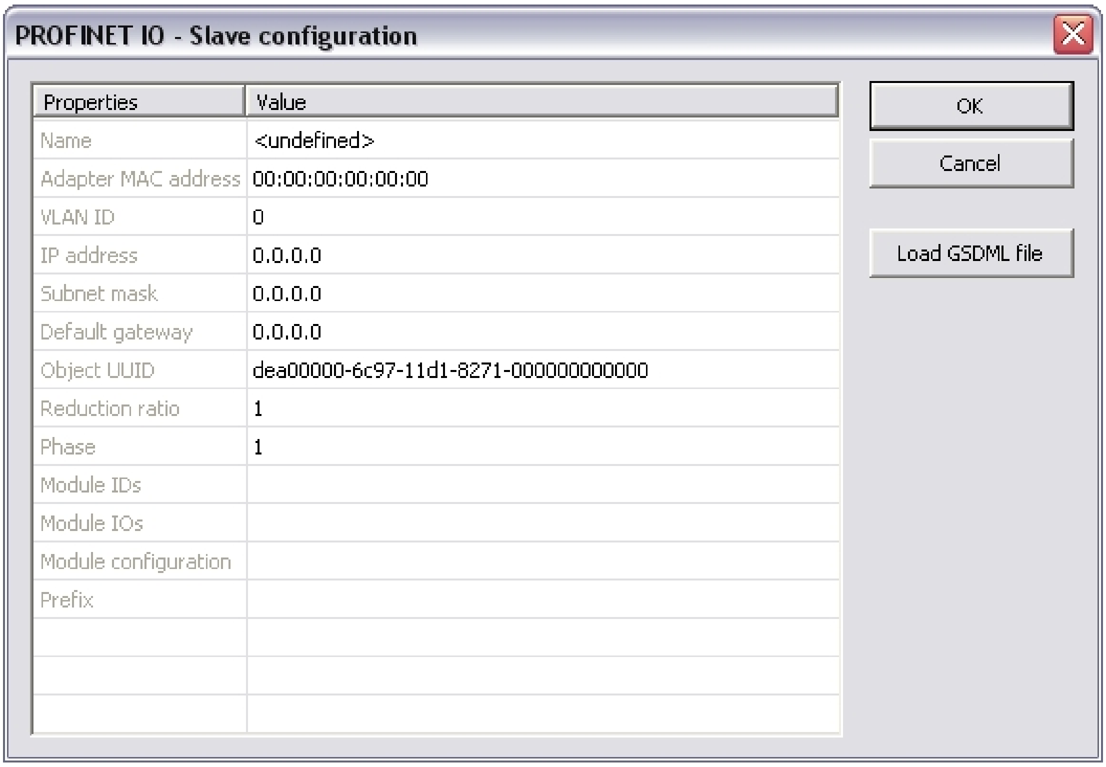


These parameter can be changed:

Parameter	Meaning
Name	• A device name can consists of labels and must follow these conventions: • 1 or more labels, separated by [.] • Total length is 1 to 240 • Label length is 1 to 63 • Labels consist of [a...z0...9-] • Labels do not start with [-] • Labels do not end with [-] • The first label does not start with "port-xyz" or "port-xyz-abcde" with a,b,c,d,e, x, y, z = 0...9 • Device names do not have the form n.n.n.n, n = 0...999 • Labels do only start with 'xn-' if RFC 3490 is applied
Adapter identification	MAC address of the Profinet IO controller or name of the network connection (Windows XP or younger and Windows CE 4.1 or younger). This must be set to `tsec0`.
Activity timeout factor [100ms]	Timeout for the connection establishment to the devices (maximum time between beginning of connection establishment and the first cyclic data exchange); Timeout factor based on 100ms.
Real-time class	Class 1: cyclic data exchange without priority tag. Class 2: data exchange with priority tag.

Parameter	Meaning
Send clock factor [31,250µs]	Send clock factor. Multiple of 31,250µs (32 = 1ms).
Watchdog factor	Watchdog factor (default 3): The Watchdog factor defines how many frames may be missing until the device is set back. Watchdog triggers: a) frames may be lost; e.g. due to bad cabling. b) frames may arrive delayed due to blocking situations in the network; e.g. due to a non-separated network. ⓘ **TIP** The watchdog may be triggered as soon as HTTP file transfer occurs between the PDMM and the IDE when the default value is used. Increase this value if you encounter frequent watchdog triggers.
Alarm timeout [100ms]	Alarm timeout (default 50).
Alarm retries	Number of alarm retries (default 3).
IP configuration delay [ms]	IP configuration delay (default 500): Defines the time to wait whether some devices are not ready after start.
Do not wait until connect hat been competed	`On`: the application starts immediately. `Off`: the application starts after all devices are connected.
Start despite disconnected slaves	`On`: the application starts despite configured but not found devices. `Off`: the application starts if all configured devices were found only.
I/O com-munication only from within cycle	`On`: Run I/O communication from within VM-cycle `Off`: Run I/O communication outside VM-cycle.

2. Mark the controller and click Insert > Insert Slave/Datablock to add a Profinet IO device.



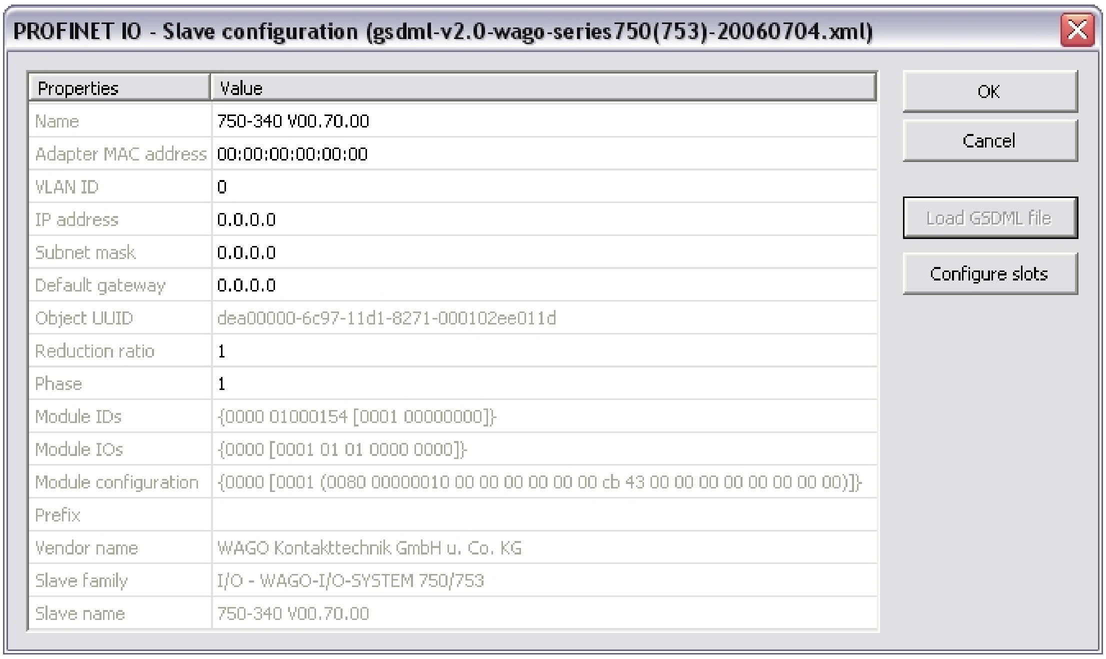
These parameter can be changed:

Parameter	Meaning
Name	• A device name can consists of labels and must follow these conventions: • 1 or more labels, separated by [.] • Total length is 1 to 240 • Label length is 1 to 63 • Labels consist of [a...z0...9-] • Labels do not start with [-] • Labels do not end with [-] • The first label does not start with "port-xyz" or "port-xyz-abcde" with a,b,c,d,e, x, y, z = 0...9 • Device names do not have the form n.n.n.n, n = 0...999 • Labels do only start with 'xn-' if RFC 3490 is applied
Adapter MAC address	MAC address of the Profinet IO device
VLAN ID	Virtual LAN ID
IP address	IP address of the device
Subnet mask	Subnet mask for the IP address of the device
Default gateway	Default gateway

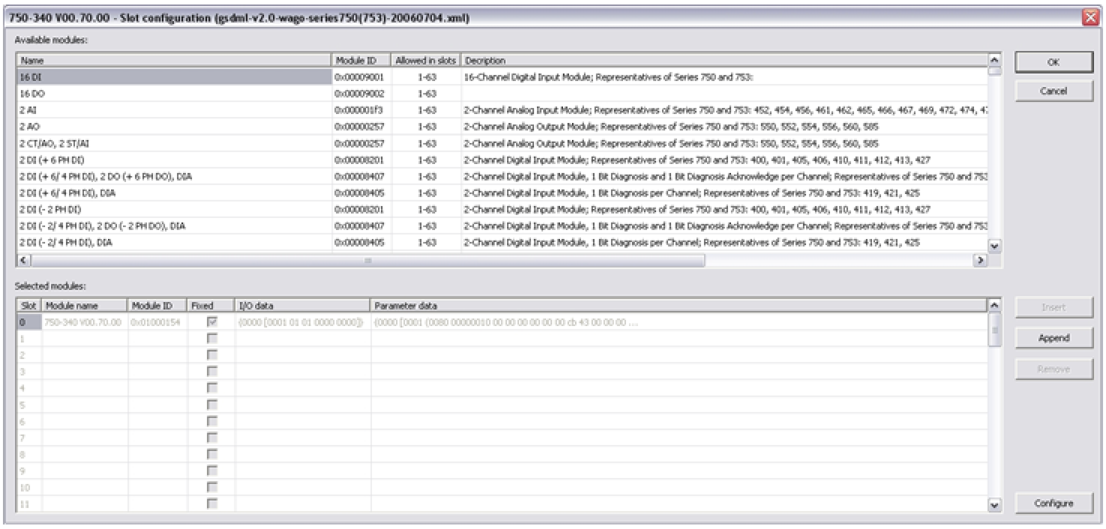
Parameter	Meaning
Object UUID	UUID of the device
Reduction ratio	Reduction ratio (default 1): The Reduction ratio defines the frequency for data to be exchanged with the device. The transfer rate is calculated by <Send clock factor> * 31,250 µs * <Reduction ratio>. The Send clock factor is a master parameter with a default value of 32 (all known devices work with this Send clock factor). With the default setting of 1 the data transfer rate is 1ms; for 2 it is 2ms...; Most devices support Reduction ratio settings of 1,2,4,8,16,32...
Phase	Phase
Module IDs	Module IDs of the device modules.
Module IOs	Module IOs of the device modules.
Module configuration	Module configuration.
Prefix	Prefix for the variables.

NOTE With the calculation of <Watchdog factor> * <Reduction ratio> * <Send clock factor> * 31,250 µs you get the time that may expire between two frames until the device is set back. I.e. for the default settings 3 * 1 * 32 * 31,250 µs the connection will be set back after 3ms of missing frames. For office networks this time is rather low. It is recommended to us a Reduction ratio of 32 and a Watchdog factor of 24 for office networks. I.e. the connection will be reset after 24 * 32 * 32 * 31,250 µs = 768 ms. If this data exchange rate is to low please separate the PROFINET IO network from the office network (e.g. by a router).

3. Click `Load GSDML file` to import the necessary GSDML file.



4. Do the slot configuration after the GSDML file import.



5. Select the modules in the upper list. With the buttons `Insert` and `Append` the modules are copied to the lower list.
   You can not configure each module. Only modules with some sub modules respectively with a sub module with parameter data can be configured. Mark the according module in the lower list and click the `Configure` button.

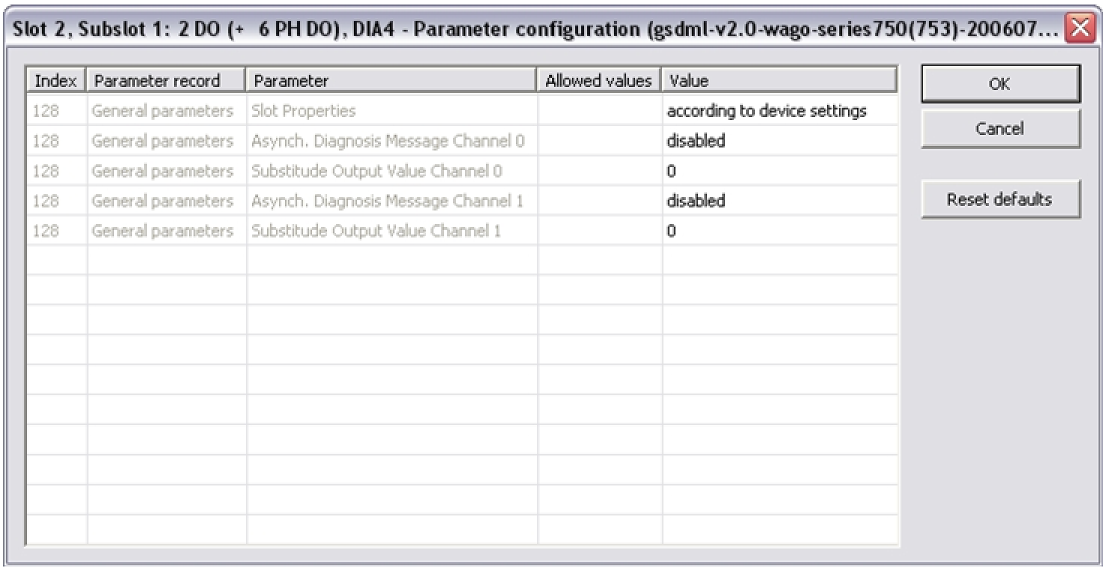
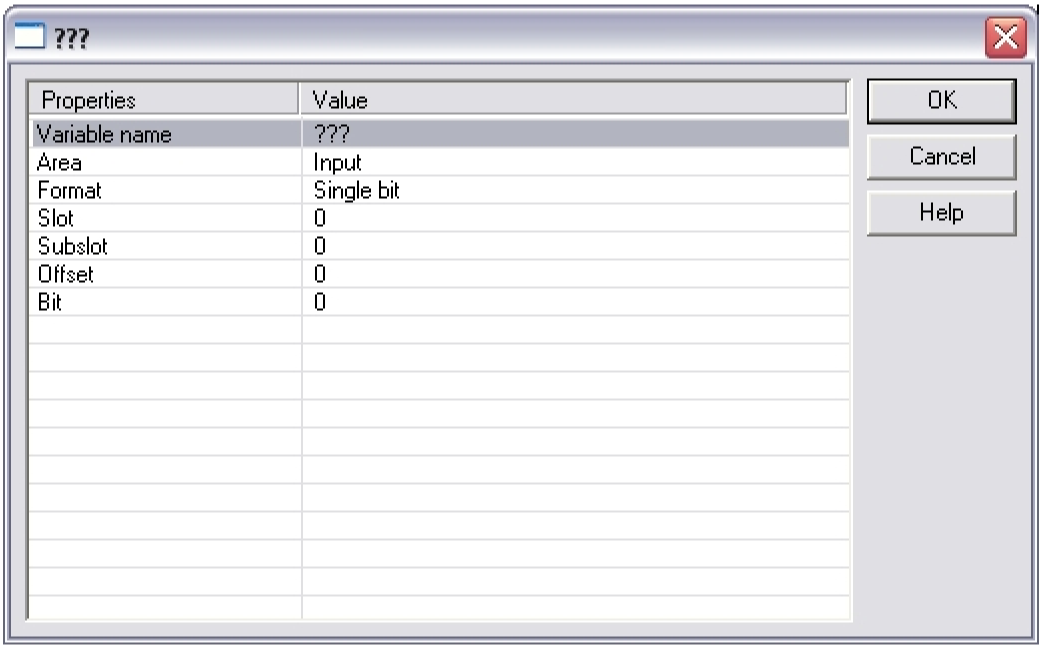


Figure 10-29: Example of configuring sub-modules.

6. Now you can connect the variables with the I/Os. Use Insert > Insert/Set Variable in order to append a variable to a device.

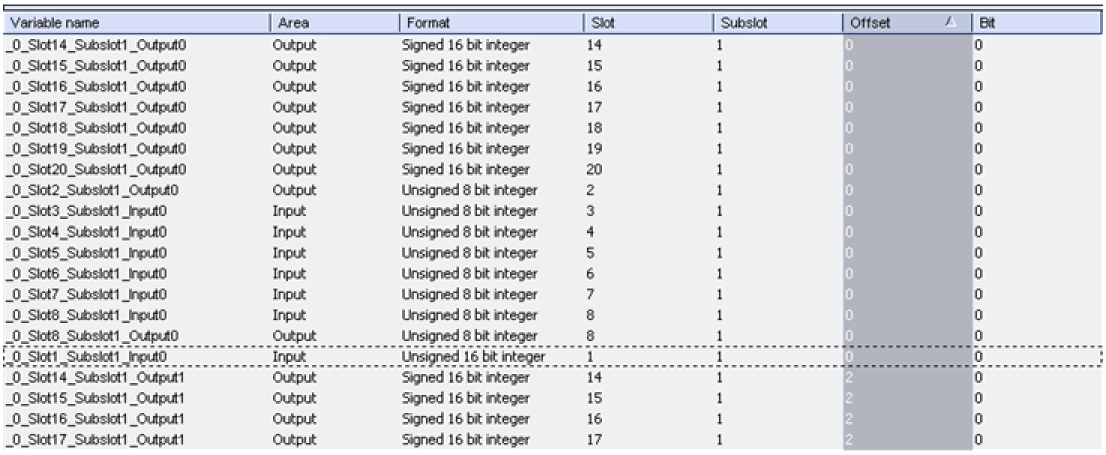


These parameter can be changed:

Parameter	Description
Variable name	Variable name following the IEC 61131-3 syntax.
Area	Output, Output IOCS, Output IOPS, Input, Input IOCS, Input IOPS, device status.
Format	32 bit float, Signed 16 bit integer, Signed 32 bit integer, Signed 8 bit integer, Single bit, Unsigned 16 bit integer, Unsigned 32 bit integer, Unsigned 8 bit integer.
Slot	Slot Number
Subslot	Subslot Number
Offset	Offset
Bit	Bit

NOTE The offset of a variable is relative to a sub module. Thus also depending from a slot and subslot. The offset of the first variable of a sub module is always 0.

7. All settings can be changed in the grid too. The information show refers to the items below of the selected item in the configuration tree.



Variable name	Area	Format	Slot	Subslot	Offset	Bit
_0_Slot14_Subslot1_Output0	Output	Signed 16 bit integer	14	1	0	0
_0_Slot15_Subslot1_Output0	Output	Signed 16 bit integer	15	1	0	0
_0_Slot16_Subslot1_Output0	Output	Signed 16 bit integer	16	1	0	0
_0_Slot17_Subslot1_Output0	Output	Signed 16 bit integer	17	1	0	0
_0_Slot18_Subslot1_Output0	Output	Signed 16 bit integer	18	1	0	0
_0_Slot19_Subslot1_Output0	Output	Signed 16 bit integer	19	1	0	0
_0_Slot20_Subslot1_Output0	Output	Signed 16 bit integer	20	1	0	0
_0_Slot2_Subslot1_Output0	Output	Unsigned 8 bit integer	2	1	0	0
_0_Slot3_Subslot1_Input0	Input	Unsigned 8 bit integer	3	1	0	0
_0_Slot4_Subslot1_Input0	Input	Unsigned 8 bit integer	4	1	0	0
_0_Slot5_Subslot1_Input0	Input	Unsigned 8 bit integer	5	1	0	0
_0_Slot6_Subslot1_Input0	Input	Unsigned 8 bit integer	6	1	0	0
_0_Slot7_Subslot1_Input0	Input	Unsigned 8 bit integer	7	1	0	0
_0_Slot8_Subslot1_Input0	Input	Unsigned 8 bit integer	8	1	0	0
_0_Slot8_Subslot1_Output0	Output	Unsigned 8 bit integer	8	1	0	0
_0_Slot1_Subslot1_Input0	Input	Unsigned 16 bit integer	1	1	0	0
_0_Slot14_Subslot1_Output1	Output	Signed 16 bit integer	14	1	2	0
_0_Slot15_Subslot1_Output1	Output	Signed 16 bit integer	15	1	2	0
_0_Slot16_Subslot1_Output1	Output	Signed 16 bit integer	16	1	2	0
_0_Slot17_Subslot1_Output1	Output	Signed 16 bit integer	17	1	2	0

If the GUI Views is online with a target system the grid shows the real-time data of the variables.

Data types

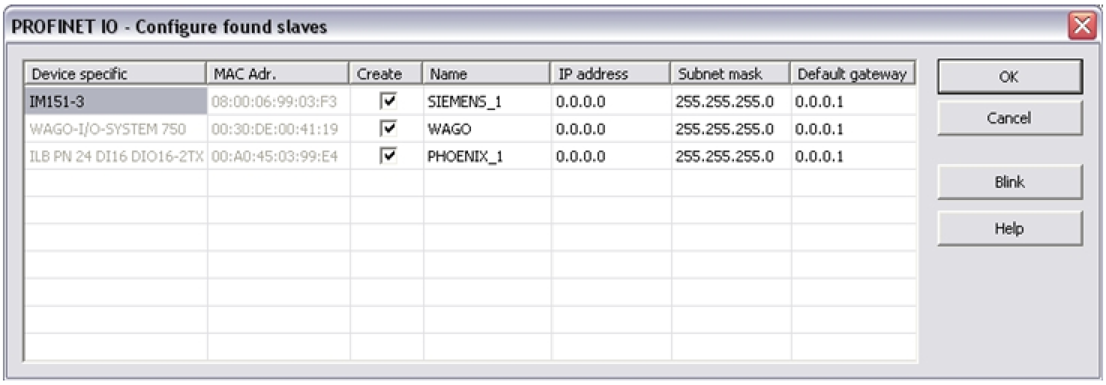
You can connect variables of any data type to the Profinet I/Os. The Runtime converts the values of the I/Os to the type of the variable. STRING variables are not supported.

Additional features

> **TIP**
>
> To use some of the additional features of the Profinet controller field bus editor, such as “Browse Network for Slaves”, you need to install the "CD Prot" driver. The installer can be found in `<Kollmorgen Automation Suite installation directory>\Bin\CDProtDriver`.
>
> Please note that your computer will need to be rebooted after installing the driver.

Browse network for slaves

This command is available in the context menu of the network adapter. All connected Profinet IO devices are listed up.



Device specific	MAC Adr.	Create	Name	IP address	Subnet mask	Default gateway
IM151-3	08:00:06:99:03:F3	☑	SIEMENS_1	0.0.0.0	255.255.255.0	0.0.0.1
WAGO-I/O-SYSTEM 750	00:30:DE:00:41:19	☑	WAGO	0.0.0.0	255.255.255.0	0.0.0.1
ILB PN 24 DI16 DIO16-2TX	00:A0:45:03:99:E4	☑	PHOENIX_1	0.0.0.0	255.255.255.0	0.0.0.1

Select the devices you want to add in the tree via the checkbox in the column `Create`.

The Network Link (or another) -LED of the marked device blinks for three seconds when pressing the `Blink` button.

Set the device names (column `Name`) here. A device name can consist of labels and must follow these conventions:

- 1 or more labels, separated by [.].
- Total length is 1 to 240.

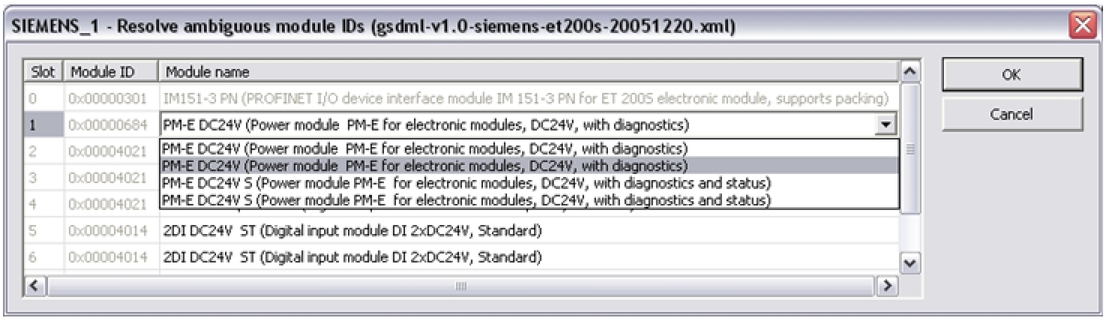
- Label length is 1 to 63.
- Labels consist of [a...z0...9-].
- Labels do not start with [-].
- Labels do not end with [-].
- The first label does not start with "port-xyz" or "port-xyz-abcde" with a,b,c,d,e, x, y, z = 0...9.
- Device names do not have the form n.n.n.n, n = 0...999.
- Labels do only start with 'xn-' if RFC 3490 is applied.

Set the IP address of the device here. By clicking into the grid the Default gateway will be set automatically.

Click `OK` after setting the IP-parameters (address, subnet mask, default gateway).

Configuration of devices

After adding the Profinet devices to the controller they need to be configured. Double click on the referring device. Load the GSDML file. If ambiguous module IDs are found they are shown in a window. Choose the right module in the combo-box.



NOTE Automatic creation of variables can be done primal after this step.

Set slave station name

With this context menu command it is possible to rename the slave names.

Read module configuration

With this context menu command it is possible to read out the module configuration again.

Create variables

Find this command in the context menu of the device. Based on the defined device modules the referring variables are generated.

Within the same dialog it is possible to define the variables for

- Device diagnosis
- IOxS

Device diagnosis

Based on the Profinet standard the referring variables can be generated:

- CycleCounter [UINT].
- Status [BOOL].
- DataValid [BOOL].
- ProviderState [BOOL].
- StationsProblemIndikator [BOOL].

Create IOxS for slave modules

Based on the defined device modules the referring IOPS- and IOCS-variables are generated.

How to Resolve Errors

Device is not found

- Check if device is switched on
- Check the network connection
- Ensure the correct name was set on the device

Error setting the IP configuration:

- Ensure that the IP configuration is valid and appropriate for your network
- Ensure the VLAN ID is setup correctly in the device settings

Timeout error

- Ensure that the IP configuration is valid and appropriate for your network
- Increase the IP configuration delay

Other errors

- Ensure you use the correct device name
- Ensure you have chosen the correct GSDML file (also check if the GSDML file version is appropriate for the firmware version of the device)

Timeout error

- Ensure that the IP configuration is valid and appropriate for your network
- Increase the IP configuration delay

Connect response error

Depending on the PNIO status error check, if the settings listed in the table below are set appropriate for your device.

PNIO status	Check the specified setting
1C010003	Slave\ObjectUUID (correct GDML file)
DB81010A	Master\Activity timeout factor
DB81010B	Master\Controller name
DB81010C	Master\Controller name
DB810207	Master\RT-Class
DB81020A	Master\Send clock factor
DB81020B	Slave\Reduction ratio
DB81020C	Slave\Phase
DB81020F	Master\Watchdog factor
DB810210	Master\Watchdog factor
DB8103,*	Slave\Module IDs,IOs
DB810407	Master\Alarm timeout
DB810408	Master\Alarm retries

Table 10-4: PNIO status error codes on connect and the related settings in the configuration

Module configuration is different

- Ensure you have configured the modules present on the device (If you have read the module configuration from the device, ensure you have selected the correct modules in the resolve ambiguous modules dialog displayed after selecting the GSDML file).

- Ensure you have chosen the correct GSDML file (also check if the GSDML file version is appropriate for the firmware version of the device).

Writing parameterization error (with status 0xDF80*)

- Ensure you have configured the modules present on the device (If you have read the module configuration from the device, ensure you have selected the correct modules in the resolve ambiguous modules dialog displayed after selecting the GSDML file).
- Ensure you have chosen the correct GSDML file (also check if the GSDML file version is appropriate for the firmware version of the device).

If your error could not be resolved using the hints above, update the firmware of your device and the GSDML file to the latest version and try again.

For further analyses record the PNIO communication using Wireshark (http://www.wireshark.org/). Start Wireshark and select Capture\Options from the menu. Choose the network interface used for the PNIO communication and enter the Capture filter below:

```
ether proto 0x8892 or (ether proto 0x8100 and ether[16:2]=0x8892)
or udp port 0x8894
```

Then click start. Start the straton RT with you application several times and then stop the Wireshark capturing by selecting Capture\Stop from the menu. You can save your recorded traffic by selecting File\Save from the menu.

Coding of PNIO status

The PNIO status is a 32 bit unsigned value. It is composed of 4 byte values, which define the meaning of the status. For positive responses PNIO status has a value of zero.

Bit	Meaning
24-31	ErrorCode
16-23	ErrorDecode
8-15	ErrorCode1
0-7	ErrorCode2

Table 10-5: Coding of PNIO status for negative responses

ErrorCode	Meaning
DB	Error in connect response
DC	Error in release response
DD	Error in control response
DF	Error in write response

Table 10-6: Meaning of ErrorCode for negative responses

ErrorDecode	Meaning
80	Read/Write service
81	Connect, Control, Release service

Table 10-7: Meaning of ErrorDecode for negative responses

ErrorCode1	Meaning
A1	write error

ErrorCode1	Meaning
A2	module failure
A3-A6	reserved
A7	busy
A8	version conflict
A9	feature not supported
AA-AF	device specific
B0	invalid index
B1	write length error
B2	invalid slot/subslot
B3	type conflict
B4	invalid area/API
B5	state conflict
B6	access denied
B7	invalid range
B8	invalid parameter
B9	invalid type
BA	backup
BB-BF	device specific
C0	read constrain conflict
C1	write constrain conflict
C2	resource busy
C3	resource unavailable
C4-C7	reserved
C8-CF	device specific

Table 10-8: Meaning of ErrorCode1 for ErrorDecode = 80

ErrorCode1	Meaning
01	Connect Parameter Error, Faulty ARBlockReq
02	Connect Parameter Error, Faulty IOCRBlockReq
03	Connect Parameter Error, Faulty ExpectedSubmoduleBlockReq
04	Connect Parameter Error, Faulty AlarmCRBlockReq
05	Connect Parameter Error, Faulty PrmServerBlockReq
06	Connect Parameter Error, Faulty MCRBlockReq
07	Connect Parameter Error, Faulty ARRPCBlockReq
08	Read Write Record Parameter, Error Faulty Record
14	IODControl Parameter Error, Faulty ControlBlockConnect
15	IODControl Parameter Error, Faulty ControlBlockPlug
16	IOXControl Parameter Error , Faulty ControlBlock after a connection establishment
17	IOXControl Parameter Error, Faulty ControlBlock after a plug alarm

ErrorCode1	Meaning
28	Release Parameter Error, Faulty ReleaseBlock
40	RMPM (Device state machines, device resources)

Table 10-9: Meaning of ErrorCode1 for ErrorDecode = 81

ErrorCode1	Meaning
00	ArgsLength invalid
01	Unknown Blocks
02	IOCR Missing
03	Wrong AlarmCRBlock count
04	Out of AR Resources
05	AR UUID unknown
06	State conflict
07	Out of Provider, Consumer, or Alarm Resources
08	Out of Memory
09-FF	Reserved

Table 10-10: Meaning of ErrorCode2 for ErrorCode1 = 40

For ErrorCode1 <> 40 ErrorCode2 refers to a field in the block specified by ErrorCode1.

ErrorCode2	Meaning
00	Block type
01	Block length
02	Block version (high byte)
03	Block version (low byte)

Table 10-11: Meaning of ErrorCode2 for ErrorCode1 <> 40

ErrorCode2	Meaning
04	AR Type
05	AR UUID
06	Session key
07	Initiator MAC address
08	Initiator Object UUID
09	AR Properties
0A	Activity timeout factor
0B	UDP RT port
0C	Station name length
0D	Station name

Table 10-12: Meaning of ErrorCode2 for ErrorCode1 = 01 (AR block request)

ErrorCode2	Meaning
04	IOCR Type

ErrorCode2	Meaning
05	Reference
06	LT
07	IOCR properties
08	IO data length
09	Frame ID
0A	Send clock factor
0B	Reduction ratio
0C	Phase
0D	Sequence
0E	Frame send offset
0F	Watchdog factor
10	Data hold factor
11	Tag header
12	IOCR multicast MAC address
13	Number of APIs
14	API
15	Number of IO data objects
16	Slot
17	Subslot
18	IO data object offset
19	Number of IOCS
20	Slot
21	Subslot
22	IOCS offset

Table 10-13: Meaning of ErrorCode2 for ErrorCode1 = 02 (IOCR block request)

ErrorCode2	Meaning
04	Number of APIs
05	API
06	Slot
07	Module ident number
08	Module properties
09	Number of submodules
0A	Subslot
0B	Submodule ident number
0C	Submodule properties
0D	Data description
0E	Data length
0F	IOPS length
10	IOCS length

Table 10-14: Meaning of ErrorCode2 for ErrorCode1 = 03 (Expected submodule block request)

ErrorCode2	Meaning
04	Type
05	LT
06	AlarmCR Properties
07	RTA timeout factor
08	RTA retries
09	Local alarm reference
0A	Maximum alarm data length
0B	Alarm CRT Tag high
0C	Alarm CRT Tag low

Table 10-15: Meaning of ErrorCode2 for ErrorCode1 = 04 (AlarmCR block request)

ErrorCode2	Meaning
04	Sequence number
05	AR UUID
06	API
07	Slot number
08	Subslot number
09	Padding
0A	Index
0B	Data length
0C	Target AR UUID

Table 10-16: Meaning of ErrorCode2 for ErrorCode1 = 8 (Read/write record block request)

ErrorCode2	Meaning
05	Padding
06	Session key
07	Padding
08	Control block command
09	Control block properties

Table 10-17: Meaning of ErrorCode2 for ErrorCode1 = 16 (IOXControl block request)

9.3.10.7 Profinet IO RT Device Configuration

The KAS IDE contains a fully integrated configurator for Profinet IO RT Device.

IMPORTANT Referring to the Profinet standard, the units of a Profinet network are named as IO Controllers (Masters) and IO Devices (Slaves).

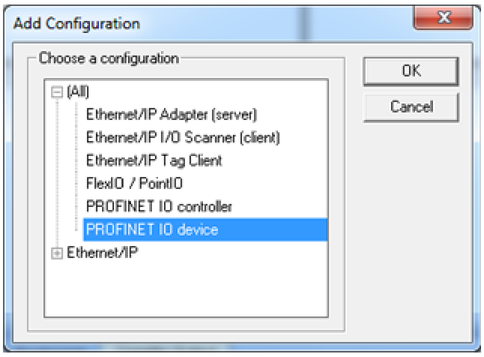
NOTE Profinet is only supported on AKD PDMMs.

ProfinetIO RT Device configuration

The Runtime manages a mapping table which contains the Profinet IO Inputs and Outputs. An appropriate configuration tool is integrated in the KAS.

To start the configuration:

1. Open the fieldbus configuration window.
2. Right click on the windows and choose **Insert > Insert Network**. The following window opens.

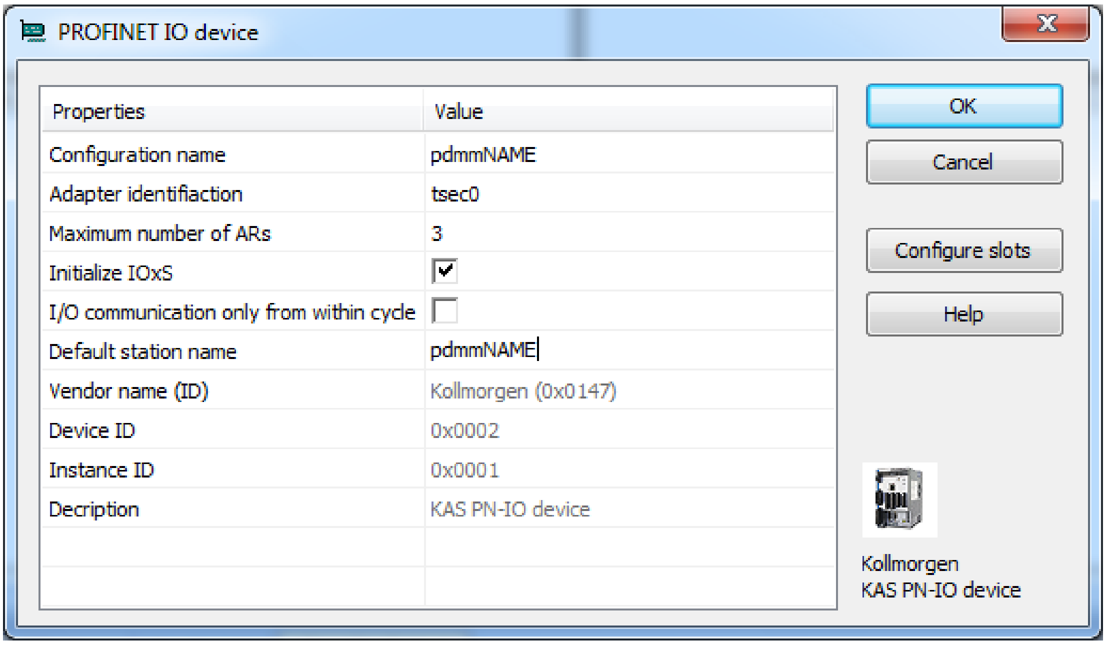


3. Select **PROFINET IO device** and click **OK**.
   The configuration is represented as a tree:
   - Profinet IO Configuration
     - Profinet IO device (*)
       - Group (*)
         - Variable (*)

   (*) These items can appear several times in the configuration (depending on the bus topology).

   The I/Os of the Profinet network must be connected to the variables via a Profinet IO device.

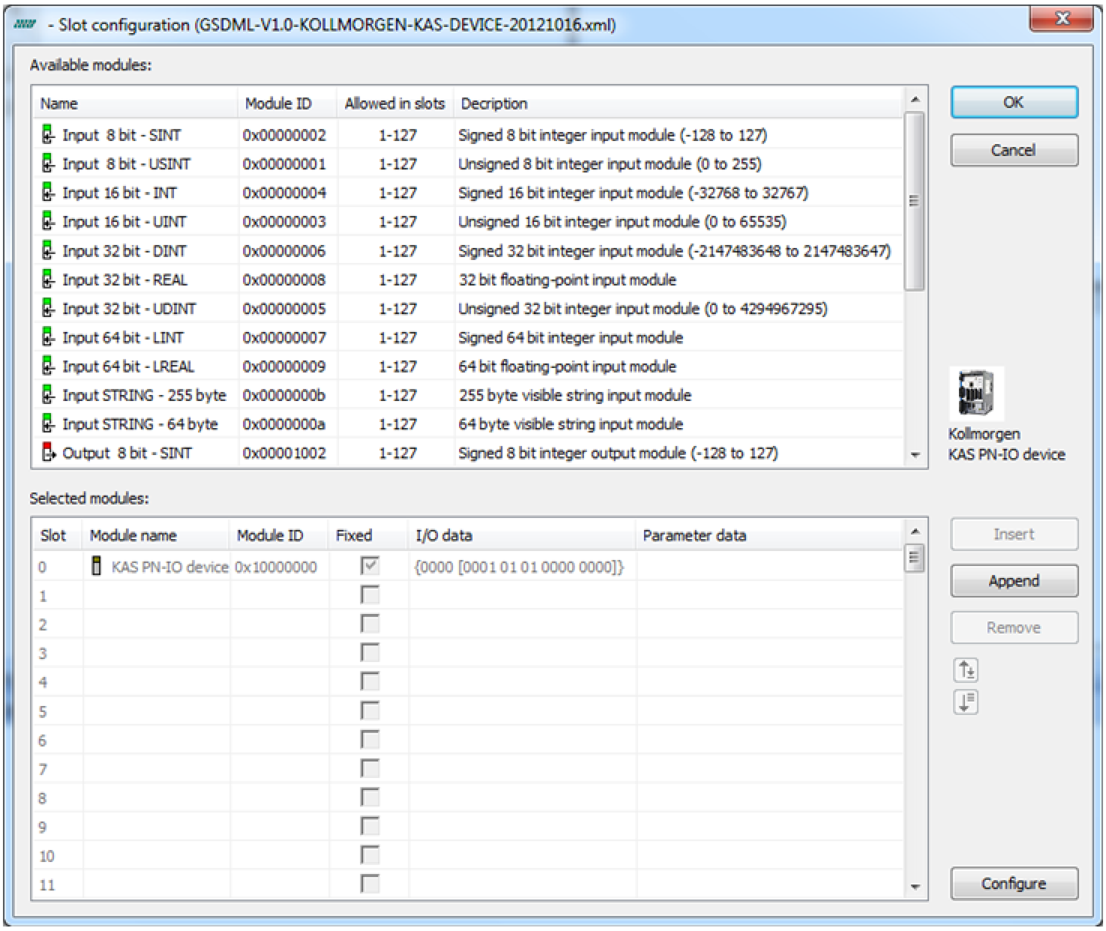
4. Start the declaration of a Profinet device by right clicking on the PROFINET IO device network and choosing Insert Master. The following window opens:



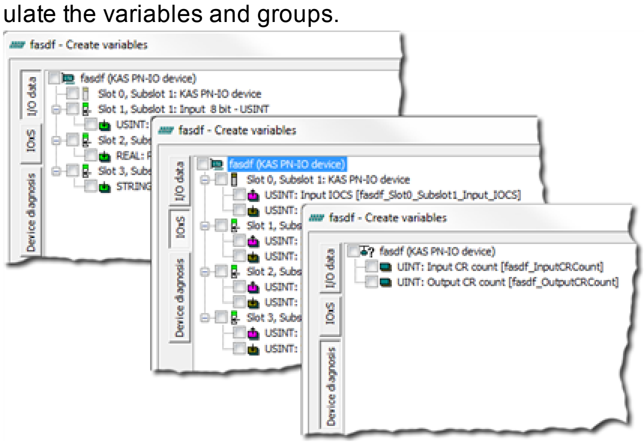
These parameters can be changed:

Parameter	Meaning
Name	A device name can consists of labels and must follow these conventions: • 1 or more labels, separated by [.] • Total length is 1 to 240 • Label length is 1 to 63 • Labels consist of [`a...z0...9-`] • Labels do not start with [-] • Labels do not end with [-] • The first label does not start with "`port-xyz`" or "`port-xyz-abcde`" with a,b,c,d,e, x, y, z = 0...9 • Device names do not have the form `n.n.n.n, n = 0...999` • Labels do only start with '`xn-`' if RFC 3490 is applied
Adapter identification	Must be "`tsec0`"
Maximum # of ARs	Maximum number of alarm retries (default 3).
Initialize IOxS	`On`: Initialize IOxS with good status. `Off`: No initialization of IOxS.
IO communication only from within cycle	`On`: Run IO communication from within VM-cycle `Off`: Run IO communication outside VM-cycle.
Default station name	Name of the station.

5. Click on Configure Slots.



6. Select the modules in the upper list. With the buttons **Insert** and **Append** the modules are copied to the lower list.
   You can not configure each module. Only modules with some sub modules respectively with a sub module with parameter data can be configured. Mark the according module in the lower list and click the **Configure** button.
7. Click OK to close the Slot Configuration and click OK to close the device window.
8. Right click on the master and select Create Variables. This will automatically populate the variables and groups.

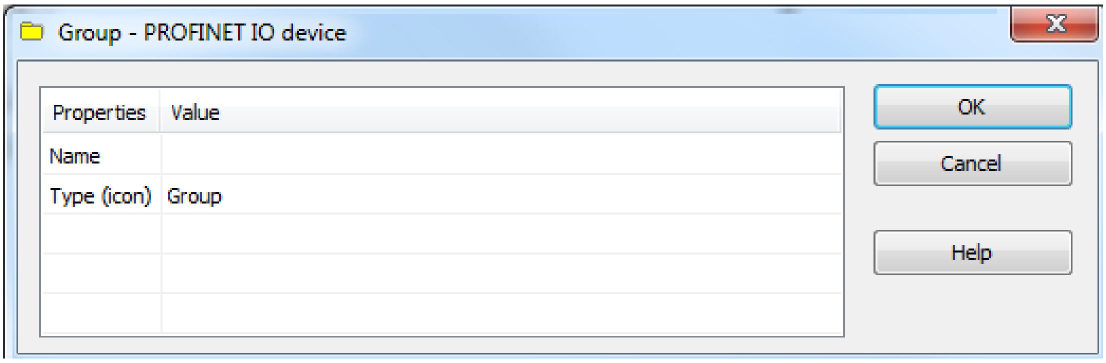


IMPORTANT Every Profinet variable is expanded to a set of boolean variables in PLC by default. A SINT slot, for example, will be mapped to eight PLC BOOL variables. Therefore,

IMPORTANT	if you have many configured slots, many PLC variables will be produced. The AKD PDMM will be slowed by a large amount of PLC variables. To avoid this, you can right click on a slot in the *Create variables* dialog and select *Pack bits*. Doing so with a SINT slot, for example, will create one SINT variable in KAS instead of eight BOOL variables. This will help reduce the number of PLC variables and reduce the load on the AKD PDMM.

TIP	The *Pack bits* action may be applied to all slots by right clicking on the root node in the *Create variables* dialog.

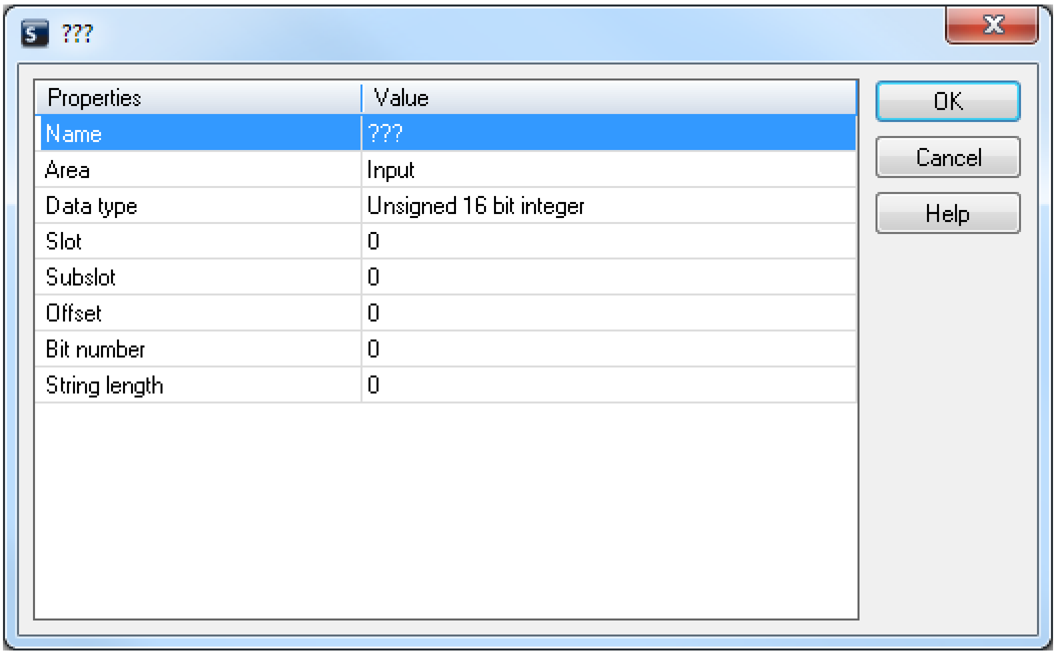
9. Mark the device and right click on it, then choose **Insert Slave** to add a group.



These parameters can be changed:

Parameter	Meaning
Name	Name of the group
Type (icon)	Icon used for the group

10. Now you can connect the variables with the I/Os. Right click on the group and choose **Insert Symbol** in order to append a variable to a device.

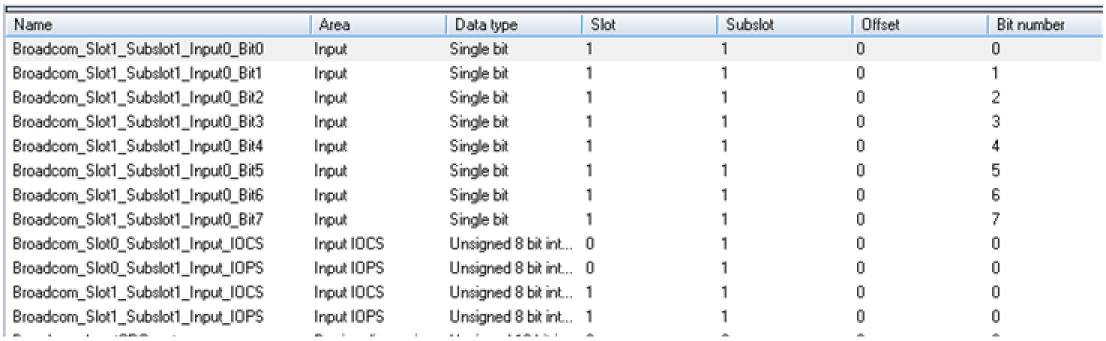


These parameters can be changed:

Parameter	Description
Variable name	Variable name following the IEC 61131-3 syntax.
Area	Output, Output IOCS, Output IOPS, Input, Input IOCS, Input IOPS, device status.
Format	32 bit float, Signed 16 bit integer, Signed 32 bit integer, Signed 8 bit integer, Single bit, Unsigned 16 bit integer, Unsigned 32 bit integer, Unsigned 8 bit integer.
Slot	Slot Number
Subslot	Subslot Number
Offset	Offset
Bit	Bit

The offset of a variable is relative to a sub module. Thus also depending from a slot and subslot. The offset of the first variable of a sub module is always 0.

All settings can be changed in the grid too. The information show refers to the items below of the selected item in the configuration tree.



Name	Area	Data type	Slot	Subslot	Offset	Bit number
Broadcom_Slot1_Subslot1_Input0_Bit0	Input	Single bit	1	1	0	0
Broadcom_Slot1_Subslot1_Input0_Bit1	Input	Single bit	1	1	0	1
Broadcom_Slot1_Subslot1_Input0_Bit2	Input	Single bit	1	1	0	2
Broadcom_Slot1_Subslot1_Input0_Bit3	Input	Single bit	1	1	0	3
Broadcom_Slot1_Subslot1_Input0_Bit4	Input	Single bit	1	1	0	4
Broadcom_Slot1_Subslot1_Input0_Bit5	Input	Single bit	1	1	0	5
Broadcom_Slot1_Subslot1_Input0_Bit6	Input	Single bit	1	1	0	6
Broadcom_Slot1_Subslot1_Input0_Bit7	Input	Single bit	1	1	0	7
Broadcom_Slot0_Subslot1_Input_IOCS	Input IOCS	Unsigned 8 bit int...	0	1	0	0
Broadcom_Slot0_Subslot1_Input_IOPS	Input IOPS	Unsigned 8 bit int...	0	1	0	0
Broadcom_Slot1_Subslot1_Input_IOCS	Input IOCS	Unsigned 8 bit int...	1	1	0	0
Broadcom_Slot1_Subslot1_Input_IOPS	Input IOPS	Unsigned 8 bit int...	1	1	0	0

If KAS is connected to a target system and the system is running, the grid shows the real-time data of the variables.

Data types

You can connect variables of any data type to the Profinet I/Os. The Runtime converts the values of the I/Os to the type of the variable. **STRING** variables are not supported.

Additional features

Create Variables

Find this command in the context menu of the device. Based on the defined device modules the referring variables are generated.

Within the same dialog it is possible to define the variables for

- Device diagnosis
- IOxS

Device Diagnosis

This retrieves the device state information. Based on the Profinet standard the referring variables can be generated:

- InputCRCount [UINT].
- OutputCRCount [UINT].

Create IOxS for Slave Modules

Based on the defined device modules the referring IOPS- and IOCS-variables are generated.

Input modules/modules without IO data:

- The IOPS state is managed by the device. This is the status sent by the module. If the data is invalid, the controller has to ignore it.
- The IOCS state is managed by the controller. The controller can indicate if it can't handle the data.

Output modules:

- The IOPS state is managed by the controller it reflects the status of the data sent by the controller. If the data is invalid the device has to ignore it
- The IOCS state is managed by the device. Le device can indicate to the controller that the data can't be handled.

How to resolve errors

Device is not found

- Check if device is switched on
- Check the network connection
- Ensure the correct name was set on the device

Error setting the IP configuration

- Ensure that the IP configuration is valid and appropriate for your network
- Ensure the VLAN ID is setup correctly in the device settings

CL-RPC Loolup (< 6.22 SP0 Build 3)

Timeout error

- Ensure that the IP configuration is valid and appropriate for your network
- Increase the IP configuration delay

Other errors

- Ensure you use the correct device name
- Ensure you have chosen the correct GSDML file (also check if the GSDML file version is appropriate for the firmware version of the device)

Connect response error

Timeout error (> 6.22 SP0 Build 3)

- Ensure that the IP configuration is valid and appropriate for your network
- Increase the IP configuration delay

Connect response error

Depending on the PNIO status error check, if the settings listed in the table below are set appropriate for your device.

PNIO Status	Check the specified setting
1C010003	Slave\ObjectUUID (correct GDML file)
DB81010A	Master\Activity timeout factor
DB81010B	Master\Controller name
DB81010C	Master\Controller name
DB810207	Master\RT-Class
DB81020A	Master\Send clock factor
DB81020B	Slave\Reduction ratio

PNIO Status	Check the specified setting
DB81020C	Slave\Phase
DB81020F	Master\Watchdog factor
DB810210	Master\Watchdog factor
DB8103,*	Slave\Module IDs,IOs
DB810407	Master\Alarm timeout
DB810408	Master\Alarm retries

Table 10-18: PNIO status error codes on connect and the related settings in the configuration

Module configuration is different

- Ensure you have configured the modules present on the device (If you have read the module configuration from the device, ensure you have selected the correct modules in the resolve ambiguous modules dialog displayed after selecting the GSDML file).
- Ensure you have chosen the correct GSDML file (also check if the GSDML file version is appropriate for the firmware version of the device).

Writing parameterization error (with status 0XDF80*):

- Ensure you have configured the modules present on the device (If you have read the module configuration from the device, ensure you have selected the correct modules in the resolve ambiguous modules dialog displayed after selecting the GSDML file).
- Ensure you have chosen the correct GSDML file (also check if the GSDML file version is appropriate for the firmware version of the device).

If your error could not be resolved using the hints above, update the firmware of your device and the GSDML file to the latest version and try again.

For further analyses record the PNIO communication using Wireshark (http://www.wireshark.org/).

- Start Wireshark and select Capture\Options from the menu.
- Choose the network interface used for the PNIO communication and enter the Capture filter below:

```
ether proto 0x8892 or (ether proto 0x8100 and
ether[16:2]=0x8892) or udp port 0x8894
```

- Then click start.
- Start the straton RT with your application several times and then stop the Wireshark capturing by selecting Capture\Stop from the menu. You can save your recorded traffic by selecting File\Save from the menu.

Coding of PNIO status

The PNIO status is a 32 bit unsigned value. It is composed of 4 byte values, which define the meaning of the status. For positive responses PNIO status has a value of zero.

Bit	Meaning
24-31	ErrorCode
16-23	ErrorDecode
8-15	ErrorCode1
0-7	ErrorCode2

Table 10-19: Coding of PNIO status for negative responses

ErrorCode	Meaning
DB	Error in connect response
DC	Error in release response
DD	Error in control response
DF	Error in write response

Table 10-20: Meaning of ErrorCode for negative responses

ErrorDecode	Meaning
80	Read/Write service
81	Connect, Control, Release service

Table 10-21: Meaning of ErrorDecode for negative responses

ErrorCode1	Meaning
A1	write error
A2	module failure
A3-A6	reserved
A7	busy
A8	version conflict
A9	feature not supported
AA-AF	device specific
B0	invalid index
B1	write length error
B2	invalid slot/subslot
B3	type conflict
B4	invalid area/API
B5	state conflict
B6	access denied
B7	invalid range
B8	invalid parameter
B9	invalid type
BA	backup
BB-BF	device specific
C0	read constrain conflict
C1	write constrain conflict
C2	resource busy
C3	resource unavailable
C4-C7	reserved
C8-CF	device specific

Table 10-22: Meaning of ErrorCode1 for ErrorDecode = 80

ErrorCode1	Meaning
01	Connect Parameter Error, Faulty ARBlockReq
02	Connect Parameter Error, Faulty IOCRBlockReq
03	Connect Parameter Error, Faulty ExpectedSubmoduleBlockReq
04	Connect Parameter Error, Faulty AlarmCRBlockReq

ErrorCode1	Meaning
05	Connect Parameter Error, Faulty PrmServerBlockReq
06	Connect Parameter Error, Faulty MCRBlockReq
07	Connect Parameter Error, Faulty ARRPCBlockReq
08	Read Write Record Parameter, Error Faulty Record
14	IODControl Parameter Error, Faulty ControlBlockConnect
15	IODControl Parameter Error, Faulty ControlBlockPlug
16	IOXControl Parameter Error , Faulty ControlBlock after a connection establishment
17	IOXControl Parameter Error, Faulty ControlBlock after a plug alarm
28	Release Parameter Error, Faulty ReleaseBlock
40	RMPM (Device state machines, device resources)

Table 10-23: Meaning of ErrorCode1 for ErrorDecode = 81

ErrorCode1	Meaning
00	ArgsLength invalid
01	Unknown Blocks
02	IOCR Missing
03	Wrong AlarmCRBlock count
04	Out of AR Resources
05	AR UUID unknown
06	State conflict
07	Out of Provider, Consumer, or Alarm Resources
08	Out of Memory
09-FF	Reserved

Table 10-24: Meaning of ErrorCode2 for ErrorCode1 = 40

For ErrorCode1 <> 40 ErrorCode2 refers to a field in the block specified by ErrorCode1.

ErrorCode2	Meaning
00	Block type
01	Block length
02	Block version (high byte)
03	Block version (low byte)

Table 10-25: Meaning of ErrorCode2 for ErrorCode1 <> 40

ErrorCode2	Meaning
04	AR Type
05	AR UUID
06	Session key
07	Initiator MAC address
08	Initiator Object UUID
09	AR Properties
0A	Activity timeout factor
0B	UDP RT port

ErrorCode2	Meaning
0C	Station name length
0D	Station name

Table 10-26: Meaning of ErrorCode2 for ErrorCode1 = 01 (AR block request)

ErrorCode2	Meaning
04	IOCR Type
05	Reference
06	LT
07	IOCR properties
08	IO data length
09	Frame ID
0A	Send clock factor
0B	Reduction ratio
0C	Phase
0D	Sequence
0E	Frame send offset
0F	Watchdog factor
10	Data hold factor
11	Tag header
12	IOCR multicast MAC address
13	Number of APIs
14	API
15	Number of IO data objects
16	Slot
17	Subslot
18	IO data object offset
19	Number of IOCS
20	Slot
21	Subslot
22	IOCS offset

Table 10-27: Meaning of ErrorCode2 for ErrorCode1 = 02 (IOCR block request)

ErrorCode2	Meaning
04	Number of APIs
05	API
06	Slot
07	Module ident number
08	Module properties
09	Number of submodules
0A	Subslot
0B	Submodule ident number
0C	Submodule properties
0D	Data description
0E	Data length

ErrorCode2	Meaning
0F	IOPS length
10	IOCS length

Table 10-28: Meaning of ErrorCode2 for ErrorCode1 = 03 (Expected submodule block request)

ErrorCode2	Meaning
04	Type
05	LT
06	AlarmCR Properties
07	RTA timeout factor
08	RTA retries
09	Local alarm reference
0A	Maximum alarm data length
0B	Alarm CRT Tag high
0C	Alarm CRT Tag low

Table 10-29: Meaning of ErrorCode2 for ErrorCode1 = 04 (AlarmCR block request)

ErrorCode2	Meaning
04	Sequence number
05	AR UUID
06	API
07	Slot number
08	Subslot number
09	Padding
0A	Index
0B	Data length
0C	Target AR UUID

Table 10-30: Meaning of ErrorCode2 for ErrorCode1 = 8 (Read/write record block request)

ErrorCode2	Meaning
05	Padding
06	Session key
07	Padding
08	Control block command
09	Control block properties

Table 10-31: Meaning of ErrorCode2 for ErrorCode1 = 16 (IOXControl block request)

9.4 Project Structure Guidelines

9.4.1 Introduction

By implementing a predefined structure for new projects, KAS tries to achieve the following goals:

- Efficiency in developing new applications
- High flexibility to keep only functionalities that are needed and to create the new ones that are required
- Safe applications due to an already tested and approved structure that optimize the resources usage (memory and processor load)
- Reliable framework that supports error, state, data and communication management
- Easier to exchange applications
- Less time needed to understand, maintain and teach an application (from a troubleshooting and support standpoint)
- Less documentation work is required since the main behavior of the Application is already documented (only the specific functionalities need some additional work)

9.4.2 External Files

Some items that belongs to your application (displayed in the **Project Explorer**) are not embedded into the project file. For the domains listed below, KAS IDE also uses some resources that are stored in external files.

Domain	Description	File
HMI 	Using Kollmorgen HMI, simply tag the variables in the PLC environment to create an export file that describes the data to be exchanged between the PLC and the HMI. Import this Modbus mapping file into the HMI programming environment and use the variables as if they are local variables	KVB Project File
PLC	The PLC programming environment gives you the possibility to create reusable components (UDFB), and template applications which can be customized to suit any given application	Create Custom Libraries Read Common Constants
Motion	The CAM editor lets you create complex CAM profiles online using a “graphical” interface. It is also possible to import existing CAM profile points into the CAM editor to allow you to reuse your existing machine building experience seamlessly	Import Cam Profile Export Softscope Data
Fieldbus	Kollmorgen Automation Suite tightly integrates the EtherCAT motion bus (standard Ethernet-based cabling) to define all the network description	Import or Export EtherCAT ENI File
Drive 	The AKD drive is fully embedded in the Kollmorgen Automation Suite but not all interwoven at one time. This makes future customization easier to get all the firmware features	Download AKD Firmware

Table 10-32: - File location

ⓘ ***TIP*** The hyperlinks bring you to the relevant topic that contains more details.

9.4.3 Application Software Structure - Definitions

9.4.3.1 Modules to build up the Structure

Structure Overview

You normally write the PLC program. Whereas Kollmorgen application team members create in most cases the motion control part.

The global software structure is built up with different modules placed on two different levels as showed on the figure below:

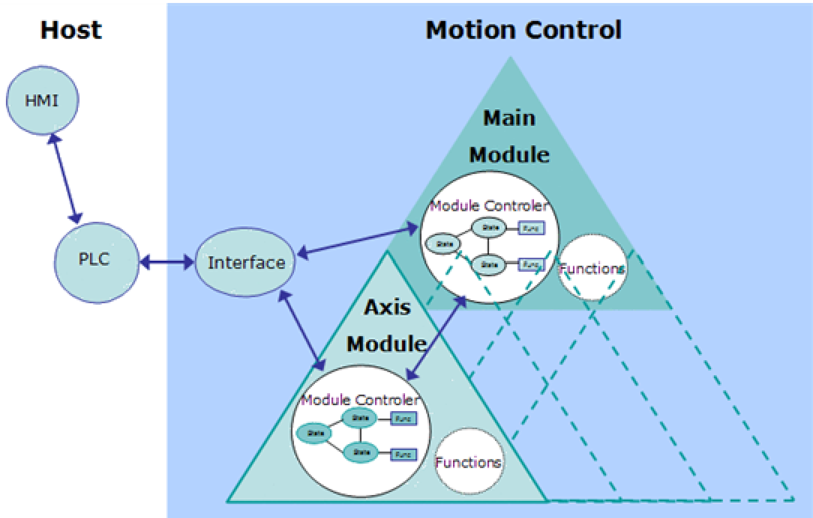


Figure 10-30: Software Structure Overview

Module Definition:

- A module is one unit of the software structure (triangle)
- It is controlled by one module from the next higher level and can in turn control several modules in the next lower level
- It never communicates with modules of the same level
- It can generally run independently from any other modules at the same or higher level

To have the structure running as a real application, it needs to be controlled by a PLC. As the PLC is not part of the application structure, only the main and axis modules are described here.

Main Module description

The main module controls the functional work that globally affect the application (e.g. multi axes functions). It receives commands from the PLC and sends back acknowledgements. The main module does not directly act on the physical axes, but controls the axis modules that are linked to them.

Communication between main and axis modules is done via internally defined data channels.

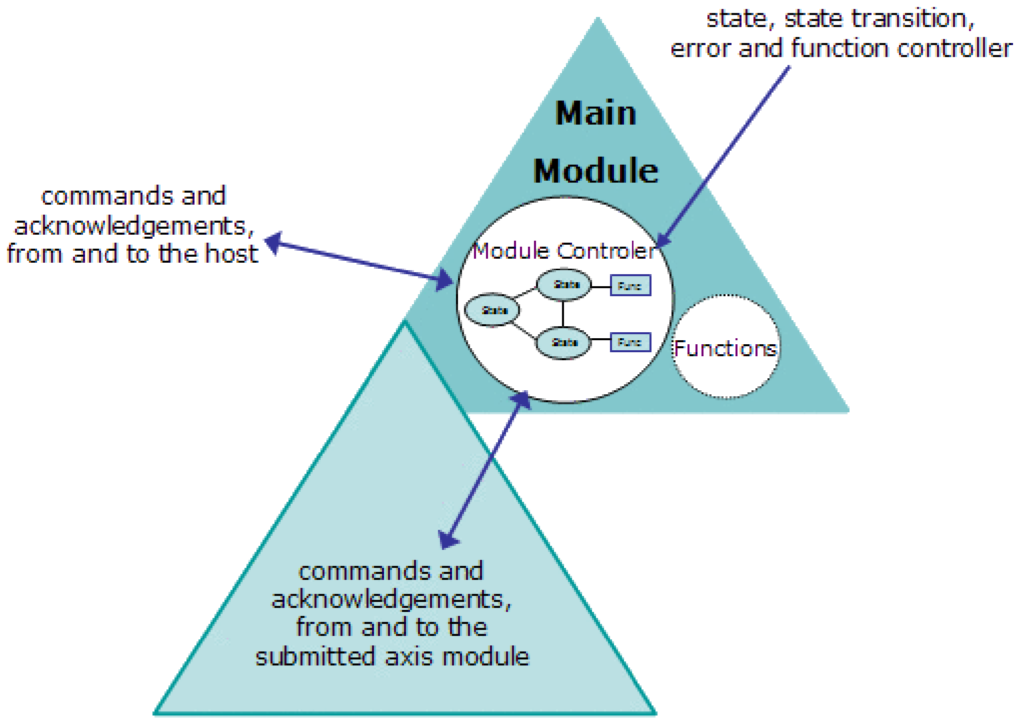


Figure 10-31: Main Module Description

As shown on the figure, the main module consists of two parts:

- the **module controller** part is responsible for state, state transition, error and functions handling. It receives state transition and function call commands from the host, performs all needed actions and sends back some acknowledgements. In case of an error it reacts by itself and sends a message to the PLC. If requested, it activates state transitions and functions in the axis modules, by sending commands to them and waiting for acknowledgement. The main module controller also manages the error status of the submitted modules and performs the needed actions.
- the **functional part** consists of all functionalities needed for the current application. These functions can be state dependant (e.g. multi axes functions) or state independent (e.g. increase a speed value).

Axis Module description

The axis module controls the functional work that affect the application one or more physical axes (e.g. single-axis functions). It receives commands from the PLC and sends back acknowledgements.

The axis module also communicates with its main module via the internally defined data channel.

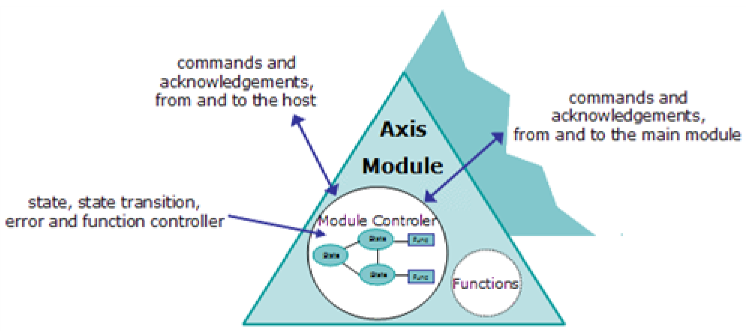


Figure 10-32: Axis Module Description

As shown on the figure, the axis module consists of the same two parts as the main module:

- the **module controller** part is responsible for state, state transition, error and functions handling. If the axis module is not connected to its main module, it receives state transition and function call commands from the host, performs all needed actions and sends back some acknowledgements. If connected, state transition commands are received from its main module and not from the host. In case of an error it only reacts by itself, if it is not connected to the main module.
- the **functional part** consists of all functionalities needed for the current physical axis. These functions can be state dependant (e.g. single axes functions) or state independent (e.g. increase a speed value).

9.4.3.2 State and Function Definitions

A state machine and some functions of general interest are implemented in the software structure. They are provided as examples of how to use the structure but can be adjusted to fulfil specific application usage (see also paragraph "How to add a new state" on page 568 and paragraph "How to add a new function" on page 570).

State transition Diagram

The following state machine has been defined.

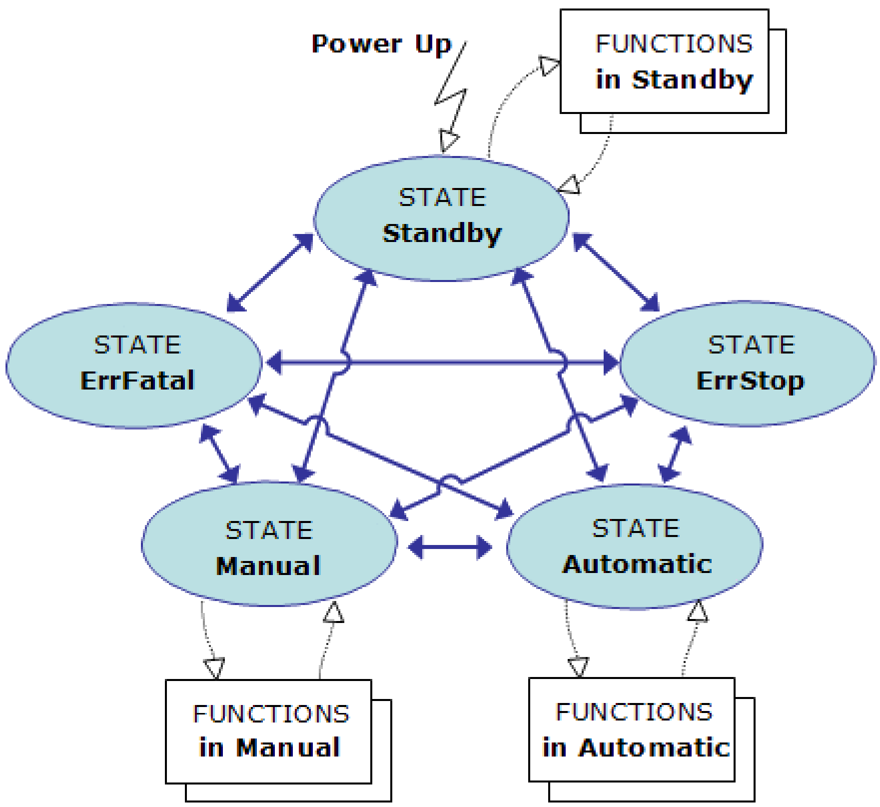


Figure 10-33: State Machine

Legend

STATE	Represents the current state of one or more modules (e.g. Standby state means that the module is waiting for commands) In addition to the static states, there are two other possible dynamic states: • **Busy**: when performing a state transition (exit old and enter new state) • **Running**: when working through a function
State Transition	It is the transition from one state to another when performing the corresponding action
FUNCTIONS	Any needed machine or single-axis functionality (e.g. single-axis jog), which can be called out of one or several states

All modules have the same states and state transitions. The state of a module is only influenced by other modules, if they are connected with each other.

State, state transitions and functions descriptions

The structure is built in such a way that state transitions are possible from the active state to any other existing states (except state ErrStop). After leaving state ErrStop (corresponding to a non-fatal error, which causes a stop and power off) the structure automatically recovers the state which was active before entering ErrStop. That means that all characteristics of the previous state are kept.

NOTE Because functionalities are always specific to the application, none are included in the structure itself.

9.4.4 Application Software Structure - Implementation

This chapter describes how the software structure described before is implemented. Insofar as all modules are implemented and behave in the same way, only the main module is described in detail here.

9.4.4.1 SFC children building up the software

The following files contain all the data to build up the application. They are all required to ensure a successful compilation.

Parent SFC

Main	System start up and SFC children call

Main module SFC children

M1_StateController	state and function controller of the main module
M1_ErrorHandling	error handling of the main module
M1_IndependentFunctions	state independent functions of the main module
M1_Interface	interface to PLC

Axis module SFC children

Ai_StateController	state and function controller of the axis module
Ai_ErrorHandling	error handling of the axis module
Ai_IndependentFunctions	state independent functions of the axis module

With i = 1... n

9.4.4.2 Variables for the Interface

List of variables

- M1_CmdState
- bM1_CallStandbyFunction1
- bM1_CallStandbyFunction2
- bM1_CallManualFunction1
- bM1_CallManualFunction2
- bM1_CallAutomaticFunction1
- bM1_CallAutomaticFunction2
- bAi_CallStandbyFunction1
- bAi_CallStandbyFunction2
- bAi_CallManualFunction1
- bAi_CallManualFunction2
- bAi_CallAutomaticFunction1
- bAi_CallAutomaticFunction2
- bErrorReset

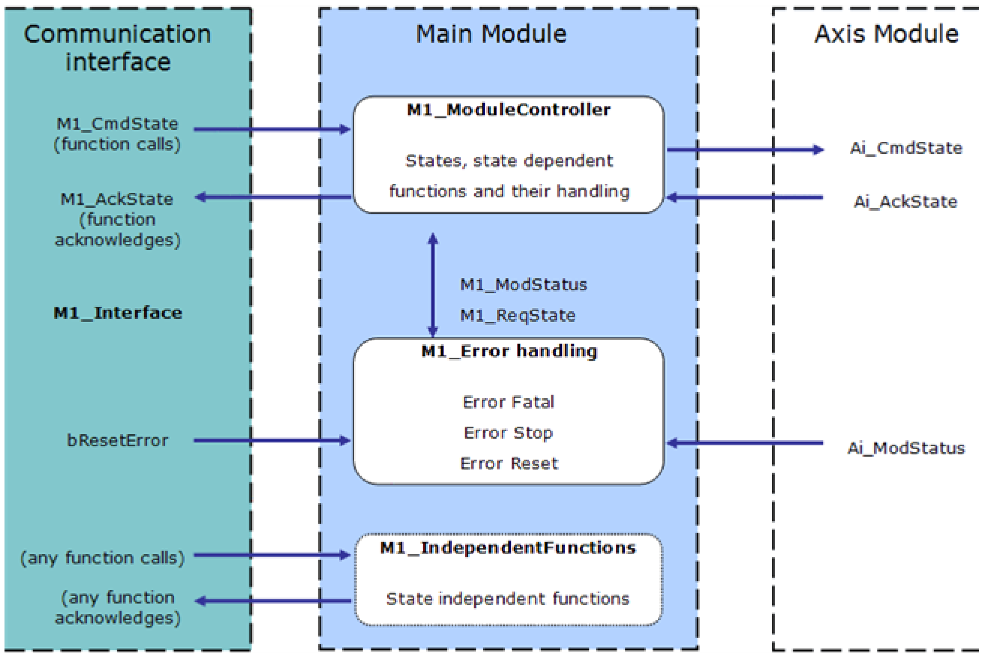
List of output variables

- M1_AckState
- M1_StatusWord
- bM1_Running
- Ai_StatusWord
- bAi_Running

9.4.4.3 Main module implementation description

In the main module, all necessary state, state transition, error and function handling facilities are implemented for this level.

Context diagram for the main module



The following objects (variables, tasks...) are defined in the structure of the main module.

M1_CmdState

Description

This internal word variable contains the actual state command value. It is automatically set to state '**Standby**' during power up.

```
//***********************************************
//**              State  Defines              **
//***********************************************
#define  DEF_StateUndefined                    0
#define  DEF_StateStandby                      1
#define  DEF_StateManual                       2
#define  DEF_StateAutomatic                    3
#define  DEF_StateBusy                        4
#define  DEF_StateErrorStop                    5
#define  DEF_StateErrorFatal                   6
```

Usage

These state commands are usually set in the communication interface (see software listing of ACT_M1_Translate and ACT_M1_SimaticSimu) and must not be set directly from the host system. If additional or different state commands are needed, then the definitions described above can be modified accordingly.

M1_AckState

Description

This internal word variable contains the actual state acknowledge value, as a result from the **M1_CmdState** state command performed with success. Possible values are the same as for the state commands (see above).

Usage

Out of this value the corresponding acknowledgements for the PLC can be created in the communication interface.

M1_ReqState

Description

This internal word variable contains the internally active state. It is used for internal purpose only, to keep the actual state value, e.g. while performing a function. Possible values are the same as for the state commands (see above).

Usage

Used by system, do not use it for application purpose.

Description

This internal word variable contains the actual module status and error information. It is automatically set to the default value during power up. The meaning of the predefined Module Error Bits are as follows:

Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0

Bits	Description
0	error stop reported by drive (drive error)
1	error fatal reported by Drive (lag error)
2	not used (motor temperature too high)
3	not used (external stop)
4	not used (negative limit switch reached)
5	not used (positive limit switch reach)
6	not used (not used)
7	not used (not used)
8	not used (state HW enable)
9	not used (state AS enable)
10	not used (axis is powered on)
11	not used (axis is homed)
12	not used (axis is running)
13	not used (pipe is connected)
14	error stop (error stop)
15	error fatal (error fatal)

Usage

While the error bits are usually set only by the error handling (M1_ErrorHandling), the mode bits can be modified where ever needed in the application program (except in the interface). Several bits can be set at the same time. Several masks have been defined to test or modify the whole word. For each module, there is one mask to define the bits causing a fatal error (e.g. **MSK_M1_StatusErrorFatal**) and one for the stop error (e.g. **MSK_M1_StatusErrorStop**). To add errors and modes, the bits not already assigned by default can be used (i.e. bits 16 to 31).

bErrorReset

Description

This internal flag variable is used as the error reset command for the main and axis modules. It is reset during power up.

Usage

Set and reset this flag to activate a reset of the module errors (M1_StatusWord, Ai_StatusWord).

M1_ErrorHandling

Description

This program is responsible for the main module error handling. If an error occurs (in the main module or a submitted axis module), the corresponding bit in the module status (M1_StatusWord) is set. This causes the error reaction bits (MSK_Mi_StatusErrorStop, MSK_Mi_StatusErrorFatal) to be set in the module status word.

Usage

Any additional error which needs to be treated has to be included in this program. Do not forget to modify the corresponding masks (**MSK_M1_StatusErrorFatal**, **MSK_M1_StatusErrorStop**) to cause the correct reaction on errors.

M1_ModuleController

Description

This program is the heart of the whole controller and contains:

- a state manager sequence
- all state sequences
- and state dependent function sequences of the main module

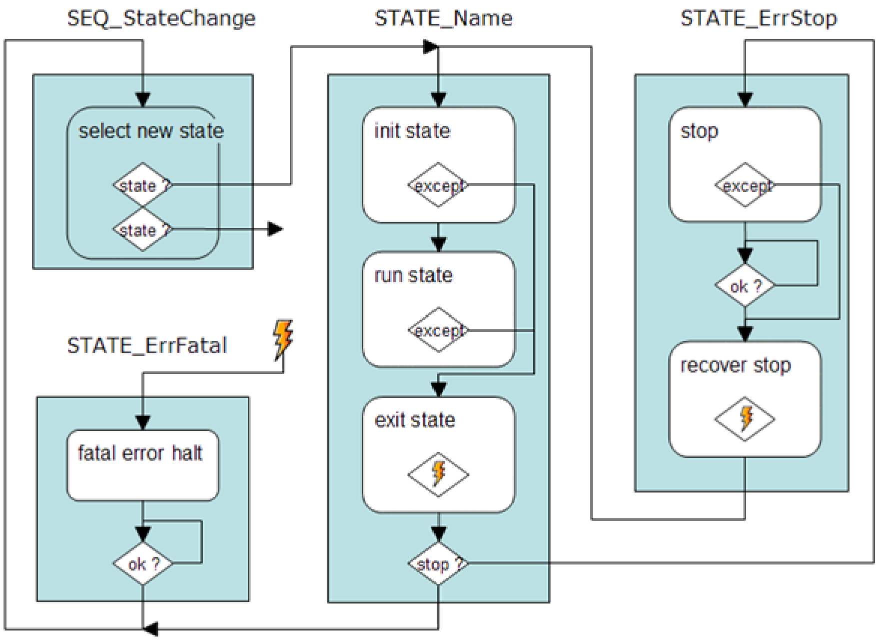
Usage

Some rules have to be followed, when using and changing states and functions (see also paragraph "How to add a new state" on page 568 and paragraph "How to add a new function" on page 570).

9.4.4.4 States and Errors

How States and Errors are treated

The figure below shows how states and errors are treated.



StateChange (state manager)

Activates the new state required by M1_ReqState

StateName (state macro)

init state	- Initializes exceptions on new state M1_CmdState <>[1] M1_ReqState and on errors set in M1_StatusWord - Goes to exit state when an exception occurs - Performs all actions to properly enter this state (init variables, pipes, ...) - Sends commands to the submitted axis modules by setting Ai_ CmdState to StateName and waits for their acknowledgement in Ai_ AckState - Acknowledges end of initialization by setting M1_AckState to M1_ ReqState
run state	- Waits for any function calls, activate function if called
exit state	- Performs all actions to properly leave this state - Acknowledges running by setting M1_AckState to '**busy**' - If error stop occurs, activates STATE_ErrStop, otherwise sets new requested state M1_ReqState to M1_CmdState and activates StateChange

How to add a new state

To add a new state, do as follows:

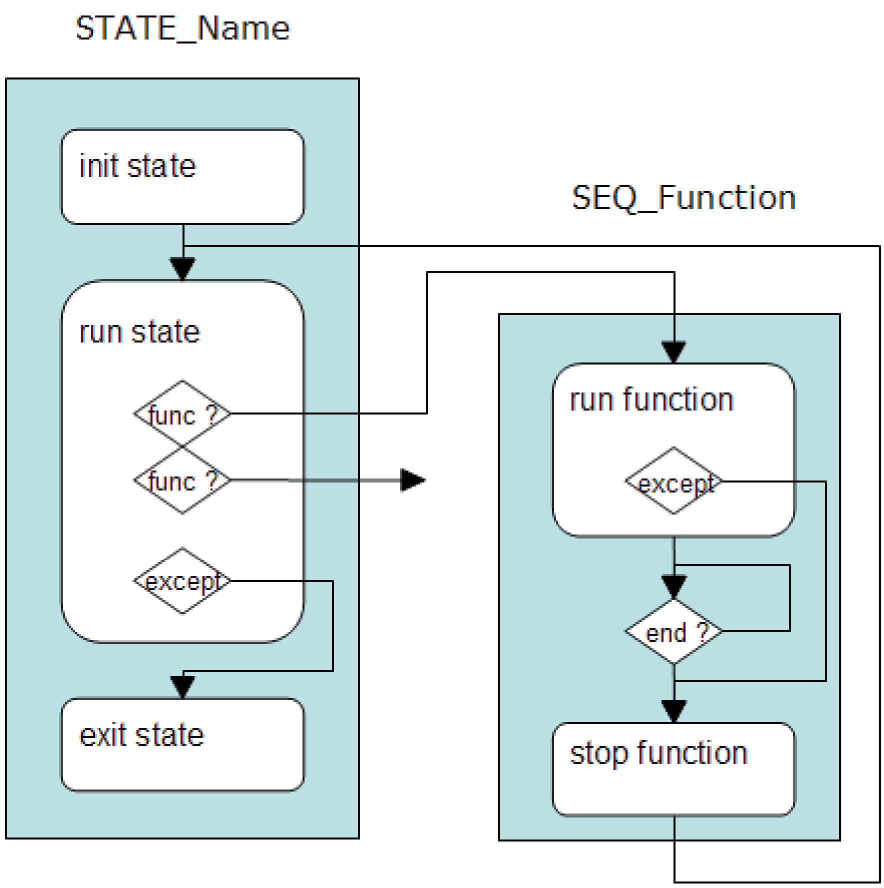
[1] <> means Not Equal

1. copy a similar existing state sequence
2. replace the old state name by the new one (e.g. 'Standby' by 'MyState')
3. modify both init and exit sections of the new state to perform the relevant actions
4. insert the needed function calls into the states run part
5. add the state call command line into the state change sequence
6. add the state definition values to the general declaration

9.4.4.5 Functions linked to states

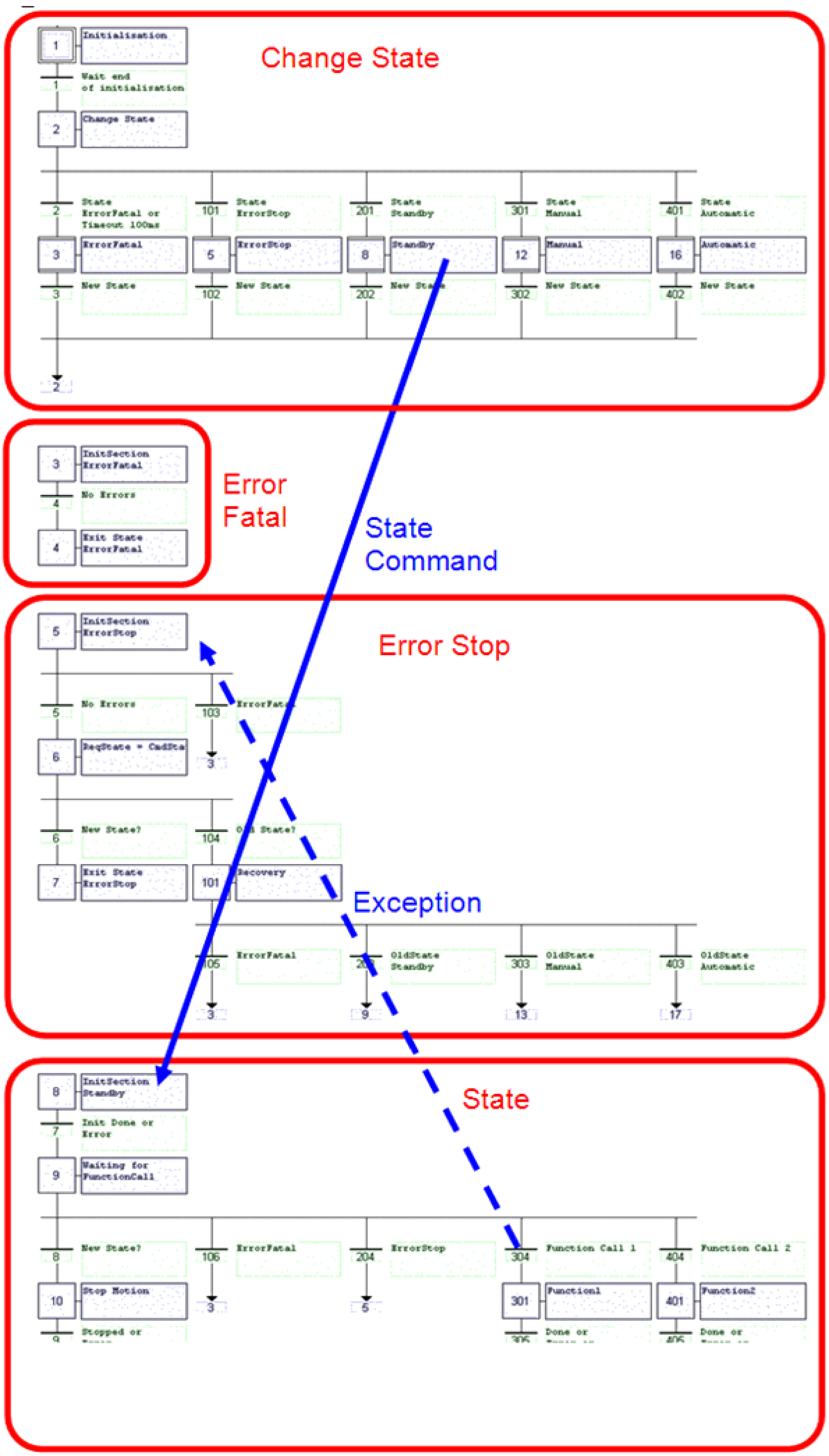
How Functions are treated

The figure below shows how functions (that are state dependent) are treated.



Function (function step)

run function	- Initializes exceptions on new state M1_CmdState <> M1_ReqState and on errors set in M1_StatusWord - Goes to exit function when an exception occurs - Acknowledges running - Performs all actions needed for the function until the function call command is reset
stop function	- Performs all actions to properly leave this function - Acknowledges end of exit, by setting M1_AckState to M1_ReqState - Returns to last state



How to add a new function

To add a new function, do as follows:

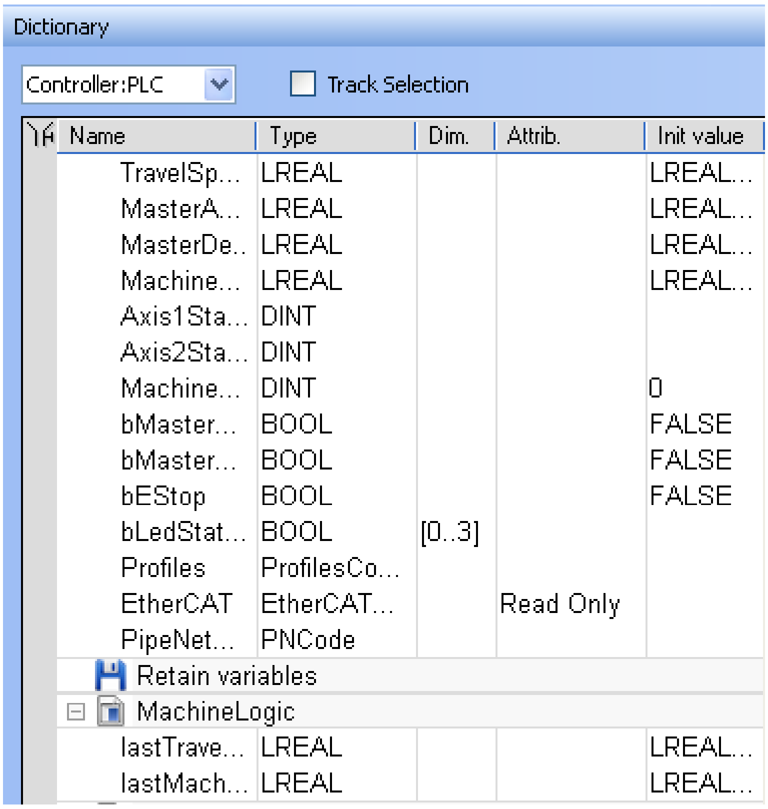
1. copy a similar existing function sequence
2. replace the old function name by the new one (e.g. 'Running' by 'MyFunction')
3. modify the exit section of the new function to perform the relevant actions
4. insert the needed function code into the run part
5. add the function call command line to the state sequence where the function is used

9.5 Templates

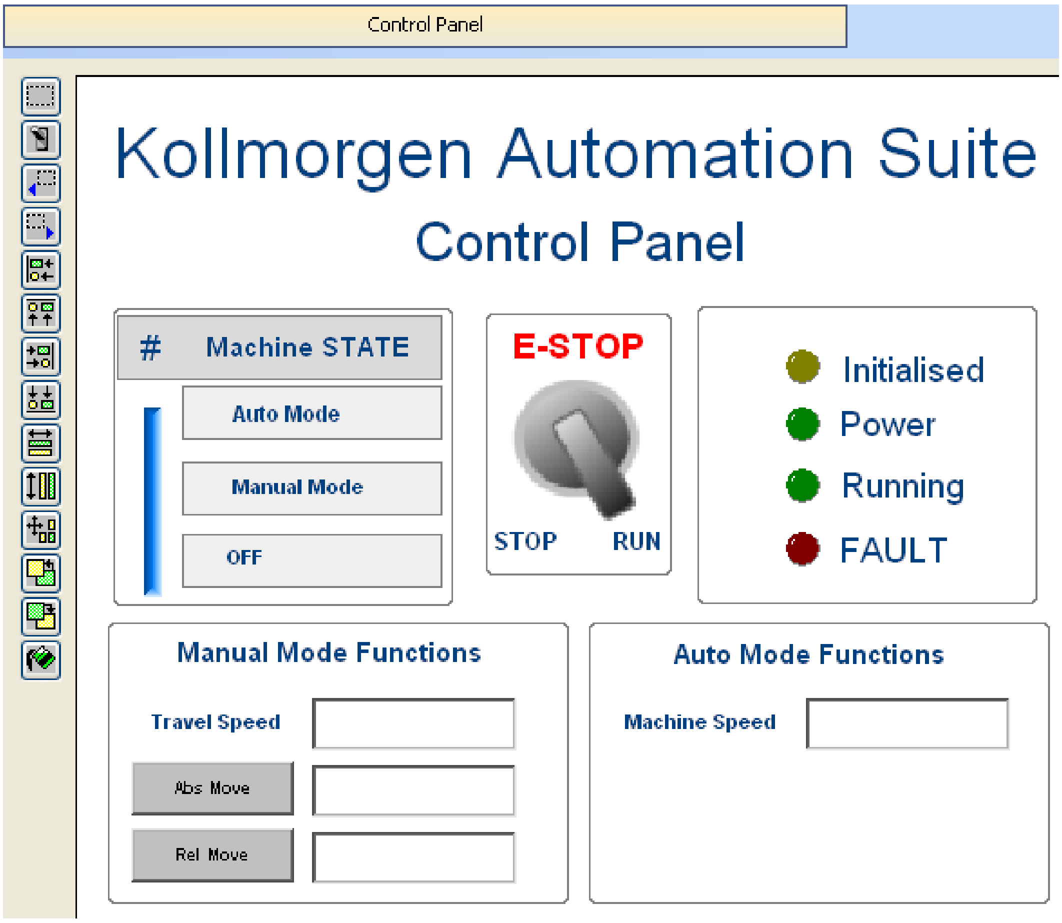
KAS provides start up templates to help you getting started (see how to use the project setup wizard here). These templates come complete with software to:

- Create two axes of servo motion
- Enabled the drives
- Perform simple motion

The templates contain variables for supporting this operation.



Additionally, they contain a Control Panel for ease of running motion.



There are templates for the Pipe Network motion engine and templates for the PLCopen motion engine:

9.5.1 Pipe Network 2-Axes Template with SFC, ST, FFLD, and FBD

9.5.1.1 PLC Programs

The 2-axes Pipe Network template has an SFC program (called **Main**) that initializes and starts the motion.

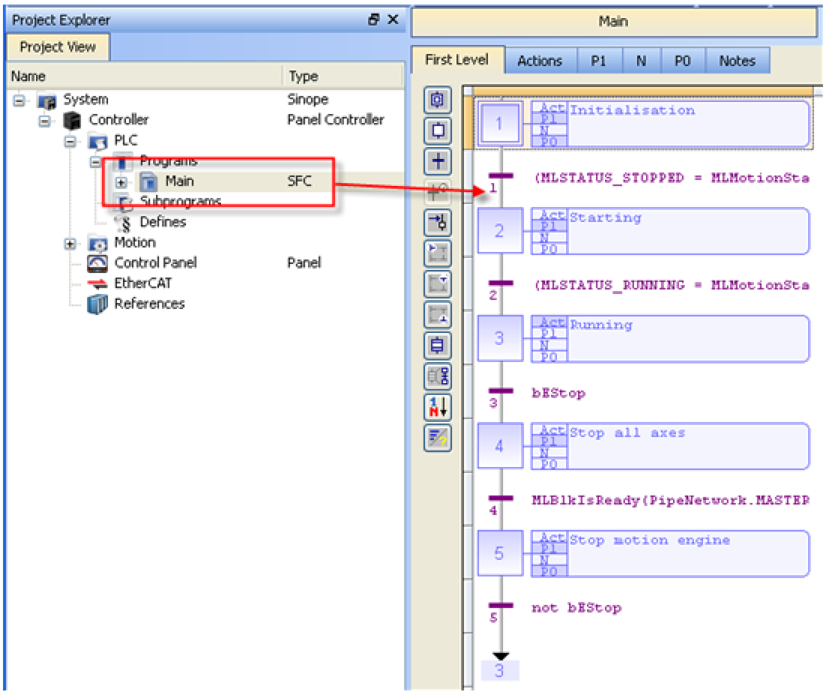


Figure 10-34: PN Template - Main

The Pipe Network Template contains an SFC child program called Machine Logic for running motion.

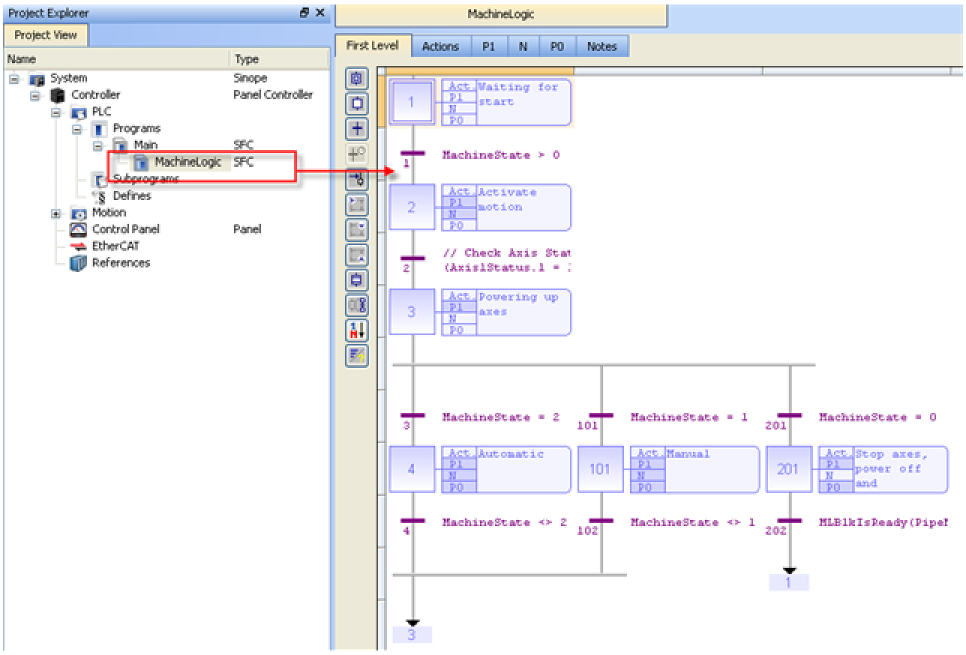
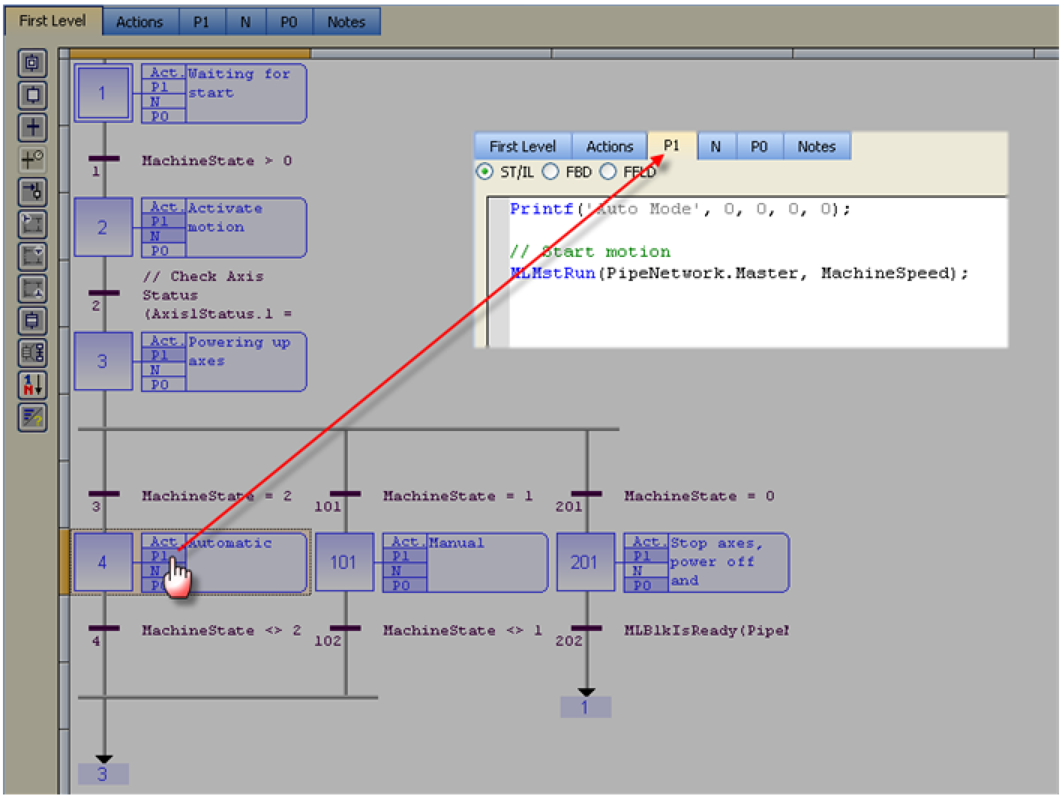
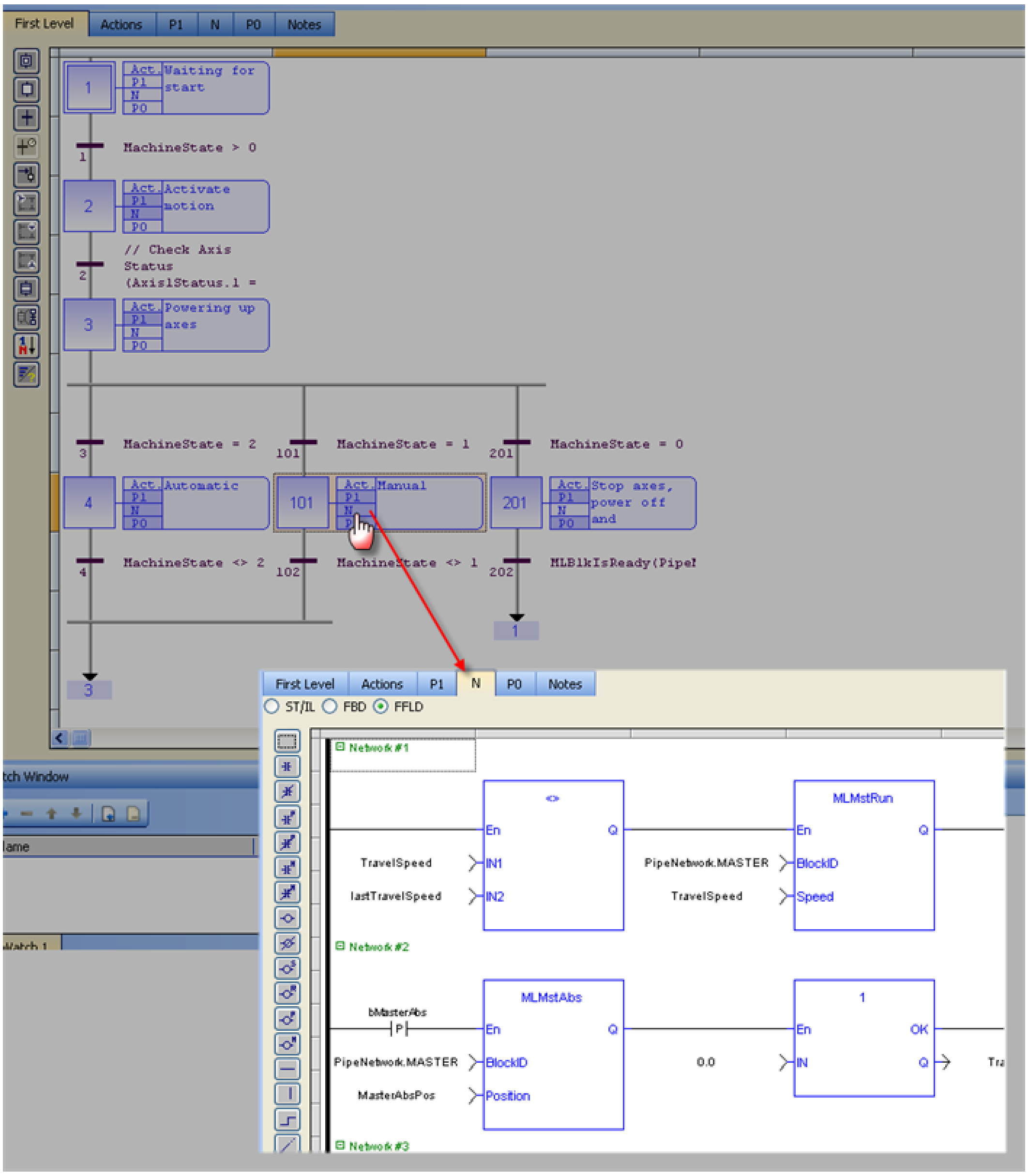


Figure 10-35: PN Template - MachineLogic

ST programs can be found in the P1 and P0 actions for many steps



FFLD programs can be found in the N action for steps 4 and 101



9.5.1.2 Motion

The template has a motion profile defined with the graphical Pipe Network editor.

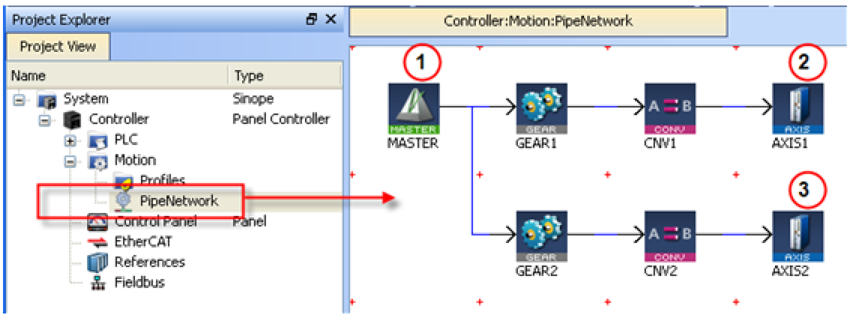


Figure 10-36: PN Template - Motion

The motion profile contains four different pipe blocks:

- The **Master** (see call out 1) is the generator that allows a synchronization between the two pipes (2 and 3).
- The **Gear** modifies (with ratio and offset) the flow of values issued from the Master.
- The **Convertor** controls the position of the axis.
- The **Axis** gives access to the physical remote drive

9.5.1.3 Control Panel

For more details, see "Design the Control Panel with the Internal Control Panel Editor" (see page 439)

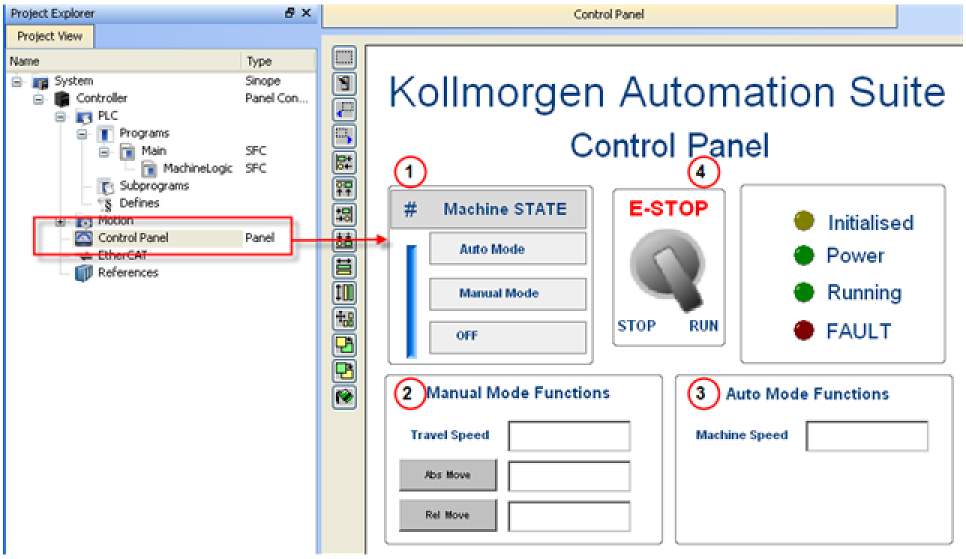


Figure 10-37: PN Template - Control Panel

Call out#	Description
1	Allows to choose how to run the axes between automatic and manual modes
2	In manual mode, you can set the speed. You can also set an absolute and relative move. When you click those commands, the two axes move to the specified position and the speed is reset to 0
3	In automatic mode, you can set the speed
4	When you click the emergency button, the machine state becomes OFF (see call out 1) and the two axes stop running

Table 10-33: PN Template - Control Panel

Based on the template, the project can be run:

- using the KAS Simulator
- with actual drives and motors (in this case, you first have to set up the axes in the EtherCAT part. For more details, click here...)

9.5.2 Pipe Network 2-Axes Template with ST only

9.5.2.1 PLC Programs

The 2-axes Pipe Network template has a ST program (called **Main**) that initializes, starts and runs the motion.

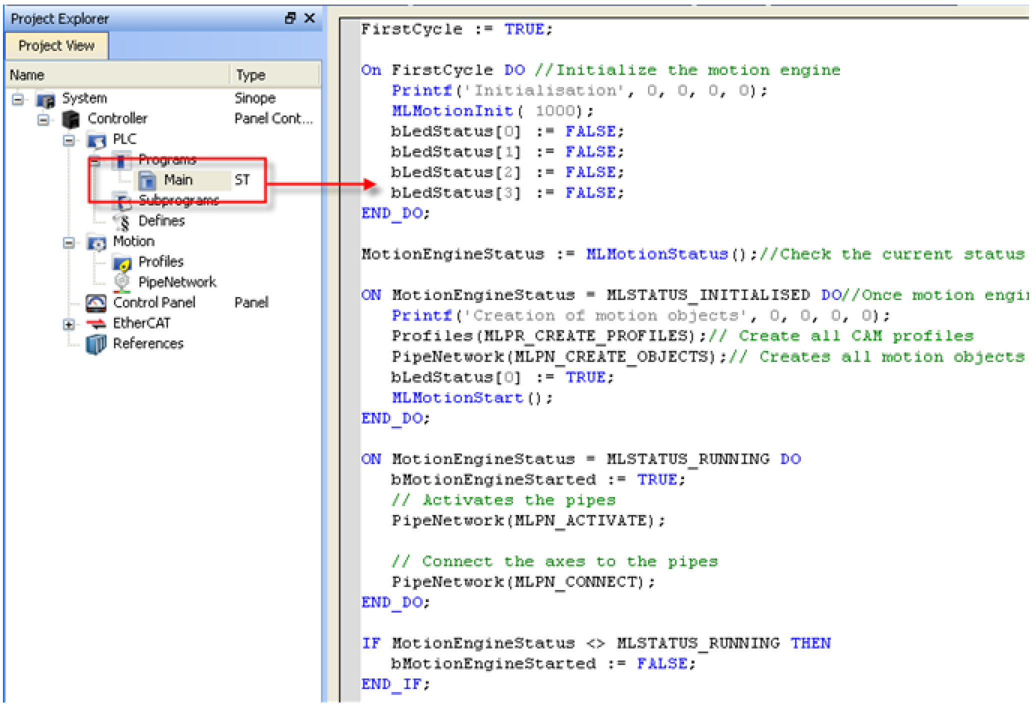


Figure 10-38: PN Template with ST - Main

9.5.2.2 Motion

The template has a motion profile defined with the graphical Pipe Network editor.

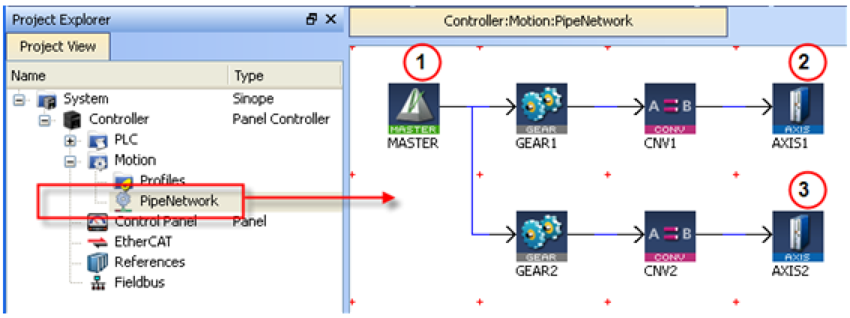


Figure 10-39: PN Template - Motion

The motion profile contains four different pipe blocks:

- The **Master** (see call out 1) is the generator that allows a synchronization between the two pipes (2 and 3).
- The **Gear** modifies (with ratio and offset) the flow of values issued from the Master.
- The **Convertor** controls the position of the axis.
- The **Axis** gives access to the physical remote drive

9.5.2.3 Control Panel

For more details, see "Design the Control Panel with the Internal Control Panel Editor" (see page 439)

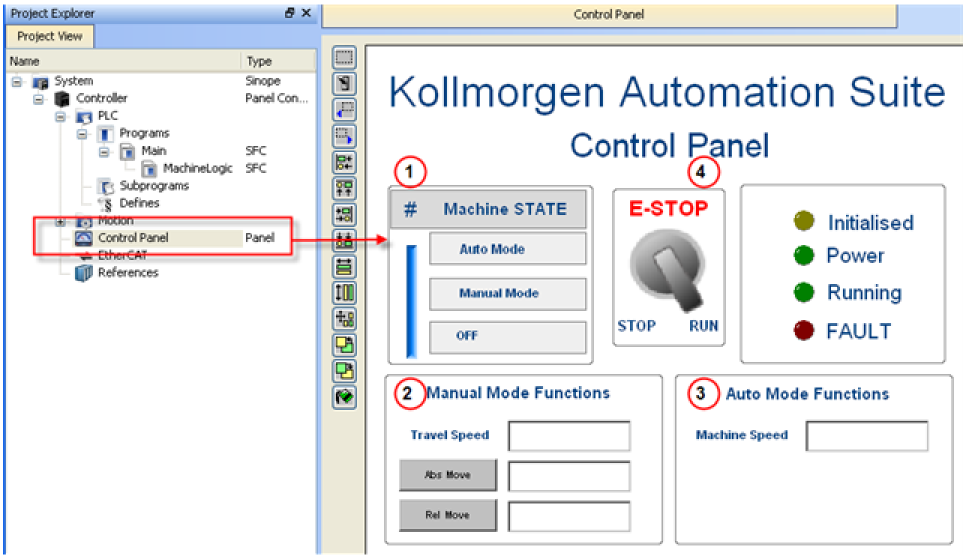


Figure 10-40: PN Template - Control Panel

Call out#	Description
1	Allows to choose how to run the axes between automatic and manual modes
2	In manual mode, you can set the speed. You can also set an absolute and relative move. When you click those commands, the two axes move to the specified position and the speed is reset to 0
3	In automatic mode, you can set the speed
4	When you click the emergency button, the machine state becomes OFF (see call out 1) and the two axes stop running

Table 10-34: PN Template - Control Panel

Based on the template, the project can be run:

- using the KAS Simulator
- with actual drives and motors (in this case, you first have to set up the axes in the EtherCAT part. For more details, click here...)

9.5.3 Pipe Network 2-Axes Template with FFLD only

9.5.3.1 PLC Programs

The 2-axes Pipe Network template has a FFLD program (called **Main**) that initializes, starts and runs the motion.

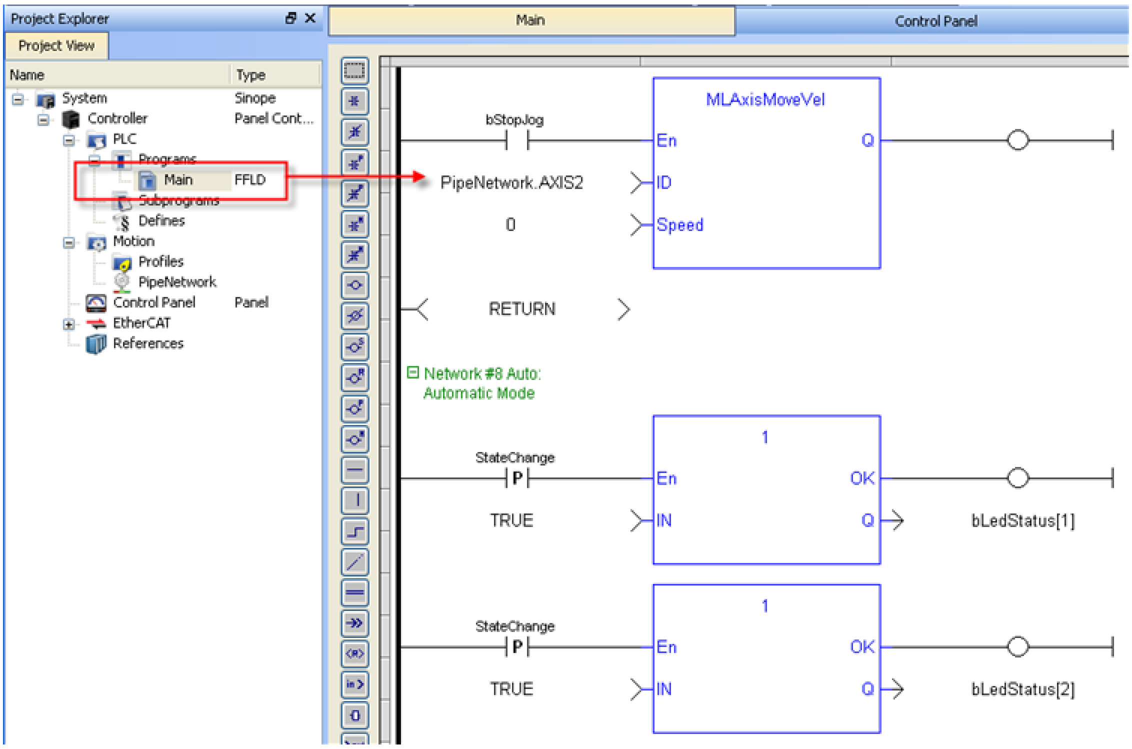


Figure 10-41: PN Template with FFLD - Main

9.5.3.2 Motion

The template has a motion profile defined with the graphical Pipe Network editor.

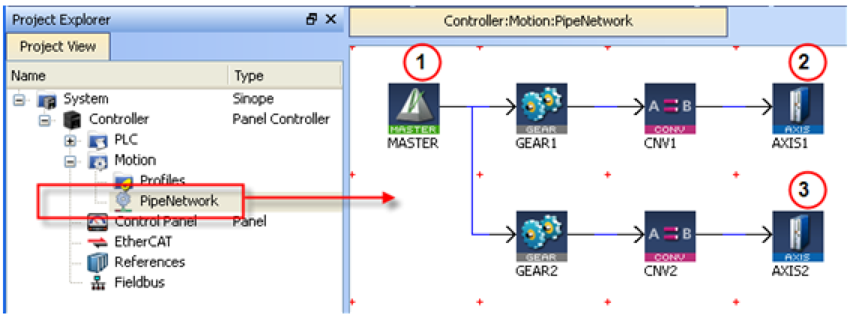


Figure 10-42: PN Template - Motion

The motion profile contains four different pipe blocks:

- The **Master** (see call out 1) is the generator that allows a synchronization between the two pipes (2 and 3).
- The **Gear** modifies (with ratio and offset) the flow of values issued from the Master.
- The **Convertor** controls the position of the axis.
- The **Axis** gives access to the physical remote drive

9.5.3.3 Control Panel

For more details, see "Design the Control Panel with the Internal Control Panel Editor" (see page 439)

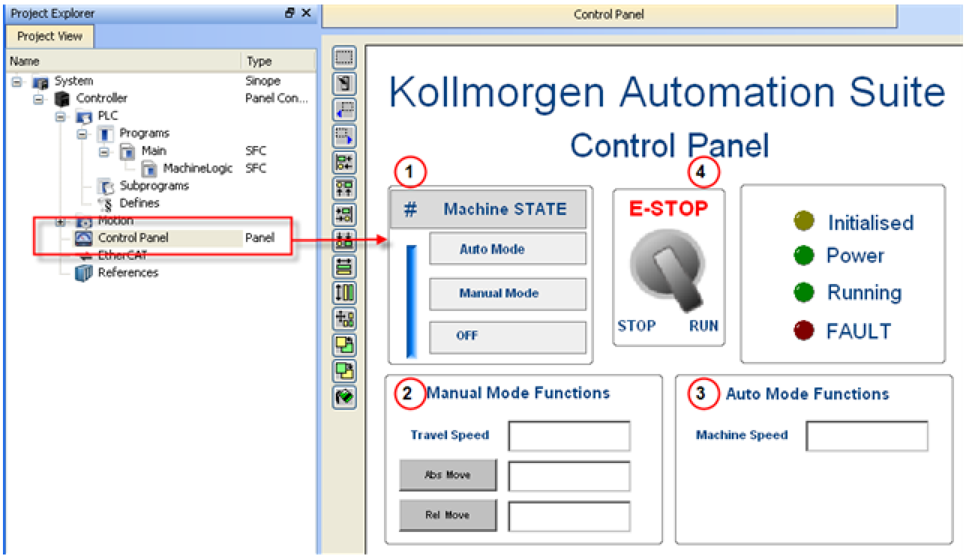


Figure 10-43: PN Template - Control Panel

Call out#	Description
1	Allows to choose how to run the axes between automatic and manual modes
2	In manual mode, you can set the speed. You can also set an absolute and relative move. When you click those commands, the two axes move to the specified position and the speed is reset to 0
3	In automatic mode, you can set the speed
4	When you click the emergency button, the machine state becomes OFF (see call out 1) and the two axes stop running

Table 10-35: PN Template - Control Panel

Based on the template, the project can be run:

- using the KAS Simulator
- with actual drives and motors (in this case, you first have to set up the axes in the EtherCAT part. For more details, click here...)

9.5.4 PLCopen 2-Axes Template with SFC and FFLD

This project contains two axes where Axis 2 is slaved to Axis 1 at a 2:1 ratio.

9.5.4.1 PLC Programs

The 2-axes PLCopen template has an SFC program (called **Main**) that initializes and starts the motion.

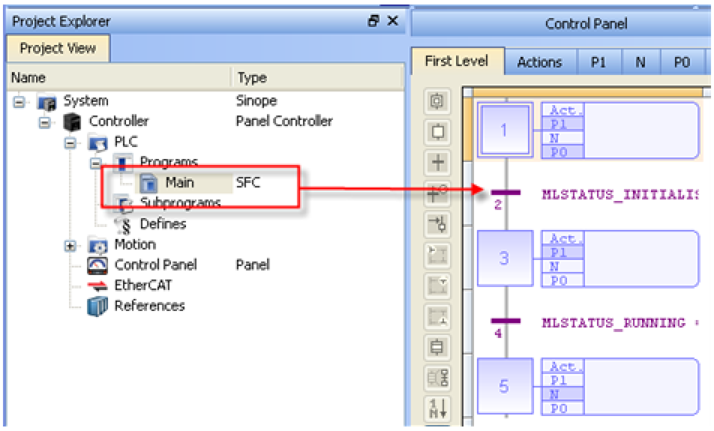


Figure 10-44: PLCopen - Template Main

Step 5 of the Main program in the PLCopen template contains the FFLD code for running the motion. As defined below with the MoveVelocity function block, the motion profile is based on a trapezoidal acceleration/deceleration.

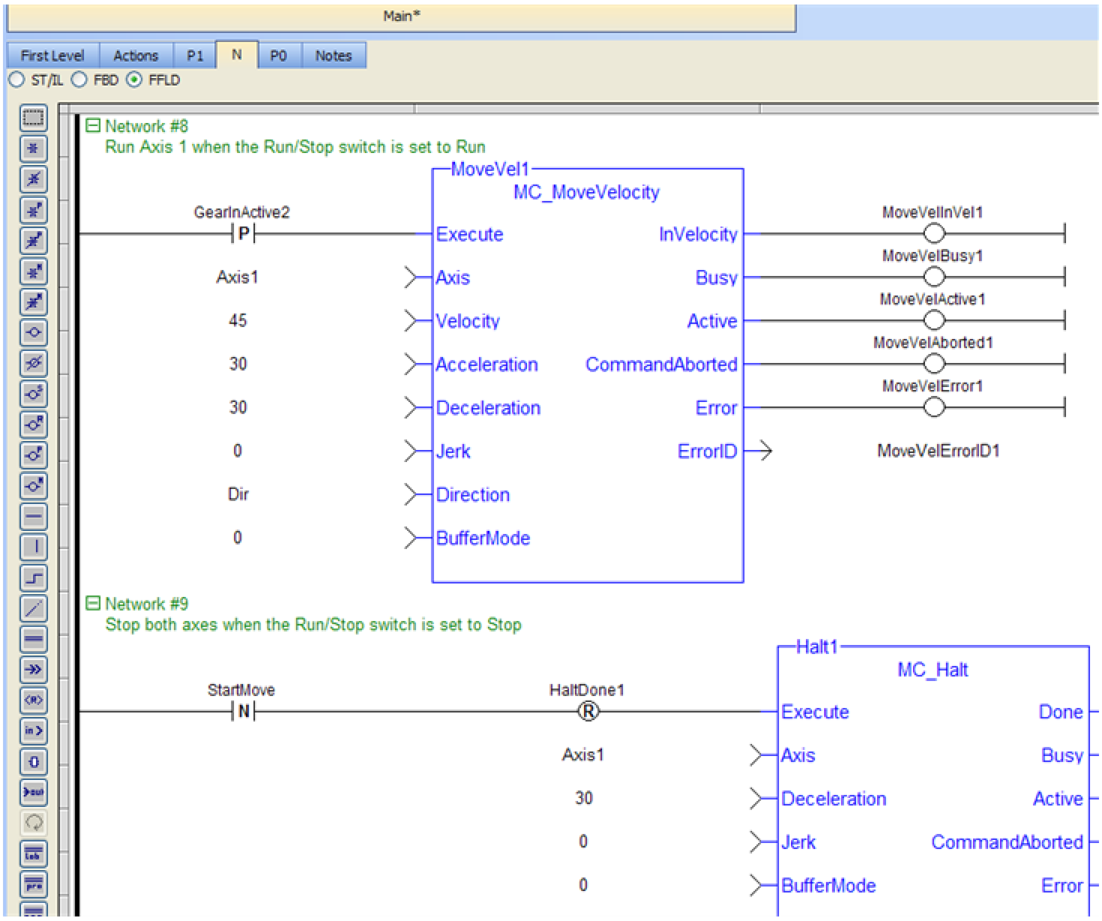


Figure 10-45: PLCopen Template - Step 5 of the Main

9.5.4.2 Motion

The template contains two PLCopen Servo axes where User Units, Update Rate, Rollover Position, and Axis Limits are defined as follows:

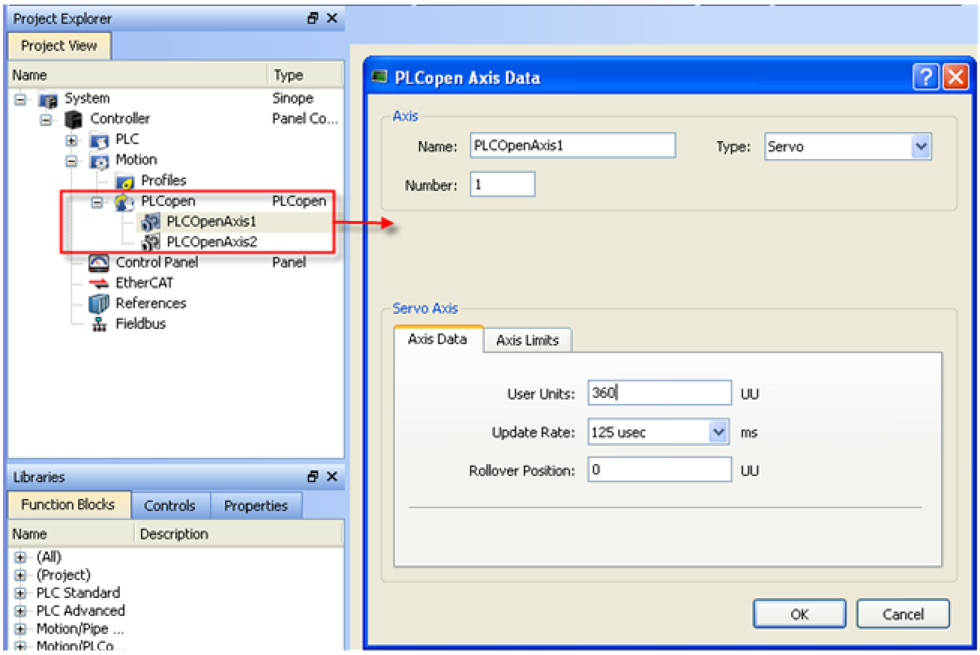


Figure 10-46: PLCopen Template - Motion

For more details on PLcopen axis parameters, see page 294

9.5.4.3 Control Panel

For more details, see "Design the Control Panel with the Internal Control Panel Editor" (see page 439)

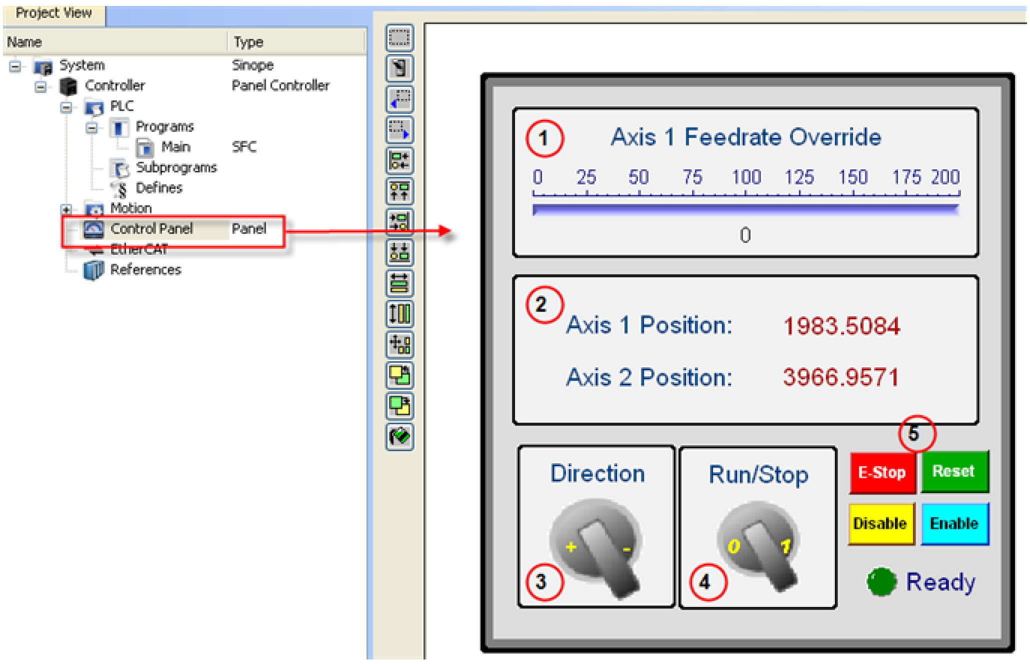


Figure 10-47: PLCopen Template - Control Panel

Call out#	Description
1	Allows you to set the speed
2	Displays the actual position for each axis
3	Select the direction of rotation clockwise (-) or anticlockwise (+)
4	Start or stop the motion on the condition that the axes are enable (the green light must be switched on)
5	Allows to enable or disable the axes. After an emergency stop, you need to select the Reset and Enable commands before running the axes

Table 10-36: PLCopen Template - Control Panel

Based on the template, the project can be run:

- using the KAS Simulator
- with actual drives and motors (in this case, you first have to set up the axes in the EtherCAT part. For more details, click here...)

9.5.5 PLCopen 2-Axes Template with ST

This project contains two axes where Axis 2 is slaved to Axis 1 at a 2:1 ratio.

9.5.5.1 PLC Programs

The 2-axes PLCopen template has a ST program (called **Main**) that initializes, starts and runs the motion.

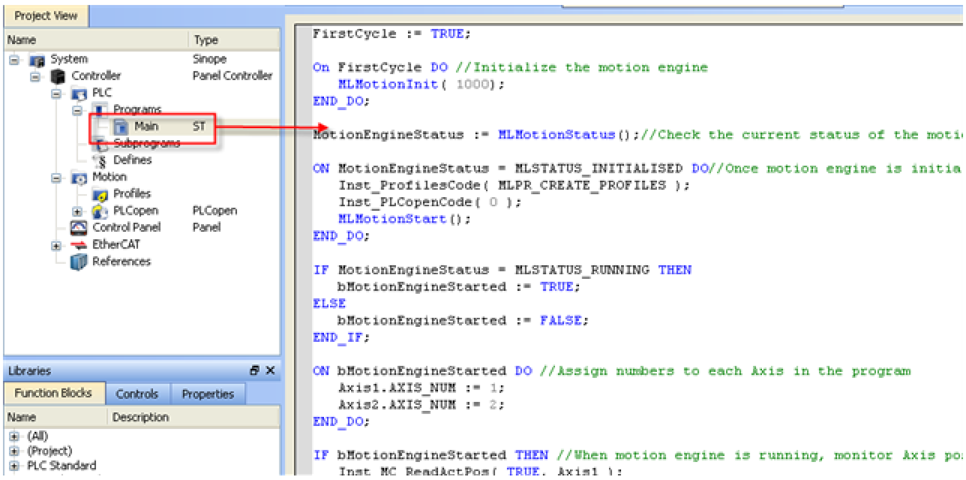


Figure 10-48: PLCopen Template with ST - Main

9.5.5.2 Motion

The template contains two PLCopen Servo axes where User Units, Update Rate, Rollover Position, and Axis Limits are defined as follows:

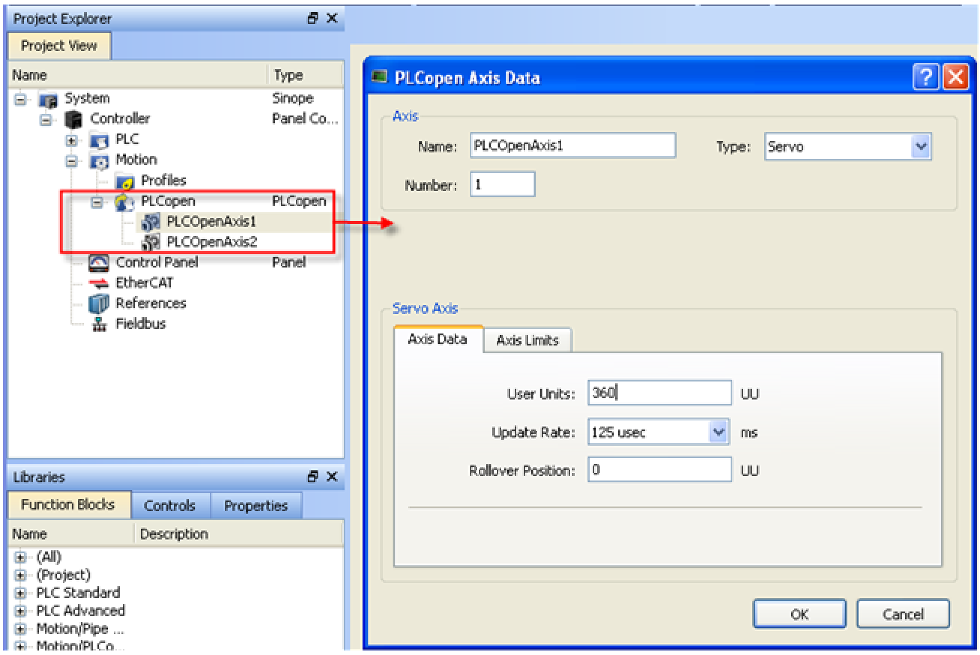


Figure 10-49: PLCopen Template - Motion

For more details on PLcopen axis parameters, see page 294

9.5.5.3 Control Panel

For more details, see "Design the Control Panel with the Internal Control Panel Editor" (see page 439)

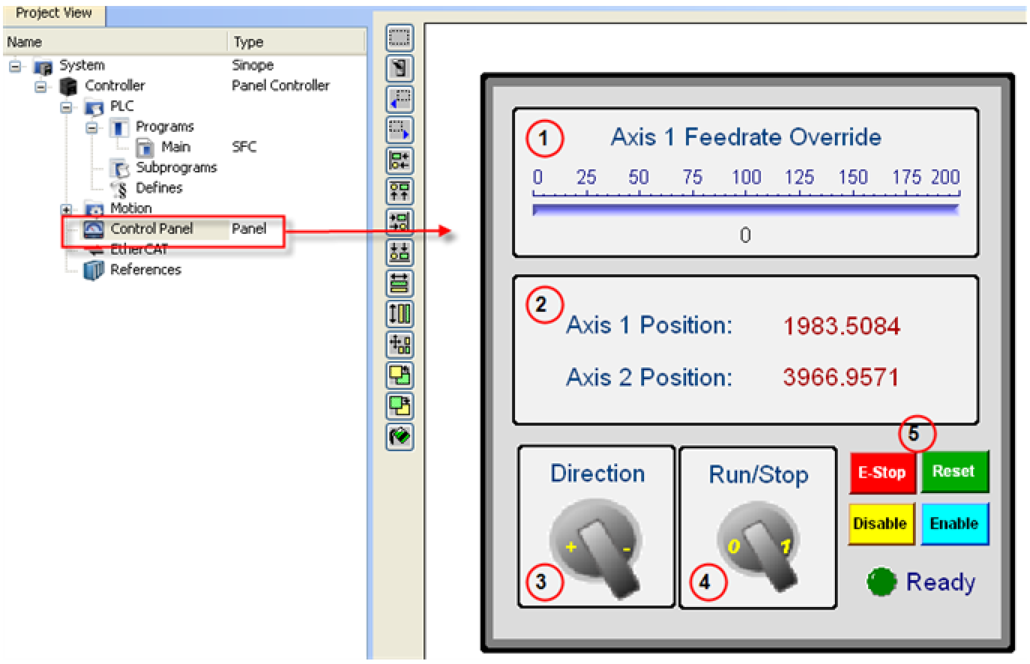


Figure 10-50: PLCopen Template - Control Panel

Call out#	Description
1	Allows you to set the speed
2	Displays the actual position for each axis
3	Select the direction of rotation clockwise (-) or anticlockwise (+)
4	Start or stop the motion on the condition that the axes are enable (the green light must be switched on)
5	Allows to enable or disable the axes. After an emergency stop, you need to select the Reset and Enable commands before running the axes

Table 10-37: PLCopen Template - Control Panel

Based on the template, the project can be run:

- using the KAS Simulator
- with actual drives and motors (in this case, you first have to set up the axes in the EtherCAT part. For more details, click here...)

9.5.6 PLCopen 2-Axes Template with FFLD

This project contains two axes where Axis 2 is slaved to Axis 1 at a 2:1 ratio.

9.5.6.1 PLC Programs

The 2-axes PLCopen template has a FFLD program (called **Main**) that initializes and starts the motion.

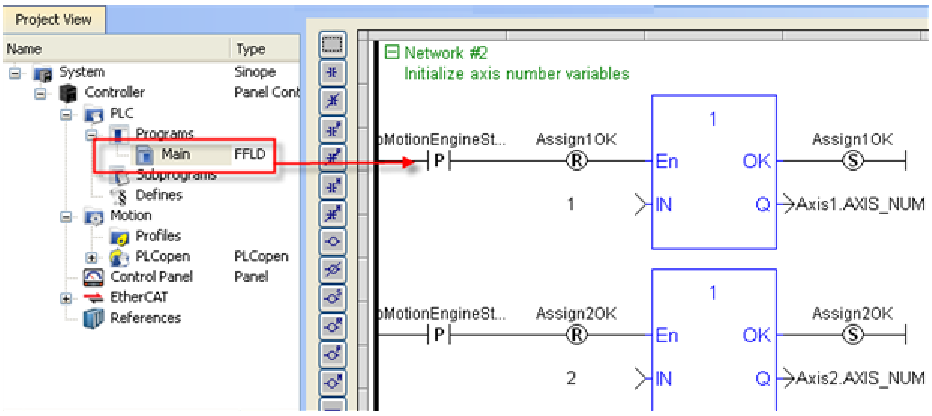


Figure 10-51: PLCopen Template with FFLD - Main

9.5.6.2 Motion

The template contains two PLCopen Servo axes where User Units, Update Rate, Rollover Position, and Axis Limits are defined as follows:

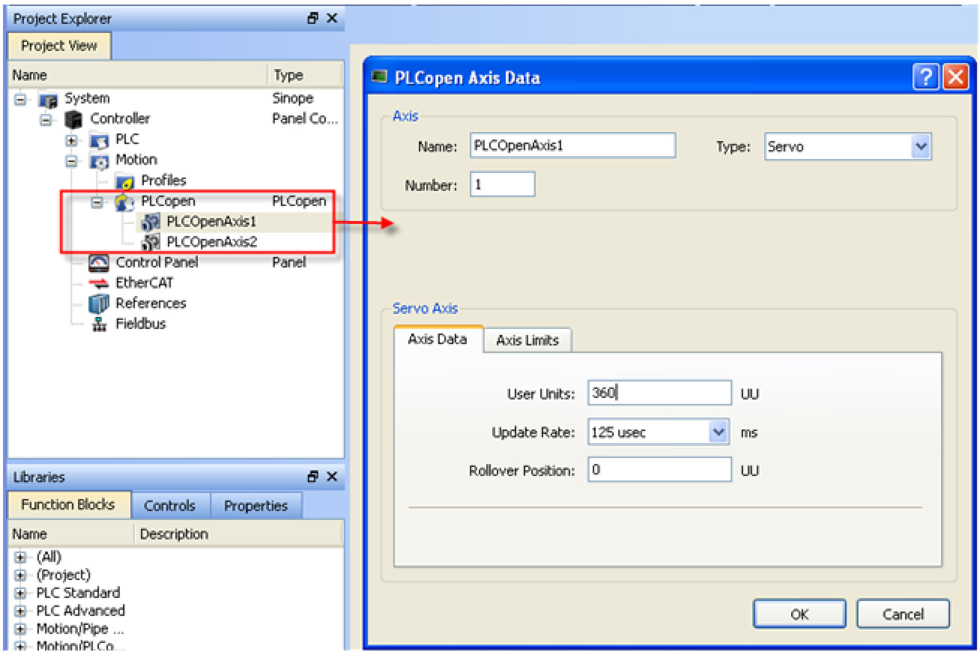


Figure 10-52: PLCopen Template - Motion

For more details on PLcopen axis parameters, see page 294

9.5.6.3 Control Panel

For more details, see "Design the Control Panel with the Internal Control Panel Editor" (see page 439)

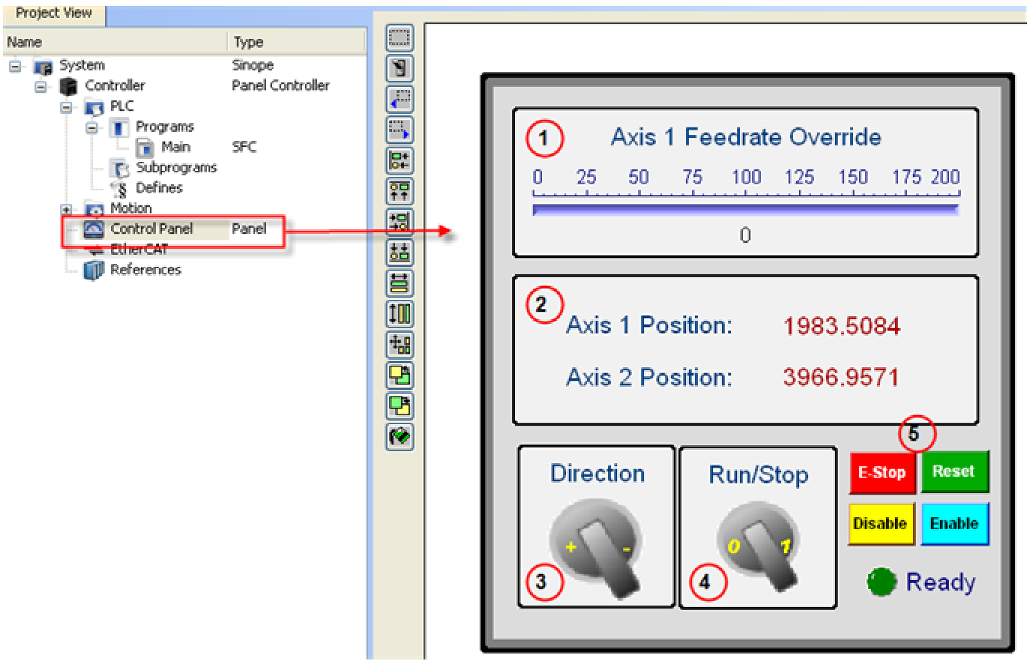


Figure 10-53: PLCopen Template - Control Panel

Call out#	Description
1	Allows you to set the speed
2	Displays the actual position for each axis
3	Select the direction of rotation clockwise (-) or anticlockwise (+)
4	Start or stop the motion on the condition that the axes are enable (the green light must be switched on)
5	Allows to enable or disable the axes. After an emergency stop, you need to select the Reset and Enable commands before running the axes

Table 10-38: PLCopen Template - Control Panel

Based on the template, the project can be run:

- using the KAS Simulator
- with actual drives and motors (in this case, you first have to set up the axes in the EtherCAT part. For more details, click here...)

9.5.7 Coordinated Motion 2-Axis Template

This project controls two axes in coordinated motion (PLCOpenAxis1 and PLCOpenAxis2). This template demonstrates ___.

9.5.7.1 Programs

The program is Sequential Function Chart (SFC) containing both Structured Text (ST) and Free Form Ladder Diagram (FFLD) code.

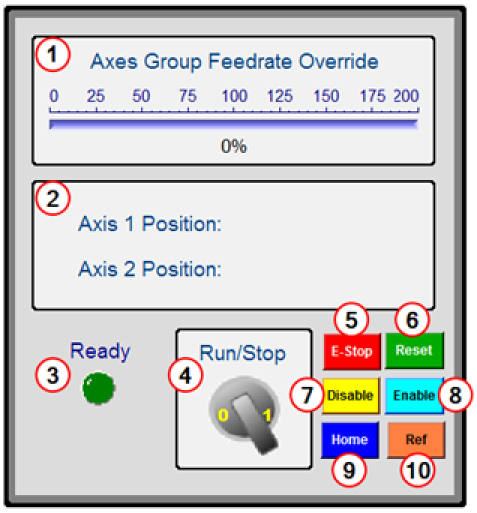
The first five steps of the SFC create and initialize the axes and the coordinated motion axes group. Step six of the SFC interfaces with the Control Panel and performs a back-and-forth coordinated motion pattern with the two axes. The program utilizes coordinated motion direct moves, linear moves, circular moves, transitions and blending.

9.5.7.2 Motion

To run the coordinated motion pattern, perform the following steps:

1. Download and start the application.
2. Press the “Enable” button to enable the axes and the axes group.
3. Press the “Home” button to move the axes to their zero position. (optional)
4. After the “Ready” light is illuminated, turn the “Cycle Start” switch to “1” and the axes will begin moving in programmed pattern.

9.5.7.3 Control Panel

#	Description	GUI
1	Allows you to change the coordinated motion feedrate from 0% to 200%	
2	Displays the axis positions	
3	Indicates that the three axes are enabled and the axes group is ready and enabled for coordinated motion	
4	Begins the coordinated motion pattern	
5	Causes an Emergency Stop for all axes. The axes will be disabled.	
6	Resets the Emergency Stop and enables the axes	
7	Disables the two axes	
8	Enables the two axes and the axes group	
9	Moves all axes to their zero position	
10	Causes a reference by setting the axes' position to zero at their current location.	

9.5.8 Coordinated Motion 3-Axis Template

This project controls two axes in coordinated motion (PLCOpenAxis1 and PLCOpenAxis2), and a third independent axis (VERTICAL_AXIS). This template demonstrates how to use coordinated motion PLCopen axes and a Pipe Network axis.

9.5.8.1 PLC Programs

The Coordinated Motion 3-Axis template has a Sequential Function Chart program (SFC) containing both Structured Text (ST) and Free Form Ladder Diagram (FFLD) code.

The first five steps of the SFC program create and initialize the axes and the coordinated motion axes group plus the Pipe Network axis.

Step 6 of the SFC interfaces with the Control Panel and performs a back-and-forth coordinated motion pattern with the two axes. The program utilizes coordinated motion direct moves, linear moves, circular moves, transitions and blending. It also performs basic moves for the third (Pipe Network) axis, to move down/up before and after the coordinated motion pattern.

9.5.8.2 Motion

This template uses both motion engines (Pipe Network and PLCopen) simultaneously.

IMPORTANT Coordinated motion can only be performed with PLCopen axes, Pipe Network axes do not support this feature. As this template demonstrates, PLCopen axes that perform coordinated motion can be mixed with independent Pipe Network

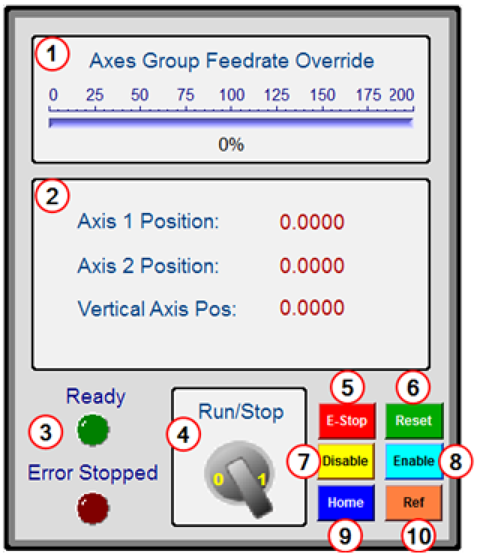
IMPORTANT axes.

There is no axis synchronization at the Motion Engine level between a PLCopen axis and a Pipe Network axis. Any synchronization between the axes must be performed inside the PLC application.

To run the complete motion pattern:

1. Download and start the application.
2. Press the “Enable” button to enable the axes and the axes group.
3. Press the “Home” button to move the axes to their zero position. (optional)
4. After the “Ready” light is on, turn the “Cycle Start” switch to “1” and the axes will begin moving in the programmed pattern.

9.5.8.3 Control Panel

#	Description	GUI
1	Allows you to change the coordinated motion feedrate from 0% to 200%	
2	Displays the axis positions	
3	Indicator lights. Ready indicates that the three axes are enabled and the axes group is ready and enabled for coordinated motion. Error Stopped indicates that the axes group has been stopped and disabled by the E-Stop button.	
4	Begins the coordinated motion pattern	
5	Causes an Emergency Stop for all axes. The axes will be disabled.	
6	Resets the Emergency Stop and clears axis errors.	
7	Disables the three axes	
8	Enables the three axes and the axes group	
9	Moves all axes to their zero position	
10	Causes a reference by setting the axes' position to zero at their current location.	

This page intentionally left blank.

10 Describing KAS Graphical User Interface

NOTE

For KAS Simulator GUI, refer to chapter "Using the KAS Simulator" on page 345
For AKD drive GUI View, refer to paragraph "AKD Drive" on page 645

10.1 Windows and Panels Overview

10.1.1 Main Window

The KAS IDE interface provides an all-in-one-window integrated workspace.

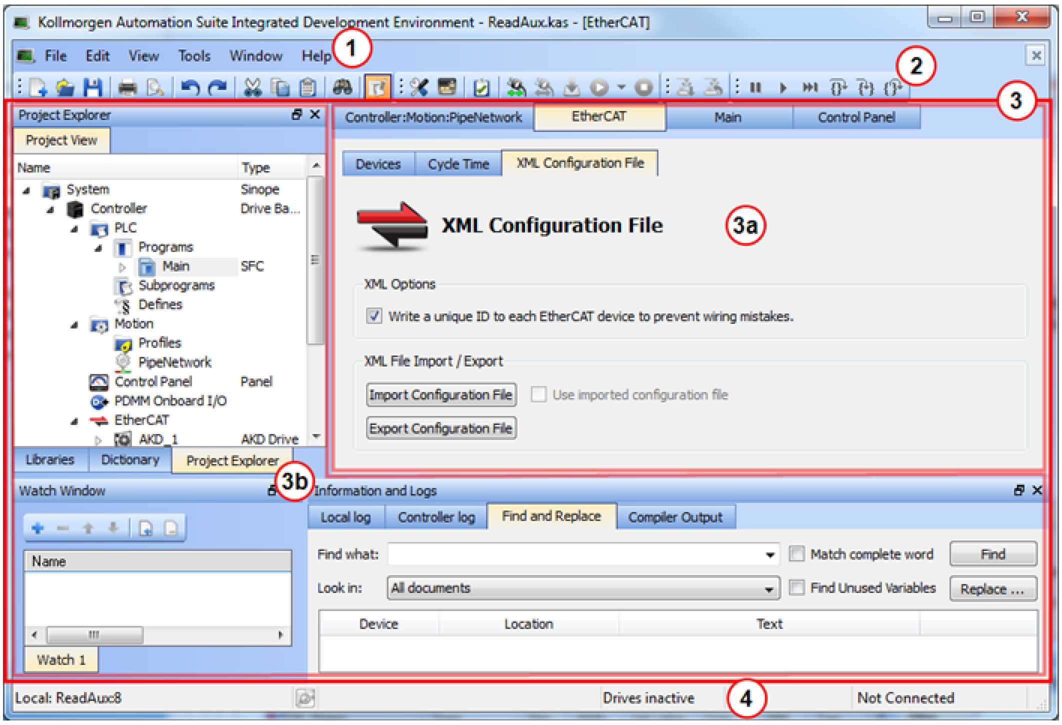


Figure 11-1: KAS IDEMain Window

The main view in the Integrated Development Environment (IDE) is a Multiple Document Interface (MDI) environment. This provides an easy-to-use and customizable view; including the capability to hide, enlarge or overlap windows in order to optimize visibility.

The main view is saved when you exit the application. This ensures that your workspace remains the same each time you open and use the KAS IDE.

The KAS IDE main window contains the following items:

- Menu bar (see call out 1)
- Toolbar 2 A toolbar is a little bar with icons which is usually located under the menu bar of a window.
- Workspace 3 which contains:
  - A specific area dedicated to displaying the workspace children windows 3a
  - Several toolboxes 3b A toolbox is a child window that provides you with some functions to perform specific tasks.
- Status bar at the bottom 4 displaying the current state of the target

10.1.1.1 About toolboxes

The available toolboxes include:

- "Project Explorer" (see page 593)
- "Libraries" (see page 604)
- "Dictionary" (see page 605)
- "Information and Logs" (see page 626)

ⓘ TIP You can hide/show each toolbox and toolbar directly from the contextual menus in any title bar (i.e. menu, toolbar or toolboxes).

10.1.2 Project Explorer

The Project Explorer toolbox is a window that displays machine application information in a tree-structure representation. This window contains all the following items used to design, implement, test, and document the application.

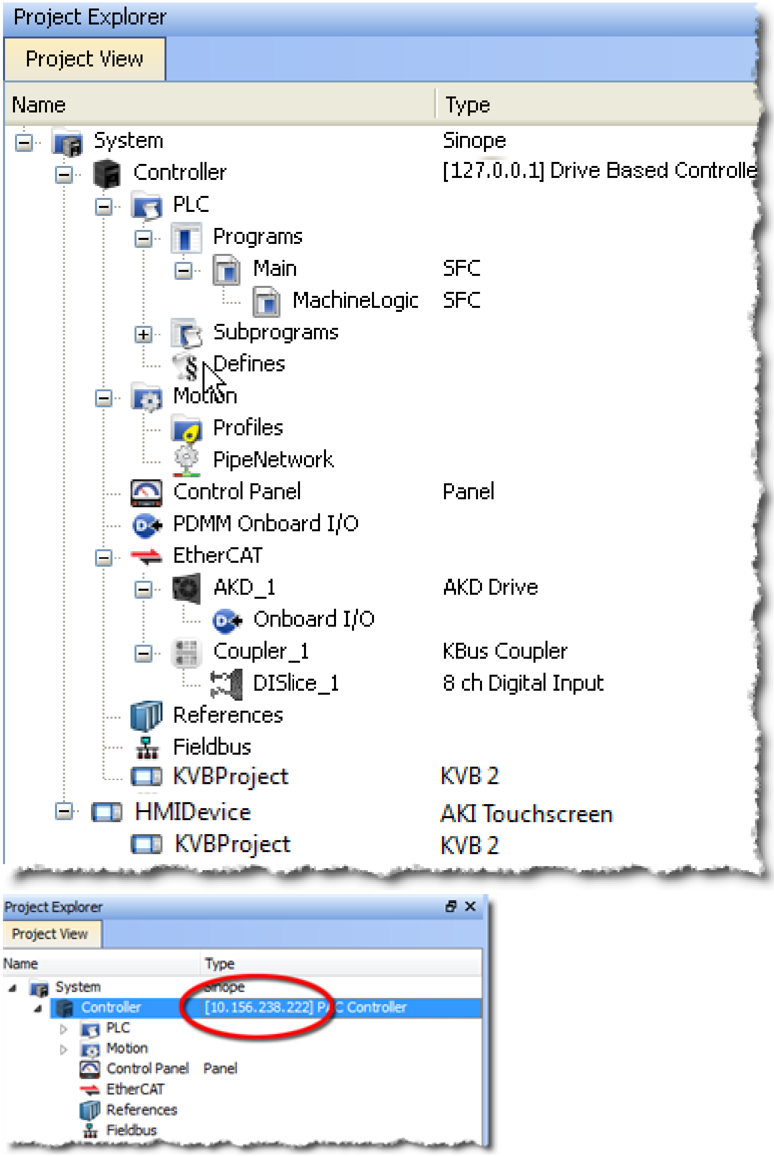


Figure 11-2: Project Explorer, PDMM and PAC versions.

Item	Description
Hardware	• Devices that make up the system such as Controllers, EtherCAT Motion Bus, servo and stepper drives, HMI devices, I/O Terminals, etc.

Item	Description
PLC (IEC 61131-3)	• Programs that control the system • User-defined Functions and Function Blocks
Motion	• Pipe Networks or PLCopen • Axis objects • Cam profiles

TIP You can navigate in the project-tree by entering the item's initial letter, or by means of the arrow keys.

A project is made of several items that are:

- "System" (see page 594)
- "Controller" (see page 594)
    - "PLC" (see page 596)
        - "Programs" (see page 597)
        - "Subprograms" (see page 598)
        - "Defines" (see page 598)
    - "Motion" (see page 598)
        - "Profiles" (see page 598)
        - "Pipe Network" (see page 599) or "PLCopen" (see page 600)
    - "Control Panel" (see page 600)
    - "AKD PDMM Onboard I/O" (see page 600)
    - "EtherCAT" (see page 600)
        - "AKD Drive" (see page 601)
        - "AKD Onboard I/O" (see page 601)
        - "Standard I/O Coupler" (see page 601)
        - "Device" (see page 602)
    - "References" (see page 602)
    - "Fieldbus" (see page 602)
    - "KVB Project" (see page 603)
- "HMI Device" (see page 602)
    - "KVB Project" (see page 603)

10.1.2.1 System

This item concerns the whole project. A right-click opens its menu that provides the following options:

Command	Description
Add HMI Device	Add a new HMI device with a KVB panel (external from the PAC). For mode details see "HMI Device" (see page 602) below.

Table 11-1: System Node - Contextual Menu

10.1.2.2 Controller

This item contains the controller of the project and displays the current IP. It is also used to "Access the WebServer From the IDE" (see page 603). The webserver functionality may be used directly within the IDE. For more information on the webserver see "Using the KAS Web Server" (see page 375).

NOTE Please note that the IP address is shown as `127.0.0.1` if the system is in

NOTE simulation mode.

Command	Description
Add Control Panel	Add a new contol panel to the controller. For more details see "Control Panel" (see page 600).
Add KVB Project	Add a new KVB panel which is embedded into the contoller. For more details see "KVB Project" (see page 603).
Import KVB Project	Import a compressed ("zipped") KVB project, which may be created in KAS or KVB. The system will validate the compressed KVB project and add the panel.
Import Control Panel	Import a pre-configured control panel for use in the project. See Import a Control Panel for more information.
Add Fieldbus	Add a node to access the Fieldbus Editor. See "Fieldbus Editor" (see page 524) for more information.
Access Web Server	This command opens the web server interface in the GUI. See "Access the WebServer From the IDE" (see page 603) and "Using the KAS Web Server" (see page 375) for more information.

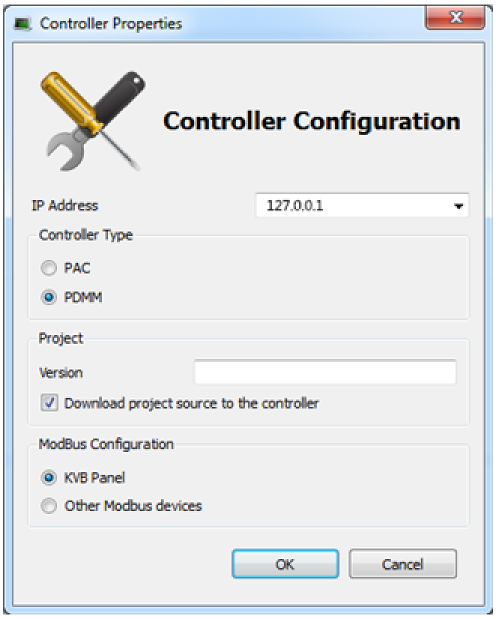
<table>
<tr><th>Command</th><th>Description</th></tr>
<tr><td>Properties</td><td>
Open a dialog box to configure the controller.


Figure 11-3: Configure the Device
Parameters Description

Address IP or COM: allows for connecting to and downloading your application to the controller. Changing this value modifies the IP address for the Controller Type.

The last 10 IPs entered are accessible and are stored with the project. Selecting an item in the list and pressing the 'Delete' key will clear the entry from the list.
When you click  to choose the simulation mode, this address is disabled.

The Controller type can be either PAC or AKD PDMM
See also "Different Implementations" on page 69
NOTE You must select the correct Controller before compiling your application (the PLC code generated for PAC and AKD PDMM have different endianness). A warning is displayed if you try to start your application to the wrong controller.
Version number is used to ensure both versions of your application on the KAS IDE and the KAS Runtime are the same

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

Table 11-2: Controller Node - Contextual Menu

A controller is composed of a PLC item, a Motion item, control panels, an EtherCAT Motion Bus and some References. These items are described in the following sections.

10.1.2.3 PLC

This item contains all the PLC (Virtual Machine) part of the controller. The following items can be present in this item:

- **Program** items
- **Subprogram** items
- Some "**Defines**"

Command	Description
Libraries	Import new libraries

10.1.2.4 Programs

Command	Description
New Program	Add new program items (SFC,ST,FBD, IL or FFLD)
Cycle	Configure the cycle of the virtual machine For mode details on Cycle, see "Define the PLC Cycle" on page 311
Import	Import a saved program

Table 11-3: Program Node - Contextual Menu

Command	Description
Add Child SFC	Add a child program to this program. Note that this is reserved for the first SFC program only.
Import Child SFC	Import a saved SFC program to the current program. Note that this is reserved for the first SFC program only. **How to import all children from one project to another?** 1. Export each program one at a time from the existing project 2. Save the program (specify a location and a name) NOTE: **Do not enter spaces** in the filename even if nothing prevents you from doing it. 3. Close the project 4. Open the project to be updated 5. Import each saved program in the project tree 6. Rename the program if needed NOTE: Only local variables are copied (not the global variables)
Export	Save the selected program to your file server. NOTE: **Do not enter spaces** in the filename even if nothing prevents you from doing it.
Rename	Rename the selected program.
Delete	Delete the selected program.
Print SFC and All Level 2	Print all PLC programs. See "Print" (see page 341) for more details.

Table 11-4: Program Item - Contextual Menu

TIP: You can double-click to open the program in the workspace.

10.1.2.5 Subprograms

Command	Description
New Function (Subprogram)	Add a new subprogram item (ST,FBD,IL or FFLD)
New UDFB	Add a new UDFB item (ST,FBD,IL or FFLD)
Import	Import a saved program

Table 11-5: Subprogram Node - Contextual Menu

You can create your own functions as well as functional blocks that are called UDFBs (User-Defined Functional Blocks). For each of them, you can use the following commands:

Command	Description
Export	Save the selected subprogram onto your file server
Rename	Rename the selected subprogram
Create Unlocked Copy	Duplicate the selected, locked subprogram. The duplicate will not be locked.
Delete	Delete the selected subprogram
In/Out Parameters	Open the Program Properties dialog box to "Declare Functions or Function Blocks" (see page 259). **NOTE** This item is disabled for locked UDFBs.

Table 11-6: Subprogram Item - Contextual Menu

10.1.2.6 Defines

This item contains all the global definitions in the scope of the corresponding device.

TIP You can double-click a **Define** item to show these global definitions. Click here to open a file of internal defines.

See also "Step 8 of 15 - Use the Defines List" on page 262

10.1.2.7 Motion

The motion item contains the motion-specific items (i.e. the Profiles and PipeNetwork items).

Command	Description
Motion Engines	Choose the motion engine for your application between PLCopen and PipeNetwork

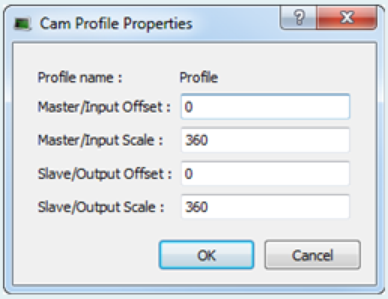
10.1.2.8 Profiles

This item contains all the cam profiles in the project.

Command	Description
New Profile	Create a new cam profile and add it to this device (*.csv, *.cam) For mode details, see page 279
Import	Import already existing cam profiles to your project
Show compiled code	Show the code corresponding to the selected cam profile

Table 11-7: Profiles Node - Contextual Menu

Right-clicking on a cam profile provides additional commands.

<table>
<tr><th>Command</th><th>Description</th></tr>
<tr><td>Rename</td><td>Provide the cam profile a unique name</td></tr>
<tr><td>Delete</td><td>Remove the cam profile from the list</td></tr>
<tr><td>Export</td><td>Save the cam profile in CAM (.cam) format</td></tr>
<tr><td>Properties</td><td>Open a dialog to modify the cam profile's Master/Input Slave/Output Offset and Scale values.

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

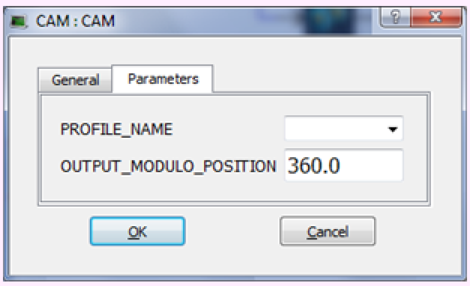
For more information on cam profiles see "Step 13 of 15 - Adding Cam Profiles" (see page 305) and "Cam Profile Editor" (see page 396).

10.1.2.9 Pipe Network

This menu applies to the Pipe Network in the project.

<table>
<tr><th>Command</th><th>Description</th></tr>
<tr><td>Import and replace</td><td>This command replaces the existing Pipe Network with a pre-saved Pipe Network. You will be presented with a dialog box to locate the pre-saved file. The Pipe Network Editor will be opened when the file is imported.

IMPORTANT The existing EtherCAT axis mapping is lost when using this command. Additionally, profiles assigned to Cam blocks are cleared at this time.

1. Double-click on EtherCAT in the Project View to open the "EtherCAT Devices" (see page 196) tab so you can reassign the axes.
2. Double-click on any Cam blocks, and set the Profile_Name parameter.

</td></tr>
<tr><td>Export</td><td>Export the Pipe Network to a file for reuse.</td></tr>
<tr><td>Show compiled code</td><td>Show the code corresponding to the Pipe Network</td></tr>
</table>

10.1.2.10 PLCopen

Command	Description
New Axis	Add a new axis to your project For mode details, see page 289
Show compiled code	Show the code corresponding to the PLCopen

Table 11-8: PLCopen Node - Contextual Menu

For each PLCopen axis you can use the following commands:

Command	Description
Properties	Open a dialog box to configure the PLCopen axis data
Delete	Delete the selected axis

Table 11-9: Axis Item - Contextual Menu

10.1.2.11 Control Panel

This item holds the Control Panel item used to provide a basic interface between you and the virtual machine.

For more details, see page 439

For a more advanced tool to build HMI, see page 603

Command	Description
Rename	Rename the selected Control panel
Delete	Delete the selected Control panel
Export	Export the control panel for use in other projects. See Export a Control Panel.

Table 11-10: HMI Control Panel Node - Contextual Menu

10.1.2.12 AKD PDMM Onboard I/O

Command	Description
Properties	Open the Properties dialog box to configure the local I/O of the AKD PDMM drive See also "Configure AKD PDMM Onboard I/O" on page 370

Table 11-11: AKD PDMM Onboard I/O Item - Contextual Menu

10.1.2.13 EtherCAT

This item gives access to all the devices linked to the EtherCAT Motion Bus.

Command	Description
Add AKD Drive	Add a new AKD drive to the EtherCAT network. See also "Step 2 of 15 - Add and Configure Drive" on page 189 Note that this command is disabled when the controller is running

Command	Description
Add Standard I/O Coupler	Add a new coupler, enabling you to connect I/O terminals. See also "Step 3 of 15 - Add and Configure I/O Terminal" on page 195 Note that this command is disabled when the controller is running
Add Device…	Add a third-party EtherCAT device to the EtherCAT node in the Project view.
Scan Devices	TheKAS Runtime sends EtherCAT messages to discover the devices present in the network See also "EtherCAT Devices" on page 196
Enable/Disable Online Configuration Mode	Toggles Online Configuration Mode on and off. See "Online Configuration Mode" (see page 660) for more information.
Properties	Open the Properties dialog box. See also "Step 4 of 15 - Configure EtherCAT Motion Bus" on page 196

Table 11-12: EtherCAT Node - Contextual Menu

See also "Add Third Party EtherCAT Devices" on page 205

10.1.2.14 AKD Drive

You can double-click an AKD to set its parameters. See also "Configure the AKD Drive" on page 190

Command	Description
Rename	Rename the selected drive
Delete	Delete the selected drive
Configuration	Opens the Configuration tab for the AKD GUI.
Properties	Select the Properties menu to access the EtherCAT device's configuration views.

Table 11-13: AKD Drive Item - Contextual Menu

10.1.2.15 AKD Onboard I/O

Command	Description
Properties	Open the Properties dialog box to configure the local I/O of the AKD drive See also "Configure Onboard I/O" on page 193

Table 11-14: AKD Onboard I/O Item - Contextual Menu

10.1.2.16 Standard I/O Coupler

The Standard I/O Coupler node gives access to its I/O slices.

Command	Description
Add I/O Slice	Add a new slice (Digital or Analog Input and Output) to the selected Standard I/O Coupler
Rename	Rename the selected coupler
Delete	Delete the selected coupler
Properties	Select the Properties menu to access the EtherCAT device's configuration views.

Table 11-15: Standard I/O Coupler Node - Contextual Menu

Note that all those commands are disabled when the controller is running.

See "EtherCAT Coupler Error Handling And Diagnosis" (see page 687) in the "Troubleshooting" (see page 681) section for information about diagnosing the coupler LEDs.

10.1.2.17 I/O Slice

Command	Description
Properties	Open the Properties dialog box to configure the I/O slice See also "Step 11 of 15 - Map Input and Output to Variables" on page 270
Rename	Rename the selected slice
Delete	Delete the selected slice

Table 11-16: I/O Slice - Contextual Menu

10.1.2.18 Device

Double-clicking a Device accesses its EtherCAT device configuration views.

Command	Description
Rename	Rename the selected device
Delete	Delete the selected device
Properties	Access the EtherCAT device's configuration views

Table 11-17: Device - Contextual Menu

10.1.2.19 References

This item allows you to **insert references** into your project. Each reference is a user-defined reference that links any kind of deliverable to your project (for more details, refer to paragraph "Use the Reference Folder" on page 343)

Command	Description
Insert Reference	Link any kind of deliverable to your current project
Delete	Delete the reference
Properties	Open the referenced file in the workspace

Table 11-18: Reference Node - Contextual Menu

10.1.2.20 Fieldbus

This item holds the Fieldbus Editor to configure the Ethernet/IP or Profinet fieldbuses. For mode details, see page 524

10.1.2.21 HMI Device

This item holds the HMI (Human Machine Interface) item used to provide an advanced interface between you and the virtual machine.

Command	Description
Add KVB Project	Add a new KVB panel to the controller. For mode details, see page 315 Note that this command is disabled when a KVB panel already exists

Command	Description
Import KVB Project	Import a compressed ("zipped") KVB project, which may be created in KAS or KVB. The system will validate the compressed KVB project and add the panel.
Rename	Rename the selected HMI device
Delete	Delete the selected HMI device

Table 11-19: HMI Device Node - Contextual Menu

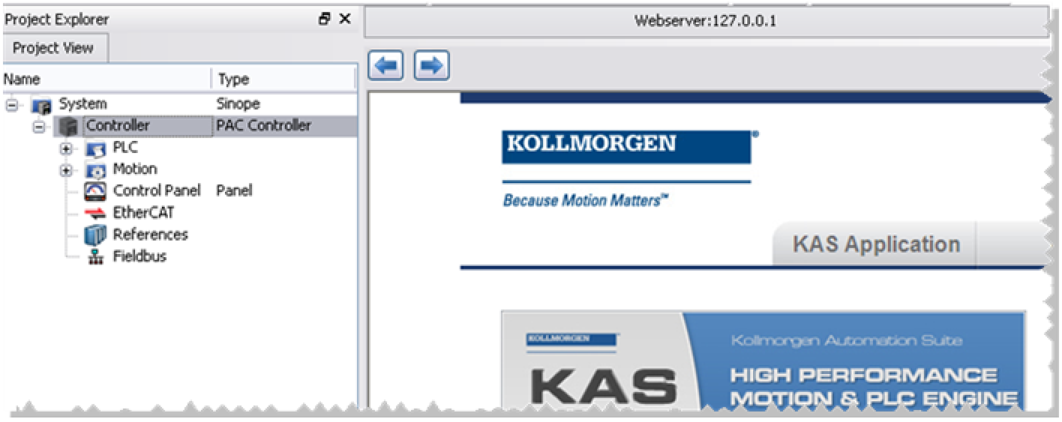
10.1.2.22 KVB Project

Command	Description
Rename	Rename the selected KVB panel
Delete	Delete the selected KVB panel
Export	Save a copy of the panel in a compressed (.zip) file.

Table 11-20: KVB Panel Node - Contextual Menu

Access the WebServer From the IDE

Double-clicking **Controller** will both expand/collapse the Controller's components as well as open the web server and automatically log into the *administrator* account. For more information on using the webserver see "Using the KAS Web Server" (see page 375).

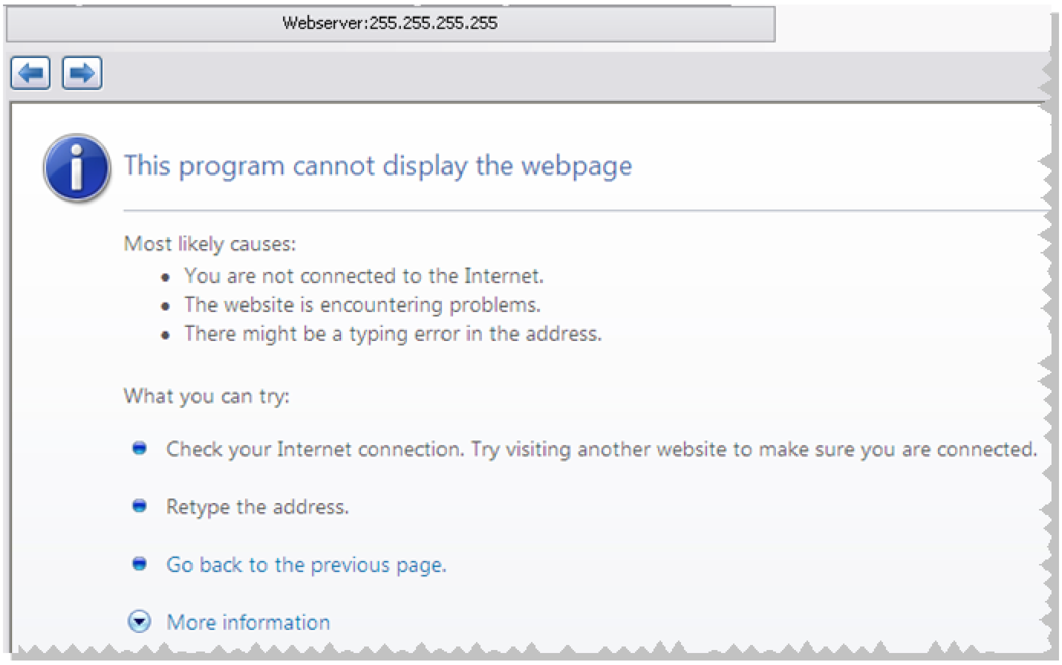


The web server can also be accessed by right-clicking the Controller node and selecting **Access webserver**.

> **TIP** IDE users are considered administrators and therefore are automatically logged into the webserver. You may change the password but there is no logout function.

By default the localhost (127.0.0.1) will be opened. To set the IP address of the controller, right click and select Properties. Enter the proper **Address** and **Controller type** then click **OK**. The page is automatically refreshed.

If an invalid or wrong IP address is entered, the following error will be displayed.



10.1.3 Libraries

This toolbox contains several tabs to access all the functions of the available libraries.

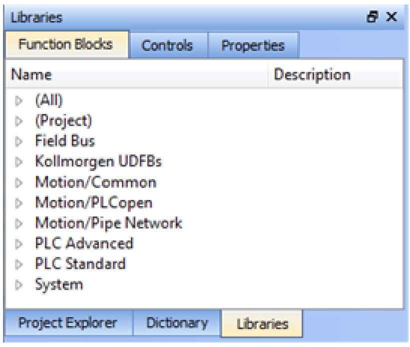


Figure 11-4: Libraries Toolbox

10.1.3.1 Function Blocks

This tab displays all the available libraries shown in a tree-structure representation and gathered by categories. You can expand a library to access all its functions. A short description of each function is also available.

The **(All)** category at the top enables you to see the full list of available functions sorted in alphabetical order.
The **(Project)** node contains all the UDFB and subprograms associated to the current project.

For more details about these libraries, refer to the following libraries description:

- PLC Standard
- PLC Advanced
- Motion/Pipe Network
- Motion/PLCopen
- Field Bus

- System
- Kollmorgen UDFBs

TIP It is possible to use the functions, UDFB or subprograms in PLC editors with a simple drag-and-drop operation.

NOTE Dragging and dropping a Kollmorgen UDFB into the "Defines" editor has no effect. However, if the Kollmorgen UDFB is already imported to the project, then it’s prototype will be seen in the editor.

10.1.3.2 Controls

This tab displays all the controls available for the HMI design.
For more details, refer to the Graphic Objects description.

10.1.3.3 Properties

This tab displays all the properties of an HMI control currently selected in the HMI editor.

More information about setting the properties of an HMI widget can be found in paragraph "Graphic Objects Properties" on page 449.

10.1.4 Dictionary

The Dictionary toolbox is used to show all the identifiers (variables, data types, sub-routines, etc.) defined within the project. There are three tabs within the Dictionary, the "Variables tab" (see page 605), the "Enum Tab" (see page 622), and the "Bit Fields Tab" (see page 624).

10.1.4.1 Variables tab

The Variables tab is used to show all the variables defined within the project. All the variable details are displayed in order to show the variable types, dimensions, attributes, etc.

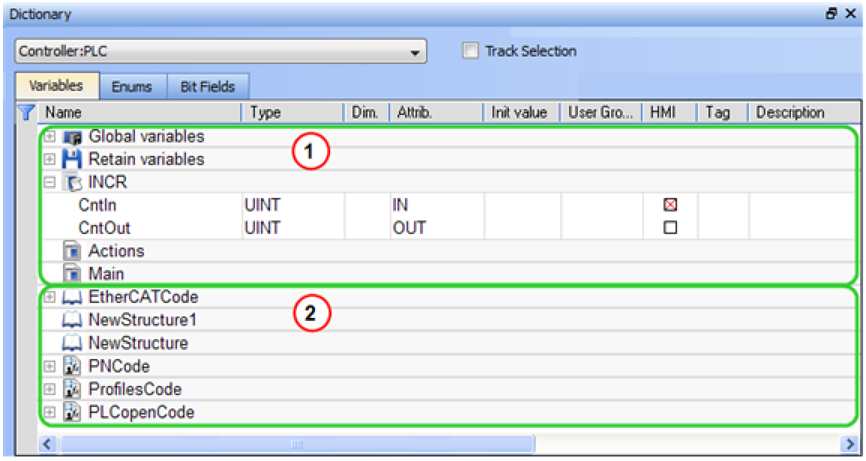


Figure 11-5: Dictionary Toolbox

The list of variables is split into two parts:

- All the **"Variables" (see page 609)** at the top 1
- All the **"Structures" (see page 611)** at the bottom 2

NOTE For more information about the procedure to create an instance of a structure,

NOTE see "Call Functions or Function Blocks" on page 262

TIP To show all the variables of all programs, select '**PLC**' in the project tree.

About the Dictionary's contextual menu.

Right-click in the Dictionary window to open the menu as follows:

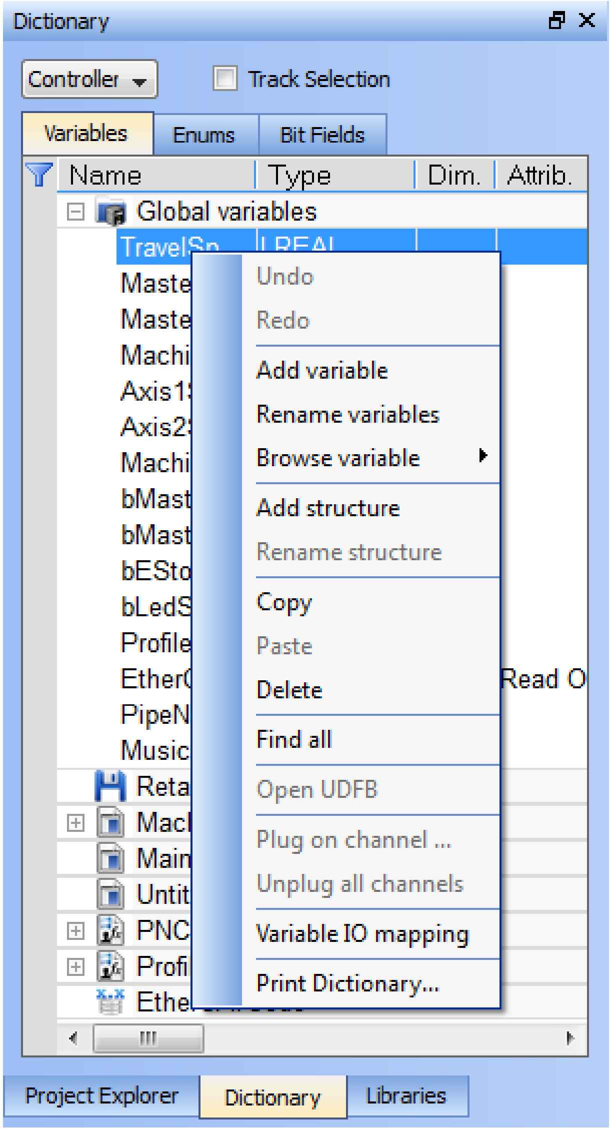


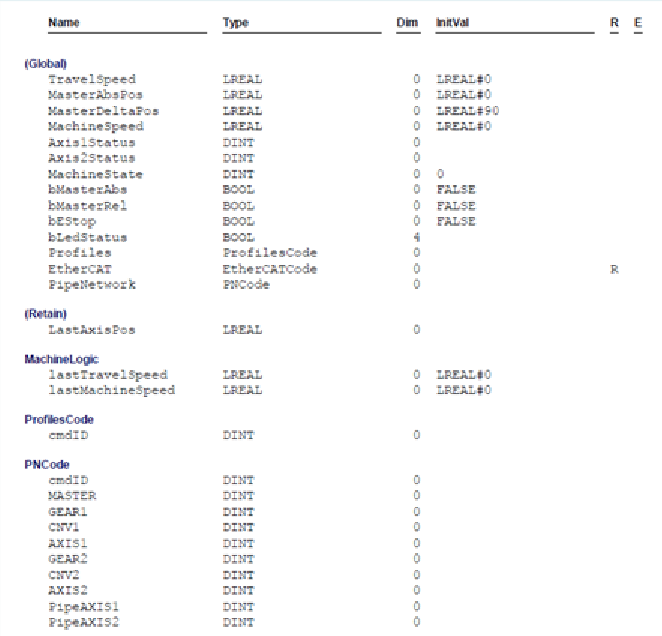
Figure 11-6: Dictionary Contextual Menu

This menu allows you to perform the following actions:

Command	Description
Undo	Undo the last action performed on the Dictionary
Redo	Redo the last undone action

Command	Description
Add Variable	Add a new variable in the selected level (Global, Retain, program). This automatically creates a new variable called NewVar with type BOOL For a Function or UDFB, you can specify input and output parameters (for more details, see "Define Parameters and Private Variables" on page 261)
Rename variables	This function can either replace a section of matching variables or append text to the variables' names. See "Rename Variables" (see page 615)
Browse variable	This function allows you to browse all instances of a variable. See "Browse Variable Tab" (see page 635) for more information.
Add Structure	Used to have a new complex type. A structure named NewStructure is created and variables can be dragged into it (for more details, see "Complex Structures" on page 256)
Rename Structure	Rename the selected structure
Copy	Copy a variable
Paste	Paste the copied variable to the selected level
Delete	Delete the selected variable. A deletion can also be performed by pressing the Delete key on the keyboard
Find all	This function will find all instances of the specified variable and open the results in the Find and Replace tab.
Open UDFB	Open the selected UDFB instance (for more details, see page 338)
Plug On Channel ...	Plug the selected variable on a channel. This command opens a dialog used to configure the variable plug operation. **NOTE** This command is enable when your application is connected and running, and if the type of variable is eligible for the softscope (i.e. BOOL, INT, SINT, DINT, LINT, UINT, USINT, UDINT, ULINT, BYTE, WORD, DWORD, LWORD, TIME and LREAL, as long as they are not in a UDFB instance)
Unplug All Channels	Unplug all plugged probes from the softscope
Variable I/O mapping	Connect a variable to an I/O.

Command	Description
Print Dictionary ...	Print all the variables displayed in the Dictionary and sorted by programs. The columns display the Name, Type, Dimension, Initial Value, and Attributes



Name	Type	Dim	InitVal	R	E
(Global)					
TravelSpeed	LREAL	0	LREAL#0		
MasterAbsPos	LREAL	0	LREAL#0		
MasterDeltaPos	LREAL	0	LREAL#90		
MachineSpeed	LREAL	0	LREAL#0		
Axis1Status	DINT	0			
Axis2Status	DINT	0			
MachineState	DINT	0	0		
bMasterAbs	BOOL	0	FALSE		
bMasterRel	BOOL	0	FALSE		
bEStop	BOOL	0	FALSE		
bLedStatus	BOOL	4			
Profiles	ProfilesCode	0			
EtherCAT	EtherCATCode	0		R	
PipeNetwork	PNCode	0			
(Retain)					
LastAxisPos	LREAL	0			
MachineLogic					
lastTravelSpeed	LREAL	0	LREAL#0		
lastMachineSpeed	LREAL	0	LREAL#0		
ProfilesCode					
cmdID	DINT	0			
PNCode					
cmdID	DINT	0			
MASTER	DINT	0			
GEAR1	DINT	0			
CNV1	DINT	0			
AXIS1	DINT	0			
GEAR2	DINT	0			
CNV2	DINT	0			
AXIS2	DINT	0			
PipeAXIS1	DINT	0			
PipeAXIS2	DINT	0			

What is the purpose of the Track Selection check box?

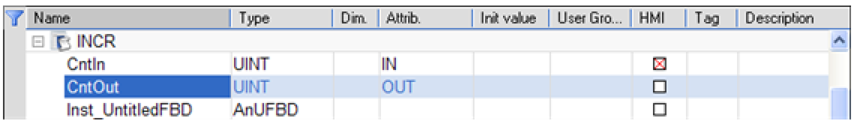
The **Track Selection** filters the displayed items in the dictionary to show only items linked to the current PLC selected program:

- **Unchecked**: All your project variables will be displayed. This is the default setting.
- **Selected**: The variables in the Dictionary are filtered to display only those that are relevant to the PLC item currently selected in the project tree. Along with the Global, retains and variables related to the selected program or UDFB, structure definitions will be displayed. The dictionary content will change accordingly if another PLC program is selected in the project tree.

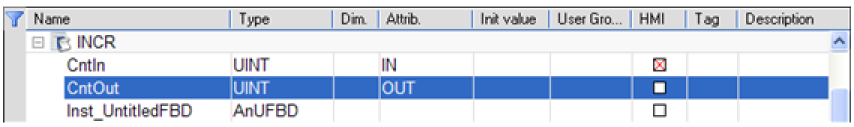
About the two editing modes for editing a variable.

There are two available modes when editing a variable in the Dictionary:

- **Cell**: only the selected cell is active



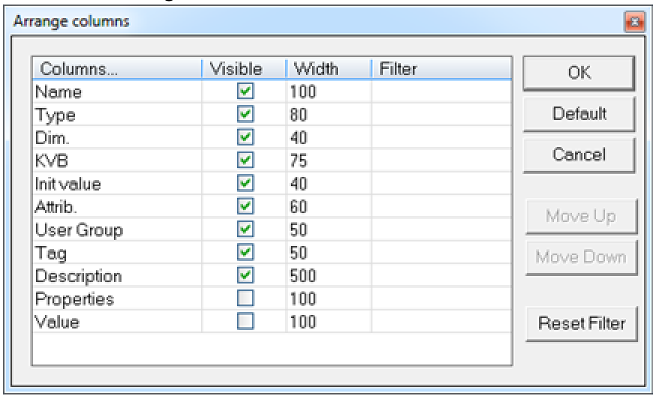
- **Row**: all the row is active



Press **Spacebar** to toggle the selection mode from cell to row (See also "Table Shortcuts" on page 671).

How can variables be arranged and/or sorted?

- The columns in the Variables tab can be reordered, resized, and hidden by double clicking on the filter icon in top left corner of the table. This opens a dialog box which allows you to modify the table's appearance. Please note that the Visible box for the Value field cannot be changed as the Value column is automatically shown in debug and online modes but hidden in edit mode.



- You can sort the list of variables in the table as follows:
  - Ensure you are in **cell** editing mode (press the **Spacebar** to toggle from one mode to the other)
  - Click the header of the column you want to use as the key sort order

How to modify parameters of a variable?

(Press **Spacebar** to toggle to the relevant edition mode).

Mode	Description
One Parameter	Assuming you are in the **cell** edition mode, double-click on the parameter
All the parameters are at the same time	Assuming you are in the **row** edition mode, double-click in any parameter to open the dialog box for variable configuration as shown below. For more details on parameters, see "Variables" on page 609.

NOTE It is not possible to modify a variable when the KAS IDE is connected to the controller.

Variables

All **variables** within the entire system project are grouped as follows:

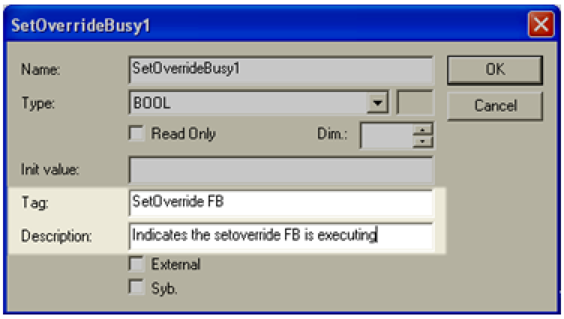
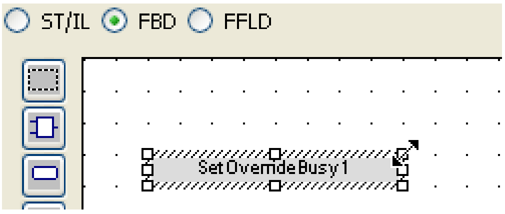
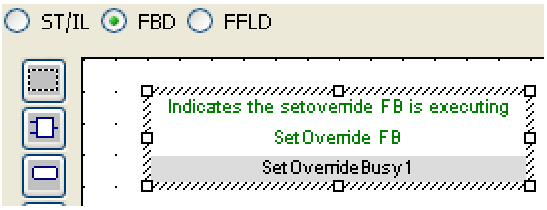
Variable	Description
Global variables	List all global variables that are used and accessible throughout the entire program
"Retain Variables" (see page 79)	List all variables that are to be retained when the system is powered down
Program variables	List the variables related to your specific selected program

For each variable, the Dictionary toolbox allows you to set the following parameters:

Field	Description
Name	The variable name

Field	Description
Value	All the variables in the Dictionary are animated with real-time values [1] Note that this column is only displayed when your application is running For more details, see "Variable Monitoring" on page 338
Type	The variable type (which can also be UDFB or complex structure)
Dim.	To declare an array, you can specify dimension(s) for an internal variable
Attrib.	The variable attributes (Read Only, External, IN, OUT) as defined below • Read Only: a variable set as Read Only is a constant (it cannot be modified in your PLC code, but it can be forced manually). Read Only variables can be mapped to Outputs but not to Inputs. This is because Inputs can change state and a Read Only variable would not be able to change its value to match the input state. • External: this attribute is not used • IN or OUT: Input or Output parameters of User Defined Function Blocks
Init value	The variable initial value when you start your application (see more details here)
User Group	The variable user group (used for sorting variables)
HMI	Select variables to be used in HMI (see procedure)

[1] To better track variables in Running mode, the KAS IDE dynamically computes their value along with the application execution and display the result in this column.

Field	Description
Tag	The variable tag is a short comment, that can be displayed together with the variable name in graphical editors. Edit the variable parameters  Add the variable to your FBD program  Resize the rectangle to make the Tag and Description visible 
Description	The variable description is a long comment text that describes the variable
Syb.	reserved

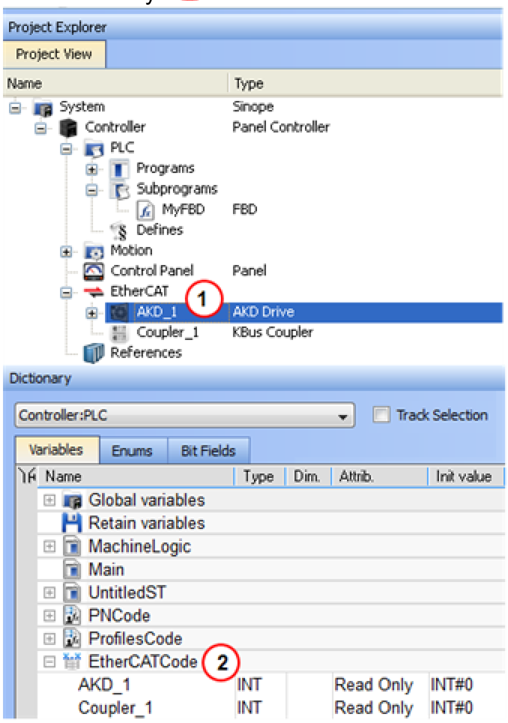
Structures

All the **structures** within the entire system project are grouped as follows:

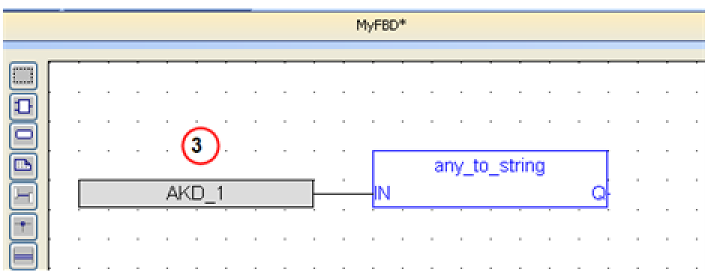
<table>
<tr><th>Structure</th><th>Description</th></tr>
<tr><td>EtherCATCode</td><td>List all the devices as PLC variables so you can use their names in FBs instance

• The devices are displayed under the EtherCAT node in the Project Explorer (see call out 1)

• The same devices are also displayed as PLC variables in the Dictionary 2



• You can use them in your PLC programs with a drag-and-drop operation 3



</td></tr>
<tr><td>PNCode</td><td>List all the variables related to the specific Pipe Network Code</td></tr>
<tr><td>ProfilesCode</td><td>List all the variables related to the specific profiles</td></tr>
<tr><td>PLCopenCode</td><td>List all the variables related to the specific PLCopen Code</td></tr>
<tr><td>UDFB variables</td><td>List the variables related to a specific UDFB</td></tr>
</table>

Variable editor

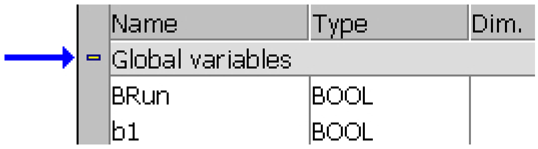
Variables are declared in the Dictionary of the KAS IDE main window.

The variable editor is a table that enables you to declare all variables of the application. Variables in the editor are sorted by groups:

- global variables
- "retain" non-volatile global variables
- I/O variables (each I/O device is a group)
- variables local to a program (including in and out parameters in case of a UDFB).

Please refer to the description of variables in the language reference for a more detailed overview.

Each group is marked with a gray header in the variable list. The "**-**" or "**+**" icon on the left of the group header can be used to expand or collapse the group:



See how to:

- Create New Variables
- Use the Variable Table List
- Define Structures
- Set Bookmarks

Create new variables

Press the INSERT key in the variable editor to create a new variable in the selected group. The variable is added at the end of the group. Variables are created with a default name. You can rename a new variable or change its attribute by using the Variable Editor.

TIP	You cannot insert a new variable in an I/O group.

In case of a group corresponding to local variables of a UDFB, pressing the INSERT key gives you the choice between:

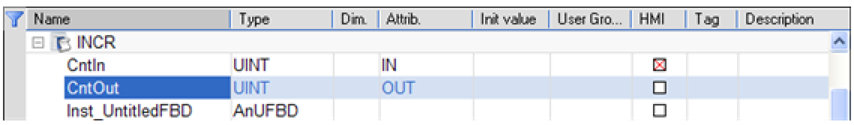
- adding an "IN" (input) parameter
- adding an "OUT" (output) parameter
- adding a private variable

IN and OUT parameters always appear at the beginning of a UDFB group.

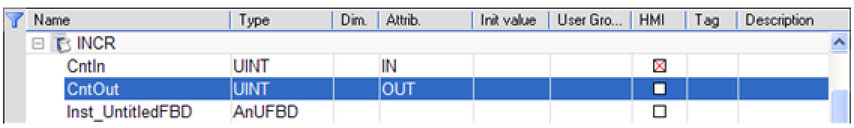
Variable Table List

There are two available modes when editing a variable in the Dictionary:

- **Cell**: only the selected cell is active



- **Row**: all the row is active

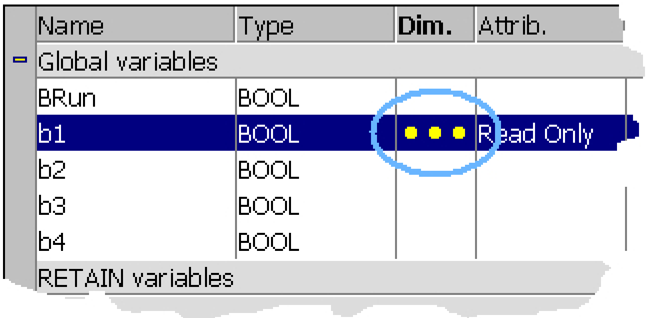


Press **Spacebar** to toggle the selection mode from cell to row (See also "Table Shortcuts" on page 671).

When the selection mode is on cell, the variable editor enables you to enter each piece of information directly in the cell.

Double-click or press the ENTER key to open the dialog box.

When the active grid is active, the name of the selected column is displayed in bold characters. The text of selected cell (or ". . ." if empty) is marked in bold yellow characters:



At any time you can drag with the mouse the column separators in the main grid header for resizing columns.

Press the following keys for browsing groups of variables:

`Ctrl + Page Up`	Move the selection to the head of the previous group
`Ctrl + Page Down`	Move the selection to the head of the following group

For Tables manipulation, see also paragraph "Windows Standard Conventions" on page 662

Sort variables

At any moment you can sort variables of a group according to their name, type or dimension. To do this, you simply need to:

1. Move the cursor to the header of the group
2. Click on the name of the column you want to sort

The KAS IDE always keeps the original order of declared variables, to allow safe online change. Each time you insert a new variable or expand/collapse a group, the original sorting is re-applied.

Define structures

To create a new type of data structure, use the "Add structure" command.

For more details of the full procedure, refer to paragraph "Complex Structures" on page 256

Each structure is represented as a group in the dictionary grid. Enter the members of the structure in its group in the same way you enter variables in another group.

New data structures are created with default names. Use the "Rename structure" command to change its name.

Use the "Move Structure Up / Down" commands in the "Edit" menu to organize the list of data structures.

If a member of a structure is an instance of another structure, the nested structure must be declared BEFORE in the list.

Name a variable

To change the name of the variable, do as follows:

1. In the Name column of the table, select the cell you want to edit
2. Press ENTER (or press the first character of the new name)
3. Enter the name in the small box
4. Press ENTER to validate the name or ESCAPE to cancel the change

A variable must be identified by a unique name within its parent group. The variable name cannot be a reserved keyword of the programming languages and cannot have the same name as a standard or "C" function or function block. A variable must not have the same name as a program or a user-defined Function Block.

The name of a variable must begin by a letter or an underscore ("_") mark, followed by letters, digits or underscore marks. It is not allowed to put two consecutive underscores within a variable name. Naming is case-insensitive. Two names with different cases are considered as the same.

Naming Physical I/Os

Each I/O channel has a predefined symbol that reflects its physical location. This symbol begins with "**%I**" for an input and "**%Q**" for an output, followed by a letter identifying the physical size of the data. Refer to the description of variables for more details.

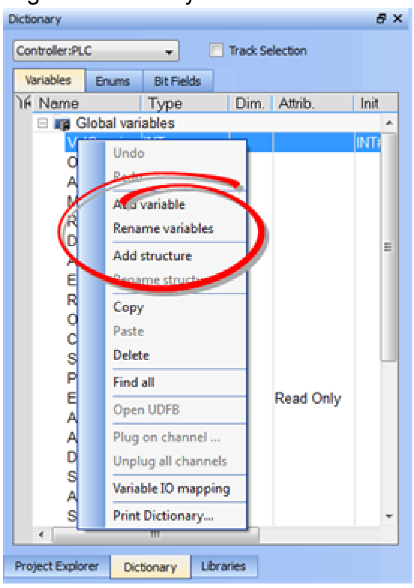
You cannot change the "%..." name of an I/O variable. This name is directly allocated according to the I/O devices defined in the I/O device list. But you can give an alias (a readable name) to each I/O channel. In that case, either the "%" name or the alias can be used in programs. The alias must fit to the same rules as a variable name.

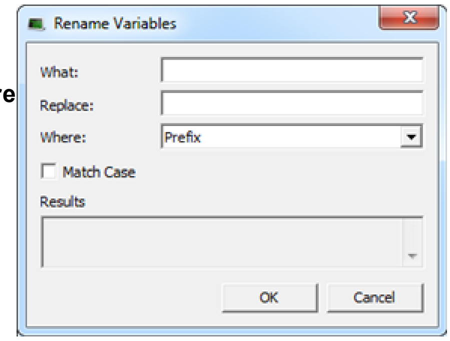
When an alias is defined for a variable, both "%..." name and alias are displayed in the "name" column of the grid.

Rename Variables

Variables may be renamed from within the Dictionary. The renaming function can either replace a section of the name or append text to variable names. The search will find and replace matches first within the Dictionary and will then continue the search within programs in the current project. You will have the option to propagate the changes to programs or not.

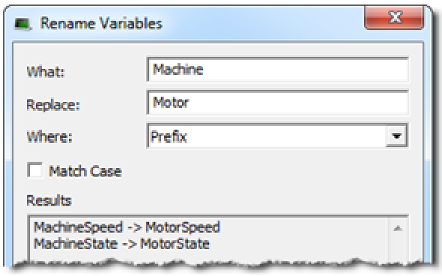
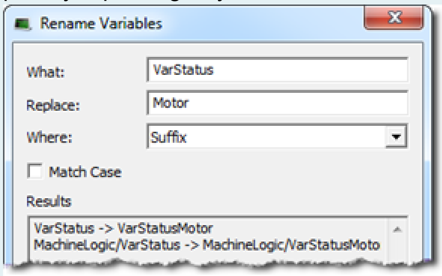
1. Right-click on any variable in the Dictionary and select **Rename Variables**.

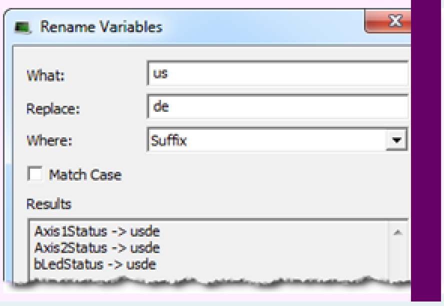
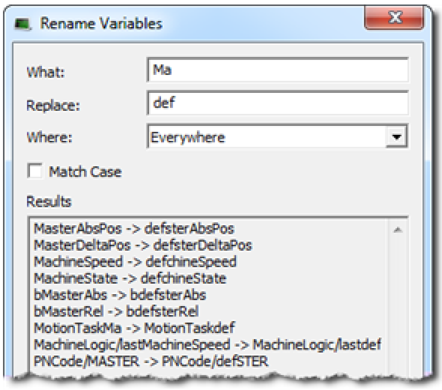




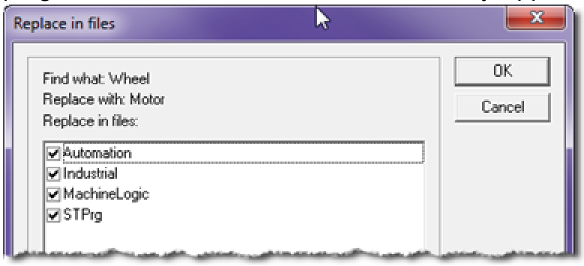
2. Enter the phrase to be replaced in **What**.
3. Enter the new phrase in **Replace**.
4. Set the replacement method using the **Where** menu.
5. Click OK to make the changes shown in the **Results** frame.
6. The **Replace in files** dialog box opens if there are matches within programs. Select which programs to modify.

Item	What it means
What	Enter the variable name/text which is to be replaced
Replace	New text which will replace current variable name text

<table>
<tr><th>Item</th><th>What it means</th></tr>
<tr><td>Where</td><td>How the replace will occurs
<table>
<tr><td>Prefix</td><td>The search/replace starts from the first character in the variable name.

</td></tr>
<tr><td>Suffix</td><td>The "Replace" text will be appended to the "What" text, completely replacing any matches.


IMPORTANT Please be aware that this search method completely replaces the search string. Use this method with caution.

</td></tr>
<tr><td>Everywhere</td><td>Searches the entire variable name.

</td></tr>
</table></td></tr>
<tr><td>Match Case</td><td>The search will be case sensitive</td></tr>
<tr><td>Results</td><td>The search results and proposed changes are displayed.</td></tr>
</table>

When matches exist within programs the **Replace in Files** dialog box displays the matches and lets you choose which programs to modify. Once **OK** is clicked, the programs will be modified in the order they appear in the list.



Initial Value of a Variable

A variable can have an initial value. The value must be a valid constant expression that fits to the data type of the variable. The initial value is displayed in red if it is not a valid expression for the selected data type.

There is no initial value for arrays and instances of function blocks.

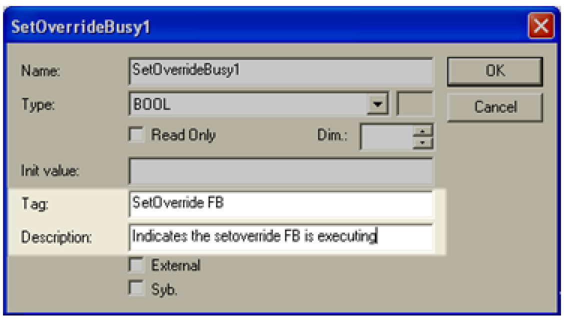
You can change the initial value of a variable by using the Variable Editor.

Variable Tag and Description

For each variable, the KAS IDE enables you to enter in the dictionary two strings that describe the variable:

- The "**Tag**" is a short comment, that can be displayed together with the variable name in graphic languages.
- The "**Description**" is a long comment text that describes the variable.

To change the tag or description of a variable, enable the modification mode to Row and move the cursor to the corresponding cell. Then press ENTER to enter the new text.



I/O devices


Press this button in the main window to open the I/O device editor.

The I/O device editor is run is a separate box. It is used for declaring and setting up I/O devices, and establish the link between the application variables and physical equipment.

The list shows the possible slot numbers between 0 and 255. Select a slot and:

- Hit ENTER for selecting or changing the type of I/O device to be put on this slot. In the selection box, I/O devices are sorted by categories. Select the "All" choice for displaying the full list of available devices. The description note of the selected device is displayed in the selection window. Double-click on a device or hit ENTER to select it. Press ESCAPE to cancel the operation.
- Run "Edit / Rename" menu command to change the name of the device. You can freely give any name to each I/O device.

- Hit Alt+ENTER or run "Edit / Properties" menu command to setup the physical properties of the device. Refer to OEM instructions for detailed explanation about I/O device properties.
- Hit **Spacebar** to set the selected I/O device as "Virtual" or "Real" (normal). A virtual device is disconnected from physical operations and is managed as group of internal variables at run-time Using virtual devices enables you to test your application even if the actual hardware is not available. Virtual devices are marked in blue and between parentheses in the device list:

There can be either simple or complex I/O devices. A simple I/O device is a group of I/O channel. All channels of the group have consistent data types, the same direction (input or output), and are numbered from 0. A complex I/O device is a list of simple devices, and generally represents a mixed type/direction equipment.

Variable properties

The KAS IDE enables you to embed in the application code extra information for each variable. Run the "Edit / Properties" when a variable is selected in the grid to edit its properties in a separate box. You also can set the "View / Properties" menu option to display variable properties in one more column in the grid.

Publishing properties

Select the "Publishing" tab to enter the pieces of information you want to embed in the target application and publish for extra embedded software. For each variable, you can embed:

- its symbol
- a numerical tag (a number between 1 and 65535)
- a profile name
- a list of OEM defined properties

The list of properties is entered in the grid at the bottom of the box, and corresponds to the selected profile. Refer to OEM instructions for further description of available profiles.

To change a value in the property list, double-click on a line, or hit the first character of the value. Press ENTER to validate a value or ESCAPE to cancel the change.

Editing variables as text

As an alternative to the user friendly grid for editing variables, it is possible to declare variables as text. Text editing applies to all the variables of a group. It cannot be an I/O group. During text editing, the group and all its variables are locked in the grid so that no change can be entered from other windows.

To edit a group of variable as text, select the group in the grid and run the "Tools / Edit variables as text" menu command.

Sereval syntaxes are available for describing variables:

IEC 61131-3	The original IEC 61131-3 syntax for declaring variables
XML tags	An easy XML structure using tags and attributes
CSV	CSV format (separator: semicolon)

Editing variables as XML tags

You can describe variable using a simple XML structure, where each variable is described as an XML tag. The file must fit the baisc XML syntax. Values of tag attributes mus be entered between double quotes. Characters < > " ' & are reserved to XML and cannot appear in values of tag attributes. Instead you should use the following sequences:

```
<   &lt;
>   &gt;
```

```
"   &quot;
'   &apos;
&   &amp;
```

Below is the tag structure for variable declaration:

```
<k5project>
  |
  +-<vargroup>
      |
      +-<var>*
          |
          +-<varinfo>*
```

(the "*" mark indicates that the tag can appear 0 or more times)

The rest of this page describes the format and meaning of each tag:

<k5project>

This tag must be entered at the top level and is unique. It is reserved for extensions (enhancement of the XML structure), and specifies the version of the syntax. Its attributes are:

version Reserved for future extensions.
This attribute is mandatory and must be be "1.0".

The `<K5Project>` tag contains one `<vargroup>` tag.

<vargroup>

This tag must appear with the `<K5Project>`, and contains all `<var>` tags for variables of the group. In this version, the tag has no attribute (the name of the group is implicit)

<var>

This tag describes the basic definition of one variable. Its attributes are:

name Symbol of the variable.
This attribute is mandatory.
type Name of the data type of the variable
This attribute is mandatory
len Maximum length if the data type is STRING.
This attribute is mandatory for STRING variables, and should not appear for other data types.
dim Dimension(s) if the variable is an array.
There are at most 3 dimensions, seperated by comas.
This attribute is optionnal.
attr Attributes of the variable, separated by comas. Possible values are:
IN : this is an INPUT parameter (for UDFBs only)
OUT : this is an OUTPUT parameter (for UDFBs only)
external : this is an external variable
constant : variable is read only
This attribute is optional.
init Initial value of the variable
Must be a valid constant expression that fits the data type
This attribute is optionnal

The `<var>` tag contains zero or more `<varinfo>` tags.

<varinfo>

This tag indicates an additional info for the variable it belongs to. Its attributes are:

type Type of information contained in the "**data**" attribute.
Possible values are:
tag : variable tag (short comment)
desc : description
profile : name of the embedded profile
embed : set of embedded properties
This attribute is mandatory.
data Data specified y the "**type**" attribute, in text format.
This attribute is mandatory

Editing variables as text in CSV format

Using CSV format, each variable is defined on one line of text. Each component of the variable definition is entered as one CSV element. CSV elements are separated by semi-colons. Each element is written between double quotes. A double quote within an element is represented by two double quotes. CSV format is an easy way to exchange variable declaration with Spreadsheet applications.

It is not mandatory that all elements (all columns) appear in the text. The first line must contain the list of columns used, using the following keywords:

name variable symbol — this item is mandatory
type name of the data type — this item is mandatory, and must appear before len, dim and init columns
len string length if the data type is STRING — this item must be empty for other data types
dim dimensions in case of an array — there are at most 3 dimensions, separated by comas
attr attribute of the variable, can be:
IN : input parameter of a UDFB
OUT : output parameter of a UDFB
external : extern variable
RO if "**YES**" indicates that the variable has the read-only attribute — (note: you can also use "TRUE" or "1" value)
init initial value of the variable — must be a valid constant expression that fits the data type
tag tag (short description text)
desc description text
profile name of the embedded profile
embedembedded properties (same syntax as displayed in the variable editor grid)

Below is an example of CSV text for the declaration of 3 variables, with some columns missing:

```
"name","type","len","attr","RO"
"MyVar","BOOL","","","NO"
"ExtVar","DINT","","external","YES"
"MyStr","STRING","10","","NO"
```

Editing variables as text using IEC 61131-3 syntax

Using IEC61131-3 syntax, variables are declared within structured blocks. Each blocks begins with "VAR", "VAR_INPUT", "VAR_OUTPUT" or "VAR_EXTERNAL" keyword and ending with "END_VAR" keyword (with no semicolon after). Below is the meaning of each keyword:

VAR	Memory variables. Can be global, local or retain depending on the edited group
VAR_INPUT	Input parameters of a block. Available only when the edited group is a UDFB.
VAR_OUTPUT	Output parameters of a block. Available only when the edited group is a UDFB.
VAR_ EXTERNAL	External variables. Can be global or local depending on the edited group

Basic syntax for declaring a variable:

To declare a variable, simply enter its symbol, followed by ":" and its data type. If the data type is STRING, it must be followed the maximum length between parentheses. Example:

```
MyVar : BOOL;
MyString : STRING(255);
```

To indicate that a variable has the "read only" attribute, insert the "`CONSTANT `" keyword at the beginning of the variable declaration:

```
CONSTANT VarName : DataType;
```

To declare an array, the data type must be preceeded by "ARRAY [dimensions] OF". There are at most 3 dimensions, separated by comas. Each dimension is specified as "0 .. MaxBound". Below are examples:

```
Array1 : ARRAY [0 .. 99] OF DINT;
Matrix : ARRAY [0 .. 9, 0 .. 9, 0 .. 9] OF REAL;
```

Additionaly, you can specify an initial value for single variables. The initial value is entered after the data type, and is preceeded by ":=". The initial value must be a valid constant expression that fits the data type. Examples:

```
MyBool : BOOL := TRUE;
MyString : STRING(80) := 'Hello';
MyLongReal : LREAL := lreal#1.0E300;
```

Additional information and description texts:

As a variable may have additional properties and comment texts in the KAS IDE, we use special directives entered as IEC comments AFTER the declaration of the variable, to specify additional info. The following directives are available:

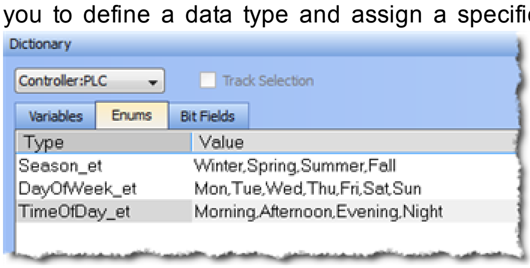
(*$tag=*Text****)**	Variable tag (short comment)
(*$desc=*Text****)**	Variable description
(*$profile=*ProfileName****)**	Variable embedded profile
(*$embed=*Text****)**	Variable embedded properties (the syntax is the one shown in the variable grid, in the "Property" column)

You can also use "//" single line comments to enter the directives:

```
//$tag=Text
//$desc=Text
//$profile=ProfileName
//$embed=Text
```

10.1.4.2 Enum Tab

This tab allows you to define enums (Enumerated Types). An enumerated type allows you to define a data type and assign a specific set of accepted values.



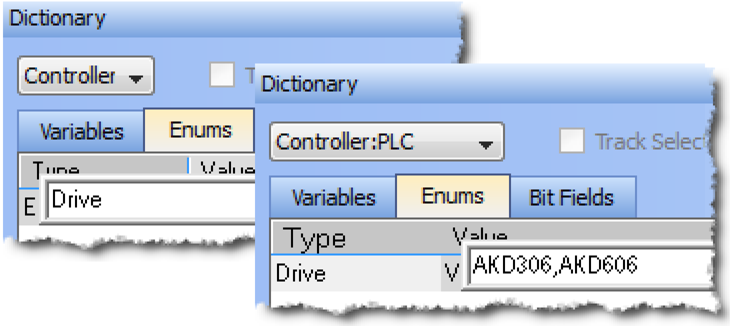
Adding Enums

An enum may be created in one of two ways:

- Right-click within the Enum tab and select **Insert Enum**.
- Press the **Insert** key while the Enum tab is active.

This creates a default enum labelled as "`EnumType`*n*" with the Value of "`V0,V1`".

Double click on the Type or Value to make changes.

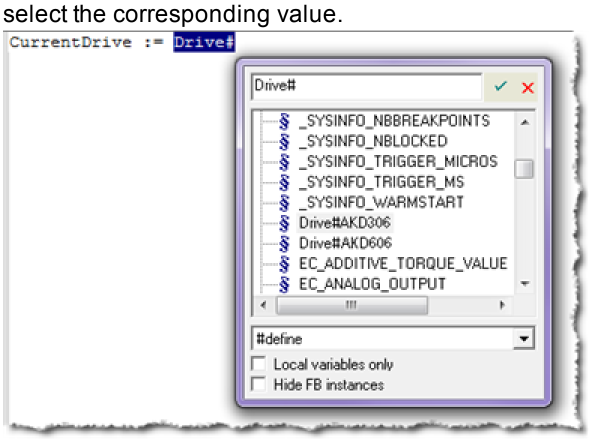


NOTE

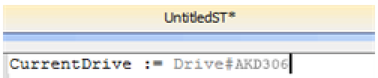
- An enum must contain two or more values. If only one value is defined, it will be not be considered and the previous value will be retained.
- An enum should not contain special characters such as #, @, etc.. If a special character is defined in the value field, it will be not be considered and the previous value will be retained.

To Use Enums

1. Enter the enum type and hash (#), the press Ctrl+Space. This opens a dialog to select the corresponding value.



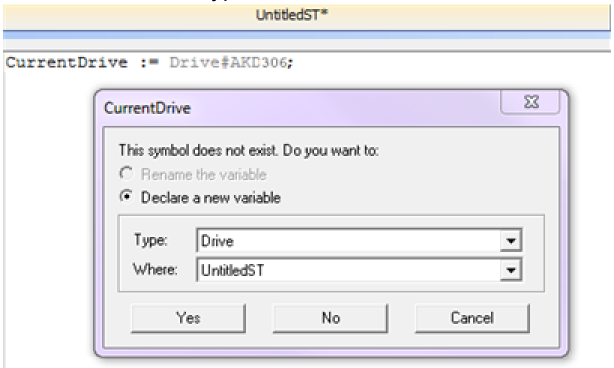
2. Select the value and click the check mark. The value is loaded into your program.



Declaring Enums

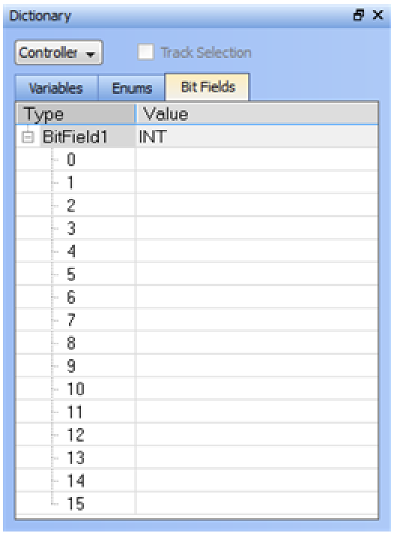
Enums may be declared in the same way as variables; at the end of the line press Enter.

As CurrentDrive is not a part of the dictionary, a dialog is opened to add the variable. Set the Types as the enum name.



Once this is added, the variable will be displayed with the selected type in the Dictionary.

10.1.4.3 Bit Fields Tab



A bit field packs multiple pieces of data together in one variable. Each field represents one piece of data. Each piece of data should have no dependency upon other fields. Bit Fields are used to define custom variable types and values.

Adding Bit Fields

A Bit Field may be created in one of three ways:

- Right-click within the Bit Fields tab and select **Insert BitFields**.
- Press the **Insert** key while the Bit Fields tab is active
- Double-click within the Bit Fields tab.

This creates a default Bit Field labelled as "`BitField`*n*" with a Value type "`INT`".

- The Bit Field type **Value** may be modified by double-clicking on value, allowing a selection from a list.

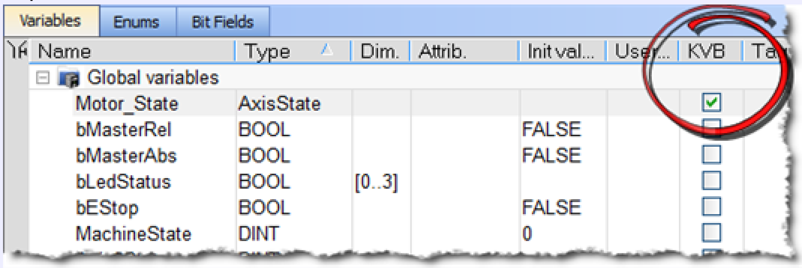
Once the Type is defined you may populate the bits.

- Double click on a bit value to change the text.

IMPORTANT To save your changes you must press **Enter**, **Tab**, the Up, or the Down arrow key.

NOTE Bit Fields are supported over MODBUS. Below is a variable with a custom

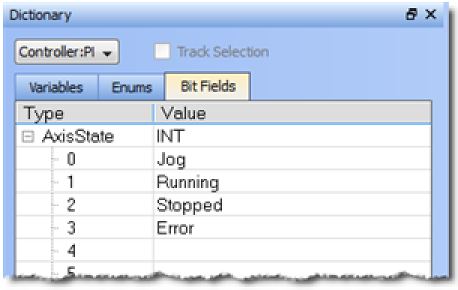
NOTE Type defined in the Bit Fields list which is flagged for KVB. It will be exported.to MODBUS.



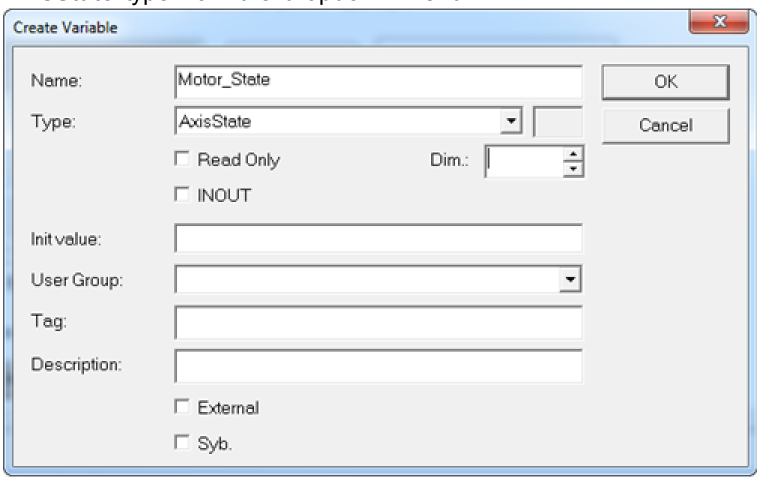
Using Bit Fields

Following is an example of setting up and testing a Bit Fields entry.

1. Define the Bit Field Type and Values.
   Here we have defined a bit called **AxisState** with four values.

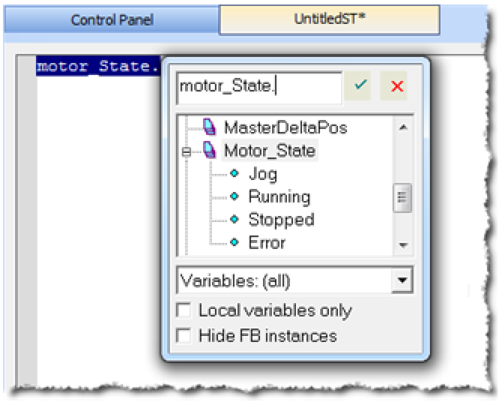


2. Create a variable whose Type that matches the Bit Field.
   Here we have defined a variable called **Motor_State**. We selected the **AxisState** type from the dropdown menu.



3. To use the bit enter `VariableName.BitName` in any editor.
   Here we are using the Structured Text editor and typed "`motor_state.`" When the dot was typed the selector opened, showing us the available bits for the

variable.



10.1.5 Information and Logs

The Information and Log window is used to identify current state status and can be used to identify operational errors, compilation errors, and also to quickly assist you in finding areas of the workspace or program variables.

This window contains different tabs that provide:

- Log messages (Local or Controller) including "Log Messages Settings" (see page 627)
- A system search function
- A list of breakpoints
- A state report on the program compiler

10.1.5.1 Log Messages

TIP Log messages are an important source of information when you are troubleshooting with KAS IDE.
When reporting an issue to Support, copy/paste the logs in your report.

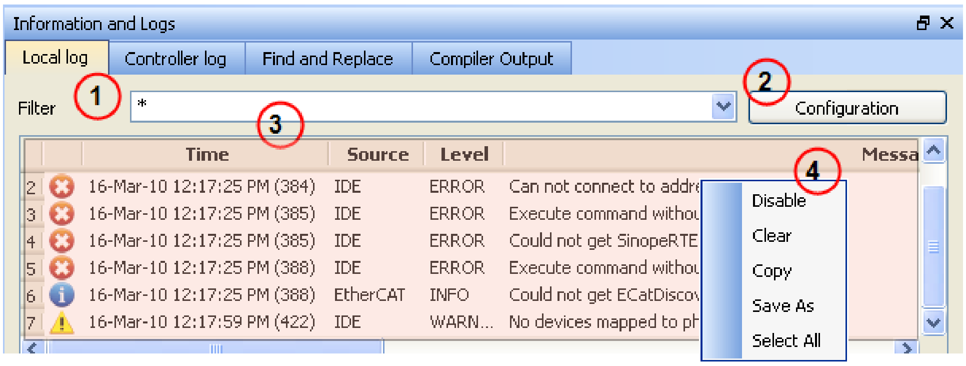


Figure 11-7: Log Messages

Log messages are displayed on two tabs, **Local log** and **Controller log**.

- The Local log tab shows all messages managed by the KAS IDE to explain the current state of the system and to help identify any operation errors encountered when developing your system.
- The Controller log shows all messages managed by the KAS Runtime.

Based on the configuration settings (see call out 2), only messages that are recorded and that match the filter 1 are displayed.

TIP The Configuration button is only available on the Controller log tab when the IDE is connected to a controller.

Every log message in the table widget 3 has the following information:

Field	Description
Time	Time when the log was recorded with the format: DD-MMMM-YY hh:mm:ss (millisecond)
Source	Identifies a software or hardware component issuing the messages. Each source is configured with a specific Level.
Level	Each message has one of the following levels with importance in ascending order: DEBUG > INFO > WARNING > ERROR > CRITICAL
Message	Text of the message issued from the source

Table 11-21: Log Messages - List of Fields

The table contains a contextual menu (see call out 4) with the following commands:

Command	Description
Disable/ Enable	You can stop the log recording at any time, so that no more messages are added
Clear	Empty the list by erasing all the messages already recorded
Copy	Copy the text of the selected messages to the clipboard (you can perform multi-selection with the Ctrl or Shift keys)
Save As	Save all the messages in a log file
Select All	Select all the messages that are displayed in the table

Table 11-22: Log Messages - List of Buttons

10.1.5.2 Log Messages Settings

The KAS IDE manages all messages according to the two following gates:

- Configuration settings define what is recorded in the database
- Filtering defines which messages are displayed in the table widget

Configuration Settings

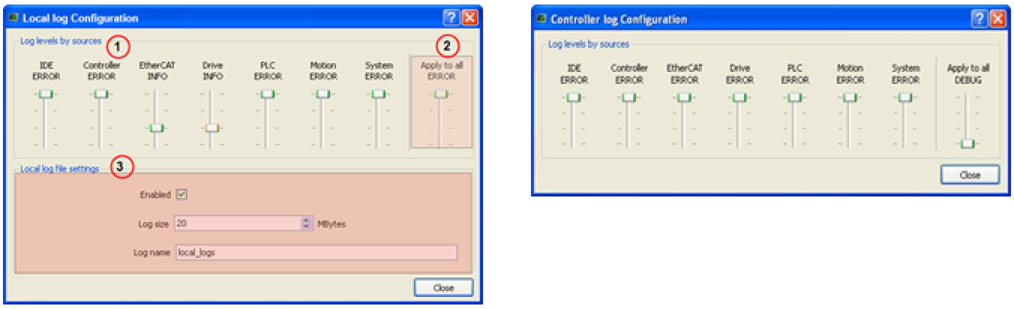
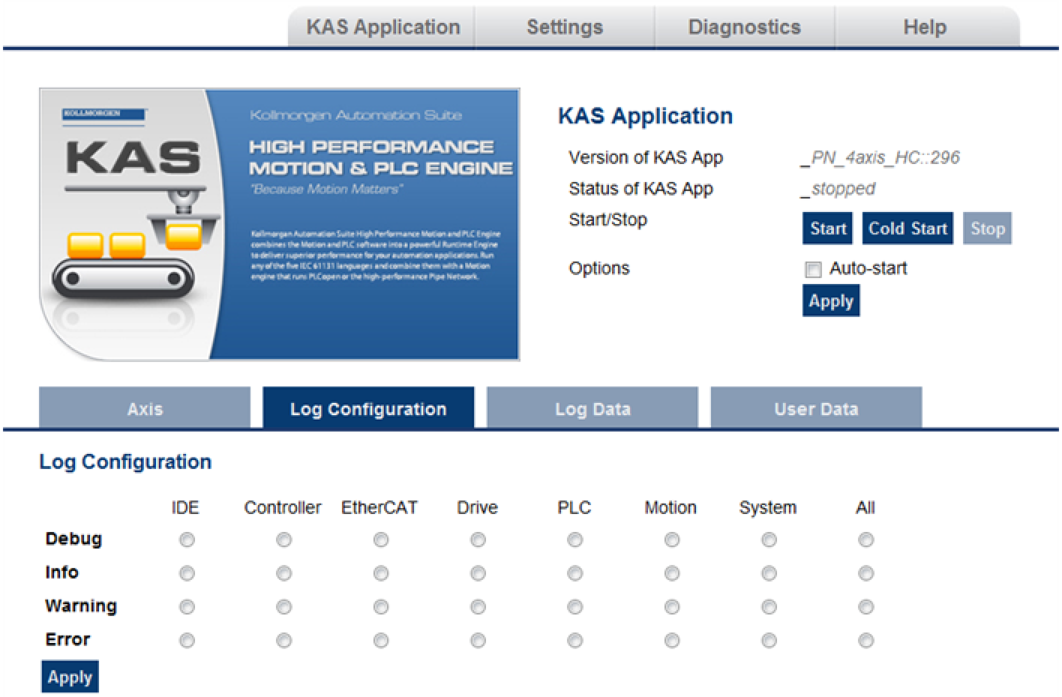


Figure 11-8: Configuration of the Local log and Controller log Messages

Call out#	Description
1	Each source can be set with its own level. It is possible to get a maximum of log details for the selected source without getting a flood of irrelevant messages from the other sources.
2	You can set or reset all the sliders with the same level value
3	Logs can be recorded on the local machine as circular files. Note that on the controller, the recording of the logs is enabled by default. For more details, see page 629

AKD PDMM and PAC generated logs may be configured through the webpage. For more information on the AKD PDMM log files, see "AKD PDMM Log Files" on page 383



NOTE It is recommended that you use either the IDE or web page method, but not both. The communication is unidirectional and the configuration is not read at runtime.

Source

Source	Apply to...
IDE	Win32 applications: the KAS IDE and the KAS Runtime Server (also called the KAS Runtime Front-end)
Controller	For the KAS Runtime items: Drivers, IOEngine, SinopEngine...
EtherCAT	For all kinds of EtherCAT items: Motion bus, I/Os
Drive	Messages from AKD drive
PLC	For application engineers to create custom log within the PLC programs (similar to printf)
Motion	Messages coming from the Motion engines: PLCopen, Pipe network or VM
System	For common API and libraries. Also includes messages issued from the operating system.

Level

Level	Icon	Description
CRITICAL		Application crashes or becomes unstable. Data is corrupted. At that point, the application behavior can be unpredictable.
ERROR		The application does not behave as expected but the processes remain stable.
WARNING		System is stable but the KAS IDE warns that an unexpected event can occur. This is the default logging level. You can ignore this log.
INFO		Information status of the current process. You can ignore this log.
DEBUG		Any information logged for development purpose. You can ignore this log.

Each message has one of the following levels, with importance in ascending order: DEBUG > INFO > WARNING > ERROR > CRITICAL

How to Choose the Appropriate Level?

When a level is set for a source, only messages with the same or higher importance are recorded. In other words, drag the level control slider **Up to reduce** the verbosity, **Down to increase** it.
When the configuration leads to lower verbosity, the treatment during the filtering is quicker.

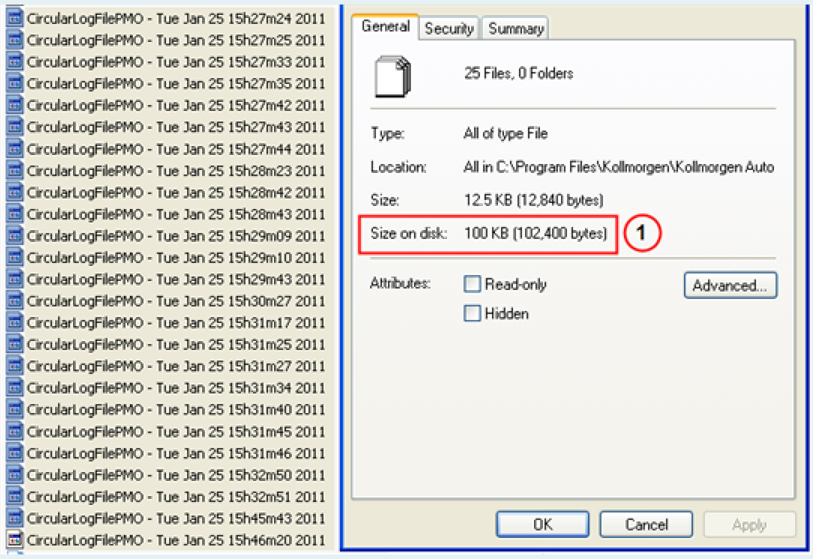
For example, if a source is set to WARNING, then all messages with levels WARNING, ERROR and CRITICAL are recorded (DEBUG and INFO messages are discarded).
In other words, DEBUG is the most verbose, whereas ERROR is the less verbose.

NOTE Critical messages are always recorded (as a consequence, the Critical level is not visible on the slider).

About Log File Settings

Log files are a group of small files where all the last logs are recorded. Each log is recorded as a separated line.

TIP You can import the log files into Microsoft Excel using drag-and-drop.

Field	Description
Enabled	The Log File Settings has to be enabled to record all the logs.
Log size	To prevent any overflow, you can define the maximum size on disk dedicated to the group of local log files. The minimum size is 1MB, the maximum log size is 2MB. At most, the system will store 20 files. When a new file is created, an old one is deleted, maintaining the 20 file threshold. In the example below, the size on disk is limited to 100 KBytes (see call out 1).  Note that the number of rows can vary for each file, depending on what is in the backlog when KAS creates the log files.
Log name	You can define the filename prefix to be used on the local machine.(on the controller, the filename prefix is: controller logs). The suffix to create the complete filename contains a timestamp with the following format: - day MMM DD HHhMMmSS YYYY nn

Where are the log files stored?

- For the local machine (IDE), the Log files are located in the following location:

OS	Location
Windows XP	`C:\Documents and Settings\User\Local Settings\Application Data\Kollmorgen\KAS\Astrolabe\logs`
Windows 7	`C:\Users\AppData\Local\Kollmorgen\KAS\Astrolabe\logs`

- For the controller, the Log files are located under: `<user->\AppData\Local\Kollmorgen\KAS\Sinope Simulator\Application\logs`
- The AKD PDMM logs are accessed via the web server page by browsing to `KAS Application > Log Data`.

Filtering

You can narrow the list of recorded messages by specifying a filter. The filter is applied on all the strings displayed on each row of the table widget (i.e Time, Source, Level and Message).

The drop-down menu gives access to some predefined filters, which can also be edited.

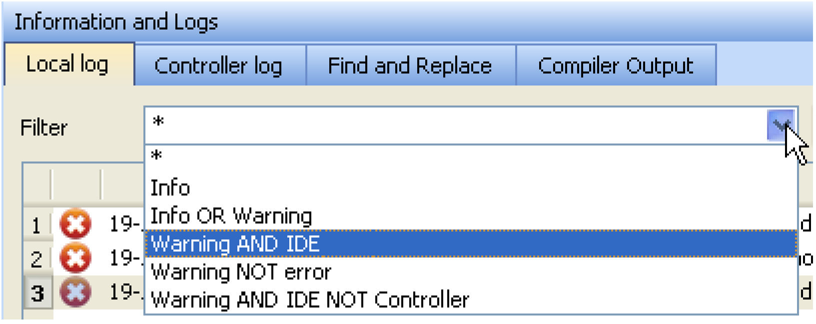


Figure 11-9: Filtering the Messages

For example, filtering with **Warning NOT error** means that only the lines including the word "warning" but not the word "error" are listed.

Filtering Rules

The following rules apply when you work with filters:

- You can combine several strings by including one of the three following boolean operands:
- OR
- AND
- NOT (or use the exclamation mark "**!**")
- Several keywords separated with spaces are considered as an exact string
- Filtering is **not** case sensitive

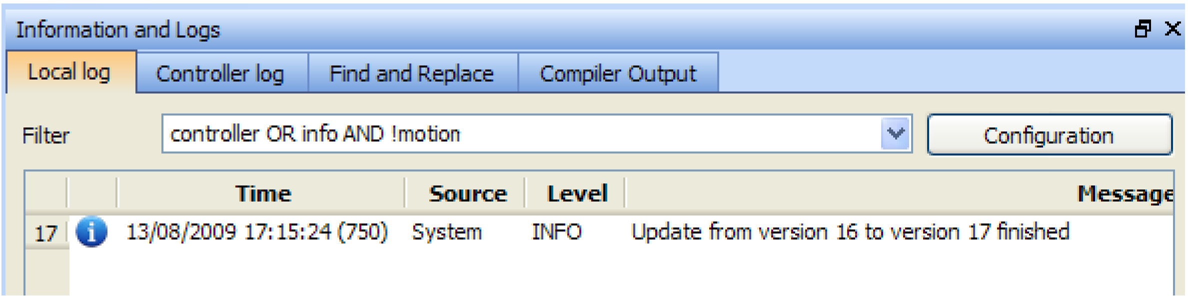


Figure 11-10: Filtering the Messages - Example

NOTE

Warning! When you apply the filter, all the currently recorded messages are parsed and displayed if they match the filter. But all the upcoming recorded messages are added as new rows at the end of the table widget with **no filtering**.

About Scrolling

If you select a message in the table, the scrolling is **stopped**.
All the upcoming recorded messages are added at the end of the list, but your selected message always remains in the same place (you have to scroll down to make the most recent messages visible).

If you select the last row of the table (shortcut: **Alt+Page Down**), the scrolling is **active**.
The last recorded message is always selected and visible at the bottom of the table.

10.1.5.3 Find and Replace

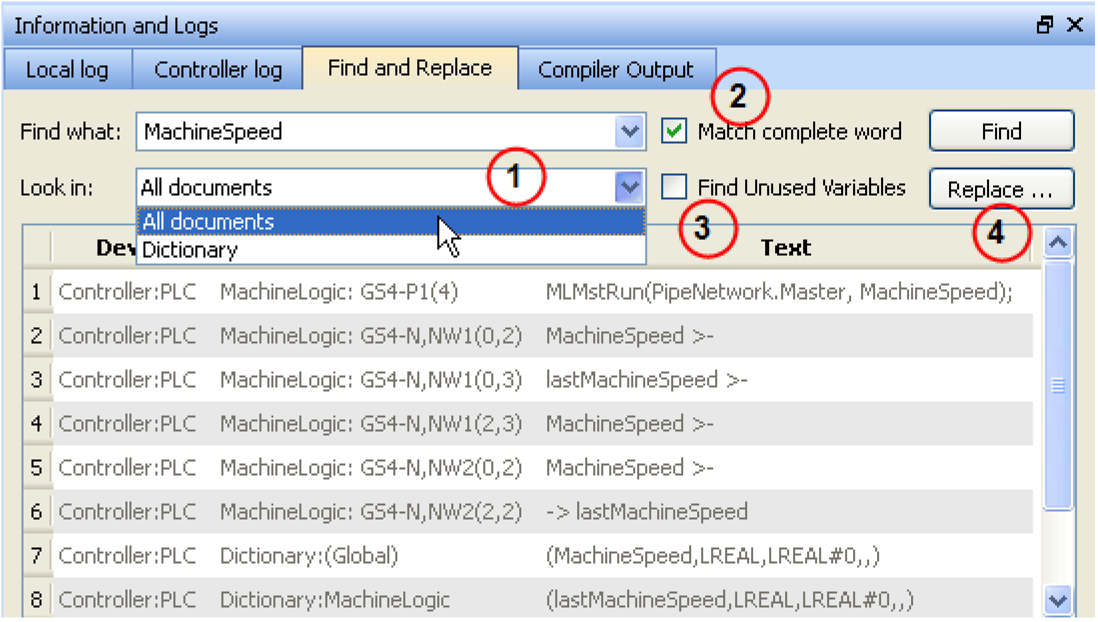


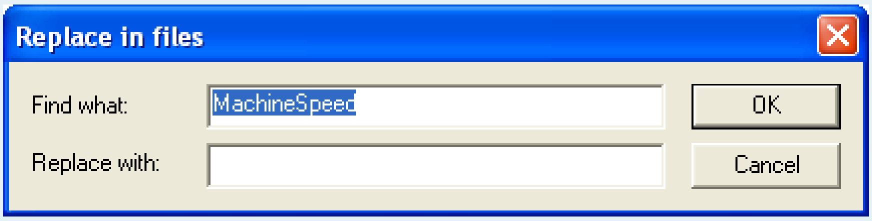
Figure 11-11: Find and Replace

This tab enables you to search for all the instances of a string of characters (search is **not** case sensitive) within the entire environment, and replace it if desired.

You can re-use one of the last ten entries or type a new text string.

Call out#	Description
1	The operation can be performed across: • Dictionary: search the dictionary • All documents: search all files of the project and the dictionary
2	The search string can be with **'complete word'** only. When selected, only the instances that match the complete words defined in the "Find what" field are selected (for example, a search for "MyVar" returns "MyVar" but not "MyVariable"). **NOTE** When this option is selected, the search is case sensitive.
3	To optimize your project, the **Find Unused Variables** option allows you to perform a search in the dictionary to locates variables not used [1] in any program. **TIP** Double-click a variable in the list within the table widget to open it in the Dictionary. Then you can delete this variable from the contextual menu.

[1]A variable is **not used** when there is no effective usage of it in your entire project.
It can still be the case even when a value is assigned to a variable (e.g. MyVar := 100. * Axis1.Velocity ;).
The variable MyVar becomes **used** when it is affected as an input argument (e.g. Velocity := MyVar ;).

Call out#	Description
4	With the **Replace...** button, you can replace in all documents a string with another one (or re-use one of the last ten entries)  The replace function is limited: it is not supported in the Dictionary, Pipe Network and HMI. For those, you have to make the change manually.

Once the search is done, the results appear in the table widget at the bottom of this tab. If a replace has been performed, the Text column provides more information about the replacement.

Double-click the item you want in the list in order to open it in its relevant location (it can be a PLC editor or the Dictionary).

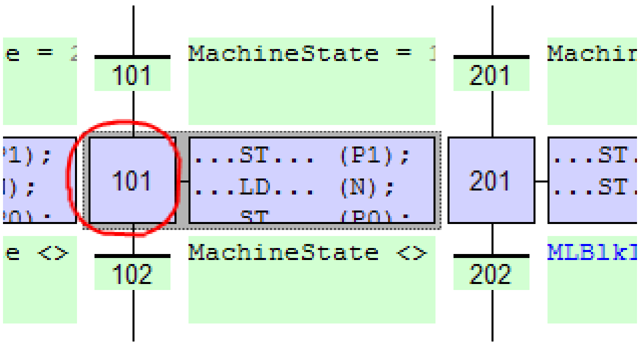
How to Understand the Details of Location?

For SFC programs

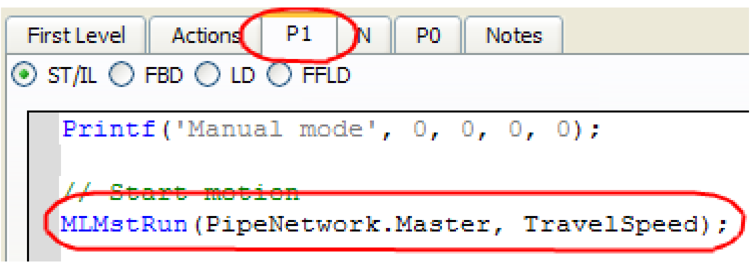
2	controller:PLC	MachineLogic: GS101-P1(4)	MLMstRun(PipeNetwork.Master, TravelSpeed);

SFC Location details

- **Controller** : **PLC** and **MachineLogic** refer to the program in the Project Explorer
- GS stands for **G**raphical and **S**tep (T is for Transition)
- 101 is the reference in the editor



- -P1(4) refers to the **P1** tab and the **4**th line in the source code

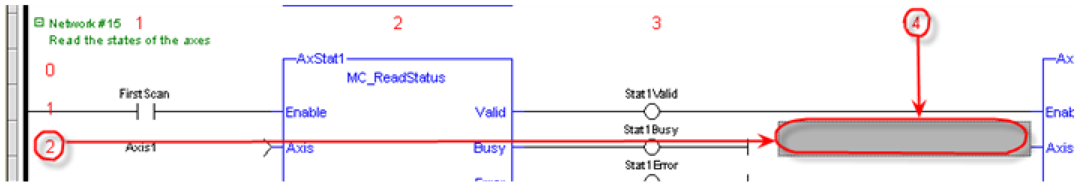


For FFLD programs

```
Controller:PLC:>> Complex variables stored in a separate segment
Controller:PLC:Loading application symbols...
Controller:PLC:Main
Controller:PLC:Main: GS5-N,NW15(4,2): Object not connected on the left
Controller:PLC:Main: GS5-N,NW15(4,0): Bad value on box input(s)
Controller:PLC:Error(s) detected
Controller: ------------ PLC failed ---------------
```

FFLD Location details

- **Controller** : **PLC** and **Main** refer to the program in the Project Explorer
- GS stands for **G**raphical and **S**tep (T is for Transition)
- 5 is the reference in the editor
- -N refers to the **N** tab
- NW15 stands for **Network** number **15**
- (4,2) correspond to the **X,Y** coordinates of the cell relative to the current network

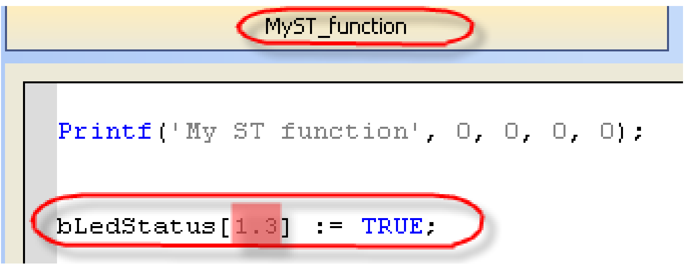


For ST programs

```
Controller:PLC:MyST_function
Controller:PLC:MyST_function: (5): Array index expected - must be a DINT expression
Controller:PLC:Error(s) detected
```

ST Location details

- **Controller** : **PLC** and **MyST_function** refer to the program in the Project Explorer
- (5) refers to the **5**th line in the source code



For more details, see "Find and Replace Operations" on page 634.

10.1.5.4 Find and Replace Operations

The Find and Replace command enables you to search for a specified string of characters within your project.

You can use any of the following methods to access this functionality:

- From the **Information and Logs** toolbox
- In the **Dictionary** panel
- From an **editor** (ST/IL, FBD, FFLD)

Information and Logs

For more details, refer to the Information and Logs toolbox.

Dictionary

Right-click on the variable name and select the **Find all** command in the menu. This command starts a search of all documents for the selected variable and displays the results in the table widget within the Information and Logs toolbox.

NOTE This opeartion selects only the instances that match the complete words (for example, a search for "MyVar" returns "MyVar" but not "MyVar1").

Editor

It is possible to perform a search and replace from a PLC editor (ST/IL, FBD, FFLD) by selecting the *Find* or *Find next* commands in the contextual menu.

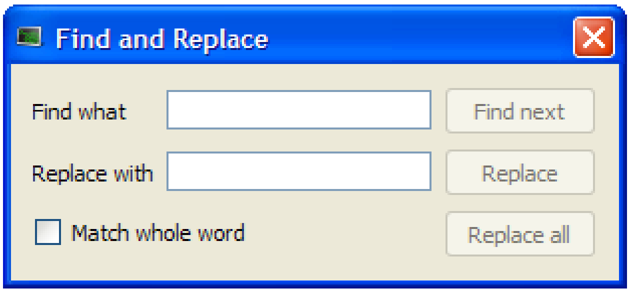


Figure 11-12: Find and Replace from an Editor

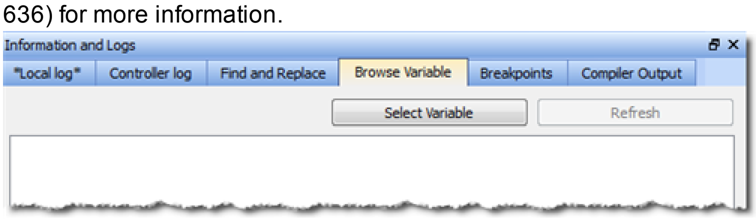
Function	Description
Match Whole Word	When selected, only the instances that match the complete words defined in the "Find what" field are selected (for example, a search for `MyVar` returns "MyVar" but not "MyVar1").
Find next	Allows you to select in the current editor the next instance of the matched string.
Replace next	Allows you to replace the next instance of the matched string.
Replace all	Allows you to replace in the current editor all instances of the matched string.

NOTE The *Find*, *Replace* and *Replace all* operations work only for *variable symbol* property of the *Control*.

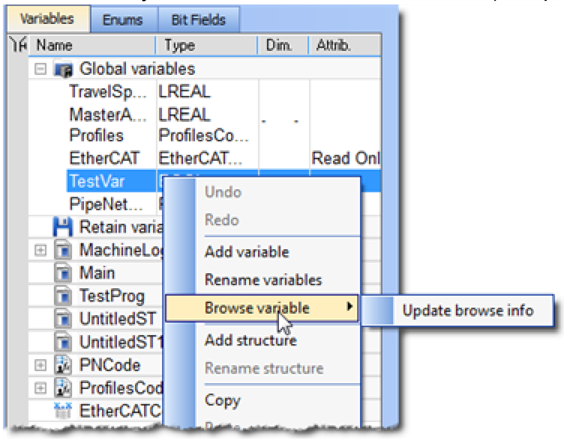
10.1.5.5 Browse Variable Tab

This tab is used to browse all instances of a variable. It will show the locations and usage of each instance of a single variable. There are two ways to populate the tab:

- Click the **Select Variable** button. See "Using the Browse Variable tab" (see page 636) for more information.

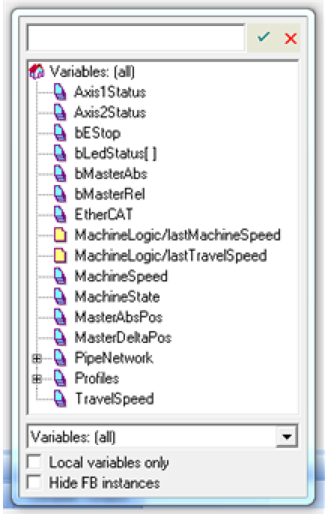


- Right-click on a variable in the Dictionary and select **Browse Variable**. See "Using the Dictionary's Browse Variable menu item" (see page 637) for more information.



Using the Browse Variable tab

1. Click the **Select Variable** button to open a variable selection pop-up window.
2. Select a variable from the pop-up list.



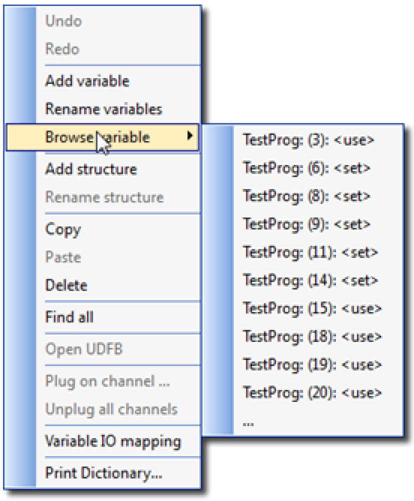
3. The Browse Variable tab is updated with a two-column table detailing the variable's usage
   - **Location** shows the file name and line number where the variable is used.
   - **Usage** shows whether the variable is being read (**use**) or written to (**set**).
4. Double-click on an entry to be taken to the correspondinglocation in the editor.

NOTE

- The process of generating the Browse Variable content after clicking Select Variable can take a significant amount of time, depending upon the number of files in the project. If there are no modifications to the project, further browse operations will not take any time.
- The browse information may not be current if changes are made to the project.
- The browser information needs to be refreshed after saving and compiling a project. A message is shown in the tab to alert you if the information may not be accurate.

Using the Dictionary's Browse Variable menu item

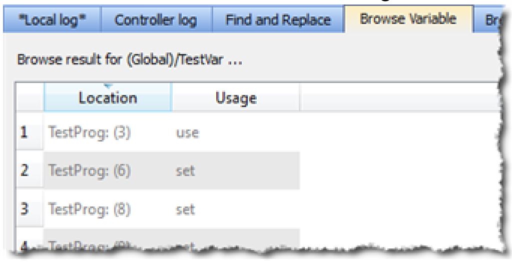
The first time this menu is accessed (per project instance) you are presented with **Update browse info**. This updates the browse data. Further selections of any variable result in the location and usage being shown directly in the menu.



- Selecting an entry will take you to the corresponding location in the editor.

When the menu has more than ten entries an ellipsis (…) is added to the bottom of the menu.

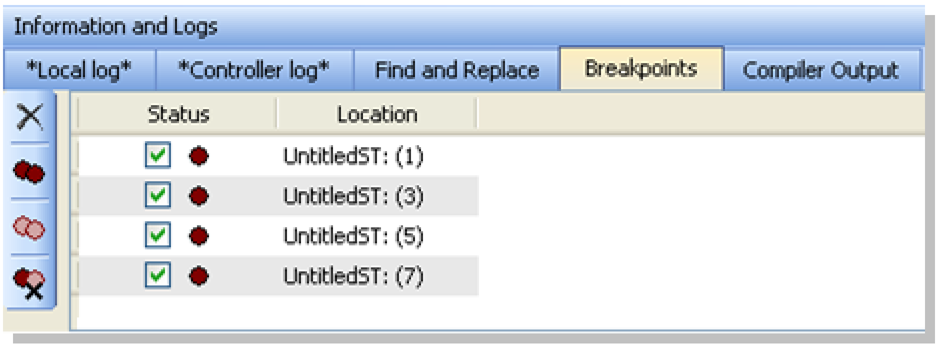
- Selecting the ellipsis (…) brings you to the Browse Variable tab, which will be populated with the variable locations and usages.



10.1.5.6 Breakpoints tab

The Breakpoints tab lists all of the breakpoints in the PLC program, including their position and status. Double-clicking on an entry will take you to that location in the editor.

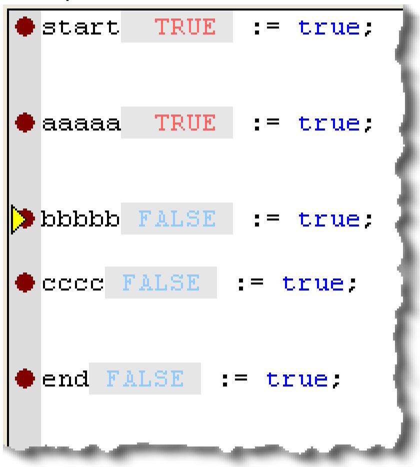
Breakpoints may be enabled and disabled singly by clicking the **Status** checkboxes. Buttons on the left of the tab provide the ability to remove single breakpoints, enable and disable all breakpoints, and remove all breakpoints.

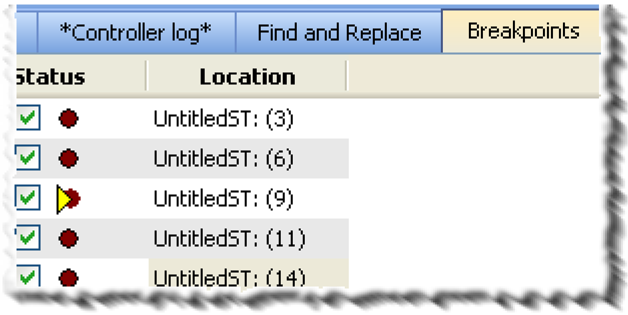


Button	Action Performed
	Delete selected breakpoint from the list and editor.
	Enable all breakpoints. This will make all breakpoints "active"
	Disable all breakpoints. This will make all breakpoints "inactive"
	Remove all breakpoints.

Right-clicking on a breakpoint entry in the list provides for enabling, disabling, deleting the entry, and going to that location in the source code.

Breakpoints (both active and inactive) which have been "hit" or reached in the code are flagged with a yellow triangle. This provides a quick and easy way to identify the breakpoint. This can be seen in both the code and the Breakpoints tab.





TIP Any program (except for an SFC program) that contains a breakpoint that gets "hit" during debugging will be automatically opened for your convenience.

As breakpoints set in SFC programs cannot be enabled or disabled, entries in the Breakpoints widget do not have a checkbox to perform these actions.

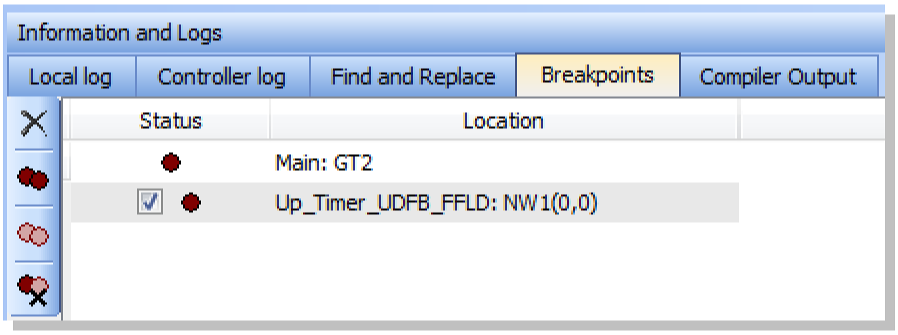


Figure 11-13: Example of a breakpoint (*Main: GT2*) set in an SFC program.

For more information on breakpoints, see "Breakpoints" (see page 329) and "Setting, Removing, Enabling, and Disabling Breakpoints" (see page 331).

10.1.5.7 Compiler Output

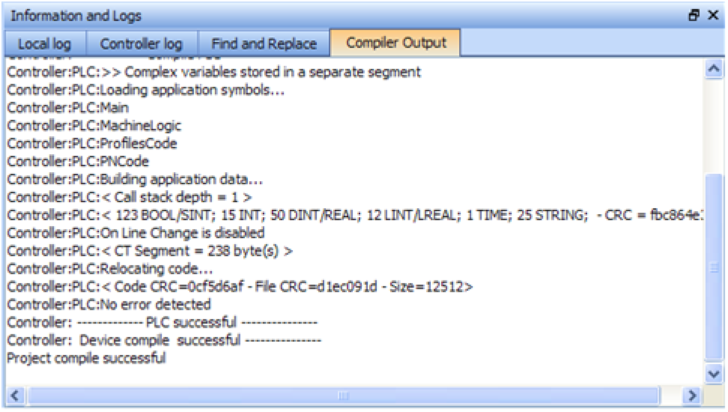
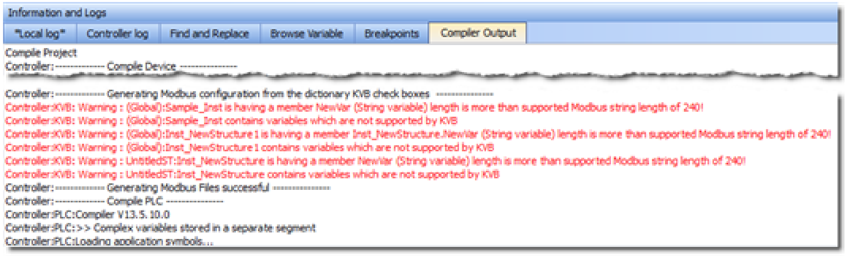


Figure 11-14: Compiler Output

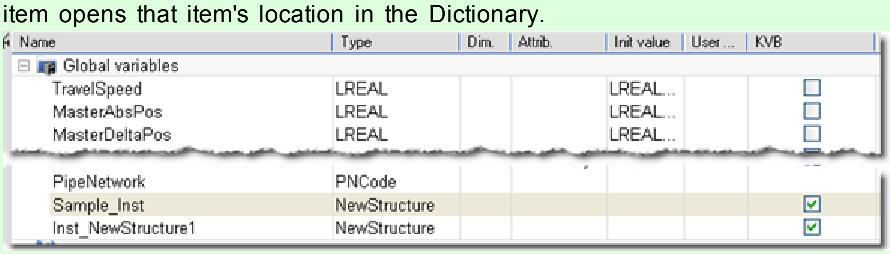
This tab displays information about the last project compilation. It shows information messages as well as Warnings and Errors (highlighted in red). Successful and unsuccessful output is reported within this tab to help identify and troubleshoot problem areas of the program development.



ⓘ **TIP**

Double-click an error to open the program in the workspace and jump directly to the relevant location in the editor. This lets you rework the program and fix the error.

In the image above, the first warning is for Sample_Inst. Double clicking that item opens that item's location in the Dictionary.



When there is a long list of statements, only the bottom part is displayed. Do not forget to scroll up.

How to Clean-up the Code?

To clean-up your application, do as follows:

1. Scroll up to start from top and locate the first error message
2. Fix the error

NOTE

Because fixing **one** piece of code can eliminate **multiple** compiler output error, it is recommended to recompile each time you correct an error.

When no more errors exist, the following messages are displayed:

- PLC successful (the IEC 61131-3 code is correct)
- Device compile successful (is related to the Motion part (e.g. CAM profiles), EtherCAT XML file...)
- Generating Modbus files (related to the variables mapped with the HMI)
- Project compile successful (the complete project is ready to be downloaded to the target)

Text displayed:

Operands of "*" or "/" must be numbers and have the same type

Meaning:

This error appears in a ST instruction when a constant does not have the expected type in a multiplication or division operation. Typically, REAL is the default precision for floating points, so you have to explicitly declare your long real constants with the LREAL# prefix when required.

How to Understand the Details of Location?

Same explanations contained in previous section **Find and Replace** are also applicable here.

10.1.6 Watch Window

This toolbox enables you to add variables to a dedicated watch window to display its value in real time.

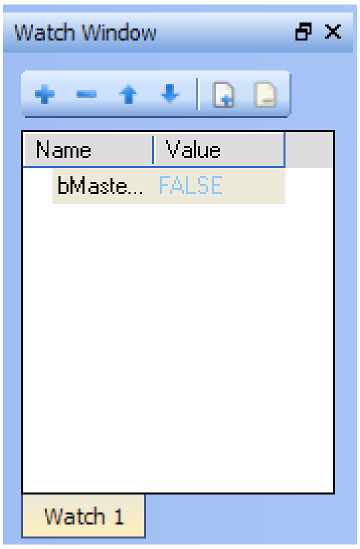


Figure 11-15: Watch Window

NOTE PLC variables viewed in the Watch Window are saved as a part of the project. This means that the next time you open the project, those variables will be pre-loaded in the watch window. This does not apply to AKD variables, which are not saved with the project.

10.1.6.1 Multiple Watch Windows

The KAS IDE allows you to group several variables in a single watch window, and to have up to 10 different watch windows. Each of them is displayed as a tab with its own label.

Explanation for each icon:

Icon	Description
	Add a variable with the PLC Variable Selector **TIP** You can also add an existing variable in the watch window directly by using **drag-and-drop** from the Dictionary or the PLC editors.
	Remove a variable
	Move up the selected variable
	Move down the selected variable
	Add a new watch window Each window is displayed as a tab with its own label (Note that there is a limit of 10 tabs)
	Remove the selected watch window

Table 11-23: Watch Window - List of Icons

Each variable in the table widget has the following information:

Field	Description
Name	Lists the variables as well as structure, arrays and expressions. You can double-click a variable (or press **F2** key when it is selected) to edit its name (except for structure and array members)
Value	When the application is running, displays the variable or expression's value. You can double-click a value to force modification of the selected variable

The contextual menu allows you to:

- Add a variable
- Remove a variable
- Remove all variables

10.1.6.2 Access Structure and Arrays

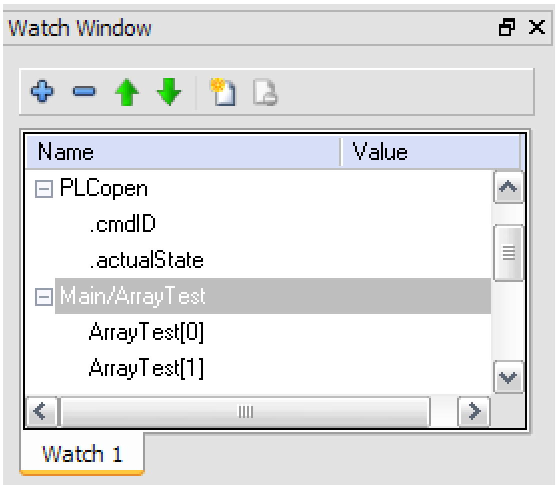


Figure 11-16: Watch Window - Accessing Arrays

When a structure or an array is in the watch window, you can expand its node to display all its members.

Note that structure or array members cannot be deleted, edited or moved up/down in the list.

10.1.6.3 Add Variable

- Double-click the nodes (**(Global)**, **Main**...) to expand their related variables

ⓘ *TIP*	Expand AKD node if you want to add AKD parameters to the Watch Window

- Select one from the list
- Click **OK**

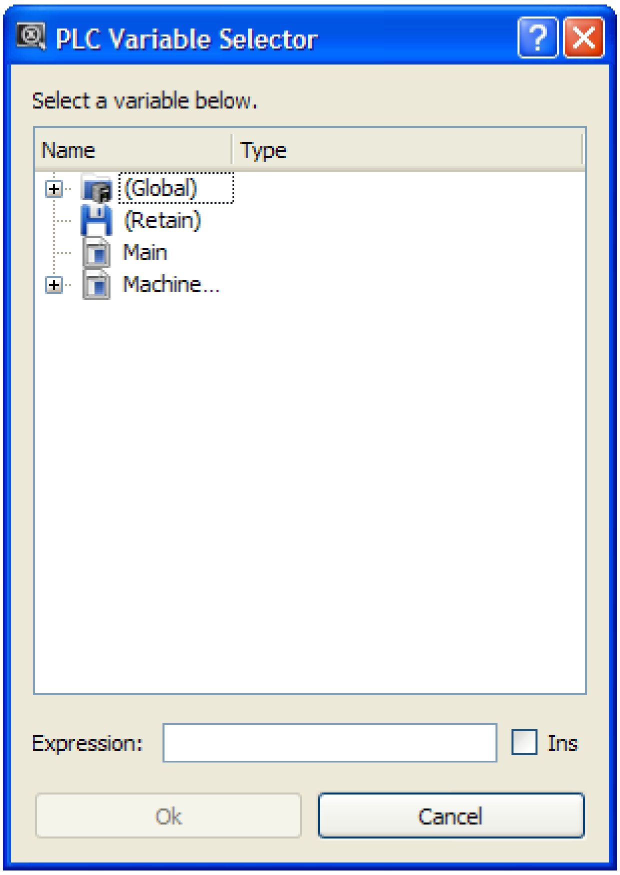


Figure 11-17: Watch Window - Selecting PLC Variable

This variable is then added to the current watch window tab.

10.1.6.4 Add an Expression

You can enter variable strings as an expression.

For example, if you want to add together two integer variables, follow these steps:

- Click the Add symbol ✚ to open the PLC Variable Selector
- Choose a variable, but do not click OK yet
  (the variable is added to the expression field where you can do any required editing)

- Select the **Ins** option
  (this option allows you to insert the next selected variable at the current cursor position in the expression edit field)

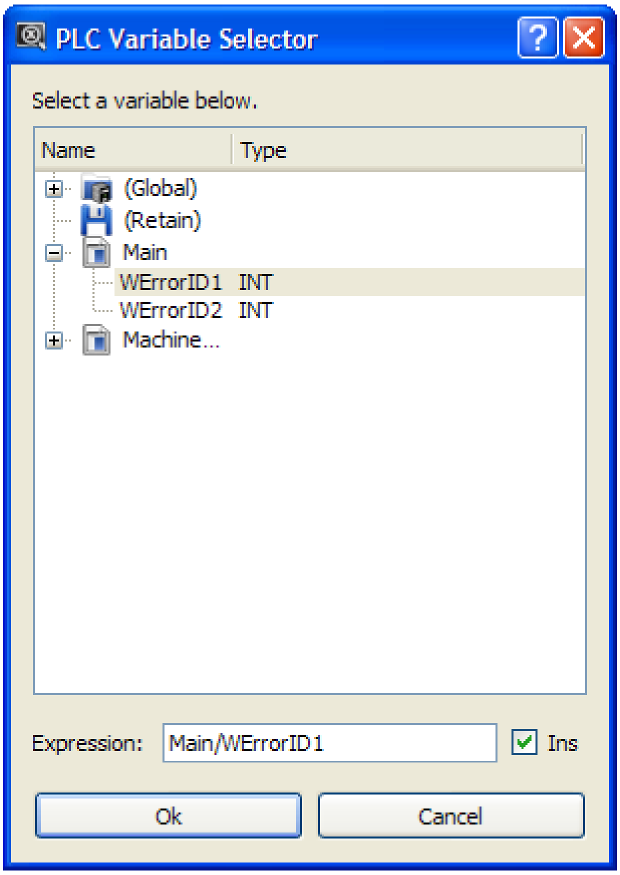
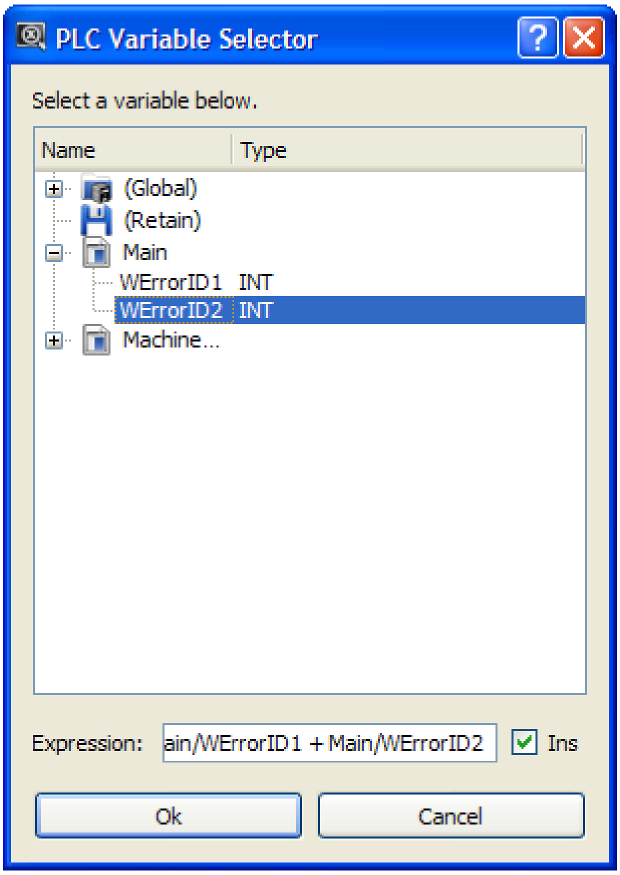


Figure 11-18: Watch Window - Creating Expression

- Press the PLUS SIGN (**+**) in the expression field
- Select another variable

- Click the **OK** button



- Then the expression is displayed into the watch window

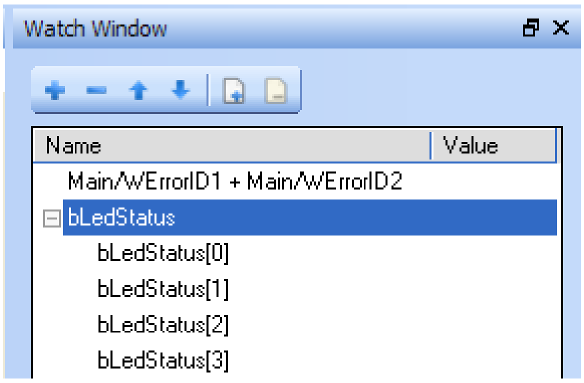


Figure 11-19: Watch Window - Displaying Expression

What you can include in a complex expression:

- Index of array
- Comparison “>”, “<”, “<>”, “=”
- Operator “+”, “*”, “-“, “/”

Please note that the DIVIDE SIGN (/) is not interpreted as an operator when used with prefixed variables (e.g. MachineLogic/lastTravelSpeed)

10.1.6.5 Force a Variable

At run-time, all variables in the table widget are animated [1] with real-time values.

You can double-click on the value of a variable (or press the **ENTER** key when it is selected) to open a pop-up window that allows you to:

- **Force**:
  change the value of the selected variable. Depending on the variable type, you have the possibility to define its value either in the text field or with the check boxes.

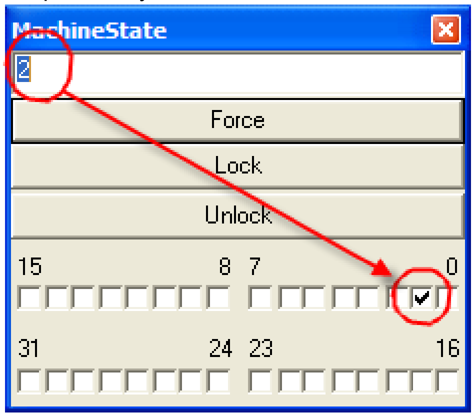


Figure 11-20: Watch Window - Forcing a variable

- **Lock**:
  When a variable is locked, its value is no longer changed by the runtime. You can then force its value from the debugger independently from the runtime operations. Note that all variables can be locked and forced at run-time.

> **NOTE** The value of a locked variable is displayed with square brackets.
>
> `ActualMachineState [[ -1 ]] := -1;`

- **Unlock**:
  Remove the lock on a variable so that it can be changed again by the runtime.

10.1.7 AKD Drive

In addition to the different views, the AKD GUI provides a toolbar and a status bar to display some extra information.

10.1.7.1 AKD GUI Toolbar

The toolbar provides access to the following:

- Enable / Disable the drive (software enable)
- Start / Stop the Service Motion
- Mode:position / torque / velocity
- Disable & Clear Faults: Click this button to clear the fault, then click the Enable button to enable the drive again

[1] To better track variables and expressions of the PLC programs in test mode, the KAS IDE dynamically computes their value along with the execution and displays the result

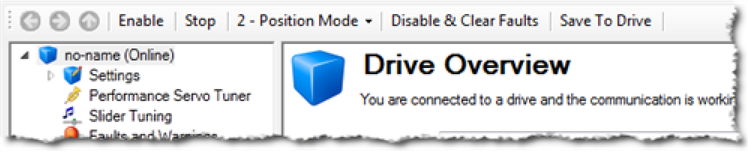


Figure 11-21: AKD Toolbar

> **TIP** To stop all the AKD drives at the same time, click on the Stop button  in the Device Toolbar.

10.1.7.2 Status Bar

The status bar provides the following information on the drive:

- A fault indicator (No Faults / x Faults) that becomes red when any AKD gets a fault You can also set the Log message to get more details on the drive messages
- The drive status: active / inactive
- The software (SW) enable status
- The hardware (HW) enable status

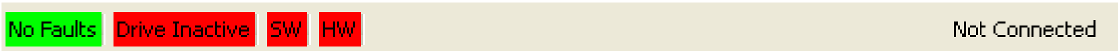


Figure 11-22: AKD Status Bar

For the SW and HW enable status indicators, the color code is:

- **Green** when it is OK (i.e. everything is ready to do motion)
- **Red** in case of errors / faults
- **Grey** for all other cases (for example when SW or HW is not enabled: status is not green because a motion could not happen, and not red because it is not an error)

10.1.8 Status Bar

A status bar located at the bottom of the KAS IDE main window displays the five following labels from left to right:

- Local version
- Controller version (application version located in the controller)
- Drives state
- Controller state (stopped/running)
- Connection state

Figure 11-23: Status Bar Labels

An icon  between the Local and Controller versions allows to show any differences (for more details, see page 336).

The space on the left of the status bar is reserved for messages.

10.1.8.1 Local Version

This label provides information about the version locally present in the KAS IDE. There are three different states:

- *Nothing displayed* (for instance when no project is loaded)
- Version information (when available)
- Compilation error (background in **red**)

TIP You can position the mouse over the text field to display a tooltip with the detailed version information.

10.1.8.2 Controller Version

This label provides information about the version present in the controller. There are three different states:

- *Nothing displayed* (when not connected)
- No Application in the controller
- Version information (when available)
  Syntax of the version label is: <project_name>:<version>

TIP When an application is active in the target, you can hold the mouse over the text field to display a tooltip with the detailed version information, including a timestamp of the compilation.

10.1.8.3 Drives state

There are three different states:

- Drives inactive (drives are disabled or your application is not connected to the target)
- Drives active (at least one drive is active)
- Drives error (at least one drive is in error)

10.1.8.4 Controller State

The Controller state label lets you know if the Controller is running or stopped. There are three different states:

- *Nothing displayed* (the label is empty when the KAS IDE is not connected to the target)
- Controller is stopped
- Controller is running

10.1.8.5 Connection State

The Connection label displays the Connection state between the KAS IDE and the Controller. There are five different states:

- Not connected
- Connecting
- Connected (background in **green**)
- Connection Error (background in **red**)
- Unexpected Disconnection (background in **red**)

TIP You can hold the mouse over the text field to display a tooltip with some detailed information about the Error, and the Controller address when connected.

10.1.8.6 Color Codes

The Local and Controller version labels has an **orange** background in case of version mismatch between the IDE and the Controller. This warns you that you have to download the new version of the application.

The Local version label has a **red** background if the compilation fails.

List of use cases for the labels of the status bar

The following table summarizes all cases for the labels of the status bar.

Connection state	Local version	Controller version	Controller status	Connection status
Disconnected				Not Connected
Disconnected	Version A			Not Connected
Connecting	Version A			Connecting…
Connected	Version A	No Application	Stopped	Connected
Connected	Version B	Version B	Stopped or running	Connected
Connected	Version B	Version A	Stopped or running	Connected
Disconnected	Compile error			Not Connected
Connected	Compile error	Version A	Stopped or running	Connected
Comm. error	Version A			Connection Error
Disconnected	Version A			Unexpected dis-connection

Table 11-24: Connection Status

10.2 Choose a Workspace Layout

10.2.1 Move Child Windows

In the integrated workspace, all child windows are integrated into a single larger application window.

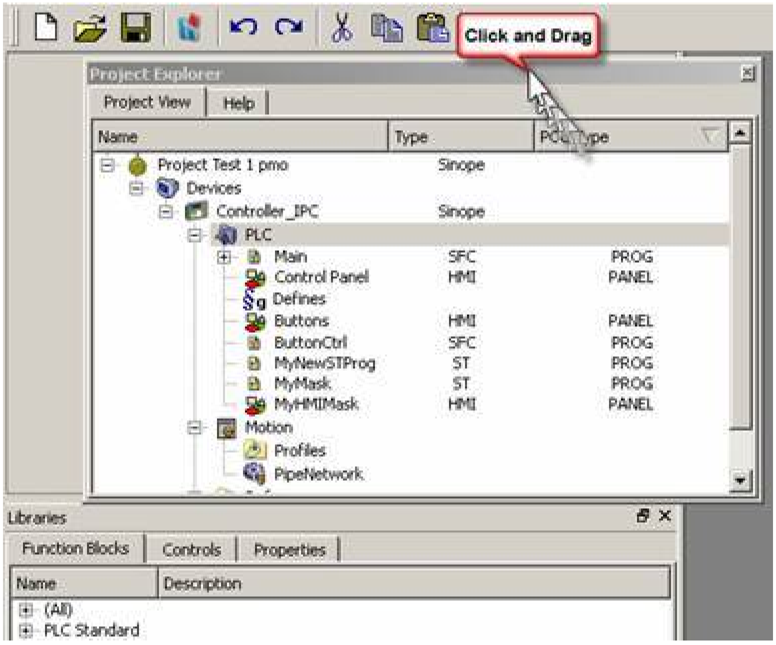
With the MDI/Tabbed workspace command in the Window menu, you can choose to display the child windows either as Tabbed Document Interface (TDI) or as Multiple Document Interface (MDI).

When in MDI mode, you can move and resize the displayed windows.

The Cascade command automatically rearranges all the windows to provide you with easier access to each of them.

10.2.2 Move Toolbox

All toolboxes can be moved within the workspace to a more appropriate location. To customize your workspace, click in the Toolbox header and move the window using drag-and-drop. The other toolboxes are adapted accordingly.



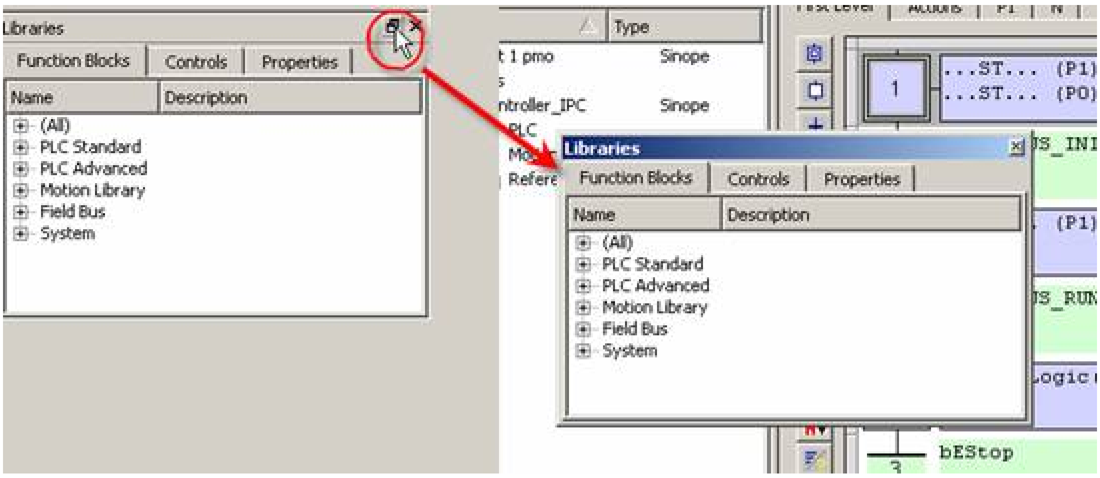
10.2.2.1 Dock Window

You can separate out a toolbox and change it to a docking window to be placed in the workspace independently of the other toolboxes.

How to change a toolbox to a Docking window?

To do so, click the icon (you can also double-click in the toolbox header).

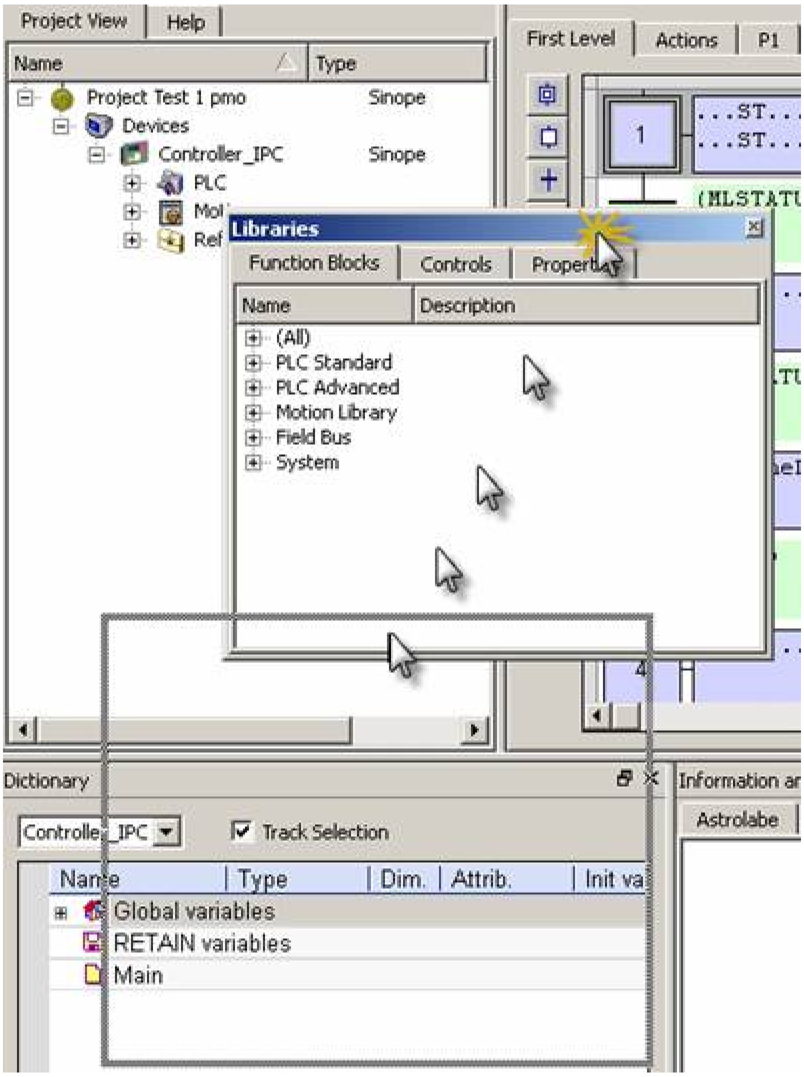
TIP Double-click to place the window back into its original position.



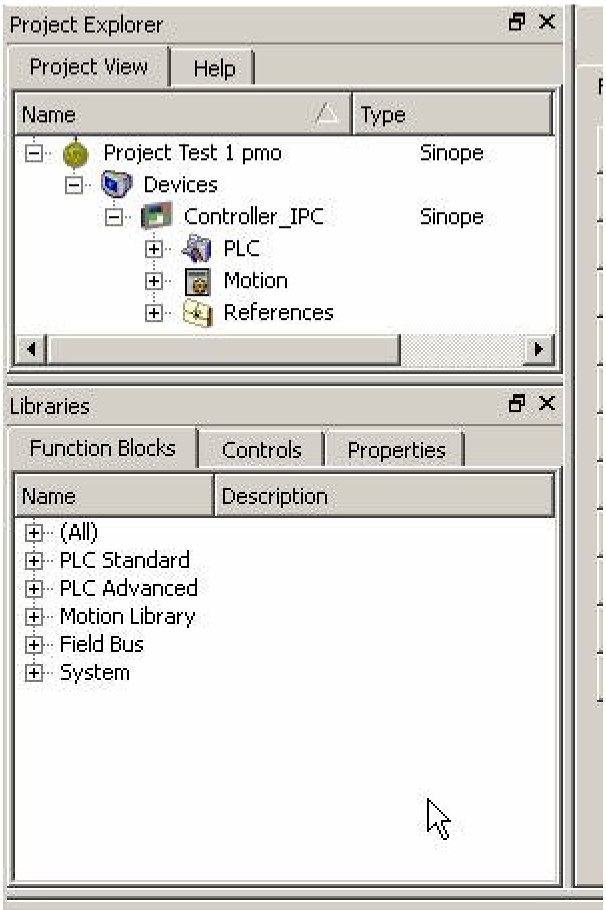
NOTE Moving a toolbox to a docking window can lead to problems which can be difficult to recover.

How to undock a window?

If problems arise, drag-and-drop the window to a toolbox border as shown below:



Dropped in the bottom border of the Project Explorer toolbox, then the **Libraries** toolbox is moved nearby.

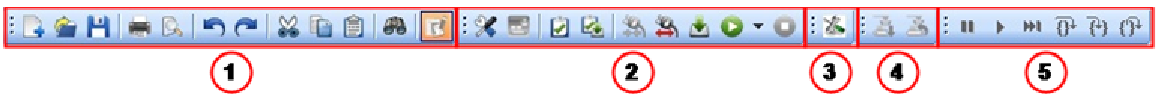


10.3 Menus and Toolbar Overview

The KAS IDE contains the five following menus:

- File
- Edit
- Tools
- Windows
- Help

...and the following toolbars:



1. Tools
2. Device
3. EtherCAT
4. Online Change
5. Debug

A specific toolbar is also available for the AKD drive.

For details about icons available in the graphical PLC editors, see these sections:

- FBD toolbar
- FFLD toolbar

- SFC toolbar

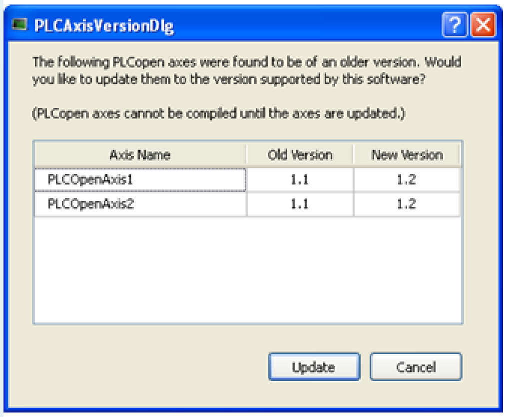
10.3.1 File Menu

Command	Description
New (Ctrl + N)	Close the current project if any, and then launch the project wizard to create a new one

Command	Description
Open... (Ctrl + O)	Open an existing project

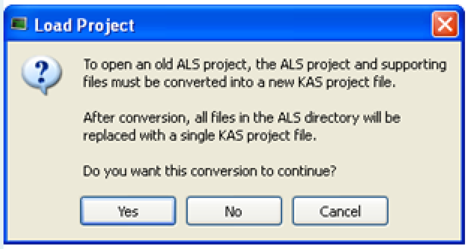
About project from older version

When you try to open a previous project, KAS IDE proposes you to do the conversion to keep the compatibility.



About ALS project

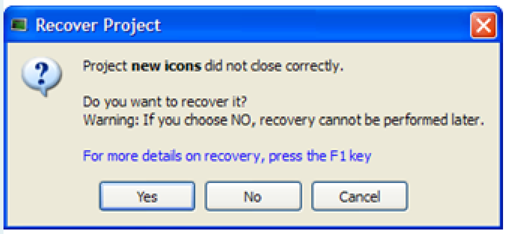
When you try to open a previous project with ALS format, the KAS IDE proposes you to convert it to the current KAS format.



After conversion, all the files located in the folder structure are replaced with a single KAS file.

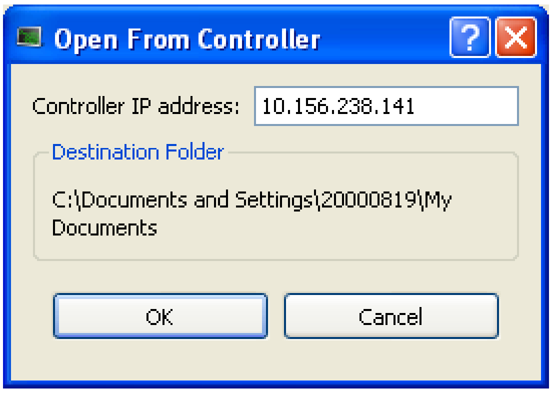
About Auto Recovery

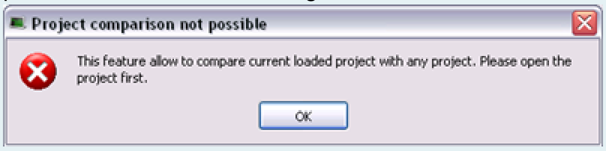
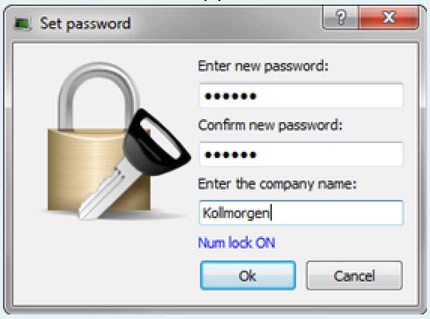
If the KAS IDE crashes when you are working with a project, you can start it again to recover the project from the last successful **Save** (or auto save) operation.



About Corruption

If your project is corrupted, KAS IDE opens a pop-up

Command	Description
	window  Or you can also open an existing library
Open From Controller (Alt+Shift+O)	Project files saved on a controller can be retrieved using this command. To retrieve the project, enter the controller's IP Address.  Projects which are successfully retrieved will be saved in *My Documents* before being opened. See "Step 5 of 6 - Download the Application" (see page 325) for information on storing projects on the controller.
Save	Save the current project **About Auto Save** At compile time, an auto-save operation is performed which allows you to recover the project if a crash occurs. **NOTE** When in Debug mode, saving your project automatically switch to the Edit mode.
Save As...	Save the current project in a location and with a name that you can define. (the project name is also reflected in the KAS IDE window's title)

Command	Description
Compare Projects (Alt+Shift+C)	This option allows for comparison between an open project and another local project. Once you have browsed for and selected the file for comparison, a window will open which displays the differences between the files. See "Compare PLC Programs" (see page 336) for more information on the comparison tool. A project must be loaded in the IDE to use this function. If a project is not present you will be presented with the following error. 
Close Project	Close the current project. (if changes have not been saved, a prompt is displayed first)
Password Protection	This option provides the ability to set, change and remove password protection on a project file to prevent unauthorized access. The menu options are available when a project is open. Passwords must be 6-20 characters in length and may consist of any alpha-numeric characters (a-z, 0-9, $, &, *, }, etc,); spaces, tabs, and apostrophes are not permitted. Additionally, a company name must be provided, this should be the name of the company which owns the application source code.  The status bar has a project security status icon in the lower left corner: protected and unprotected. **NOTE** To protect a project which is stored on the controller, the Project must be compiled and downloaded.
Page Setup...	Define page setup, margins and header/footer
Print... (Ctrl + P)	Print the project element currently open in the workspace
Print Preview...	Display a printout on the screen so you can preview it before printing

Command	Description
Print Project...	Select among the complete project's elements those you want to print
Recent Projects	List the most recently used projects
Exit	Quit KAS IDE

Table 11-25: File Menu Commands

10.3.2 Edit Menu

Command	Description
Cut	Cut selected data and copy it to the clipboard
Copy	Copy selected data to the clipboard
Paste	Paste the data currently stored in the clipboard
Undo	Undo last command **NOTE** This action is not possible for all operations.
Redo	Redo last command
Find…	Show the Find and Replace tab in the Information and Logs toolbox

Table 11-26: Edit Menu Commands

10.3.3 Tools Menu

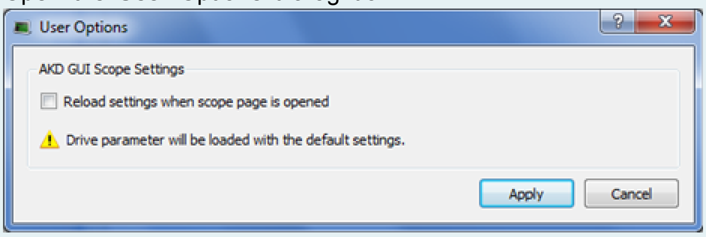
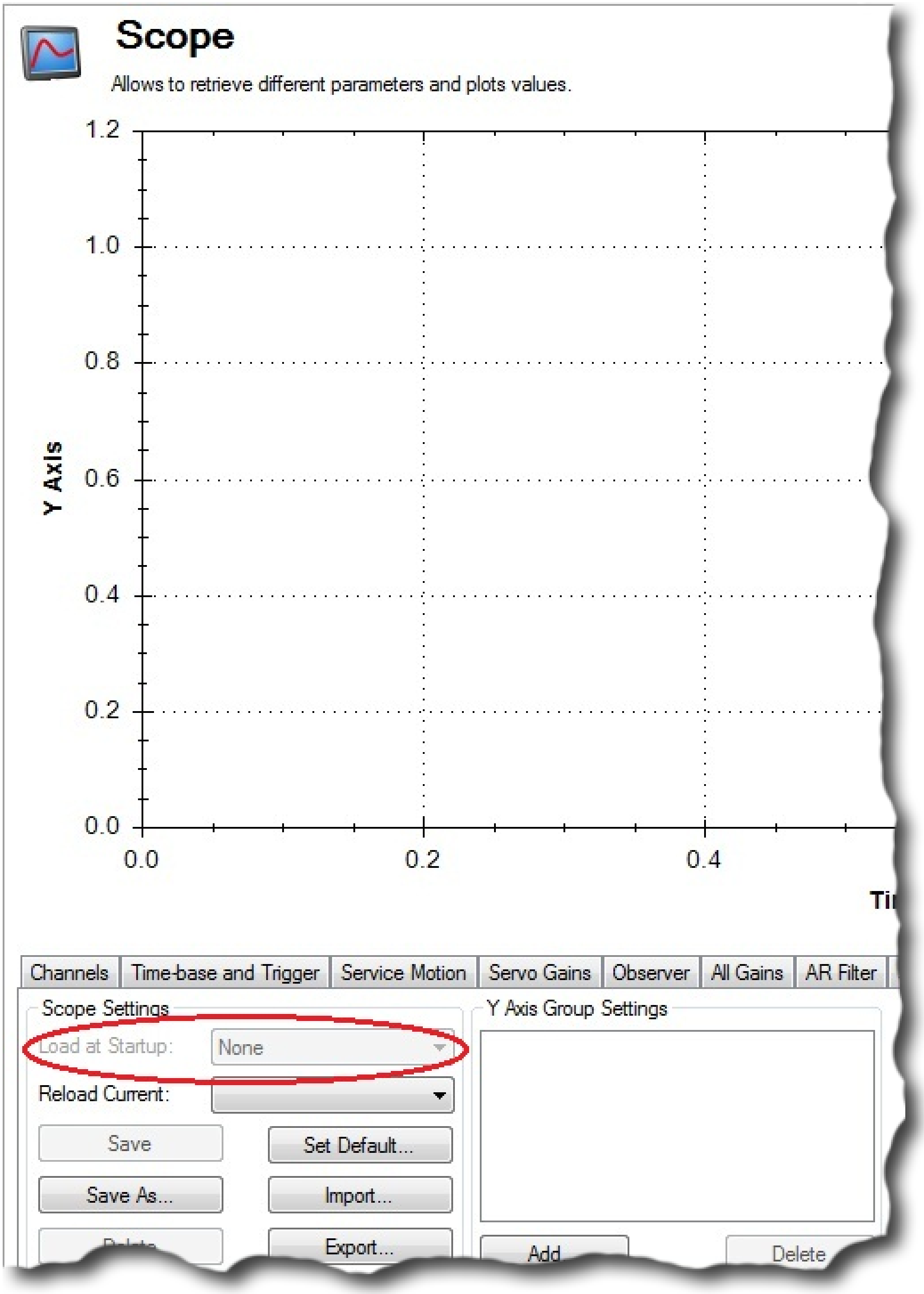
Command	Description
Custom IO Editor	Show the I/O Editor
Oscilloscope	Show the soft oscilloscope
Compile (Ctrl + B)	Compile the whole project
User Options	Open the User Options dialog box. 

Table 11-27: Tools Menu Commands

10.3.3.1 User Options

The check box found in this dialog enables/disables the “Load at startup” option in the AKD GUI Scope “settings” tab.



10.3.4 Windows Menu

Command	Shortcut	Description
MDI/Tabbed Workspace	ALT+W	Toggle the workspace between the MDI and the tabbed mode

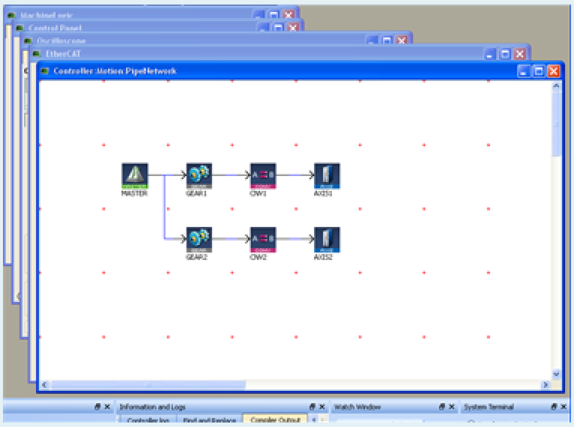
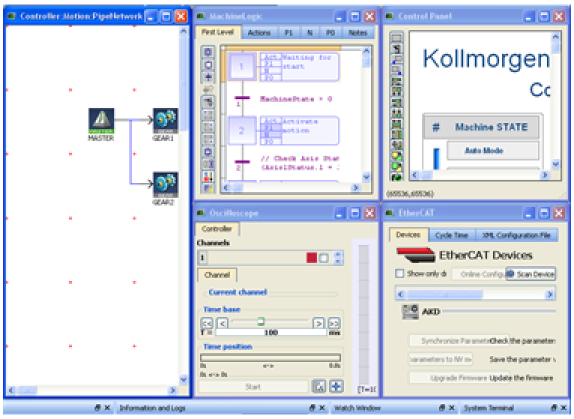
Command	Shortcut	Description
Cascade	ALT+C	Re-arrange all workspace children windows in cascade mode 
Tile	ALT+T	Re-arrange all workspace children windows in tile mode 

Table 11-28: Windows Menu Commands

10.3.5 Help Menu

Command	Description
Documentation	Opens the help system.
About	Show version numbers and other information about the KAS IDE See also "View Version Information" on page 183

Table 11-29: Help Menu Commands

10.3.6 Toolbar

The main toolbar of the KAS IDE (Tools) contains the following icons:

Icon	Description
	Create a new project
	Open an existing project
	Save the project

Icon	Description
	Print the project item currently open in the workspace For more details, refer to paragraph "Print" on page 341
	Print preview
	Undo
	Redo
	Cut
	Copy
	Paste
	Find
	Toggle Edit/Debug mode (enabled when the application is running) For more details, refer to paragraph "PLC Online Change" on page 491

Table 11-30: Main Toolbar Icons

10.3.7 Device Toolbar

Each icon provided in this toolbox has a brief explanation provided below in order to explain the functionality.

<table>
<tr><th>Icon</th><th>Description</th></tr>
<tr><td></td><td>
Show the parameters of the PLC Compilation options that can be modified for the target

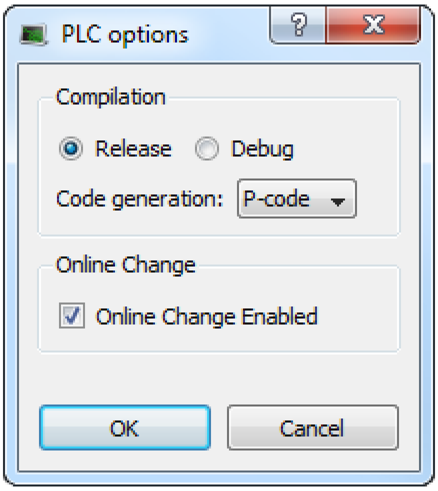


Figure 11-24: PLC Options - Online Change

For more details, see "Step 1 of 6 - Set the Compilation Options" on page 318

In the Online Change frame, it is possible to allow or forbid Online Changes. For more details, see "PLC Online Change" on page 491
</td></tr>
<tr><td></td><td>Change the Controller IP address to connect with KAS Simulator
For more details, refer to paragraph "Using the KAS Simulator" on page 345</td></tr>
</table>

Icon	Description
	Compile project
	Compile and download project
	Establish a connection with the target Controller (for possible statuses, see page 648)
	Close connection with the target Controller
	Download the application to the targeted Controller (Note that the application must **not** be running). For more details, refer to paragraph "Step 5 of 6 - Download the Application" on page 325
	Start the application. It can be either a **Warm** or **Cold** start.
	Stop the application

Table 11-31: Device Toolbar Icons

10.3.8 EtherCAT Toolbar

Each icon provided in this toolbox has a brief explanation provided below in order to explain the functionality.

Icon	Name	Description
	Online Configuration Mode	This action is only available when the KAS Runtime is connected with the target Controller. *TIP* Click this toggle button to change the mode (ON / OFF) After the scan has been performed, and your project is compliant with the physical devices on the EtherCAT network, you can activate the **Online Configuration Mode**. Online Configuration Mode allows setting up AKD drives in an EtherCAT installation. In this mode, KAS IDE communicates with the AKD drives through the integrated views of the AKD Setup Screens or with the AKD Setup Wizard. Additionally, KAS IDE displays a quick status overview of all the drives. The AKD Setup Screens allow functions such as enabling/disabling the drive, service motion, tuning, and a scope where you can plot up to six different parameters from the drive. For more details on the AKD Setup Wizard, see page 192 *NOTE* After a scan, if you choose to map a physical devices to **none**, then the Online Mode cannot be activated because your project is not compliant See also "FAQ" for a potential issue when resetting the factory parameters.

Table 11-32: EtherCAT Toolbar Icons

10.3.9 Online Change Toolbar

Each icon provided in this toolbox has a brief explanation provided below to explain the functionality.

Icon	Description
	When Online Change has been activated in the PLC options; the new code is loaded even if the application is running. See also the Warning in paragraph "How to Activate Online Change" on page 494
	Revert your changes done after an Online Change, and go back to the previous application

Table 11-33: Debug Toolbar Icons

10.3.10 Debug Toolbar

Each icon provided in this toolbox has a brief explanation provided below in order to explain the functionality.

Icon	Description
	Pause application in Cycle to Cycle mode
	Restart application in normal execution mode
	Execute a cycle step
	Step Over the next instruction: If the next instruction is a call of a function block or a sub-program, the execution passes over to the following instruction.
	Step Into the next instruction: The next step will be at the beginning of the called block (if the next instruction is **not** a call of a function block or a sub-program, then the Step Into behaves like the Step Over)
	Step Out the current block: If the current stepping position is in a called function block or a sub-program, the execution continues up to the end of the current block. Otherwise, the Step out behaves like the Step Over.

Table 11-34: Debug Toolbar Icons

10.3.11 Help Toolbar

The help toolbar contains the following icons:

Tool	Description
	Allows you to open the topic that was viewed previously
	Allows you to open the next topic in a previously viewed sequence
	Allows you to open the Help at the start page
	Lets you open the **Print** dialog so that you can send the open topic to the printer
	Allows you to add the active topic to the Favorites pane so that you can quickly access the topic in future

Tool	Description
	Allows you to toggle between hiding and showing the navigation pane in the output window
	Allows you to **expand** all elements such as togglers, drop-down effects, and expanding text effects in a topic (if they are not yet expanded)
	Allows you to **collapse** all elements such as togglers, drop-down effects, and expanding text effects in a topic (if they are expanded)
	After you perform a quick search in a topic, the search text found in the topic is highlighted. This button lets you turn the highlights off
	From the position of the current topic in your Table of Contents (TOC), opens the **previous** topic after it
	From the position of the current topic in your Table of Contents (TOC), opens the **next** topic after it

ⓘ ***TIP*** To perform a search in the active topic, use the local find (Ctrl + F)

10.4 Windows Standard Conventions

10.4.1 Windows Manipulation

The following standards apply to the KAS IDE windows:

- Move
- Resize
- Minimize
- Maximize
- Close (Alt+F4)

Press **Esc** to exit a pop-up window.

10.4.2 Mouse Manipulation

Double-click an item to open it (e.g. double-click a program in the Project Explorer to open it in the appropriate editor)

Right-click to open the menu and give access to the relevant commands (e.g. to add a variable to the Dictionary)

10.4.3 Table Manipulation

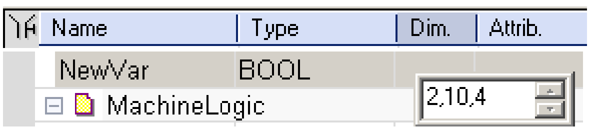
10.4.3.1 Sorting Items

If the sort feature is implemented, you can click in the column header to sort all the items according to one of the available parameters.

Click again to alternately sort in ascending or descending order.

10.4.3.2 Selecting a Cell

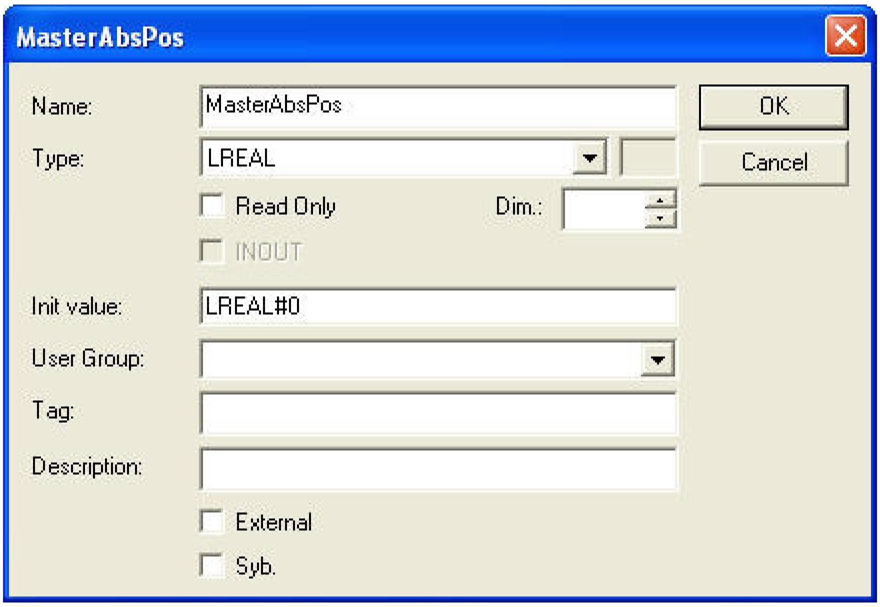
Click a cell in the table to select it. Once selected, press **F2** to edit the value.



TIP A double-click directly opens the pop-up window for editing.

10.4.3.3 Selecting a Row

When available, press the **Spacebar** to toggle the selection mode from cell to row. Then click a cell in the table to select the entire row. Once selected, press **F2** to edit the values of the row.



10.4.3.4 Resizing a Column

If you want to enlarge a column width to make more content visible, put the mouse in the table header between two columns so the cursor change to the following and move right or left to resize your column.
After this operation, you need to scroll horizontally to see the other columns.

10.5 Keyboard Shortcuts

List of accelerator keys sorted by context:

- "Common Shortcuts" (see page 663)
- FBD Editor
- FFLD Editor
- SFC Editor
- ST Editor
- Graphic Editor
- Table shortcuts
- CAM Editor

NOTE A shortcut can be unavailable depending on the context.

10.5.1 Common Shortcuts

Shortcut	Command
Alt + Return	Edit properties

Shortcut	Command
Ctrl + A	Select All
Ctrl + Alt + E	Open an Explorer window on the project folder
Ctrl + C	Copy
Ctrl + F	Find
Ctrl + F3	Find next assignment
Ctrl + F4	Close
Ctrl + F7	Build program
Ctrl + G	Display / hide grid
Ctrl + Insert	Copy
Ctrl + L	List of windows
Ctrl + N	New
Ctrl + O	Open
Ctrl + P	Print
Ctrl + S	Save
Ctrl + Shift + F6	Previous tab
Ctrl + F6	Next tab
Ctrl + V	Paste
Ctrl + X	Cut
Ctrl + Y	Redo
Ctrl + Z	Undo
Alt + Shift + O	Open from controller
Alt + Shift + C	Compare projects
Del	Delete
F1	Display the help
F2	Rename
F3	Find next
F6	Next window
F7	Build project
Shift + Del	Cut
Shift + F6	Previous window
Shift + Insert	Paste
-	Collapse
+	Expand
Page Up/Down	Scroll Page up/down
RETURN	Equivalent to double-click

Table 11-35: List of Common Shortcuts

10.5.2 Debugging

Shortcut	Command
Ctrl + Alt + F4	On line change
Ctrl + F5	Debug
Ctrl + F8	Step Out

Shortcut	Command
F4	Pause/resume
F5	Simulation
F8	Step In
F9	Set/Remove breakpoint
F11	Download
Ctrl + Shift + F4	Start/stop application
Shift + F4	One cycle
Shift + F8	Step Over

10.5.3 FBD Editor Shortcuts

- "FBD Editor (common)" (see page 665)
- "FBD Editor (when editing)" (see page 666)
- "FBD Editor (during debug)" (see page 666)

10.5.3.1 FBD Editor (common)

Shortcut	Command
Arrows	Scroll window
Ctrl + d	Display FBD execution order
Ctrl + F2	Toggle bookmark
Ctrl + page UP/DOWN	Go to previous/next section
Escape	Cancel linking/resizing/dragging if selection: deselect if no selection: select mode active
Page UP/DOWN	Scroll page up/down
Return	Equivalent to double-click
Ctrl + Shift + End	Select all items from the cursor position to the end of the document
Ctrl + Shift + Home	Select all items from the begin to the cursor position
Shift + F2	Go to next bookmark
Tab	Select next position item
Tab + shift	Select previous position item
Ctrl+F2	Toggle Bookmark (Note that you first have to select the Network header)
Shift+F2	Go to Next Bookmark
Ctrl+Shift+F2	Go to Previous Bookmark

Table 11-36: List of FBD Shortcuts

10.5.3.2 FBD Editor (when editing)

Shortcut	Command
char	Start editing a symbol (variable, constant, instance) On jump/comment/break: open dialog box to enter text
Ctrl + arrows	Align selected items
Del	Delete selection
Shift + arrows	Move selection
Shift + page UP/DOWN	Move selection (4 cells)
Spacebar Ctrl + Shift + down	Swap item style Insert blank lines at the position of the mouse

10.5.3.3 FBD Editor (during debug)

Shortcut	Command
Spacebar	Swap TRUE/FALSE boolean value
*	Lock var
/	Unlock var

10.5.4 FFLD Editor Shortcuts

- "FFLD Editor (when editing)" (see page 666)
- "FFLD Editor (during debug)" (see page 669)

10.5.4.1 FFLD Editor (when editing)

List of accelerator keys (sorted by action types)

Insert

Shortcut	Command
Ctrl+Shift+D	Insert Coil De-Energize
Ctrl+Shift+E	Insert Coil Energize
Ctrl+Shift+R	Insert Coil Reset (Unlatch)
Ctrl+Shift+S	Insert Coil Set (Latch)
Ctrl+Shift+K	Insert a positive coil to the destination cell
Ctrl+Shift+L	Insert a negative coil to the destination cell
Ctrl+Shift+C	Insert Contact NC
Ctrl+Shift+A	Insert Contact NC, Negative Transition
Ctrl+Shift+I	Insert Contact NC, Positive Transition
Ctrl+Shift+O	Insert Contact NO
Ctrl+Shift+N	Insert Contact NO, Negative Transition

Shortcut	Command
Ctrl+Shift+P	Insert Contact NO, Positive Transition
Ctrl+Shift+F	Insert Data In
Ctrl+Shift+W	Insert Data In Inverted
Ctrl+Shift+Q	Insert Data Out
Ctrl+Shift+B	Insert Wire (both)
Ctrl+Shift+H	Insert Horizontal Wire
Ctrl+Shift+V	Insert Vertical Wire
Shift+Insert	Insert Network
Ctrl+Shift+J	Insert Jump
Ctrl+Shift+T	Insert Return
Insert Key	Insert Row
F8	Insert FB

Table 11-37: List of FFLD Shortcuts

Trace

Shortcut	Command
Ctrl+J	Trace Horizontal Wire Left
Ctrl+K	Trace Horizontal Wire Right
Ctrl+M	Trace Vertical Wire Down
Ctrl+I	Trace Vertical Wire Up

Move

Shortcut	Command
Ctrl+End	Go to End of Network
Ctrl+End followed by Ctrl+End	Go to End of Ladder
Ctrl+Home or Home followed by Home	Go to Top of Network
Ctrl+Home followed by Ctrl+Home	Go to Top of Ladder
Ctrl+Page Up	Go to Previous Network
Ctrl+Page Down	Go to Next Network
Ctrl+Left Arrow or Home	Move focus to begin of row.
Ctrl+Right Arrow or End	Move focus to end of row.
Tab	Move focus cell right
Shift+Tab	Move focus cell left
Arrows	Move focus cell or scroll through ladder
Page up	Scroll 1 page up
Page Down	Scroll 1 page down

Select

Shortcut	Command
Shift+Arrow	Multiselect cells
Shift+left Arrow	Select current cell and one cell to left

Shortcut	Command
Shift+right Arrow	Select current cell and one cell to right
Ctrl+Shift+ right Arrow or Shift+End	Select from current cell to end of line
Ctrl+Shift+ End	Select from current cell to end of network (Bottom element of network and the furthest to the right)
Ctrl+Shift+ left Arrow or Shift+Home	Select from current cell to beginning of line
Ctrl+Shift+ Home	Select from current cell to beginning of network
Shift+up Arrow	Select Cell above or below when focus is on cell.
Shift+down Arrow	Select Row above or below when focus is on left rail
Ctrl+A	Select the contents of a network/rung
Ctrl+A followed by Ctrl+A	Select the entire ladder
Shift+Page Up	Selection Page-Up
Shift+Page Down	Selection Page-Down

Edit

Shortcut	Command
Ctrl+C	Copy Item
Ctrl+X	Cut Item
Ctrl+V	Paste Item
Return	Equivalent to double click
Space	Change contact or coil
Ctrl+Y	Redo
Ctrl+Z	Undo
Ctrl + mouse-wheel up or PLUS Sign (+) on the keypad	Zoom in
Ctrl + mouse-wheel down or MINUS Sign (-) on the keypad	Zoom out
Ctrl+S	Save
Esc or Shift-ESC	Close the rename widget. Exit Dialog

Find

Shortcut	Command
F3 or Ctrl+F	Find
Ctrl+F3	Find Next
Alt+F3	Find and Replace

Delete

Shortcut	Command
Delete Key	Delete cell, selection, or row
Shift+Delete	Delete Network

Bookmark

Shortcut	Command
Ctrl+F2	Toggle Bookmark (you must first select the Network header)
Shift+F2	Go to Next Bookmark
Ctrl+Shift+F2	Go to Previous Bookmark

10.5.4.2 FFLD Editor (during debug)

Shortcut	Command
Spacebar	Swap TRUE/FALSE boolean value
*	Lock var
/	Unlock var

10.5.5 SFC Editor Shortcuts

Shortcut	Command
?	Show/Hide notes
arrows	Move caret
Page UP/DOWN	Scroll page up/down
Return	Equivalent to double-click
Shift + arrows	Select multiple cells
Shift + Home	Select from left to caret
Shift + Page Up/Down	Selection Page Up/down
b or B	Insert macro body
c or C	Insert convergence
Ctrl + return	Edit reference
d or D	Insert divergence
Del	Delete selection
i or I	Insert step initial
j or J	Insert jump
m or M	Insert macro
s or S	Insert step
Spacebar	Swap item style
t or T	Insert transition
x or X	Insert the left side corner of a divergence/convergence

Table 11-38: List of SFC Shortcuts

10.5.6 ST Editor Shortcuts

- "ST Editor (common)" (see page 670)
- "ST Editor (when editing)" (see page 670)
- "ST Editor (during debug)" (see page 670)

10.5.6.1 ST Editor (common)

Shortcut	Command
Arrows	Move caret
Shift + arrows	Selection
Ctrl + left/right arrow	Go to previous/next word
Shift + Ctrl + left/right arrow	Select previous/next word
Ctrl+F2	Toggle Bookmark (Note that you first have to select the Network header)
Shift+F2	Go to Next Bookmark
Ctrl+Shift+F2	Go to Previous Bookmark

Table 11-39: List of ST Shortcuts

10.5.6.2 ST Editor (when editing)

Shortcut	Command
.	Select member of a structure or instance
Ctrl + Spacebar	Auto completion or Open the variable selector dialog
Ctrl + J	Auto completion or Open the variable selector dialog (an alternative method)

10.5.6.3 ST Editor (during debug)

Shortcut	Command
*	Lock variable
/	Unlock variable
Shift + double-click	Force a variable
Spacebar	Toggle boolean value or bring the dialog to force, lock, unlock the variable (equivalent of Shift + double click)

10.5.7 Graphic Editor Shortcuts

Shortcut	Command
Ctrl + mouse-wheel down or Shift+MINUS Sign (-) on the numerical keypad	Zoom out
Ctrl + mouse-wheel up or Shift+PLUS Sign (+) on the numerical keypad	Zoom in
Arrow	Scroll
Ctrl + F2	Toggle bookmark
Ctrl + arrow	Align on main selected item
Del	Delete selection
Escape	Cancel resizing/dragging if selection: unselect if no selection: select mode active

Shortcut	Command
Ctrl + Shift + End	Select all items from the cursor position to the end of the document
Ctrl + Shift + Home	Select all items from the begin to the cursor position
Shift + F2	Go to next bookmark
Shift + Page UP/DOWN	Offset selection
Shift + Arrow	Move selection
Tab	Select next position item
Tab + shift	Select previous position item

Table 11-40: List of Graphics Editor Shortcuts

10.5.8 Table Shortcuts

Shortcut	Command
Arrows	Move selection
Shift + Tab	Move selection to the left
Spacebar	Line selection/cell selection
Tab	Move selection to the right
Ctrl + Home/End	Go to previous/next group

Table 11-41: List of Table Shortcuts

10.6 Bookmarks

Bookmarks are used for navigating in a document. You can insert bookmarks anywhere in a document. Then you can jump from one bookmark to another with a single command for browsing the document. Bookmarks are supported in all program editors and the Variable editor.

Below are the available commands for using bookmarks:

`Ctrl + F2` Toggle the bookmark at the current position
`Shift + F2` Go to the next bookmark

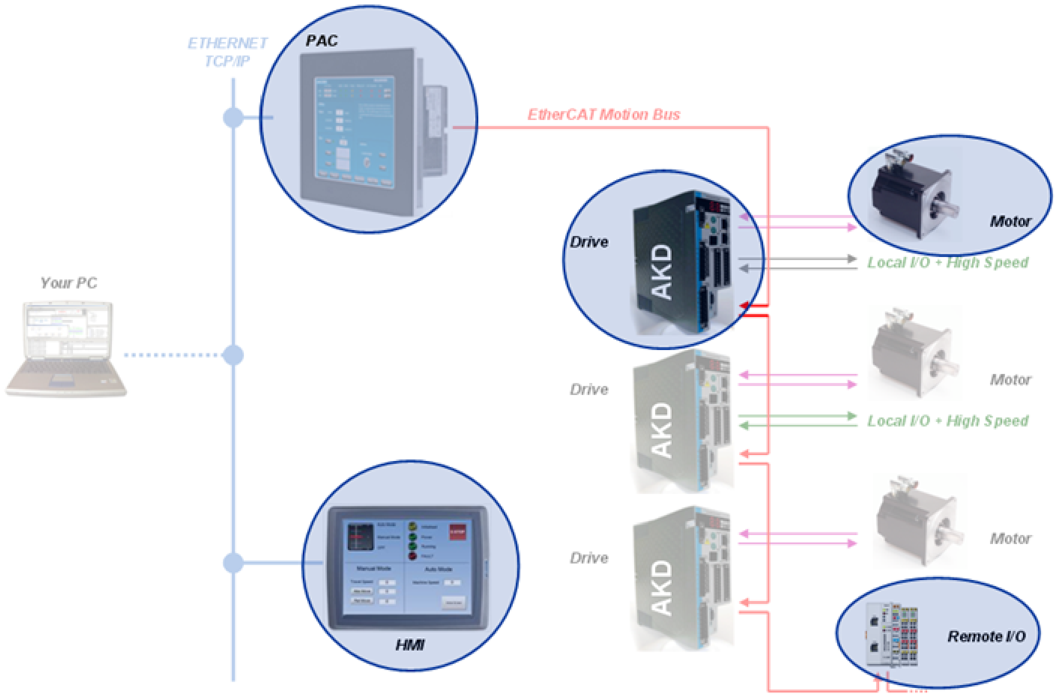
According to the type of document, the possible locations for a bookmark are:

- In the text editor, a bookmark is placed on a line of text.
- In the SFC editor, a bookmark is placed on an SFC symbol (step, transition, jump...).
- In the FBD editor, a bookmark is placed on any FBD object (not on a line).
- In the FFLD editor, a bookmark is placed on a rung header.
- In the Variable editor, a bookmark is placed on any line of the grid (variable or group).

NOTE Bookmarks are valid only while the editing window is open; they are not stored in the document when the window is closed.

This page intentionally left blank.

11 KAS Component Manuals



11.1 HMI

HMI part number	Description	Tech. Manual
AKI-CDA-MOD-04T-000	Graphical Display 4.3" TFT, LCD, 16.7M Colors, Touch Screen	
AKI-CDA-MOD-07T-000	Graphical Display 7" TFT, LCD, 262K Colors, Touch Screen	
AKI-CDA-MOD-10T-000	Graphical Display 10.4" TFT, LCD, 16.7M Colors, Touch Screen	
AKI-CDB-MOD-07T-000	Graphical Display 7" TFT, LCD, 262K Colors, Touch Screen	
AKI-CDB-MOD-12T-000	Graphical Display 12.1" TFT, LCD, 262K Colors, Touch Screen	
AKI-CDB-MOD-15T-000	Graphical Display 15.4" TFT, LCD, 262K Colors, Touch Screen	
AKI-CDC-MOD-12T-000	Graphical Display 12.1" TFT, LCD, 16M Colors, Touch Screen	
AKI-CDC-MOD-15T-000	Graphical Display 15.4" TFT, LCD, 16M Colors, Touch Screen	
AKI-CDC-MOD-21T-000	Graphical Display 21.5" TFT, LCD, 16M Colors, Touch Screen	

Table 12-1: List of KAS HMI

NOTE

Refer to our Web site for up-to-date information and material that are available in the **Automation Component Solutions** section.
Note that you first need to log in before accessing KAS Literature.
Scan this QR code to access our web site with your mobile device.

11.1.1 HMI Accessories

Various accessories are available to compliment the HMI, including:

- Programming Cable, RS232 to HMI Terminal RS232
- Programming Cable, USB to Ethernet
- Key cover for 5.7" Graphical Display
- Key cover for 10.4" Graphical Display
- Touch cover for 3.5" Touchscreen Graphical Display
- Touch cover for 5.7" Touchscreen Graphical Display
- Touch cover for 10.4" Touchscreen Graphical Display
- Touch cover for 15.1" Touchscreen Graphical Display
- 512 MB Compact Flash Industrial Grade
- 1 GB Compact Flash Industrial Grade

11.2 Controllers - PAC

PAC part number	Description	Tech. Manual
AKC-PLC-C1-224-00N-00-000	Box Controller, Celeron 1.2GHz CPU, 2G RAM, 4G CF, 128KB NVRAM	
AKC-PLC-D2-224-00N-00-000	Box Controller, Dual Core 2.26GHz CPU, 2G RAM, 4G CF, 128KB NVRAM	
AKC-PNC-C1-224-10N-00-000	10" Panel Controller, Celeron 1.2GHz CPU, 2G RAM, 4G CF, 128KB NVRAM	
AKC-PNC-C1-224-15N-00-000	15" Panel Controller, Celeron 1.2GHz CPU, 2G RAM, 4G CF, 128KB NVRAM	
AKC-PNC-D1-224-15N-00-000	15" Panel Controller, C2D 1.86GHz CPU, 2G RAM, 4G CF, 128KB NVRAM	
AKC-PNC-D1-224-17N-00-000	17" Panel Controller, C2D 1.86GHz CPU, 2G RAM, 4G CF, 128KB NVRAM	
AKC-RMC-D2-224-00N-00-000	Rackmount Controller, Dual Core 2.26GHz CPU, 2G RAM, 4G CF, 128KB NVRAM	

Table 12-2: List of KAS PAC

11.2.1 NVRAM

KAS uses the NVRAM (non-volatile memory) to save "Retain Variables" (see page 79).

Hardware Type	NVRAM Size Allocation
Old generation PAC	32 Kbytes
New generation PAC	128 Kbytes
Simulator	128 Kbytes
AKD PDMM	32 Kbytes

Table 12-3: NVRAM Size Depending on Hardware

IMPORTANT Part of the NVRAM allocation is reserved to store some internal data (144 bytes).
As a consequence, not all the complete physical NVRAM is available for the

IMPORTANT retain variables.

If the size is big enough, KAS updates the non-volatile memory to store the retain variables values. This operation is performed in the background every 20 seconds (frequency increases to each 2 seconds when the application is running), and when you shutdown the application.

NOTE Using the retain variables is highly cycle time consuming. As a consequence, Kollmorgen strongly recommends to carefully monitor the system load with the TraceTimes command.

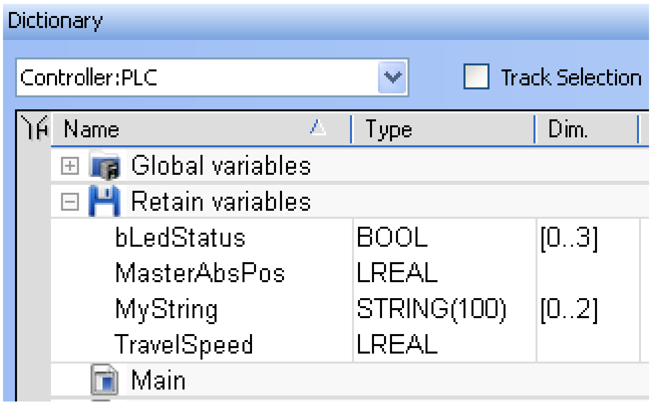
11.2.1.1 How can I check the NVRAM space is enough to store my retain variables?

To calculate the NVRAM space, you have to add the size of each retained variable according to:

- its data type as described here
- the numbers of elements in case you declare the variable as an array

Do not forget to add the 144 bytes as stated in the Warning above.

In the following example, the total size is: **3684 bits** (which is less than 0.5 Kbytes)



Variable	Size / element	Element no.	Total Size / variable
bLedStatus	1 bit	4	4
MasterAbsPos	64 bits	1	64
MyString	800 bits (100 bytes)	3	2400
TravelSpeed	64 bits	1	64
Internal data	*1152 bits (144 bytes)*	*na*	*1152*

11.3 Remote Input/Output (I/O Terminals)

KAS remote I/Os provide a complete spectrum of bus couplers, digital and analog inputs, digital and analog outputs, stepper, counter, and thermocouple modules.

Related Documents

Please find in the table below the list of each I/O component available.

I/O terminal part number	Description	Tech.Manual
AKT-ECT-000-000	ETHERCAT Coupler	
AKT-DNH-008-000	8 Channel Digital Input Module, 24 VDC 0.2ms	
AKT-DN-008-000	8 Channel Digital Input Module, 24 VDC 3ms	
AKT-DNH-004-000	4 Channel Digital Input Module, 24 VDC 0.2ms	
AKT-DN-004-000	4 Channel Digital Input Module, 24 VDC 3ms	
AKT-DT-008-000	8 Channel Digital Output Module, 24 VDC 0.5A	
AKT-DT-004-000	4 Channel Digital Output Module, 24 VDC 0.5A	
AKT-DT-2RT-000	2 Channel Relay Output Module, 250 V AC 2.0A Rel.2NO Pot.-Free	
AKT-AN-420-000	4 Channel Analog Input Module, 0-20 mA	
AKT-AN-410-000	4 Channel Analog Input Module, 0-10 VDC	
AKT-AN-820-000	8 Channel Analog Input Module, 0-20 mA	
AKT-AN-810-000	8 Channel Analog Input Module, 0-10 VDC	
AKT-AN-200-000	2 Channel Thermocouple Input Module	
AKT-AN-400-000	4 Channel Thermocouple Input Module	
AKT-AT-220-000	2 Channel Analog Output Module, 0-20 mA	
AKT-AT-420-000	4 Channel Analog Output Module, 0-20 mA	
AKT-AT-410-000	4 Channel Analog Output Module, 0-10 VDC	
AKT-AT-820-000	8 Channel Analog Output Module, 0-20 mA	
AKT-AT-810-000	8 Channel Analog Output Module, 0-10 VDC	
AKT-EM-000-000	End Module	
AKT-IM-000-000	Isolation Module	
AKT-PS-024-000	Power Supply, 24 VDC	
AKT-PSF-024-000	Fused Power Supply with diagnostics, 24 VDC	

Table 12-4: List of KAS I/O Terminals

11.4 Drives

This section details the following drives:

AKD part number	Description
AKD-B00106	120/240 VAC 1.5A Drive
AKD-B00306	120/240 VAC 3A Drive
AKD-B00606	120/240 VAC 6A Drive
AKD-B01206	120/240 VAC 12A Drive
AKD-B02406	120/240 VAC 24A Drive
AKD-B04806	120/240 VAC 48A Drive
AKD-B00107	240/480 VAC 1.5A Drive
AKD-B00307	240/480 VAC 3A Drive
AKD-B00607	240/480 VAC 6A Drive
AKD-B01207	240/480 VAC 12A Drive
AKD-B02407	240/480 VAC 24A Drive

Table 12-5: List of AKD Drives

Related Documents

For further information on drives, refer to the following manuals:

Drives Guide	Description
AKD Quick Start	Contains all information needed to safely install and setup an AKD drive
AKD and AKD PDMM Installation Manual	Covers the most important points to install the drive hardware and software Provides instructions for basic drive setup and connection to a network
AKD User Manual	Describes how to use your drive in common applications. It also provides tips for maximizing your system performance with the AKD
AKD Accessories Manual	Includes technical data and dimensional drawings of accessories such as cables, brake resistors, and mains supplies
AKD EtherCAT Manual	Describes the installation, setup, range of functions, and software protocol for the EtherCAT AKD product series
AKD CANopen Communication	This manual includes setup information for the CAN interface and describes the CANopen profile
AKD EtherNet/IP Communications Manual	This manual contains information for using an AKD EtherNet/IP drive.
AKD EtherNet/IP with RSLogix Manual	This manual contains information for using an AKD EtherNet/IP drive with RSLogix.

Drives Guide	Description
AKD Profinet Communication Manual	This manual contains information for using an AKD Profinet drive.
AKD SynqNet Communication Manual	This manual contains information for using an AKD SynqNet drive.
AKD HMI Modbus Communication Manual	This manual contains information on communication between an AKD and HMI through Modbus.
AKD sercos III Communication Manual	This manual contains information for using an AKD sercos III drive.
S300 Reference Documentation	Kollmorgen website that gives access to all **S300** manuals

Table 12-6: List of Drives' Manuals

NOTE

The AKD manuals are located under:
C:\Program Files\Kollmorgen\AKD WorkBench 1.0.x.y\WebHelp
(x.y must be replaced with the version number)
(this location differs if you chose another location when installing AKD).

This page intentionally left blank.

12 Troubleshooting

12.1 FAQs

Why does the Installer not Start when I insert the CD?

Your Autorun feature may be deactivated. Open an Explorer window to see the autorun.exe file and use the Run command in the contextual menu to manually start the installer.

Why does the KAS IDE not display all the items in the Project Explorer when I create a new project based on a template?

A side effect with some remaining files that were not deleted properly can interfere with your new project. To fix this issue:

- Close your current project without saving
- Open Windows Explorer and go to **C:\Documents and Settings\(user)\Local Settings\Application Data\Kollmorgen\KAS\Project** , where “(user)” is the Windows' username you are currently logged in with
- Delete all the remaining files and folders
- You can now create your new project

How can I restore IPC Backup Image?

This procedure (as well as Backup creation) is fully described in the chapter Getting Started.

How can I prevent file corruption in my CompactFlash memory?

File system corruptions happens when the IPC is not properly shutdown. It is strongly recommended that one of the possibilities be put in place:

- Use a UPS (uninterruptible power supply) solution
- Rely on Microsoft Enhanced Write Filter (EWF).

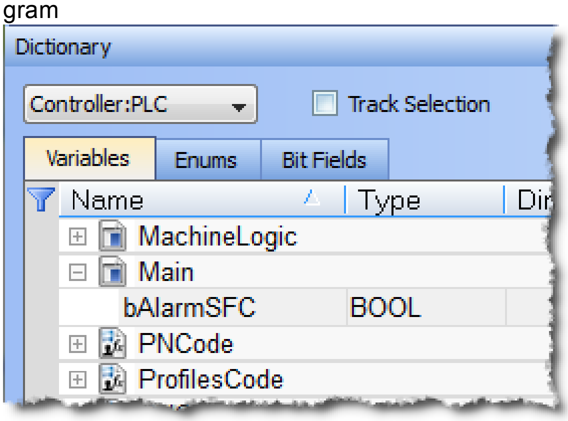
How can I download new Firmware to my AKD Drive?

How can I control the time execution for an SFC step?

When you want to check the maximum time execution for an SFC step, you have to program this action based on the SFC alarm capability.

To show this status, you have to:

- In the Dictionary, declare a Boolean PLC variable linked to the related SFC program



- Add the instruction in the Actions tab related to the SFC step, with first parameter set to **A** (for Alarm) as shown below

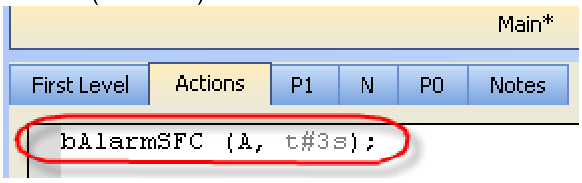


Figure 13-1: SFC Step - Timeout Alarm

How can I fix the Library Access issue?

If you open a project containing a link on a library which is no longer available, a warning is displayed. To fix this issue, refer to paragraph "What happens when a library no longer exists?" on page 269

How are fieldbuses connected to the KAS Runtime?

As depicted in figures found in paragraph "Different Implementations" on page 69, the fieldbus serially links all the drives to the industrial PC.

NOTE When the KAS IDE is used to deploy an automation system on a master drive (also known as programmable drive), the fieldbus serially links all the drives to the master drive.

Is EtherCAT limited to Master/Slave applications?

No. As with every real-time Industrial Ethernet system, one device (the master) has to be in charge of the network management and organize the Medium Access Control. With EtherCAT, Slave-to-Slave communication is supported in two ways:

- topology dependent within one communication cycle ("upstream" device talks to "downstream" device)
- topology independent within two cycles.

Since EtherCAT is so much faster than competing systems, slave-to-slave communication using two cycles is faster, too.

How does Kollmorgen Automation Suite communicate with a Host?

As described in paragraph "Communication and Fieldbus" on page 64, KAS can communicate with outside world through Ethernet, Profibus, CANopen, DeviceNet.

Why is the PLC execution rate not the same as the EtherCAT rate with the KAS Simulator?

When the application runs on the KAS Simulator, the PLC execution rate is approx. 10 milliseconds. KAS Simulator cannot execute the PLC programs faster because Windows is not able to handle timing less than 10ms.

When can I expect my SDO command to be completed?

If you need to rely on SDO communication to set the parameter of an EtherCAT device, you can do this with the ECATWriteSdo FB .

Being asynchronous and based on the EtherCAT mailbox, the SDO communication is not deterministic. So the EtherCAT master uses a polling mode to ensure the SDO command is completed. Note that in operational mode, this polling is performed every 50 cycles [1] . As a consequence, you can expect the acknowledgement of your SDO command usually before less than 100 ms. So, a good practice is to set the update rate for SDO communication in your PLC application each 25 cycles.

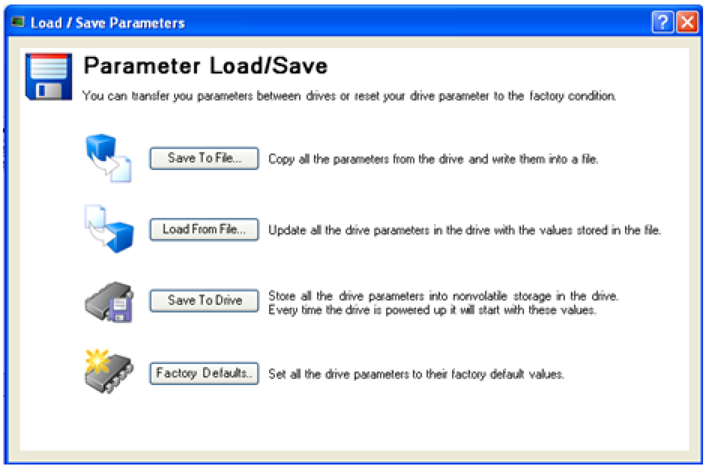
See also "EtherCAT Motion Bus Concepts" on page 156 for more details.

Why does Online Configuration Mode not work after I reload the drive's factory default parameters?

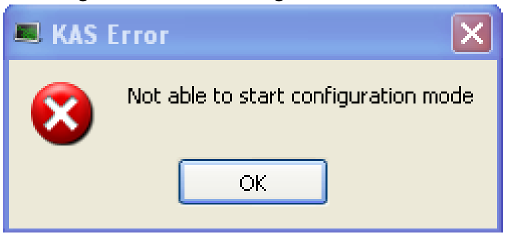
Description

This issue occurs when you perform the following

- Connect to the controller and download your application
- In the project explorer, open the EtherCAT properties
- Click the Online Configuration Mode
- In the project explorer, right-click on the AKD_1 and select **Load/Save Parameter…**
- Then select the **Factory Defaults…** command to reset the drive to its default parameters



- Clicking the Online Configuration Mode leads to the following error



Reason

If you set the drive to its default parameters, then all the AKD parameters are restored and the unique ID (FBUS.PARAM03) used to identify the drive is lost.

Solution

You have to perform a new scan operation after setting the parameters to its default values

[1] To avoid overloading the controller, this rate is set according to the communication load, as well as the duration the AKD takes to process commands

TIP You can also clear the **Write a unique ID** option in the XML configuration tab (for more details, see page 215)

How can I fix security issues?

If you encounter any security issues during execution of Kollmorgen Automation Suite, refer to your IT department to set your proxy properly.

Firewall

You may have to define your firewall settings to allow accessing the IP addresses used by KAS (for instance, IP address of the target system, or localhost IP address for the KAS Runtime Simulator: 127.0.0.1).

Port numbers

Port numbers have to be set properly in your firewall settings to avoid any trouble during communication, such as when downloading the application to the target, or plugging a probe to the softscope. Kollmorgen strongly recommends opening port numbers over 1024, as well as the range 502 to 520.

What is the Fast Input?

The Fast Input allows an application to get information about the occurrence of an external event at a higher resolution than the cycle time.

For more details, refer to paragraph "Fast Inputs with Pipe Network" on page 498

How do I implement feedback?

There are two kinds of feedback:

Primary feedback
With a S300 drive you can use a resolver for primary feedback.

Secondary feedback
If a secondary feedback is required with your S300 drive, you can use a BiSS feedback device.

If you use the same setup with an S300 drive, the S300's EXTPOS parameter has to be set to **-11**.
IMPORTANT: do not omit the negative sign!

To use the secondary feedback, you have to rely on a SAMPLER Pipe Network block. To configure the block use the MLSmpConnectEx function. The arguments must be:

- The Pipe Network block ID being configured
- The string 'EtherCATDriver'
- A string of the form '<EtherCAT address>: Position actual value 2'. For the first EtherCAT node, this string would be '1001:Position actual value 2'

How do I implement Torque Feed-forward?

Current drives that support torque feed-forward are: S300 and AKD drives.

To use torque feed-forward, you have to rely on a CONVERTER Pipe Network block. To configure the block use the MLCNVConnectEx function. The arguments must be:

- The Pipe Network block ID being configured
- The ID of the axis to which the torque feed-forward is applied

- The constant EC_ADDITIVE_TORQUE_VALUE
- An ignored integer value (usually set to zero)

For more details, refer to the three following links:

- Torque Feed-forward
- Guidelines for Choosing feed-forward Control in Industrial Applications
- Tuning with Feed-forwards
- Measurement-based Feed-forward Tuning

How is Torque Feed-forward Scaled?

If I measure a number e.g. 500 as an input at the converter block which is connected with the PDO object (Additive Torque Value 0x60B2), how many Amps are fed in the current loop at the AKD?

Current loop feed-forward value = Rated current x IL.KBUSFF x input at converter block / 1000

For example, with an AKD where:

Rated current:	3 A
IL.KBUSFF:	1.0
Additive Torque (PDO object):	500 Units

Then

IL.FF = 3 x 1.0 x 500 / 1000

IL.FF = 1.5 A

How many axes can the KAS IDE manage in 1 ms?

This number is mainly dependent upon the application and your PC's computing power. An average number would be 20 axes/ms

What are the limitations with cams?

There is no limitation with the cams, the number of cams, the number of cam points, etc.. The limitation is only given by the processing power of your PC.

If a variable is associated with an I/O point value, would it get automatically updated?

Yes, I/O points represent the state of real world values.

How can I see the CPU load between the PLC and motion parts?

This procedure allows you to determine if your controller is overloaded due to the PLC program or motion system load.

You can use the Softscope and the **Trace Times** button to display the following CPU loads:

- CycleJitter (microseconds)
- Motion execution time (microseconds)

- PLC execution time (microseconds)
- Real Time Margin (microseconds)

To view the load, do as follows:

- Open the Softscope
- Plug four probes to any kind of data (see procedure here)
- In the Control Panel, click the TraceTimes button

How does the Pipe Network engine interact with a PLC program?

This item is explained here

How can I check the if there is enough NVRAM space to store my Retain Variables?

For explanation, see page 676

Where can I get the latest User Manuals?

The documentation is embedded in Kollmorgen Automation Suite package in e-format.

See also "Learning Kollmorgen Automation Suite" on page 44

How can I keep track of my latest searches in the Online Help?

When you enter a search criteria, you can save it as a favorite for further re-uses. For more details, see page 42

Why can I not move to the next animated lesson when I click the button?

If you encounter some issues when moving to the next lesson, you have to check the flash settings on your computer, as follows:

- Open the animated lessons in the Internet Explorer window
- Do a right-click somewhere on the animation and select the **About Adobe Player** command
- A new window comes up
- Under **Support** (located at the right-side of the window), select **Settings Manager**
- Then, select **Global Security Settings panel** (located at the left-side of the window)
- Check the **Always allow** (the radio button is located in the drawing)
- Close the window and reload the animated lessons in your Internet Explorer window
- Try again the button to move to the next animated lesson

12.2 EtherCAT Coupler Error Handling And Diagnosis

This section provides information about the diagnostic LEDs for the ETHERCAT Coupler.

NOTE This section is an excerpt of the EtherCAT Coupler Technical Manual.

12.2.1 EtherCAT Diagnostic LEDs

After switching on, the ETHERCAT Bus Coupler immediately checks the connected configuration. Error-free start-up is indicated when the red I/O ERR LED goes out. If the I/O ERR LED blinks, an error in the area of the terminals is indicated. The error code can be determined from the frequency and number of blinks. See below for more information.

The ETHERCAT Bus Coupler has respectively a green and yellow LED at the RJ45 plug sockets, which indicate the state of the fieldbus (Figure 4.1). The RUN and ERROR LEDs (upper middle) indicate the state of the EtherCAT State Machine.

On the upper right hand side of the Bus Couplers are two more green LEDs that indicate the supply voltage. The left hand LED indicates the presence of the 24 V supply for the Bus Coupler. The right hand LED indicates the presence of the supply to the power contacts.

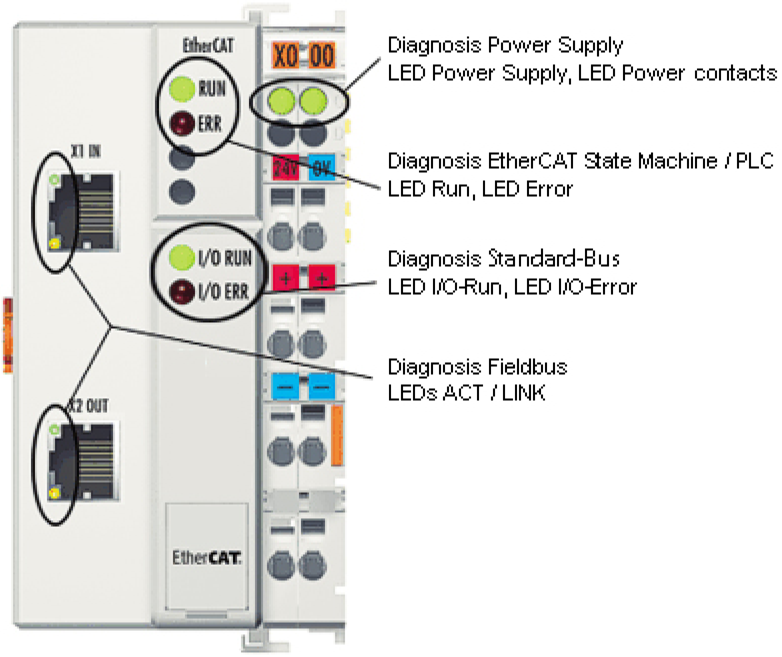


Figure 13-2: ETHERCAT Coupler Diagnostic LEDs

12.2.2 EtherCAT LED Power Supply Diagnosis

LED		Display	Description
Power Supply	Green	Off	No operating voltage connected
		On	24 VDC operating voltage connected
Power Contacts	Green	Off	No 24 VDC power connected to the power contacts
		On	24 VDC power connected to the power contacts

12.2.3 EtherCAT LED Off Power Supply Diagnosis

LEDs	
Left Green LED Off	Bus coupler has no power
Right Green LED Off	No 24 VDC power connected to the power contacts

12.2.4 LEDs for EtherCAT State Machine/PLC Diagnosis

LEDs		Display	Status	Description
Run	Green	Off	Init	State of the EtherCAT State Machine: INIT = Initialization
		Blinking	Pre-operational	State of the EtherCAT State Machine: PREOP = Pre-Operational
		Single Flash	Safe operational	State of the EtherCAT State Machine: SAFEOP = Safe-Operational
		On	Operational	State of the EtherCAT State Machine: OP = Operational
		Flashes	Bootstrap	State of the EtherCAT State Machine: BOOT = Bootstrap (Update of the coupler firmware)
Error	Red	Off	—	No errors
		Blinking	Err-Operational No Communication	PLC error / Lost frames

12.2.5 LEDs for Fieldbus Diagnosis

LEDs		Display	Status	Description
LINK (X1 IN)	Yellow	Off		No connection with the previous EtherCAT client
		On	Linked	Previous EtherCAT-client connected
ACT (X1 IN)	Green	Blinking	Active	Communication with the previous EtherCAT client
		Off		No connection with the previous EtherCAT client
		On		No communication with the previous EtherCAT client
LINK (X2 OUT)	Yellow	Off	Linked	Next EtherCAT client connected
		On	Active	Next EtherCAT client connected
ACT(X2 OUT)	Green	Blinking	Active	Communication with the next EtherCAT client
		Off		No connection with the next EtherCAT client
		On		No communication with next previous EtherCAT client

12.2.6 LEDs for Standard-Bus Diagnosis

LEDs		Display	Status	Description
I/O-Run	Green	Off	—	Standard-Bus inactive
		On	—	Standard-Bus active

LED Red; I/O Error	Error Code Argument	Description	Remedy
Persistent, continuous blinking		EMC problems	• Check power supply for overvoltage or under voltage peaks • Implement EMC measures • If a Standard-Bus error is present, it can be localized by a restart of the coupler (by switching it off and then on again)

LED Red; I/O Error	Error Code Argument	Description	Remedy
1 Pulse	0	EEPROM checksum error	Set manufacturer’s setting with the configuration software
	1	Code buffer overflow	Insert fewer Bus Terminals. The programmed configuration has too many entries in the table Software update required for the Bus Coupler
	2	Unknown data type	Software update required for the Bus Coupler
2 Pulses	0	Programmed configuration has an incorrect table entry	Check programmed configuration for correctness
	n (n > 0)	Table comparison (Bus Terminal n)	Incorrect table entry
3 Pulses	0	Standard-Bus command error	• No Bus Terminal inserted • One of the Bus Terminals is defective; halve the number of Bus Terminals attached and check whether the error is still present with the remaining Bus Terminals. Repeat until the defective Bus Terminal is located.
4 Pulses	0	Standard-Bus data error, break behind the Bus Coupler	Check whether the n+1 Bus Terminal is correctly connected; replace if necessary.
	n	Break behind Bus Terminal n	Check whether the Bus End Terminal is connected.
5 Pulses	n	Standard-Bus error in register communication with Bus Terminal n	Exchange the nth Bus Terminal
14 Pulses	n	nth Bus Terminal has the wrong format	Start the Bus Coupler again, and if the error occurs again then exchange the Bus Terminal
15 Pulses	n	Number of Bus Terminals is no longer correct	Start the Bus Coupler again. If the error occurs again, restore the manufacturers setting using the configuration software
16 Pulses	n	Length of the Standard-Bus data is no longer correct	Start the Bus Coupler again. If the error occurs again, restore the manufacturers setting using the configuration software

12.3 Connect Remotely

You can connect to an AKD PDMM or AKC from an external network using VPN or other tunneling protocol. To do so you must open the following ports which are used by KAS to connect to a controller.

Port	Component/Protocol using the port
9900	Command Server
4002	Probes
4003	Copalp
502	Modbus TCP/IP
80	HTTP
23	Telnet (optional)

12.4 How to Give Feedback

- After every crash of either the development tools or the Windows part of the Runtime engine, the KAS IDE proposes you to send via email a crash report back to the development team of Kollmorgen Automation Suite.
- An automatic tool has been designed to regularly check crash report email account and to populate a crash report database with all new incoming crashes. This database allows us to make statistics on received crash reports and to focus on solving the most frequent ones. After being reported in our database, the status with all its relevant information will be available on the intranet.

This page intentionally left blank.

13 Annexes

13.1 List of Figures

13.2 List of Tables

13.3 List of How-Tos

13.3.1 PLC Code How-Tos

- Declare an Array
- Control an SFC Child
- Draw SFC divergences
- Create SFC Parallel Branches
- Toggle a FBD Connection to make it Negative
- Change a Link in the Pipe Network
- Create a PLCopen Axis
- Read Output of a MC Function Block in ST
- Sort the Variables in the Dictionary
- Understand the Location Details in the Find and Replace window

13.3.2 EtherCAT Fieldbus How-Tos

- Map EtherCAT Devices
- Map I/Os to PLC variables

13.3.3 Advanced Motion How-Tos

- Use Fast Inputs in PLC Programs
- Implement the Torque Feed-forward

13.3.4 Run the Application How-Tos

- Choose the Appropriate Level for Log Messages
- Plug a Probe in the Softscope
- Plug Motion Variables in the Softscope
- Plug PLC Variables in the Softscope
- Export the Softscope Data
- Set Breakpoints
- Activate Online Change
- Change Priority among Programs
- Specify the Duration of a Cycle

13.3.5 Hardware How-Tos

- Download a new Firmware to my AKD Drive
- Check the NVRAM space is enough to store my retain variables
- Download your Application on the HMI device (AKI)
- Download your Application on the PAC (AKC)

This page intentionally left blank.

Acronyms

Term	Definition	Description
AKA	Also Known As	Provides an alias to a name
AKC	Advanced Kollmorgen Controller	see page 675
AKD	Advanced Kollmorgen Drive	see page 678
AKI	Advanced Kollmorgen Interface	see page 674
AKT	Advanced Kollmorgen Terminal	see page 676
ANSI	American National Standards Institute	ANSI is a private, nonprofit organization that oversees the development of voluntary consensus standards for products, services, processes, systems, and personnel
ASFB	Application Specific Function Block	Library that can be written to provide a specific application task
ASIC	Application-Specific Integrated Circuit	An ASIC is an integrated circuit (IC) customized for a particular use, rather than for general-purpose use. Modern ASICs often include entire 32-bit processors, memory blocks including ROM, RAM, EEPROM, Flash and other large building blocks
BiSS	Bi-directional Serial Synchronous interface	An open-source communication protocol for feedback devices. With BiSS, all of the computation for interpolation in regard to position occurs on the ASIC directly in the encoder
CAM	Computer-Aided Manufacturing	CAM means the use of a wide range of computer-based software tools that assist engineers and CNC machinists in the manufacture or prototyping of product components
CAN	Controller Area Network	CAN is a broadcast, differential serial bus standard developed for connecting electronic control units. Each node is able to send and receive messages, but not simultaneously.
CF	Compact Flash	CF is a mass storage device format used in portable electronic devices
CIP	Common Industrial Protocol	The Common Industrial Protocol allows complete integration of control with information, multiple CIP Networks, and Internet technologies
CRC	Cyclic Redundancy Check	A CRC is a type of function that takes as input a data stream of any length and produces as output a value of a certain fixed size. The term CRC is often used to denote either the function or the function's output. A CRC can be used as a checksum to detect accidental alteration of data during transmission or storage

Term	Definition	Description
CSV	Comma-Separated Values	CSV file format is a file type that stores tabular data
DMA	Dynamic Memory Allocation	DMA is the allocation of memory storage for use in a computer program during the run-time of that program. It can be seen also as a way of distributing ownership of limited memory resources among many pieces of data and code
ENI	EtherCAT Network Information	A network configuration file in XML format, the ENI file describes the network topology, the initialization commands for each device, and commands which have to be sent cyclically. The ENI file is provided to the master, which sends commands according to this file. The KAS IDE creates the ENI file after a network discovery, which can be exported or imported. A scan and compile should be redone, if the network changes, in order to regenerate the ENI file.
ERP	Enterprise Resource Planning	ERP integrates (or attempts to integrate) all data and processes of an organization into a unified system
ESI	EtherCAT Slave Information	A device description in XML format. This is a fixed file provided by the supplier of a given EtherCAT device. The ESI file contains information about the device's functionality and settings. EtherCAT device vendors must provide an ESI file, which is used by the KAS IDE to compile the network information (e.g. process data structures, initialization commands) and create the ENI file.
FBD	Function Block Diagram	A function block diagram describes a function between input variables and output variables. A function is described as a set of elementary blocks
FFLD	Free Form Ladder Diagram	Free Form Ladder logic is a method of drawing electrical logic schematics. It is now a very popular graphical language for programming Programmable Logic Controllers (PLCs). It was originally invented to describe logic made from relays. The name is based on the observation that programs in this language resemble ladders, with two vertical "rails" and a series of horizontal "rungs" between them
FoE	File over EtherCAT	This very simple protocol, similar to TFTP, enables access to any data structure in the device. Standardized firmware upload to devices is therefore possible, irrespective of whether or not they support TCP/IP

Term	Definition	Description
FPGA	Field-Programmable Gate Array	FPGA is a semiconductor device that can be configured by the customer or designer after manufacturing; hence the name "field-programmable"
FSoE	FailSafe over EtherCAT	The protocol FSoE was specified for the transmission of safety relevant data. It is used to send input information of safety sensors (such as safety light curtains or emergency stop buttons) to a safety logic controller. Based on these inputs, this controller computes the commands for the safe outputs (such as contactors or safety relevant drives) and thus controls the safety functionality of the machine
GUI	Graphical User Interface	A GUI is a type of user interface which allows people to interact with a computer and computer-controlled devices
HMI	Human-Machine Interfaces	Also known as computer-human interfaces (CHI), and formerly known as man-machine interfaces, they are usually employed to communicate with PLCs and other computers, such as entering and monitoring temperatures or pressures for further automated control or emergency response
IC	Integrated Circuits	Miniaturized electronic circuits (consisting mainly of semiconductor devices, as well as passive components) that have been manufactured in the surface of a thin substrate of semiconductor material
IDE	Integrated Development Environment	An integrated development environment is a type of computer software that assists computer programmers in developing software. IDEs normally consist of a source code editor, a compiler and/or interpreter, build-automation tools, and a debugger
IDN	Identification Number	An IDN preceded by the prefix "P", specifies a product specific (manufacturer) IDN in short-hand notation. The actual IDN number for a product-specific IDN, can be obtained by adding 32768 to the short-hand numeric value. For convenience, the actual IDN number is given in parentheses following the short hand notation. For example, P2 is a manufacturer-specific IDN whose actual IDN number is 32770
IEC	International Electrotechnical Commission	IEC is a not-for-profit, non-governmental international standards organization that prepares and publishes International Standards for all electrical, electronic and related technologies
IEC 61131		IEC standard for Programmable logic controllers (PLCs)

Term	Definition	Description
IEC 61131-3		IEC 61131-3 is the third part of the open international standard IEC 61131. The current (second) edition was published in 2003. IEC 61131-3 currently defines five programming languages for programmable control systems It deals with programming languages and defines two graphical and two textual PLC programming language standards
IL	Instruction List	It is a low-level language and resembles assembly
IPC	Industrial PC	Industrial PC is the x86 PC-based computing platform for industrial applications. Industrial PC offers features different from the consumer PC on the reliability, compatibility, expansibility and long term supply. KAS IPC usually includes a touch-screen display as a combined input and output device.
IRQ	Interrupt Request	An interrupt request refers to the act of interrupting the bus lines used to signal an interrupt
JTAG	Joint Test Action Group	JTAG is used for accessing sub-blocks of integrated circuits, and is also useful as a mechanism for debugging embedded systems, providing a convenient "back door" into the system. When used as a debugging tool, an in-circuit emulator - which in turn uses JTAG as the transport mechanism - enables a programmer to access an on-chip debug module which is integrated into the CPU via the JTAG interface. The debug module enables the programmer to debug the software of an embedded system
KAS	Kollmorgen Automation Suite	Umbrella name for a software package including the KAS IDE and the KAS Runtime software
KAS IDE	Kollmorgen Automation Suite - Integrated Development Environment	The KAS IDE is the GUI View environment. It is a Windows integrated design environment (IDE) containing all the tools and editors (based on the different IEC 61131 languages) that users need during the entire life cycle of the machine
KAS Runtime	Kollmorgen Automation Suite - Runtime	The KAS Runtime is the engine that provides a soft PLC and a motion controller
KVB IDE	Kollmorgen HMI Development Environment	Kollmorgen Visualization Builder is an editor that allows the end-user to control the KAS Runtime
LD	Ladder Diagram	see page 712

Term	Definition	Description
LSB	Least Significant Bit	Sometimes abbreviated as LSB, the least significant bit is the lowest bit in a series of numbers in binary; the LSB is located at the far right of a string. For example, in the binary number: 10111001, the least significant bit is the far right "1".
MDI	Multiple Document Interface	Graphical computer applications with an MDI are those whose windows reside under a single parent window (usually with the exception of modal windows), as opposed to all windows being separate from each other (single document interface). **Advantages:** - With MDI, a single menu bar and/or toolbar is shared between all child windows, reducing clutter and increasing efficient use of screen space - An application's child windows can be hidden/shown/minimized/maximized as a whole - Features such as "Tile" and "Cascade" can be implemented for the child windows
ML	Motion Library	The Motion Library is the interface between the IEC61131-3 logical application and the motion engine. It gives access from IEC61131-3 to pipe and Pipe Blocks parameters and methods as well as to higher levels of functionalities such a homing, tensioning, dynamic correction, etc.
MSB	Most Significant Bit	Sometimes abbreviated as MSB, the most significant bit is the bit position in a binary number having the greatest value
NAT	Network Address Translation	In computer networking, NAT is the process of modifying network address information in datagram (IP) packet headers while in transit across a traffic routing device for the purpose of remapping a given address space into another.
NIC	Network Interface Controller	A network interface controller (or card) is a hardware device that handles an interface to a computer network and allows a network-capable device to access that network

Term	Definition	Description
NVRAM	Non-Volatile Random Access Memory	NVRAM is the general name used to describe any type of random access memory which does not lose its information when power is turned off. This memory is in contrast to the most common forms of random access memory today, which both require continual power in order to maintain their data. NVRAM is a subgroup of the more general class of non-volatile memory types, the difference being that NVRAM devices offer random access, like hard disks. The best-known form of NVRAM memory today is flash memory
OEM	Original Equipment Manufacturer	A term that refers to containment-based re-branding, namely where one company uses a component of another company within its product, or sells the product of another company under its own brand. OEM refers to the company that originally manufactured the product
OPC	OLE for Process Control	OPC is the original name for an open standard to specify the communication of real-time plant data between control devices from different manufacturers
PAC	Programmable Automation Controller	PAC is a compact controller that combines the features and capabilities of a PC-based control system with that of a typical programmable logic controller (PLC). A PAC thus provides not only the reliability of a PLC, but also the task flexibility and computing power of a PC. Additionally, because they function and communicate over popular network interface protocols, PACs are able to transfer data from the machines they control to other machines and components in a networked control system
PCI	Peripheral Component Interconnect	The PCI specifies a computer bus for attaching peripheral devices to a computer motherboard
PD	Programmable Drive	(Also known as Servo Amplifiers or Servo Drive) A Drive can be programmable, which means it has an open hardware and software architecture to make it ready for nearly all conceivable customer-specific modifications
AKD PDMM	Programmable Drive Multi-axis Master	Programmable drive let you control up to seven EtherCAT slave drives and I/O

Term	Definition	Description
PDO	Process Data Object	PDO is a type of protocol frame used in some fieldbuses. A PDO contains one or more object dictionary entries, which define the application data transferred between devices. EtherCAT uses the same communication mechanisms (PDO and SDO) as CANopen. EtherCAT transfers the process data between the master and slave device cyclically. PDOs have several attributes to define their properties. • PDOs can be exclusive, meaning that no additional PDOs may be assigned to a device if an exclusive PDO is assigned. • If the PDO type is fixed (Fixed attribute = 1) then the PDO's content cannot be changed by users.
PID	Proportional-Integral-Derivative	A PID controller is a generic control-loop feedback mechanism widely used in industrial control systems. An "error" occurs when an event or a disturbance triggers off a change in the process variable. A PID controller attempts to correct the error between a measured process variable and a desired setpoint by calculating and then outputting a corrective action that can adjust the process accordingly
PLC	Programmable Logic Controller	A Programmable Logic Controller, PLC, or Programmable Controller is a digital computer used for automation of industrial processes, such as control of machinery on factory assembly lines. Used to synchronize the flow of inputs from (physical) sensors and events with the flow of outputs to actuators and events
PNE	Pipe Network Engine	The Pipe Network concept is an innovative solution to solve axis synchronization problems. It is based on Pipe Blocks representing the whole mechanical system by analogy
POU	Programmable Organization Unit	An application is a list of programs. Programs are executed sequentially within the target cycle according to the order defined by the user and displayed in the Project View
Profibus	Process Field Bus	Profibus is one of the most popular type of fieldbus used worldwide
Qwt	Qt Widgets	Qwt is a graphics extension to the Qt GUI application framework from Trolltech ASA

Term	Definition	Description
RTC	Real-Time Computing	RTC is the study of hardware and software systems which are subject to a "real-time constraint" (i.e., operational deadlines from event to system response)
RTOS	Real-Time Operating System	RTOS is a multitasking operating system intended for real-time applications
S300	Servostar 300 drive	see page 726
S700	Servostar 700 drive	see page 726
SCADA	Supervisory Control And Data Acquisition	SCADA systems are typically used to perform data collection and control at the supervisory level. Some SCADA systems only monitor without doing control, these systems are still referred to as SCADA systems
SDO	Service Data Object	The SDO protocol is used to read and write values across fieldbuses . The SDO data is defined by the object dictionary. EtherCAT uses the same communication mechanisms (PDO and SDO) as CANopen. SDO data is non-cyclic and is applicable for non-deterministic data transfers.
SFC	Sequential Function Chart	It can be used to program processes that can be split into steps. The main components of SFC are: - Steps with associated actions - Transitions with associated logic conditions - Directed links between steps and transitions
SPLC	Software version of a PLC	Usually working on PC-based hardware
ST	Structured Text	A high-level language which is block structured and syntactically resembles Pascal
TDI	Tabbed Document Interface	TDI allows multiple documents to be contained within a single window, using tabs to navigate between them
TMP	Trapezoidal Motion Profile	This Pipe Block is a source block that frequently serves as a virtual master for a system composed of several pipes. Generally, a trapezoidal motion profile generator is used to generate a flow of values with a first derivative which produces a trapezoidal trajectory
UDFB	User-Defined Function Block	UDFB can be used as a sub-Function Block in another program of the application. It is described using FBD, LD, ST or IL language. Input/output parameters of a UDFB (as well as private variables) are declared in the variable editor as local variables of the UDFB

Term	Definition	Description
UDP	User Datagram Protocol	UDP is a network protocol used for the Internet. This protocol assumes that the Internet Protocol (IP) is used as the underlying protocol. This protocol provides a procedure for application programs to send messages to other programs with a minimum of protocol mechanism. The protocol is transaction oriented, and delivery and duplicate protection are not guaranteed.
USB	Universal Serial Bus	USB is a serial bus standard to interface devices
UTF8	Unicode Transformation Format (8-bit)	UTF-8 is a variable-length character encoding for Unicode. It is able to represent any character in the Unicode standard, yet the initial encoding of byte codes and character assignments for UTF-8 is backward-compatible with ASCII
UU	User Units	A coordinate value or length expressed in user units represents a coordinate value or length in the current user coordinate system. Thus, 10 user units represent a length of 10 units in the current user coordinate system.
XML	Extensible Markup Language	XML is a general-purpose markup language. It is classified as an extensible language because it allows its users to define their own tags
VDK	VisualDSP Kernel	Operating system supported by Blackfin microprocessors
VLAN	Virtual LAN	A VLAN is a group of hosts with a common set of requirements that communicate as if they were attached to the Broadcast domain, regardless of their physical location. A VLAN has the same attributes as a physical LAN, but it allows for end stations to be grouped together even if they are not located on the same network switch. Network reconfiguration can be performed using software instead of physically relocating devices

Term	Definition	Description
XPe	Windows XP Embedded	XPe is a componentized version of the Professional edition of Windows XP. An original equipment manufacturer is free to choose only the components needed, thereby reducing operating system footprint and also reducing attack area as compared with XP Professional. Unlike Windows CE, Microsoft's operating system for portable devices and consumer electronics, XP Embedded provides the full Windows API, and support for the full range of applications and device drivers written for Microsoft Windows. The system requirements state that XPe can run on devices with at least 32MB Compact Flash, 32MB RAM and a P-200 microprocessor
WUI	Web User Interface	WUI is the set of means by which people interact with a particular machine, device, computer program or other complex tool via the Web

Glossary

Terms in this Glossary are provided for informational purposes only and can describe features not included in your particular license.

Term	Definition
Actuator	A mechanical device for moving or controlling a mechanism or system. An actuator typically is a mechanical device which transforms an input signal (usually an electrical signal) into motion
Bandwidth	In computer networking, bandwidth often refers to a data rate measured in bits/s, for example, network throughput. The reason for the connection of data rate with the term bandwidth is that the limit to the data rate of a physical communication link is related to its bandwidth in hertz
Cam profiling	The position of a slave axis is mathematically linked to the position of a master axis. A good example of this would be in a system where two rotating drums turn at a given ratio to each other. A more advanced case of electronic gearing is electronic camming. With electronic camming, a slave axis follows a profile which is a function of the master position. This profile need not be linear, but it must be a mathematical function
CANopen	CANopen is a communication protocol and device profile specification for embedded systems used in automation for fieldbuses working in real-time
Caret	The term caret is also sometimes used in graphical user interface terminology where it means a text insertion point indicator, frequently represented by a blinking vertical bar. In this context, it can be used interchangeably with the word **cursor**, although the latter term is often reserved for a mouse pointer
Casting	For Typecasting, see page 728
COM	COM is the original name of the serial port interface. It does not only refer to physical ports, but also to virtual ports, such as ports created by bluetooth or USB-to-Serial adapters
Contactor	A contactor is an electrically controlled switch (relay) used for switching a power circuit. A contactor is activated by a control input which is a lower voltage/current than that which the contactor is switching. Unlike a circuit breaker, a contactor is not intended to interrupt a short-circuit current
Datagram	A datagram is a basic transfer unit in which the delivery arrival time and order are not guaranteed. A datagram consists of header and data areas. The source and destination addresses as well as a type field are found in the header of a datagram.

Term	Definition
DeviceNet	DeviceNet is a communication protocol (based on Controller Area Network) used in the automation industry to interconnect control devices for data exchange. Typical applications are information exchange, safety devices, and large I/O control networks
Drive	In electrical engineering, a drive is an electronic device providing power to a motor or servo, and controlling it through the current and timing in its coils
Driver	In computing and electronics, a driver is a software component allowing higher-level computer programs to interact with a computer hardware device. A driver typically communicates with the device through the computer bus or communications subsystem to which the hardware is connected
Endian	Big-endian and little-endian describe the order in which a sequence of bytes are stored in computer memory. Big-endian is an order in which the "big end" (most significant value in the sequence) is stored first (at the lowest storage address). Little-endian is an order in which the "little end" (least significant value in the sequence) is stored first For example the decimal integer 56789652 (0x03628a94 in hexadecimal) is stored as follows: • 0x03 0x62 0x8a 0x94 on big-endian • 0x94 0x8a 0x62 0x03 on little-endian KAS applications can be downloaded to big-endian or little-endian processor targets
Environment	Environment objects are global objects that exist before the execution of the script. Typically, they are global objects of the KAS IDE that can be accessed from the script
EtherCAT	"Ethernet for Control Automation Technology" EtherCAT is an open, high-performance Ethernet-based fieldbus system. The development goal of EtherCAT was to apply Ethernet to automation applications which require short data update times (also called cycle times) with low communication jitter (for synchronization purposes) and low hardware costs
Ethernet	Ethernet is a large, diverse family of frame-based computer networking technologies that operate at many speeds for local area networks (LANs)
EtherNet/IP	An open industrial application layer protocol for industrial automation applications. The EtherNet/IP application layer protocol is based on the CIP layer

Term	Definition
Fast Inputs	The inputs are taken into account at each cycle depending on the system periodicity (for example each millisecond). Under certain circumstances it can be insufficient when more accuracy is needed, or if a quick response is required from the system. To fill the gap, a drive can have some Fast Input connections (generally one or two). When an event happens that triggers a Fast Input (e.g. when a sensor sends a rising edge), the detection of a signal occurs faster (which can be 1000 times more accurate than the system periodicity). Then the timestamp associated with this input can be provided to the IPC to take corrective action
Feedback Device	A process whereby some proportion of the output signal of a system is passed (fed back) to the input. In automation, a device coupled to each motor to provide indication of the motor's shaft angle, for use in commutating the motor and controlling its speed and position
feed-forward	This describes an element or pathway within a control system which passes a controlling signal from a source in the control system's external environment, often a command signal from an external operator, to a load elsewhere in its external environment
Fieldbus	A Fieldbus is an industrial network protocol used for distributed control (e.g. EtherCAT, CAN, Profibus or Sercos). It is a way of connecting instruments in a plant design
Flash Memory	A Flash memory is a non-volatile computer storage chip that can be electrically erased and reprogrammed. In addition to being non-volatile, flash memory offers fast read access times, as fast as dynamic RAM, although not as fast as static RAM or ROM. Its mechanic shock resistance explain the popularity over hard disks in portable devices; so does its high durability, being able to withstand high pressure, temperature, immersion in water etc.
Frame	In networking dialect, a message is called a frame
Front-end	In software design, the front-end is the part of a software system that interacts directly with the user
Homing	The Homing procedure allows, based on a position measurement, to set a position offset to the motor in order to ensure it is physically at the home position. The homing offset is saved in the controller.
Interrupt	An interrupt is an asynchronous signal from hardware indicating the need for attention or a synchronous event in software indicating the need for a change in execution

Term	Definition
Intime	INtime software combines deterministic, hard real-time control with standard Windows operating systems (including Windows XP, Windows XP Embedded, Windows 2000, Windows Server 2003, Vista and Windows 7) without requiring additional hardware. INtime was designed specifically to take advantage of the powerful capabilities of the x86 processor architecture. Therefore, real-time and non real-time applications run in separate virtual machines on a single computer, for cost-effective, reliable control which is easy to develop and maintain
Jerk	In physics, jerk is the rate of change of acceleration; more precisely, the derivative of acceleration with respect to time
Latch	The control word is used to activate the drive's latch status machine. The latch control word is processed independently of the EtherCAT bus cycle. The status word is used to return the drive's latch status
MAC address	A Media Access Control address (MAC address) is a quasi-unique identifier assigned to most network adapters or network interface cards (NICs) by the manufacturer for identification. If assigned by the manufacturer, a MAC address usually encodes the manufacturer's registered identification number
ModBus	ModBus is a serial communications protocol and is now the most commonly available means of connecting industrial electronic devices. ModBus is often used to connect a supervisory computer with a remote terminal unit in supervisory control and data acquisition (SCADA) systems. Versions of the ModBus protocol exist for serial port and Ethernet (it is widely used with TCP/IP over Ethernet)
Motion Bus	A Motion bus is an industrial network protocol used for **real-time** distributed control (e.g. EtherCAT).
Motion control	Motion control is a sub-field of automation, in which the position and/or velocity of machines are controlled using some type of device such as a hydraulic pump, linear actuator, or an electric motor, generally a servo. Motion control is an important part of robotics and CNC machine tools; however, it is more complex than in the use of specialized machines, where the kinematics is usually simpler. The latter is often called General Motion Control (GMC). Motion control is widely used in the packaging, printing, textile and assembly industries
Motor	An actuator focused to a movement, converting electrical energy in a force or torque

Term	Definition
Non-volatile	Information is stored in a specific memory to remain accessible even when the application has been powered off
OpenGL	OpenGL (Open Graphics Library) is a standard specification defining a cross-language, cross-platform API for writing applications that produce 2D and 3D computer graphics. The Softscope uses this API to implement graphical manipulations
P-code	P-code machine or pseudo-code machine is a specification of a CPU whose instructions are expected to be executed in software rather than in hardware. Programs that have been translated to P-code are executed (interpreted) by a software program that emulates the behavior of the CPU specification
PDO	PDO is a type of protocol frame used in some fieldbuses. A PDO contains one or more object dictionary entries, which define the application data transferred between devices. EtherCAT uses the same communication mechanisms (PDO and SDO) as CANopen. EtherCAT transfers the process data between the master and slave device cyclically. PDOs have several attributes to define their properties. • PDOs can be exclusive, meaning that no additional PDOs may be assigned to a device if an exclusive PDO is assigned. • If the PDO type is fixed (Fixed attribute = 1) then the PDO's content cannot be changed by users.
Periodicity	The period of execution of a pipe is the time spent between two successive computations of set values for the same pipe. The period of execution of a pipe is specified by the PERIOD parameter of the input Pipe Block
PLCopen	A vendor -and product- independent worldwide association active in Industrial Control and aiming at standardizing PLC file formats based on XML
Pragma	A compiler directive communicating additional "pragmatic" information. Pragmas are processed at compile time, not at run-time. They pass information to the compiler

Term	Definition
Precedence	In arithmetic and algebra, when a number or expression is both preceded and followed by a binary operation, a rule is required for which operation must be applied first. From the earliest use of mathematical notation, multiplication took precedence over addition, whichever side of a number it appeared on. Thus 3 + 4 × 5 = 5 × 4 + 3 = 23. To change the order of operations, we use parentheses (). Thus, if we want to force addition to precede multiplication, we write (3 + 4) × 5 = 35
Probe	For Softscope -Probe, see page 726
Profibus	see page 717
Pulse	When the step gets activated, the action is activated for a single execution, and possibly once again when the step is deactivated
Reference Counting	In computer science, reference counting is a technique of storing the number of references, pointers, or handles to a resource such as an object or block of memory. It is typically used as a means of deallocating objects which are no longer referenced
Rising Edge	A rising edge is the transition of a digital signal from low to high. It is also called positive edge
Run-time	In computer science, run-time (or run time) describes the operation of a computer program, the duration of its execution, from beginning to termination (compare compile time)
Sensor	A sensor is a type of transducer that converts one type of energy into another for various purposes including measurement or information transfer
Service Port	UDP applications use datagram sockets to establish host-to-host communications. An application binds a socket to its endpoint of data transmission, which is a combination of an IP address and a service port. A port is a software structure that is identified by the port number, a 16 bit integer value.
Servo Drive	A servo drive is a special electric amplifier used to power electric servo motors. It monitors feedback signals from the motor and continually adjusts for deviation from expected behavior
Setpoint	Setpoint is the target value that an automatic control system (for example a PID controller) aims to reach
Softscope - Channel	A Channel is used by the softscope to acquire the evolution of a variable which is plugged on it
Softscope - Probe	A device that uses onboard instruments to gather and relay a variety of measurement to controllers from remote locations. Probes can return their data over radio links or be physically tethered to controllers or another device, or to collect and return physical samples

Term	Definition
Softscope - Sampling	To acquire the variable's evolution, samples are taken at fixed intervals. The accuracy to create the trace depends on the resolution of the acquisition. The sampling frequency must be higher than 2 times the highest frequency in the input signal. It is called the Nyquist frequency. Theoretically it is possible to reconstruct the input signal with more than 2 samples per period. In practice, 10 to 20 samples per period are recommended to be able to examine the signal thoroughly
Softscope - Time-base	The time-base allows you to set the speed at which all the lines for each channel are drawn, and is calibrated in milliseconds per division
Softscope - Trace	The trace is the resulting graph of a variable's evolution against time, with the more distant past on the left and the more recent past on the right
Synchronization	Combines an axis or axes group (as slave) with an axis as master so that the slave executes its path with synchronization to the progress of the master, meaning linked to a one-dimensional source for synchronization
SynqNet	SynqNet is a digital machine control network. Built on the 100BT physical layer, SynqNet provides a synchronous real-time connection between motion controllers, servo drives, stepper drives, I/O modules, and custom devices
Tag	In the HMI context, objects connected to tags can change values in a controller, and controller values can be reflected by changing object appearance in various ways. A tag has a symbolic name and can be of different data types. Tags can belong to a connected controller, be internal or belong to the system.
Timestamp	A timestamp is a sequence of characters denoting the date and/or time at which a certain event occurred
Torque	Torque is the tendency of a force to rotate an object about an axis. Just as a force is a push or a pull, a torque can be thought of as a twist. The SI unit for torque is the newton metre (N.m).

Term	Definition
Typecasting	In computer science, type conversion or typecasting refers to changing an entity of one data type into another. It is done to take advantage of certain features of type hierarchies. For instance, values from a more limited set, such as integers, can be stored in a more compact format and later converted to a different format enabling operations not previously possible, such as division with several decimal places' worth of accuracy. There are two types of conversion: implicit and explicit. The term for implicit type conversion is coercion. The most common form of explicit type conversion is known as casting. Explicit type conversion can also be achieved with separately defined conversion routines such as an overloaded object constructor

Index

B

C

D

N

O

P

S

T

14 Licenses

14.1 AdvXMLParser

14.2 AjaxFileUpload — MPL License

See www.mozilla.org/MPL/2.0/ for the Mozilla Public License.

14.3 Apache log4net — Apache License

marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

(a) You must give any other recipients of the Work or Derivative Works a copy of this License; and

(b) You must cause any modified files to carry prominent notices stating that You changed the files; and

(c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

(d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

14.4 bsdtar & libarchive 2 — BSD License

bsdtar.exe and libarchive2.dll for Windows

14.5 bzip2.dll — BSD License

bzip2.dll

14.6 DockPanel Suite — MIT License

DockPanel

The MIT License

14.7 FileHelpers

```
-----------------------
  FileHelpers
-----------------------

  http://www.filehelpers.com  - SF Home: http://sourceforge.net/projects/filehelpers

  The FileHelpers are an easy to use library to import/export data from fixed length or delimited files.

  If you want to start using the library go directly to the Quick Start Guide in the CHM.

----------------------------------------
  Who needs the File Helpers Library ?
----------------------------------------

  In almost every project there is a need to read/write data from/to a file of a specified format.

  For example for log parsing, data warehouse and OLAP applications, communication between systems, file format trans-
formations (for example from a fixed length to a CSV file).

  This library aims to provide an easy and reliable way to accomplish this task.

-----------
  History
-----------

   Check The docs for the History (is hard to mantain two copies =)

-----------
  Licence
-----------

 FileHelpers Library is @ Copyright 2005-2006 to Marcos Meli but it's source code and the binaries are free for com-
mercial and non commercial use.

 LGPL license description is available here: http://www.opensource.org/licenses/lgpl-license.php


---------------------
  Contact and Ideas
---------------------

 If you find that there is a feature that I must include, or you have a new idea (for the API, Source Code or
Examples), only let me know, sending an e-mail to marcos@filehelpers.com or entering the FileHelpers Forums at


         http://www.filehelpers.com/forums/

----------------------------
  Full Sources and Updates
----------------------------

 If you want to help in the develpment of the library please go to

    http://sourceforge.net/project/showfiles.php?group_id=152382&package_id=169468

 and download the AllInOne DevPack with the binaries of NUnit, NAnt and NDoc.
```

14.8 jQuery Cookies

14.9 jQuery File Tree — MIT License

JQueryFileTree

14.10 Mongoose — MIT License

14.11 MVVM Light Toolkit — MIT License

MVVM Light Toolkit

http://mvvmlight.codeplex.com/

License: The MIT License (MIT)

COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

14.12 Qwt — GNU Lesser General Public License

GNU GENERAL PUBLIC LICENSE
Version 3, 29 June 2007

Copyright (C) 2007 Free Software Foundation, Inc. <http://fsf.org/>
Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Preamble

The GNU General Public License is a free, copyleft license for software and other kinds of works.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program--to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

To protect your rights, we need to prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

Developers that use the GNU GPL protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS

0. Definitions.

"This License" refers to version 3 of the GNU General Public License.

"Copyright" also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

"The Program" refers to any copyrightable work licensed under this License. Each licensee is addressed as "you". "Licensees" and "recipients" may be individuals or organizations.

To "modify" a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a "modified version" of the earlier work or a work "based on" the earlier work.

A "covered work" means either the unmodified Program or a work based on the Program.

To "propagate" a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To "convey" a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

An interactive user interface displays "Appropriate Legal Notices" to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

The "source code" for a work means the preferred form of the work for making modifications to it. "Object code" means any non-source form of a work.

A "Standard Interface" means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The "System Libraries" of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A "Major Component", in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The "Corresponding Source" for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions.

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law.

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.

You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

a) The work must carry prominent notices stating that you modified it, and giving a relevant date.

b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices".

c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.

d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.

b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.

c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.

d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.

e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A "User Product" is either (1) a "consumer product", which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information" for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

"Additional permissions" are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or

b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or

c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or

d) Limiting the use for publicity purposes of names of licensors or authors of the material; or

e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or

f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered "further restrictions" within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An "entity transaction" is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

A "contributor" is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's "contributor version".

A contributor's "essential patent claims" are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, "control" includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To "grant" such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. "Knowingly relying" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom.

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Use with the GNU Affero General Public License.

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

14. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty.

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16.

If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively state the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

```
<one line to give the program's name and a brief idea of what it does.>
Copyright (C) <year> <name of author>
```

This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program. If not, see <http://www.gnu.org/licenses/>.

Also add information on how to contact you by electronic and paper mail.

If the program does terminal interaction, make it output a short notice like this when it starts in an interactive mode:

```
<program> Copyright (C) <year> <name of author>
This program comes with ABSOLUTELY NO WARRANTY; for details type `show w'.
This is free software, and you are welcome to redistribute it
under certain conditions; type `show c' for details.
```

The hypothetical commands `show w' and `show c' should show the appropriate parts of the General Public License. Of course, your program's commands might be different; for a GUI interface, you would use an "about box".

You should also get your employer (if you work as a programmer) or school, if any, to sign a "copyright disclaimer" for the program, if necessary. For more information on this, and how to apply and follow the GNU GPL, see <http://www.gnu.org/licenses/>.

The GNU General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Lesser General Public License instead of this License. But first, please read

<http://www.gnu.org/philosophy/why-not-lgpl.html>.

14.13 ZedGraph — LGPL License

For easy distribution, this archive provides the dll form of ZedGraph only.
This distribution of the ZedGraph source code includes two versions:

Version 4.6.x is the .Net 1.1 compatible version
Version 5.1.x is the .Net 2.0 compatible version

Complete source code for ZedGraph is available on sourceforge here:

http://sourceforge.net/projects/zedgraph/

A wiki providing help, samples, etc. is available here:

http://zedgraph.org

A tutorial on using ZedGraph is available here:

http://www.codeproject.com/csharp/zedgraph.asp

****

GNU LESSER GENERAL PUBLIC LICENSE
Version 2.1, February 1999

[This is the first released version of the Lesser GPL. It also counts as the successor of the GNU Library Public License, version 2, hence the version number 2.1.]

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public Licenses are intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users.

This license, the Lesser General Public License, applies to some specially designated software packages--typically libraries--of the Free Software Foundation and other authors who decide to use it. You can use it too, but we suggest you first think carefully about whether this license or the ordinary General Public License is the better strategy to use in any particular case, based on the explanations below.

When we speak of free software, we are referring to freedom of use, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish); that you receive source code or can get it if you want it; that you can change the software and use pieces of it in new free programs; and that you are informed that you can do these things.

To protect your rights, we need to make restrictions that forbid distributors to deny you these rights or to ask you to surrender these rights. These restrictions translate to certain responsibilities for you if you distribute copies of the library or if you modify it.

For example, if you distribute copies of the library, whether gratis or for a fee, you must give the recipients all the rights that we gave you. You must make sure that they, too, receive or can get the source code. If you link other code with the library, you must provide complete object files to the recipients, so that they can relink them with the library after making changes to the library and recompiling it. And you must show them these terms so they know their rights.

We protect your rights with a two-step method: (1) we copyright the library, and (2) we offer you this license, which gives you legal permission to copy, distribute and/or modify the library.

To protect each distributor, we want to make it very clear that there is no warranty for the free library. Also, if the library is modified by someone else and passed on, the recipients should know that what they have is not the original version, so that the original author's reputation will not be affected by problems that might be introduced by others.

Finally, software patents pose a constant threat to the existence of any free program. We wish to make sure that a company cannot effectively restrict the users of a free program by obtaining a restrictive license from a patent holder. Therefore, we insist that any patent license obtained for a version of the library must be consistent with the full freedom of use specified in this license.

Most GNU software, including some libraries, is covered by the ordinary GNU General Public License. This license, the GNU Lesser General Public License, applies to certain designated libraries, and is quite different from the ordinary General Public License. We use this license for certain libraries in order to permit linking those libraries into non-free programs.

When a program is linked with a library, whether statically or using a shared library, the combination of the two is legally speaking a combined work, a derivative of the original library. The ordinary General Public License therefore permits such linking only if the Public License permits more lax criteria for linking other code with the library.

We call this license the "Lesser" General Public License because it does Less to protect the user's freedom than the ordinary General of an advantage over competing non-free programs. These disadvantages are the reason we use the ordinary General Public License for many libraries. However, the Lesser license provides advantages in certain special circumstances.

For example, on rare occasions, there may be a special need to encourage the widest possible use of a certain library, so that it becomes a de-facto standard. To achieve this, non-free programs must be allowed to use the library. A more frequent case is that a free library does the same job as widely used non-free libraries. In this case, there is little to gain by limiting the free library to free software only, so we use the Lesser General Public License.

In other cases, permission to use a particular library in non-free programs enables a greater number of people to use a large body of free software. For example, permission to use the GNU C Library in non-free programs enables many more people to use the whole GNU operating system, as well as its variant, the GNU/Linux operating system.

Although the Lesser General Public License is Less protective of the users' freedom, it does ensure that the user of a program that is linked with the Library has the freedom and the wherewithal to run that program using a modified version of the Library.

The precise terms and conditions for copying, distribution and modification follow. Pay close attention to the difference between a "work based on the library" and a "work that uses the library". The former contains code derived from the library, whereas the latter must be combined with the library in order to run.

GNU LESSER GENERAL PUBLIC LICENSE
TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License Agreement applies to any software library or other program which contains a notice placed by the copyright holder or other authorized party saying it may be distributed under the terms of this Lesser General Public License (also called "this License"). Each licensee is addressed as "you".

A "library" means a collection of software functions and/or data prepared so as to be conveniently linked with application programs (which use some of those functions and data) to form executables.

The "Library", below, refers to any such software library or work which has been distributed under these terms. A "work based on the Library" means either the Library or any derivative work under copyright law: that is to say, a work containing the Library or a portion of it, either verbatim or with modifications and/or translated straightforwardly into another language. (Hereinafter, translation is included without limitation in the term "modification".)

"Source code" for a work means the preferred form of the work for making modifications to it. For a library, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the library.

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running a program using the Library is not restricted, and output from such a program is covered only if its contents constitute a work based on the Library (independent of the use of the Library in a tool for and what the program that uses the Library does.

1. You may copy and distribute verbatim copies of the Library's complete source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and distribute a copy of this License along with the Library.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Library or any portion of it, thus forming a work based on the Library, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) The modified work must itself be a software library.

b) You must cause the files modified to carry prominent notices stating that you changed the files and the date of any change.

c) You must cause the whole of the work to be licensed at no charge to all third parties under the terms of this License.

d) If a facility in the modified Library refers to a function or a table of data to be supplied by an application program that uses the facility, other than as an argument passed when the facility is invoked, then you must make a good faith effort to ensure that, in the event an application does not supply such function or table, the facility still operates, and performs whatever part of its purpose remains meaningful.

(For example, a function in a library to compute square roots has a purpose that is entirely well-defined independent of the application. Therefore, Subsection 2d requires that any application-supplied function or table used by this function must be optional: if the application does not supply it, the square root function must still compute square roots.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Library, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Library, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Library.

In addition, mere aggregation of another work not based on the Library with the Library (or with a work based on the Library) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may opt to apply the terms of the ordinary GNU General Public License instead of this License to a given copy of the Library. To do this, you must alter all the notices that refer to this License, so that they refer to the ordinary GNU General Public License, version 2, instead of to this License. (If a newer version than version 2 of the ordinary GNU General Public License has appeared, then you can specify that version instead if you wish.) Do not make any other change in these notices.

Once this change is made in a given copy, it is irreversible for that copy, so the ordinary GNU General Public License applies to all subsequent copies and derivative works made from that copy.

This option is useful when you wish to copy part of the code of the Library into a program that is not a library.

4. You may copy and distribute the Library (or a portion or under the terms of Sections 1 and 2 above provided that you accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange.

If distribution of object code is made by offering access to copy source code from the same place satisfies the requirement to distribute the source code, even though third parties are not compelled to copy the source along with the object code.

5. A program that contains no derivative of any portion of the Library, but is designed to work with the Library by being compiled or inked with it, is called a "work that uses the Library". Such a work, in isolation, is not a derivative work of the Library, and therefore falls outside the scope of this License.

However, linking a "work that uses the Library" with the Library creates an executable that is a derivative of the Library (because it contains portions of the Library), rather than a "work that uses the library". The executable is therefore covered by this License. Section 6 states terms for distribution of such executables.

When a "work that uses the Library" uses material from a header file that is part of the Library, the object code for the work may be a derivative work of the Library even though the source code is not. Whether this is true is especially significant if the work can be linked without the Library, or if the work is itself a library. The threshold for this to be true is not precisely defined by law.

If such an object file uses only numerical parameters, data structure layouts and accessors, and small macros and small inline functions (ten lines or less in length), then the use of the object file is unrestricted, regardless of whether it is legally a derivative work. (Executables containing this object code plus portions of the Library will still fall under Section 6.)

Otherwise, if the work is a derivative of the Library, you may distribute the object code for the work under the terms of Section 6. Any executables containing that work also fall under Section 6, whether or not they are linked directly with the Library itself.

6. As an exception to the Sections above, you may also combine or link a "work that uses the Library" with the Library to produce a work containing portions of the Library, and distribute that work under terms of your choice, provided that the terms permit modification of the work for the customer's own use and reverse engineering for debugging such modifications.

You must give prominent notice with each copy of the work that the Library is used in it and that the Library and its use are covered by this License. You must supply a copy of this License. If the work during execution displays copyright notices, you must include the copyright notice for the Library among them, as well as a reference directing the user to the copy of this License. Also, you must do one of these things:

a) Accompany the work with the complete corresponding machine-readable source code for the Library including whatever changes were used in the work (which must be distributed under Sections 1 and 2 above); and, if the work is an executable linked with the Library, with the complete machine-readable "work that uses the Library", as object code and/or source code, so that the user can modify the Library and then relink to produce a modified executable containing the modified Library. (It is understood that the user who changes the contents of definitions files in the Library will not necessarily be able to recompile the application to use the modified definitions.)

b) Use a suitable shared library mechanism for linking with the Library. A suitable mechanism is one that (1) uses at run time a copy of the library already present on the user's computer system, rather than copying library functions into the executable, and (2) will operate properly with a modified version of the library, if the user installs one, as long as the modified version is interface-compatible with the version that the work was made with.

c) Accompany the work with a written offer, valid for at least three years, to give the same user the materials specified in Subsection 6a, above, for a charge no more than the cost of performing this distribution.

d) If distribution of the work is made by offering access to copy from a designated place, offer equivalent access to copy the above specified materials from the same place.

e) Verify that the user has already received a copy of these materials or that you have already sent this user a copy.

For an executable, the required form of the "work that uses the Library" must include any data and utility programs needed for reproducing the executable from it. However, as a special exception, the materials to be distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

It may happen that this requirement contradicts the license restrictions of other proprietary libraries that do not normally accompany the operating system. Such a contradiction means you cannot use both them and the Library together in an executable that you distribute.

7. You may place library facilities that are a work based on the Library side-by-side in a single library together with other library facilities not covered by this License, and distribute such a combined library, provided that the separate distribution of the work based on the Library and of the other library facilities is otherwise permitted, and provided that you do these two things:

a) Accompany the combined library with a copy of the same work based on the Library, uncombined with any other library facilities. This must be distributed under the terms of the Sections above.

b) Give prominent notice with the combined library of the fact that part of it is a work based on the Library, and explaining where to find the accompanying uncombined form of the same work.

8. You may not copy, modify, sublicense, link with, or distribute the Library except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, link with, or distribute the Library is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

9. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Library or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Library (or any work based on the Library), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Library or works based on it.

10. Each time you redistribute the Library (or any work based on the Library), the recipient automatically receives a license from the original licensor to copy, distribute, link with or modify the Library subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties with this License.

11. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Library at all. For example, if a patent license would not permit royalty-free redistribution of the Library by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Library.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply, and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

12. If the distribution and/or use of the Library is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Library under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

13. The Free Software Foundation may publish revised and/or new versions of the Lesser General Public License from time to time. but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Library specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Library does not specify a license version number, you may choose any version ever published by the Free Software Foundation.

14. If you wish to incorporate parts of the Library into other free programs whose distribution conditions are incompatible with these, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

15. BECAUSE THE LIBRARY IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE LIBRARY, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE LIBRARY "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE LIBRARY IS WITH YOU. SHOULD THE LIBRARY PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE LIBRARY AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE LIBRARY (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE LIBRARY TO OPERATE WITH ANY OTHER SOFTWARE), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Libraries

If you develop a new library, and you want it to be of the greatest possible use to the public, we recommend making it free software that everyone can redistribute and change. You can do so by permitting redistribution under these terms (or, alternatively, under the terms of the ordinary General Public License).

To apply these terms, attach the following notices to the library. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

<one line to give the library's name and a brief idea of what it does.>
Copyright (C) <year> <name of author>

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

Also add information on how to contact you by electronic and paper mail.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the library, if necessary. Here is a sample; alter the names:

Yoyodyne, Inc., hereby disclaims all copyright interest in the library `Frob' (a library for tweaking knobs) written by James Random Hacker.

<signature of Ty Coon>, 1 April 1990
Ty Coon, President of Vice

That's all there is to it!

14.14 Zlib1.dll — BSD License

zlib1.dll

Copyright © 1995-2004 Jean-loup Gailly and Mark Adler

This software is provided 'as-is', without any express or implied warranty. In no event will the authors be held liable for any damages arising from the use of this software.

Permission is granted to anyone to use this software for any purpose, including commercial applications, and to

This page intentionally left blank.

Global Support Contacts

North America
KOLLMORGEN
203A West Rock Road
Radford, VA 24141 USA

Web:	www.kollmorgen.com
Mail:	support@kollmorgen.com
Tel.:	+1 - 540 - 633 - 3545
Fax:	+1 - 540 - 639 - 4162

Europe
KOLLMORGEN Europe GmbH
Pempelfurtstraße 1
40880 Ratingen, Germany

Web:	www.kollmorgen.com
Mail:	technik@kollmorgen.com
Tel.:	+49 - 2102 - 9394 - 0
Fax:	+49 - 2102 - 9394 - 3155

Asia
KOLLMORGEN
Rm 2205, Scitech Tower, China
22 Jianguomen Wai Street

Web:	www.kollmorgen.com
Mail:	sales.asia@kollmorgen.com
Tel.:	+86 - 400 666 1802
Fax:	+86 - 10 6515 0263