

IndraDrive

IndraMotion MLD (2G)
as of MPx-18

Application Manual
R911338914

Edition 05



Title	IndraDrive IndraMotion MLD (2G) as of MPx-18
Type of Documentation	Application Manual
Document Typecode	DOK-INDRV*-MLD3-**VRS*-AP05-EN-P
Internal File Reference	RS-77461bce0f8a914b0a6846a501d9a219-5-en-US-3
Purpose of Documentation	This documentation describes the specific functionalities of IndraMotion MLD contained in the firmware MPx-18 and above. Starting with this firmware version, IndraLogic 2G is supported. Therefore, this documentation additionally explains how to convert IndraLogic 1.x projects to IndraLogic 2G projects.
Record of Revisions	Edition 05, 2019-05 Editions see ""Editions of this documentation" on page 1"
Copyright	© Bosch Rexroth AG 2019 All rights reserved, also regarding any disposal, exploitation, reproduction, editing, distribution, as well as in the event of applications for industrial property rights.
Liability	The specified data is intended for product description purposes only and shall not be deemed to be a guaranteed characteristic unless expressly stipulated in the contract. All rights are reserved with respect to the content of this documentation and the availability of the product.
Editorial Department	Dept. DC-AE/EPI5

Table of Contents

	Page
1 Introduction.....	1
1.1 About this documentation.....	1
1.2 Structure of the documentation.....	4
1.3 Documentations.....	4
1.3.1 Overview.....	4
1.3.2 Drive systems, system components.....	5
1.3.3 Motors.....	5
1.3.4 Cables.....	6
1.3.5 Firmware.....	7
1.3.6 MLD technology functions.....	9
1.4 Reference documentations.....	9
1.5 Product presentation.....	9
1.5.1 Introduction and overview.....	9
1.5.2 Brief description.....	10
1.5.3 IndraMotion MLD-S.....	11
1.5.4 IndraMotion MLD-M.....	12
1.5.5 MLD-M extensions.....	13
1.5.6 Terms, basic principles.....	15
1.5.7 Function overview and system properties.....	17
 2 Important directions for use	 29
2.1 Appropriate use	29
2.1.1 Introduction.....	29
2.1.2 Areas of use and application.....	29
2.2 Inappropriate use.....	30
 3 Safety instructions for electric drives and controls.....	 31
3.1 Definitions of terms.....	31
3.2 General information.....	32
3.2.1 Using the Safety instructions and passing them on to others.....	32
3.2.2 Requirements for safe use.....	32
3.2.3 Hazards by improper use.....	33
3.3 Instructions with regard to specific dangers.....	34
3.3.1 Protection against contact with electrical parts and housings.....	34
3.3.2 Protective extra-low voltage as protection against electric shock	35
3.3.3 Protection against dangerous movements.....	36
3.3.4 Protection against electromagnetic and magnetic fields during operation and mounting.....	37
3.3.5 Protection against contact with hot parts.....	38
3.3.6 Protection during handling and mounting.....	38
3.3.7 Battery safety.....	38
3.3.8 Protection against pressurized systems.....	39
3.4 Explanation of signal words and the Safety alert symbol.....	40

	Page
4 Basic functions of Rexroth IndraMotion MLD.....	41
4.1 Overview.....	41
4.2 Explanation of terms.....	41
4.3 Device control.....	43
4.3.1 General information.....	43
4.3.2 Master communication.....	43
4.3.3 Axis control.....	43
4.3.4 Axis.....	43
4.3.5 PLC runtime.....	44
4.3.6 MLD options for influencing axis control.....	44
4.4 Axis commanding.....	47
4.4.1 Basics on axis control.....	47
4.4.2 Implementing motion commanding in the axis.....	52
4.4.3 Basic functional principle of motion function blocks in conformity with PLCopen.....	53
4.4.4 Axis addressing.....	55
4.4.5 Notes on application and programming.....	70
4.5 Task system.....	72
4.5.1 Definition of terms.....	72
4.5.2 Brief description.....	72
4.5.3 Task properties.....	74
4.5.4 Motion task	74
4.5.5 Task monitoring (watchdog).....	77
4.5.6 Task stack verification.....	78
4.5.7 Runtime measurements.....	79
4.5.8 Task configuration.....	83
4.5.9 Task cycle times and timing.....	83
4.6 IndraMotion MLD error handling.....	84
4.6.1 IndraMotion MLD operating behavior.....	84
4.6.2 General (for MLD-S and MLD-M).....	85
4.6.3 Configuring the error reaction in MLD-S.....	90
4.6.4 MLD-M error handling and reaction.....	91
5 MLD communication interfaces and data channels.....	97
5.1 Introduction and overview.....	97
5.2 IndraMotion MLD data channels.....	99
5.2.1 Cyclic data channels.....	99
5.2.2 Acyclic data channels / interfaces.....	117
5.2.3 Configuring the inputs and outputs (I/O configuration of MLD).....	131
5.2.4 Cyclic data in MLD-M system mode.....	140
5.3 Sercos I/O.....	145
6 Visualization, engineering and connection via OCI.....	147
6.1 Overview.....	147
6.2 Interfaces for connecting communication partners.....	149
6.2.1 Ethernet interface (TCP/IP).....	149

	Page
6.2.2 Master communication interface.....	151
6.3 Connecting IndraControl V.....	152
6.3.1 Introduction and overview.....	152
6.3.2 Configuring the symbols for variable/parameter access.....	154
6.3.3 Connecting IndraControl VE* via Ethernet.....	155
6.4 Open Core Interface for drives.....	158
6.4.1 Target platforms.....	160
6.4.2 Supported devices.....	160
6.4.3 Application options.....	160
6.4.4 SDK: Requirements and installation.....	162
6.4.5 SDK: Contents.....	162
6.4.6 Application examples.....	164
7 Notes on commissioning and application.....	175
7.1 Requirements for using Rexroth IndraMotion MLD.....	175
7.1.1 Firmware and hardware requirements.....	175
7.1.2 Software requirements.....	175
7.1.3 Requirements for communication	175
7.2 Programming in IndraWorks MLD.....	176
7.2.1 Using technology functions.....	177
7.2.2 Free programming of Rexroth IndraMotion MLD.....	180
7.2.3 Using the "AxisInterface" for programming Rexroth IndraMotion MLD.....	192
7.2.4 Automated program generation with "GAT compact".....	193
7.3 PLC project structure.....	198
7.3.1 General information.....	198
7.3.2 Saving projects in the drive.....	199
7.3.3 Boot project.....	200
7.3.4 File system in the drive.....	201
7.3.5 External memory card as a storage medium.....	202
7.3.6 IndraWorks package.....	202
7.3.7 Libraries.....	203
7.4 Programming and commissioning steps.....	203
7.4.1 Creating a new project.....	203
7.4.2 Importing a PLC Project.....	204
7.4.3 Exporting an IndraLogic project (*.project file) from IndraWorks MLD.....	205
7.4.4 Loading the IndraLogic project to the drive.....	205
7.4.5 Archiving / restoring an IndraWorks project.....	206
7.4.6 Loading the source code.....	207
7.4.7 Opening a project in the control unit.....	207
7.4.8 Archiving an MLD project.....	207
7.4.9 Importing an exported MLD project.....	217
7.4.10 Updating the IndraLogic device version.....	219
7.4.11 Changing the libraries valid for a PLC project	219
7.4.12 Importing an external library file.....	220
7.5 Drive as PLC device.....	220
7.5.1 General information.....	220

Table of Contents

	Page
7.5.2 Sercos III drive as PLC device.....	221
7.6 Supply unit as PLC device.....	230
7.6.1 General information.....	230
7.6.2 Parameterization.....	230
7.6.3 Configuring the cyclic data.....	231
7.6.4 PLC programming.....	233
7.7 Comparing the "IndraDrive Cs" range with the "IndraDrive C"/"IndraDrive M" ranges.....	235
7.7.1 Introduction.....	235
7.7.2 Physical interfaces.....	235
7.7.3 Byte order.....	236
7.7.4 MLD performance comparison.....	240
8 Programming information.....	241
8.1 Industrial standards for programming	241
8.1.1 General information.....	241
8.1.2 IEC 61131.....	241
8.1.3 PLCopen.....	241
8.2 Libraries for Rexroth IndraMotion MLD.....	242
8.2.1 General properties of the libraries.....	242
8.2.2 Libraries.....	243
8.3 Retain data backup and restore.....	247
8.3.1 Manual retain data backup and restore.....	247
8.3.2 Saving persistent variables upon "Reset origin" and initializing with saved persistent data upon PLC restart.....	251
8.4 Accessing files on the optional memory card.....	255
8.5 Start behavior and boot project.....	256
8.5.1 General information.....	256
8.5.2 Brief description.....	256
8.5.3 Notes on application and programming.....	257
8.6 Guidelines for programming with IndraLogic.....	258
8.6.1 Overview.....	258
8.6.2 Structuring PLC projects.....	258
8.6.3 Programming languages.....	259
8.6.4 Global data.....	260
8.6.5 Program header.....	260
8.6.6 History.....	261
8.6.7 Type identifiers.....	262
8.6.8 Instance identifiers.....	264
8.6.9 Definition of standard interfaces at function blocks.....	266
8.6.10 Error handling.....	270
8.6.11 Version assignment of libraries.....	270
8.6.12 Code optimization.....	271
8.7 VCS – Version Control System.....	272

	Page
9 Diagnostic and service functions.....	275
9.1 Overview and introduction.....	275
9.2 Device editor.....	275
9.2.1 Device editor, general information.....	275
9.2.2 Status.....	275
9.2.3 Files.....	276
9.2.4 Task deployment.....	276
9.2.5 Log.....	276
9.2.6 PLC settings.....	278
9.2.7 PLC shell.....	279
9.2.8 Information.....	279
9.3 Diagnostic functions.....	279
9.3.1 Standard drive diagnostic functions.....	279
9.3.2 MLD diagnostic functions.....	280
9.4 Service functions.....	284
9.4.1 Service functions of the PLC (IndraMotion MLD).....	284
9.4.2 Firmware replacement.....	285
9.4.3 Firmware version upgrade.....	286
10 Converting projects.....	291
10.1 Introduction.....	291
10.2 Requirements in MLD-1G.....	291
10.3 Adding to MLD-2G.....	292
10.3.1 Adding a project.....	292
10.3.2 Compiling errors after converting projects.....	293
10.3.3 Adjustments in MLD-2G.....	295
10.4 Converting projects MPx18 --> MPx20.....	295
11 Service and support.....	297
Index.....	299

1 Introduction

1.1 About this documentation

Editions of this documentation

Edition	Release date	Notes
05	2019-05	- General corrections, amendments and improvements - Documentation list of MLC technology functions included
04	2018-11	- In MPB-21 and above, the internal code memory (RAM in control section) is reduced to 1.2 MB (see "chapter "Amount of resources (memory)" on page 19") - Mentioned "Sequential Motion Control (SMC)" - General corrections, amendments and improvements
03	2017-06	- Extended by "IndraDrive ML" ("PSB" firmware) - Extended by MPx-19 - Extensions and changes regarding MPx-20 - New libraries (see "Supported libraries") - In case of synchronization function blocks, VmAxis1, RmAxis1, LinkAxis1 can also be used as master axis in MLD-S (see "Axis addressing") - Added description of "AxisInterface" - Added description of "GAT compact" - Added description of "Version Control System (VCS)" - Added short descriptions of "IndraControl S20" and of "Drive or frequency converter as Sercos peripherals" - Added descriptions of "Drive as PLC device" and of "Supply units as PLC device" - Added description of "Open Core interface for drives"
02	2014-08	- Corrections - Information on available memory - Amendments: - WinStudio configuration required when connecting an HMI (IndraControl VE*) via Ethernet - Description of context menu for "Application" - Information: "Advanced" performance level cannot be configured when "MLD-M" system mode has been configured for MPC
01	2014-01	First edition

Tab. 1-1: Record of revisions

Means of representation in this documentation

To facilitate reading of this documentation, the table below contains the means of representation and notations of recurring terms.

What?	How?	For example...
Important facts that should be highlighted in the body text	Boldface	"IndraDrive Cs" provides an analog input . There is no analog output .
Parameter names, diagnostic message names, function designations	Quotation marks	In the I/O configuration (IndraWorks), the inputs were linked to parameter "P-0-1390, PLC input WORD0".

Tab. 1-2: Conventions of notation

Helpful information and tips are highlighted in the text. A symbol tells you what kind of information is used in the text:

	This box contains important information that should be taken into consideration.
	This symbol highlights useful tips and tricks.

Signal words in accordance with ANSI Z535.6-2011 draw the reader's attention to hazards (see "[Explanation of signal words and the safety alert symbol](#)").

Your feedback

Your experience is important to improve our products and documentations.

If you want to report errors or suggest changes to this documentation, please send your feedback to the following e-mail address:

Dokusupport@boschrexroth.de

Please include the following information in your feedback:

- The number indicated on the back of the front page under "Internal File Reference".
- The page number.

1.2 Structure of the documentation

This documentation is structured for various audiences to make navigation as quick and easy as possible. The documentation is divided into the following main chapters, each of which is tailored to a defined audience or use:

- [chapter 1.5 "Product presentation" on page 9](#)
Provides an overview of the main features and the intended use of "IndraMotion MLD"
- [chapter 7 "Notes on commissioning and application" on page 175](#)
Notes on the commissioning and use of "IndraMotion MLD"
- [chapter 9 "Diagnostic and service functions" on page 275](#)
This chapter explains troubleshooting options if a problem occurs in conjunction with "IndraMotion MLD"
- [chapter 10 "Converting projects" on page 291](#)
This chapter describes how to convert IndraLogic 1.x projects (MLD-1G) to IndraLogic 2G projects (MLD-2G)

1.3 Documentations

1.3.1 Overview

The description of the functionality of "IndraMotion MLD" is divided as follows:

- **Application Manual for Rexroth IndraMotion MLD-2G** (this documentation)
Describes the specific functionalities of "IndraMotion MLD", explains how the PLC is integrated into the drive, and the technical options and functions
- **Getting Started with Rexroth IndraMotion MLD-2G**
Explains how to get started when using Rexroth IndraMotion MLD for the first time (preparation and setup, commissioning, programming)
- **Library for Rexroth IndraMotion MLD-2G**
 - Describes the functions and function blocks specially programmed for "IndraMotion MLD" and compiled into a special library
 - Contains references to the **system libraries** used by several systems
 - Contains references to the **basic libraries** used by several systems



You can easily access the descriptions of the system and basic libraries via the online help feature in IndraWorks.

- **MLD technology functions**
MLD technology functions are implemented in the drive by the drive-integrated PLC (IndraMotion MLD). Technology functions are made available as a compiled PLC project (*.par files) which provides the technology functions like a firmware function. Using the extended diagnostic functions made available as a compiled PLC project does not require any programming knowledge.
An individual documentation is available for each MLD technology function.

- **Operating and Programming Guide** for IndraLogic 2G
This documentation specifically describes the programming environment, as well as the programming and creation of PLC programs
- **Parameter description** for IndraDrive
Apart from the PLC-specific parameters, all other drive parameters are documented in the Reference Book of parameters
- **Description of diagnostic messages** for IndraDrive
Apart from the PLC-specific diagnostic messages, all other diagnostic drive messages are documented in the Reference Book of diagnostic messages (also called "Troubleshooting guide")

1.3.2 Drive systems, system components

Title	Type of documentation	Document typecode ¹⁾	Material number R911...
Rexroth IndraControl VR 21 Operator Panel	Operating Instructions	DOK-SUPPL*-VR21**-.01*-ITxx-EN-P	339476
IndraControl VH 2110.01 Hand-Held Terminal	Operating Instructions	DOK-SUPPL*-VH*21**-.01*-ITxx-EN-P	346750
Rexroth IndraControl VEP / VEH	Project Planning Manual	DOK-SUPPL*-VEH/VEP****-PRxx-EN-P	309682
Rexroth IndraControl S20	Operating Instructions	DOK-CONTRL-S20*SYS*INS-APxx-EN-P	335988
Automation Terminals of the Rexroth Inline Rexroth Inline	Application Manual	DOK-CONTRL-ILSYSINS***-AWxx-EN-P	317021

1) In the document typecodes, "xx" is a placeholder for the current edition of the documentation (e.g.: AW01 is the first edition of an Application Manual)

Tab. 1-3: Documentations – overview

1.3.3 Motors

Title	Type of documentation	Document typecode ¹⁾ DOK-MOTOR*-...	Material number R911...
MAD / MAF Asynchronous Motors MAD / MAF	Project Planning Manual	MAD/MAF****-PRxx-EN-P	295781
MBS-H Synchronous Kit Spindle Motors	Project Planning Manual	MBS-H*****-PRxx-EN-P	297895
MLF Synchronous Linear Motors	Project Planning Manual	MLF*****-PRxx-EN-P	293635
MCL Ironless Linear Motors MCL	Project Planning Manual	MCL*****-PRxx-EN-P	330592

Title	Type of documentation	Document typecode ¹⁾ DOK-MOTOR*-...	Material number R911...
MKE Synchronous Motors Synchronous Servo Motors for Potentially Explosive Areas acc. to ATEX and UL / CSA	Project Planning Manual	MKE*GEN2***-PRxx-EN-P	297663
MSK Synchronous Servo Motors	Project Planning Manual	MSK*****-PRxx-EN-P	296289
MSK Synchronous Servo Motors for Potentially Explosive Areas	Project Planning Manual	MSK*EXGIIK3-PRxx-EN-P	312709
MSM Synchronous Servo Motors	Data Sheet	MSM*****-DAxx-EN-P	329338
MS2E Synchronous Servo Motors acc. to ATEX Directive 2014/34/EU	Project Planning Manual	MS2E*****-PR01-EN-P	394140
MS2N Synchronous Servo Motors	Project Planning Manual	MS2N*****-PRxx-EN-P	347583
MBT Synchronous Torque Motors	Project Planning Manual	MBT*****-PRxx-EN-P	298798

1) In the documentation typecodes, "xx" is a placeholder for the current edition of the documentation (e.g.: PR01 is the first edition of a Project Planning Manual)

Tab. 1-4: Documentations – motors

1.3.4 Cables

Title	Type of documentation	Document typecode ¹⁾ DOK-CONNEC-...	Material number R911...
Rexroth Connection Cables IndraDrive and IndraDyn	Selection Data	CABLE*INDRV-CAxx-EN-P	322949

1) In the document typecodes, "xx" is a placeholder for the current edition of the documentation (example: CA02 is the second edition of the documentation "Selection Data")

Tab. 1-5: Documentations – cables

1.3.5 Firmware

Title	Type of documentation	Document typecode ¹⁾ DOK-INDRV*-...	Material number R911...
IndraDrive MPx-21 Functions	Application Manual	MP*-21VRS**-APxx-EN-P	385758
IndraDrive MPx-20 Functions	Application Manual	MP*-20VRS**-APxx-EN-P	345608
IndraDrive MPx-20 Version Notes	Release Notes	MP*-20VRS**-RNxx-EN-P	345606
IndraDrive Power Supply Basic PSB-21 Functions	Application Manual	PSB-21VRS**-APxx-EN-P	385754
IndraDrive Power Supply Basic PSB-21 Version Notes	Release Notes	PSB-21VRS**-RNxx-EN-P	385752
IndraDrive Power Supply Basic PSB-20 Functions	Application Manual	PSB-20VRS**-APxx-EN-P	345610
Rexroth IndraDrive Power Supply Basic PSB-19 Functions	Application Manual	PSB-19VRS**-APxx-EN-P	345602
Rexroth IndraDrive MPx-18 Functions	Application Manual	MP*-18VRS**-APxx-EN-P	338673
Rexroth IndraDrive MPx-18 Version Notes	Release Notes	MP*-18VRS**-RNxx-EN-P	338658
Rexroth IndraDrive MPx-17 Functions	Application Manual	MP*-17VRS**-APxx-EN-P	331236
Rexroth IndraDrive MPx-17 Version Notes	Release Notes	MP*-17VRS**-RNxx-EN-P	331588
Rexroth IndraDrive MPx-16 Functions	Application Manual	MP*-16VRS**-APxx-EN-P	326767

Introduction

Title	Type of documentation	Document typecode ¹⁾ DOK-INDRV*-...	Material number R911...
Rexroth IndraDrive MPx-16 Version Notes	Release Notes	MP*-16VRS**-RNxx-EN-P	329272
IndraDrive MPx-16 to MPx-21 and PSB Parameters	Reference Book	GEN1-PARA**-RExx-EN-P	328651
IndraDrive MPx-16 to MPx-21 and PSB Diagnostics	Reference Book	GEN1-DIAG**-RExx-EN-P	326738
Rexroth IndraDrive Integrated Safety Technology "Safe Torque Off" (as of MPx-16)	Application Manual	SI3-**VRS**-APxx-EN-P	332634
IndraDrive Integrated Safety Technology "Safe Motion" (as of MPx-18)	Application Manual	SI3*SMO-VRS-APxx-EN-P	338920
Rexroth IndraDrive Rexroth IndraMotion MLD Libraries as of MPx-17	Reference Book	MLD-SYSLIB2-RExx-EN-P	332627
IndraDrive Rexroth IndraMotion MLD Libraries as of MPx-18	Reference Book	MLD-SYSLIB3-RExx-EN-P	338916
Rexroth IndraDrive Rexroth IndraMotion MLD as of MPx-17	Application Manual	MLD2-**VRS*-APxx-EN-P	334351
IndraDrive IndraMotion MLD MPx-18 and above	Application Manual	MLD3-**VRS*-APxx-EN-P	338914

1)

In the document typecodes, "xx" is a placeholder for the current edition of the documentation (e.g.: RE02 is the second edition of a Reference Book)

Tab. 1-6:

Documentations – firmware

1.3.6 MLD technology functions

Title	Type of documentation	Document typecode ¹⁾	Material number
MLD Technology Function "Process Controller"	Application Manual	DOK-MLD***-TECFUNC*PC*-APxx-EN-P	R911397138
MLD Technology Function "Vibration Damping and Avoiding Vibration"	Application Manual	DOK-MLD***-TECFUNC*VDA-APxx-EN-P	R911397135
MLD technology function "Kinetic Buffering"	Application Manual	DOK-MLD***-TECFUNC*KB*-APxx-EN-P	R911397139

1) In the document typecodes, "xx" is a placeholder for the current edition of the documentation (e.g.: AP01 is the first edition of an Application Manual)

Tab. 1-7: Documentations – MLD technology functions

1.4 Reference documentations

Title	Type of documentation	Document typecode ¹⁾	Material number
IndraWorks 14VRS Basic libraries IndraLogic 2G	Library	DOK-IL*2G*-BASLIB**V14-LIxx-EN-P	R911338920
IndraWorks 14VRS PLC Programming System IndraLogic 2G	Application Manual	DOK-IWORKS-IL2GPRO*V14-APxx-EN-P	R911343571
IndraMotion for Metal Forming 14VRS Sequential Motion Control	Reference Book	DOK-IM*MF*-TF*SMC**V14-RExx-EN-P	R911343865
Rexroth IndraMotion for Handling 13VRS Functional Description	Reference Book	DOK-IM*HA*-FUNC****V13-RExx-EN-P	R911343861

1) In the document typecodes, "xx" is a placeholder for the current edition of the documentation (e.g.: RE02 is the second edition of a Reference Book)

Tab. 1-8: Reference documentations

1.5 Product presentation

1.5.1 Introduction and overview

"IndraMotion MLD" (Motion Logic Drive-based) is a PLC integrated in the drive provided as a functional firmware package for the "Rexroth IndraDrive" range.

"IndraMotion MLD" combines motion and PLC functions in a modern, open automation platform for modular machine designs. The decentralized control architecture forms a compact Motion Logic system based on the scalable IndraDrive platform, eliminating the need for higher-level control units.

- Electronic synchronization of up to 10 servo axes
- Up to 4 Sercos III I/O nodes are supported
- Visualization via "IndraControl Vxx"

- Applications: Belt conveyors, roll feeds, feed straightening machines, unwinders, straightening machines

"IndraMotion MLD" is available in the characteristics MLD-S (single-axis "Motion Logic Control") and MLD-M (multi-axis "Motion Logic Control").

	IndraMotion MLD (drive-integrated PLC)	
	IndraMotion MLD-S: Single-axis "Motion Logic Control"	IndraMotion MLD-M: Multi-axis "Motion Logic Control" ("MPC" firmware only)
"IndraDrive C" / "IndraDrive M" (with Cxx02 control sec- tions)	X	X
"IndraDrive Cs"	X	X
"IndraDrive Mi"	X	-
"IndraDrive ML"	X	X



This document describes the functionality and use of the "IndraMotion MLD" drive-integrated PLC with "IndraDrive Cs" devices.

1.5.2 Brief description

"IndraMotion MLD" is a characteristic of the "IndraMotion" range that is embedded in the drive controllers of the "Rexroth IndraDrive" range. It does not require any additional hardware option, just the licensing of a functional firmware package ["ML", "MA" or "TF" (MPx-20 and above)].

See also Functional description of firmware "Functional packages"

Other characteristics of the "IndraMotion" range:

- IndraMotion MLC
- IndraMotion MTX



All IndraMotion variants may be configured and operated with the "IndraWorks Engineering" Engineering tool.

The "IndraWorks MLD" Engineering tool is available for using only the "IndraMotion MLD" variant.

Use the "IndraLogic" programming tool embedded in IndraWorks for **programming** IndraMotion.

The programs created with "IndraLogic" are portable as long as the general requirements are met (see "[Programming information](#)").

Main features of "IndraMotion MLD"

These are the main features of "IndraMotion MLD":

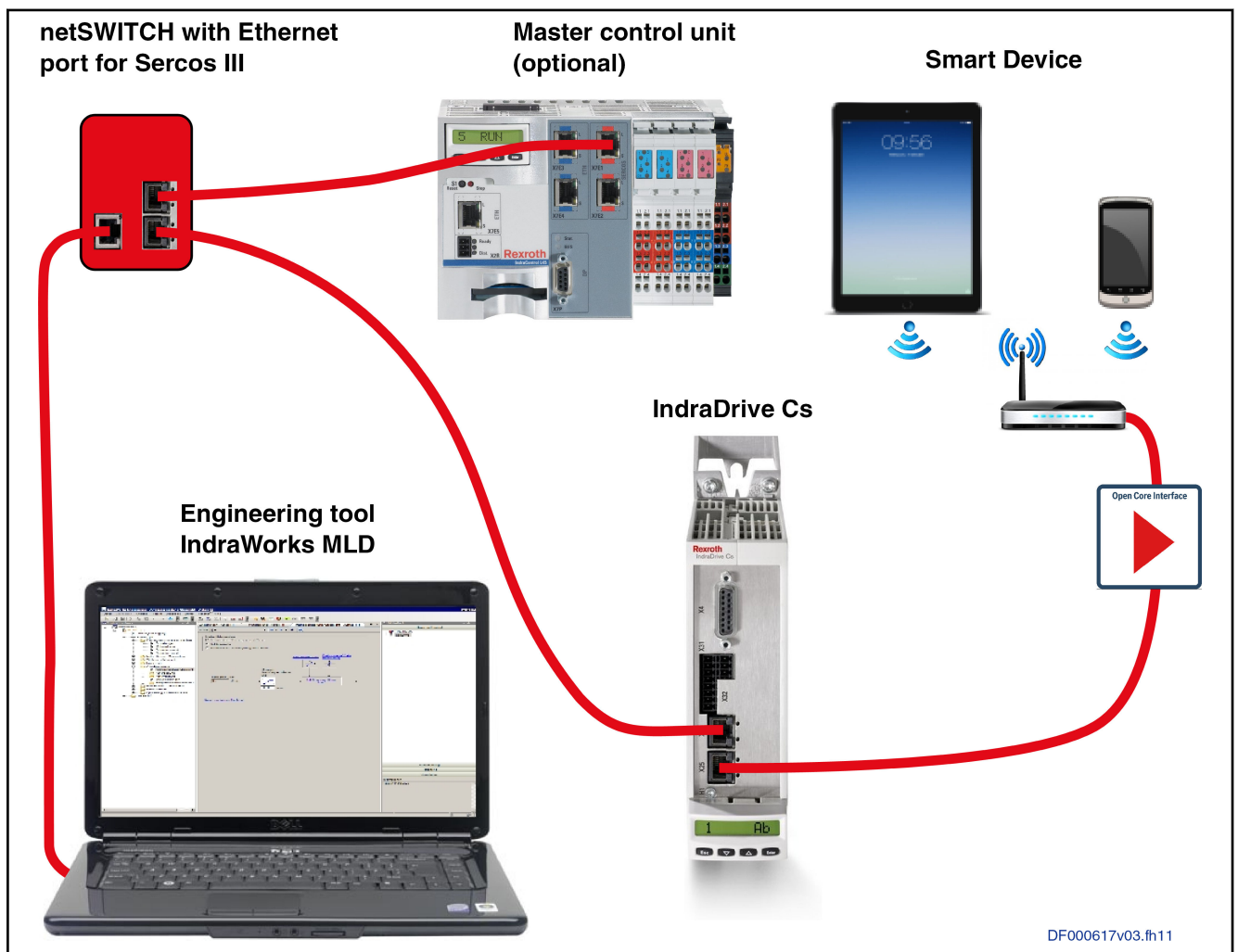
- "IndraMotion MLD" is perfect for efficiently executing motion and logic tasks with drive firmware functions (no additional hardware required)
- "Open drive", i.e., direct and transparent access to all drive functions and parameters
- **Open Core Interface (OCI)**: Allows for the use of own, real time-compatible applications directly on the control or individual IT applications on external devices (firmware version 20 and above)

- Programming functional extensions or custom motion solutions in accordance with **IEC 61131-3** based on comprehensive standard libraries and additional libraries ("technology libraries")
- Firmware version 20 and above: Simplified motion programming with the help of "AxisInterface" and "GAT compact"
- Flexible communication extension via on-board interfaces

"IndraMotion MLD" is available with the following characteristics:

- IndraMotion MLD-S (single-axis solution; "MPB" and "MPC" firmware)
 - As an intelligent servo axis (drive functionality extension)
 - As stand-alone single-axis "Motion Logic Control"
- IndraMotion MLD-M (multi-axis solution; "MPC" firmware only)
 - As a stand-alone multi-axis "Motion Logic Control" using CCD (Cross Communication Drives) based on Sercos III

1.5.3 IndraMotion MLD-S



Connection Please refer to the appropriate Project Planning Manual for the interfaces used to interconnect the components

Fig. 1-1: "IndraMotion MLD-S" basic system structure

Features of "IndraMotion MLD-S"

"IndraMotion MLD-S" comes with the following main features:

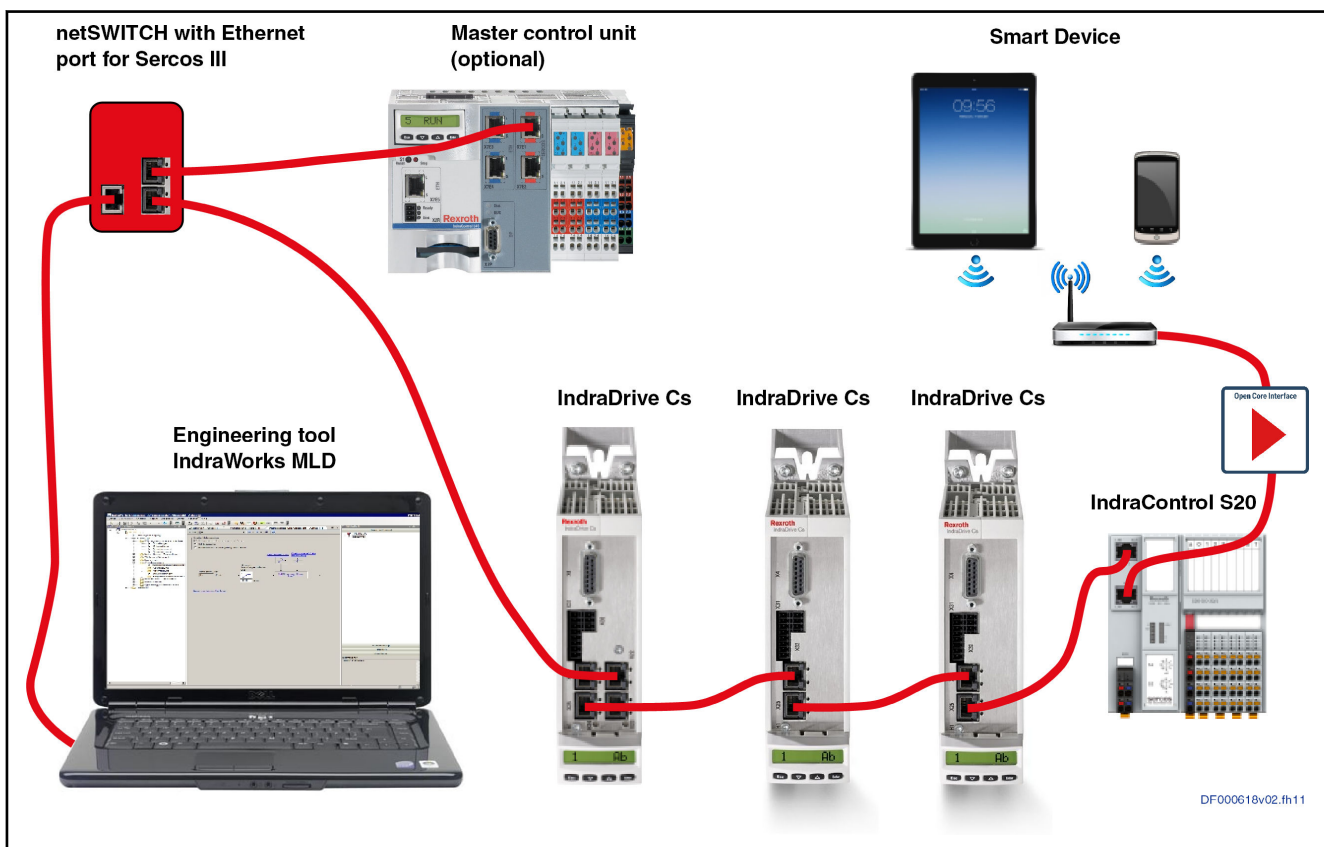
- "Intelligent servo axis" characteristic (extension of drive functionality) or "stand-alone single-axis "Motion Logic Control""
- Commanding local axis with direct access to the drive's device control
- Direct access to all drive parameters using system-wide PLC variables, functions, function blocks or a configurable synchronous channel
- Direct access to the drive's digital inputs/outputs and analog input (local axis)

Also included in the "stand-alone single-axis "Motion Logic Control"" characteristic:

- Motion Control functionality for 1 axis by commanding the local axis with a corresponding PLCopen-based Motion Control library

1.5.4 IndraMotion MLD-M

"IndraMotion MLD-M" not only controls the local axis but also remote axes (so-called "CCD slaves").



Connection Please refer to the appropriate Project Planning Manual for the interfaces used to interconnect the components

Fig. 1-2: "IndraMotion MLD-M" basic system structure

Features of "IndraMotion MLD-M"

"IndraMotion MLD-M" comes with the following additional main features:

- Only available with "MPC" firmware
- Motion Control functionality for up to 10 axes by corresponding PLCopen-based Motion Control library

- CCD cross communication interface to "remote axes" based on Sercos III
 - For commanding local and remote axis motion
 - For connecting more Sercos peripherals ("Rexroth Inline", "Rexroth IndraControl S20", drive, frequency converter)

(see "[MLD-M extensions](#)")
- Direct access to the drives' digital and analog inputs/outputs (local and remote axes)
- All drive parameters (local and remote axes) can be accessed
- Sercos III interface to the slave axes:
 - **Number of axes and cycle time**

Max. 9 slaves (T=500 µs with 1 slave, T=1000 µs with 2 slaves up to T=4000 µs with 9 slaves)
 - Cyclic data channel (MDT, AT) with maximum 48 bytes and 16 parameters each
 - Parameter or service channel (4-byte info container)

1.5.5 MLD-M extensions

Rexroth Inline

The "Rexroth Inline" product range provides digital and analog I/O extensions in two variants:

- Modular I/Os: I/O modules that can be added modularly ("Rexroth Inline Modular IO")
- Block I/Os: compact remote I/O modules ("Rexroth Inline Block IO")

The "Rexroth Inline Block IO" and "Rexroth Modular IO" I/O modules can be used in mixed mode. A total number of 4 block and modular I/O modules (bus couplers) cannot be exceeded.

The I/O modules can only be operated under the following conditions:

- The I/O modules can only be operated in conjunction with an MLD-M system (Advanced device with MPC firmware).
- A functional firmware package for "IndraMotion MLD" has to be enabled for operation.
- The I/O modules have to be configured correctly in the project.
- The MLD project has been loaded to the control unit, i.e., the I/O configuration is included in the MLD project.
- The Sercos ring is in Sercos phase 4. (IOs can only be updated in Sercos phase 4.)



In MLD-2G, the parameters of the I/O modules can be read via the service channel for diagnostics; however, it is not possible to apply the process data of the I/O modules to the cyclic communication of the master communication. In cyclic mode, the I/O modules can only be read/written via the MLD.



The signals applied to the "Inline Block I/O digital fast" I/O modules are typically read/written via a CCD-synchronous task, which requires the MLD-M system mode to be configured (P-0-1800.0.1, CCD: Configuration).

"Rexroth Inline Block IO" in conjunction with "IndraMotion MLD"

"Rexroth Inline Block IO" is the ideal solution for applications with limited I/O extent and low complexity. With its compact design, it is the perfect solution for installing in flat control boxes.

A maximum of 4 "Rexroth Inline Block IO" devices can be operated at one drive controller.

A latch-on inline connector is used for wiring. No additional terminal levels are required thanks to its multi-conductor connection technology.

A number of variants of the "Rexroth Inline Block IO" are available for Sercos III connection:

- Inline Block IO with **16 digital inputs and 16 configurable digital inputs or outputs** (R-ILB S3 24 DI16 DIO16)
Documentation: "Rexroth Inline Block IO for Sercos III With Digital Inputs and Outputs", DOK-CONTRL-S3DI16DIO16-KB, R911170500
- Inline Block IO with **4 analog inputs and 2 analog outputs** (R-ILB S3 24 AI4 AO2)
Documentation: "Rexroth Inline Block IO for Sercos III With Analog Inputs and Outputs", DOK-CONTRL-S3AI4AO2***-KB, R911170558
- Inline Block IO with **16 digital high-speed inputs and 16 digital high-speed inputs or outputs, timestamp and oversampling** (R-ILB S3 24 DI16 DIO16/F)
Documentation: In preparation
- Inline Block IO with **6 analog high-speed inputs and 4 analog high-speed outputs, oversampling** (R-ILB S3 AI6 AO4/F)
Documentation: In preparation

"R-ILB S3 24 DI16 DIO16" / "R-ILB S3 24 DI16 DIO16/F"

The Inline Block IO "R-ILB S3 24 DI16 DIO16" has been designed for use within Sercos III. It is used to register digital input signals and output digital signals. "R-ILB S3 24 DI16 DIO16" features 16 digital inputs and 16 freely configurable digital inputs or outputs.

"R-ILB S3 24 DI16 DIO16/F" also features 16 digital inputs and 16 freely configurable digital inputs or outputs. With its quick signal processing, it is suitable for use in time-critical applications.

"R-ILB S3 24 AI4 AO2" / "R-ILB S3 24 AI6 AO4/F"

The Inline Block IO "R-ILB S3 24 AI4 AO2" has been designed for use within Sercos III. It can be used to register current or voltage signals as well as temperature sensors. It also provides the option of outputting analog current and voltage signals. "R-ILB S3 24 AI4 AO2" features 4 analog inputs and 2 analog outputs.

"R-ILB S3 24 AI6 AO4/F" features 6 analog inputs and 2 analog outputs. With its quick signal processing, it is suitable for use in time-critical applications.



For general information on the "Rexroth Inline" product range, refer to the documentation "Automation Terminals Of The Rexroth Inline Product Range", document typecode "DOK-CONTRL-ILSY-SINS***-AWxx-EN-P".

"Rexroth Inline Modular IO" in conjunction with "IndraMotion MLD"

A maximum of 4 bus couplers for Sercos III with a maximum of 16 modules (nodes) each can be evaluated at a CCD master.

All modules are supported that can be operated as simple digital or analog I/Os.

The following devices ("Rexroth Inline" product range) are available:

- Bus coupler for Sercos III with digital inputs and outputs (R-IB IL S3 BK DI8 DO4-PAC)
- Modules with analog inputs (R-IB IL AI xx) or analog outputs (R-IB IL AO xx)
- Modules with digital inputs (R-IB IL 24 DI xx) or digital outputs (R-IB IL 24 DO xx)
- System terminals for different functions e.g. PWM, serial communication RS232, RS485, encoder terminals for SSI encoders, etc.



For general information on the "Rexroth Inline" product range, refer to the documentation "Automation Terminals Of The Rexroth Inline Product Range", document typecode "DOK-CTRL-ILSY-SINS***-AWxx-EN-P".

Rexroth IndraControl S20

Setup of an IndraControl S20 station

"Rexroth IndraControl S20" is an I/O system in modular design for the control cabinet. Open for all Ethernet-based communication protocols, IndraControl S20 allows for highest flexibility. Apart from that, IndraControl S20 features fast reaction time and installation, robust design and mechanics while being particularly easy to handle. It is used for transmitting the process signals to a higher-level control unit. In this connection, different networks are supported.

An IndraControl S20 station comprises individual modules that are latched onto a mounting rail. A control unit or a bus coupler is the head of the station. To it, I/O modules are added. Bus base modules are used to connect the individual modules among each other and to the station head. The modules are clicked into each other in the mounting rail and form the local bus.

For more information, refer to the application description "Rexroth IndraControl S20" (part no. R911335988).

Drive or frequency converter as Sercos peripherals

Up to 4 IndraDrive devices or frequency converters of "EFC" type can be configured as Sercos III peripherals. This increases the maximum number of configurable I/Os.

The function blocks required for controlling "IndraDrive" or "EFC" as Sercos III peripherals are contained in the "RMB_SercosIII_Util.library" library.



A total of 4 more peripherals can be configured. The peripherals include drives, frequency converters, Inline Block IO or IO bus couplers (Inline Modular IO, IndraControl S20, ...).

1.5.6 Terms, basic principles

General information

- A **technology function** or **technology package** is a self-contained, comprehensive functionality (e.g., preventive diagnostic system).
- Based on technology function blocks and packages, Bosch Rexroth also provides application-specific complete solutions, such as "Sequential Motion Control (SMC)" and "IndraMotion for Handling".
 - "Sequential Motion Control (SMC)" for IndraDrive extends the IndraDrive servo drives by sequential programming and axis coupling functions. This preconfigured system solution for all sequen-

tial motion tasks provides comprehensive functions for automatic and manual modes.

- "IndraMotion for Handling" is a control-based, preconfigured and at the same time open Motion Control solution. You may use the system for all simple and Cartesian motion tasks in automation.

PLC programming

The paragraph below contains definitions of some basic terms used in PLC programming:

- **Resources** are objects for organizing a project, tracing variable values and configuring the project for use in the target and in the network (global variables, variable configuration, sampling trace, PLC configuration, task configuration, watch and recipe manager).
 - The **variable configuration** is used for configuring global variables, which can be used in the entire project or network.
 - **Sampling trace** provides a functionality similar to an oscilloscope with memory function: PLC data (variable values) can be recorded and traced.
 - The **PLC configuration** makes it possible to configure the PLC hardware. "IndraMotion MLD-S" does not require a PLC configuration since the device inputs/outputs can be distributed between drive and PLC and addressed with the commissioning software.
 - It is possible, but not required, to control the processing of a project (control the program) via tasks; if no **task configuration** is available, the project has to contain the "PLC_PRG" function block.
 - The **watch manager** allows the values of the entered variables to be displayed in online mode. The **recipe manager** allows variables to be preset with constant values.
- A **task** is a time unit in the processing of an IEC program. It is defined by a name, a priority and a type. The type determines which condition will trigger the start of the task. This condition can either be defined by a time (cyclic interval, freewheeling) or by an internal or external event that is to trigger the task, e.g., the rising edge of a global project variable or an interrupt event of the control unit.
- A **function** is a block with the following properties:
 - A function returns exactly one piece of data (which can also contain multiple elements, such as arrays or structures) when it is executed.
 - A function can be called in textual languages as an operand used in expressions.
 - A function has no "memory", i.e., the contents of variables are not stored.
 - A function does not need an instance.
- A **function block** has the following properties:
 - A function block can return multiple values.
 - A function block has a "memory", i.e., the contents of variables are stored.
 - A function block requires an instance to be specified.
- **Programs** are globally known in the entire project. Programs are not instantiated. The variables of the program only exist once. Programs can

be called in a task. A task can call several programs and a program can call functions, function blocks or other programs.

- The **boot project** is a project loaded on the target (in this case a drive controller of the "Rexroth IndraDrive" range with the functional firmware package for "IndraMotion MLD") and started when booting up the drive controller depending on the setting (cf. "P-0-1367, PLC configuration").

PLCopen

PLCopen is an organization in which several manufacturers have collaborated to standardize programming interfaces, commissioning and maintenance, and therefore harmonize programming access to motion control.

Independence from the control architecture is achieved by encapsulating functionality and function block data, i.e., applications programmed according to the standard developed by PLCopen ("PLCopen Motion Control Profile", a set of IEC 61131-3 function blocks) are independent of the executing hardware and therefore can be reused beyond platform limits.

IEC 61131

As is common in IEC 61131, libraries are provided with the IndraWorks packages of "IndraMotion MLD". They contain ready-made functions, function blocks and variables which are used as black boxes. There are two types of libraries:

- **Firmware libraries:** They do not contain any code, just declarations. The code is hard coded in the firmware.
- **IEC libraries:** They have been programmed in accordance with IEC 61131-3 and, if required, their codes can be write- or read-protected.

1.5.7 Function overview and system properties

General basic functions of the integrated PLC

Freely programmable PLC in accordance with IEC 61131

By defining the industry standard for programming in automation, IEC 61131-3 has for the first time established the basics of combined logic and motion control. These are, in essence:

- Defining the **SFC (Sequential Function Chart)** as a powerful programming tool for data flow-oriented programs, such as controllers
- Defining the Pascal-like **ST (Structured Text)** high-level language as a powerful instrument even for programming complex (e.g., mathematical) functionalities
- Defining **basic data types** from BOOL to DINT (32-bit integer) or REAL (32-bit float), as well as arrays and structures
- Defining **function blocks that can be instantiated**
- Defining **standard function blocks** (counter, timer, edge detection, etc.), as well as standard functions (mathematical functions, string processing, comparisons, etc.)

The trend of combined logic and motion control has been reinforced by the official release of the specification by the "PLCopen Motion Task Force". This specification defines functions and function blocks for single-axis positioning, as well as electronic gear functions. In addition, a basic procedure for implementing motion-triggering function blocks was defined.

Programming languages

"IndraMotion MLD" can be programmed using the following programming languages:

- Instruction List (IL)
- Structured Text (ST)
- Ladder Diagram (LD)
- Function Block Diagram (FBD)
- Sequential Function Chart (SFC)
- Continuous Function Chart (CFC)

See "[Programming languages](#)"

The "IndraLogic 2G" programming environment that has been integrated in IndraWorks can be used for programming (see "IndraLogic 2G Programming Guide").

Supported libraries

The libraries supported by "IndraMotion MLD-2G" are described in the documentation "DOK-INDRV*-MLD-SYSLIB3-RExx-EN-P".

NOTICE

Property damage by including unsupported libraries.

Only the libraries listed in the above-mentioned documentation can be included.

Supported interfaces and access mechanisms

see "MLD communication interfaces and data channels"

Amount of resources (memory)

"IndraMotion MLD" has the following resources:

- Internal **code memory** (RAM in control section):



About 33% of the memory resource specified here are reserved for the "Online Change" function, i.e., about 66% are available.

- Up to MPB-20: **2 MB** (non-storing), from MPB-21: **1.2 MB** (non-storing)
- MPC: **12 MB** (non-storing)
- **Program size** (boot program):
 - about **650 kB** with Basic control panel and Advanced control panel in parameter memory
 - about **8 MB** with Advanced control panel and MicroSD memory card (optional memory card has to be enabled via P-0-1367)



The **boot project** is loaded to the **code memory** (RAM) from the program parameters when the device is switched on. This is why the **program memory** has to be big enough to store the project.

- **Data memory:**



About 10% of the memory resource specified here are reserved for the "Online Change" function, i.e., about 90% are available.

MPB: 3 MB

MPC: 9 MB

There are several data areas with different uses (instances are variables):

- **Global:** Memory for global variables
2048 kB
Note: The instance data are mapped via global variables. All function data are not stored in this segment but in the stack.
- **Memory:** All flag variables addressed with "%M"
32 kB
- **Input:** All input variables addressed with "%I"
16 kB
- **Output:** All output variables addressed with "%O"
16 kB
- **Retain variables and persistent variables**
(VAR_GLOBAL PERSISTENT RETAIN):
 - 472 bytes on Basic control panel (MPB)
 - 31704 bytes on Advanced control panel (MPC) in FRAM



Retain variables and persistent variables can only be declared in a "Global Persistent Variable List" which belongs to an application. Only one such list can be added per application.

Persistent variables are only reinitialized in case the control unit is reset.



The functionality of the retain variables and persistent variables is described in the "IndraLogic 2G, PLC Programming System" documentation: "Remanent variables - RETAIN, PERSISTENT".

- **Sockets**

Sockets are required to establish a connection and for data transfer with TCP (Transmission Control Protocol) and UDP (User Datagram Protocol).

20 sockets are available. The sockets available for the user program are applicable if, for example, IndraWorks with MLD, HMI are active. If more Ethernet communication channels (e.g., FTP) are used, the number of sockets available for MLD is reduced.

- **Symbols**

The number of symbols depends on where the boot project is stored.

- Boot project in parameters:

With MPB and MPC firmware, up to approx. 2,000 symbol variables are possible depending on the data type when the boot project is stored in parameters. This requires no programmed code and only symbol variables declared. The more code the application contains, the smaller the maximum possible number of symbol variables.

- Boot project on MicroSD card:

Compared to storing the boot project in parameters, with MPC firmware many more symbol variables are possible when the boot project is stored on the MicroSD card, because the code memory sets the limit. The following example can be used as a guideline in practice: 10,000 symbol variables require approx. 1.5 MB of code memory.

- **Parameters for general purpose**

- 32-bit parameters

- 32 buffered parameters (P-0-1316 to P-0-1331 and P-0-1370 to P-0-1385)

- 32 unbuffered parameters (P-0-1270 to P-0-1301)

- 32-bit list parameters

- "P-0-1368, PLC Global Register AL0" (list register with 8,192 values, unbuffered)

- "P-0-1389, PLC Global Register GL0" (list register with 1,024 values, buffered)

- "P-0-1311, PLC Global Register GL1" (list register with 1,024 values, buffered)

- "P-0-1312, PLC Global Register GL2" (list register with 1,024 values, buffered)

- Two global **text registers** with 255 characters each
- Display format of the parameters for general purpose can be parameterized via "P-0-1386, PLC display format Global Register"
- Name, unit and limit values of the parameters for general purpose can be configured via PLC functions
- **Process input image (PII)** (max. 72 bytes)
 - "P-0-1390, PLC input WORD0 AT %IB0" to "P-0-1397, PLC input WORD7 AT %IB14"
 - "P-0-1398, PLC input WORD8 AT %IB16" to "P-0-1409, PLC input WORD19 AT %IB38"
 - "P-0-1440, PLC input DWORD25 AT %IB100" to "P-0-1447, PLC input DWORD32 AT %IB128"
- **Process output image (POI)** (max. 40 bytes):
 - "P-0-1410, PLC output WORD0 AT %QB0" to "P-0-1417, PLC output WORD7 AT %QB14"
 - "P-0-1418, PLC output WORD8 AT %QB16" to "P-0-1429, PLC output WORD19 AT %QB38"

Restrictions

The following restrictions and limitations apply:

- The code memory contains the currently running project and not the "boot project". When starting the drive or possibly switching from parameter mode (PM) to operating mode (OM), the boot project is loaded to the code memory from parameters P-0-1352 to P-0-1358.
- If "Online Change" is executed, the changed program is loaded to the second code area and then the system dynamically switches to the second code area.
- The retain memory is automatically used when variables have been defined with keyword "RETAIN". If the number of retain variables created exceeds the available retain memory, an error message is generated when attempting to download.

Performance data

"IndraMotion MLD" has the following performance data:

- Number of function blocks: The value is permanently set to the maximum value (2,048 function blocks)
- Data types: Individual bits up to 32 bits float, as well as specific types and structures
- The minimum PLC cycle time does not depend on the control section and is 1 ms.

See also Functional description of firmware "Performance data"

- The performance depends on ...
 - ... the time slice available for the PLC task
 - ... the code size
 - ... the type of operation: REAL, DWORD, WORD, BYTE, BIT, etc. (The processing speed of REAL is very high due to the integrated FPU [Floating Point Unit] on the processor.)



The processing time depends on multiple factors, such as load from control/operation mode, type of arithmetic operations. At best, the processing time for 1,000 lines of IL is less than 4 µs; at worst, it is more than 100 µs.

The following processing times were measured in a test program used to process 1,000 mixed instructions:

- BASIC control section: 99 µs
- ADVANCED control section: 49 µs

Restrictions

The following restrictions and limitations apply:

- The time slice available for the PLC depends on the control and workload of the drive caused by the operation modes.
- For "Online Change", there is a short delay in the PLC because the cache is dumped and has to be reloaded.
- Due to the system, the bit processing is much slower than the processing of 8-, 16- or 32-bit values.

Instructions for use and applications

General information

The following sections contain the requirements for using "IndraMotion MLD". They also contain instructions for use and applications of "IndraMotion MLD".

The table below gives an overview of all dependencies relevant for using "IndraMotion MLD":

	Device (power section)	Firmware	IndraMotion MLD-S		IndraMotion MLD-M	
IndraDrive Cs	HCS01.1	MPB18VRS and above	ML	MA	-	-
IndraDrive Cs	HCS01.1	MPC18VRS and above	ML	MA	ML	MA
IndraDrive C / IndraDrive M with CSB02.x control section	Note: CSB02.x control sections are only supported by power sections produced from 2007 on	MPB18VRS and above	ML	MA	-	-
IndraDrive C / IndraDrive M with CSH02.3 control section	Note: CSH02.x control sections are only supported by power sections produced from 2007 on	MPC18VRS and above	ML	MA	ML	MA
IndraDrive Mi	KMS02 KSM02	MPB18VRS and above	ML	MA	-	-

ML freely programmable PLC (IEC61131)

MA freely programmable PLC (IEC61131) for complex tasks

Tab. 1-9: System requirements for using "IndraMotion MLD"

Hardware Requirements

Using the drive-integrated PLC (Rexroth IndraMotion MLD) requires the following hardware:

Power section / control section: IndraDrive Cs

- HCS01.1E-Wxxxx-A-0y-B... (as of FWA-INDRV-MPB18VRS)

- HCS01.1E-Wxxxx-A-0y-A-... (as of FWA-INDRV-MPC18VRS and MLD-M)



MLD cannot be used with Economy devices (HCS01.1E-Wxxxx-A-0y-E-...)!

MLD is not available for multi-axis devices (HCQ01.1 and HCT01.1).



Using the PLC functionality does not require any special optional module or any specific control section configuration, because it is a PLC that is running in parallel in the real-time kernel of the drive processor.

Firmware requirements

Using the drive-integrated PLC (Rexroth IndraMotion MLD) requires the base package of the firmware and a functional firmware package (optional expansion package). The functional firmware package must be enabled / licensed (see [Delivery and licensing of the PLC](#)).

Additive functional package

The following designs of the additive functional package are available:

- The **"TF"** design allows self-contained PLC programs (technology functions) by Rexroth to be loaded and used (see "Technology functions").
- The **"ML"** design allows Rexroth IndraMotion MLD-S / MLD-M to be freely programmed.
- The **"MA"** design adds the use of self-contained technology function blocks (library) to the "ML" design.
- The **"ME"** design adds an Ethernet/IP scanner functionality to the "MA" design that is provided in the form of an MLD library.



A technology function is a complete (compiled) PLC project that can consist of multiple function blocks (conforming to IEC). A project is created by combining IEC or firmware function blocks and mostly provides a complex and comprehensive functionality to solve a specific technological application.

The technology functions can be loaded via IndraWorks (see also MLD Application Manual: "Using technology functions").

Examples of technology functions: Following-on cutting devices, pick&place, process controller (register controller, winding computation,...), preventive maintenance, Sytronix...

The table below contains an overview of the base packages of the firmware and the available designs of the additive functional package "IndraMotion MLD":

Introduction

Firmware version	Characteristic	MLD-S				MLD-M			
	Option	ML	MA	ME ¹⁾	TF	ML	MA	ME	TF
FWA-INDRV*-MP*18VRS	MPB	X	X	-	-	-	-	-	-
	MPC	X	X	-	-	X	X	-	-
	MPE	-	-	-	-	-	-	-	-
	MPM	-	-	-	-	-	-	-	-
FWA-INDRV*-MP*19VRS	MPB	X	X	-	X	-	-	-	-
	MPC	X	X	-	X	X	X	-	-
	MPE	-	-	-	-	-	-	-	-
	MPM	-	-	-	-	-	-	-	-
FWA-INDRV*-MP*20VRS / FWA-INDRV*-MP*21VRS	MPB	X	X	X	X	-	-	-	-
	MPC	X	X	X	X	X	X	X	-
	MPE	-	-	-	-	-	-	-	-
	MPM	-	-	-	-	-	-	-	-
FWA-INDRV*-PSB19VRS	PSB	X	X	-	X	-	-	-	-
FWA-INDRV*-PSB20VRS	PSB	X	X	-	X	-	-	-	-

- 1) IndraDrive packages with closed-loop drive controller. "ME" (including "MA" and "ML") Ethernet IP scanner
 2) Function packages always set
 X Additive functional package possible for this firmware
 - Additive functional package not possible for this firmware

Tab. 1-10: Overview of the base packages of the firmware and the available designs of the additive functional package "IndraMotion MLD":

See also Functional description of firmware "Firmware types"

See also Functional description of firmware "Additive functional packages"

Delivery and licensing of the PLC

When the firmware base package is ordered together with the functional firmware package, "IndraMotion MLD" is already enabled at the factory. If the functional firmware package is ordered subsequently, this requires so-called additional licensing. That is to say the user subsequently has to enable or license the PLC functionality.



If the PLC has not been enabled, the drive refuses to communicate with the PLC programming system. A message will be generated. Furthermore, please keep in mind that the guarantee for the drive expires if you use the PLC without licensing!

See also Functional description of firmware "Enabling of functional packages"

Software requirements

The "IndraWorks MLD" engineering tool is required for parameterizing, commissioning and programming a drive with integrated PLC.

Possible applications of IndraMotion MLD

Overview

"IndraMotion MLD" can be used to implement simple programs for extending the drive functionality or making it more flexible ("intelligent servo axis"), or even as a motion system for commanding one or several axes (stand-alone

"Motion Logic Control") or controlling supply units (firmware version 20 and above). Various libraries and technology functions are available for this purpose.

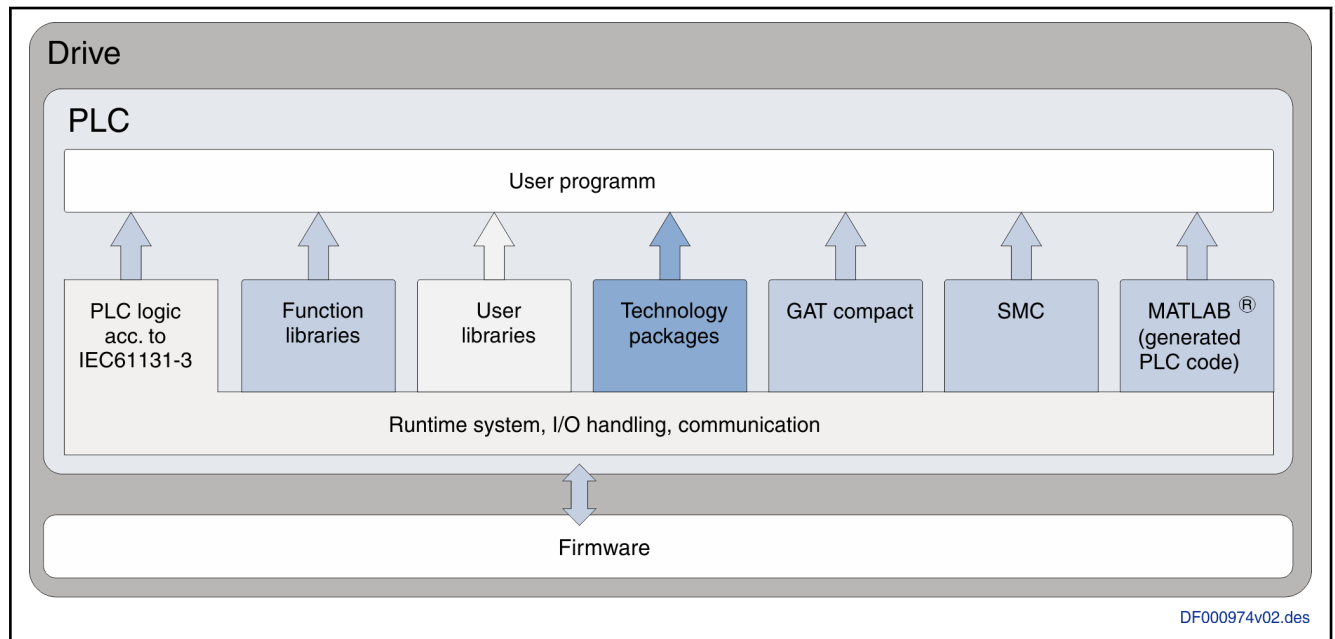


Fig. 1-3: Overview of possible applications in the drive controller

IndraMotion MLD as intelligent servo axis

Extending drive functionality or making it more flexible is most easily achieved with a function block or a simple program.

Features

The "intelligent servo axis" characteristic comes with the following features:

- The drive is managed and controlled by a higher-level control unit (NC/ PLC).
- The higher-level control unit generates the cyclic command values and determines the operation mode.
- The "IndraWorks MLD" engineering tool allows a function to be loaded from the macro libraries to the drive. The customer can then parameterize the loaded function like a "normal firmware function". For the user, macro functions work like implemented firmware functions.
- The drive can temporarily get control to take over motion control.

Example

An intelligent error reaction can be implemented by a function block or a simple program:

When an error occurs, the PLC switches to internal control and carries out a reaction with drive motion. The PLC then cedes control back to the master communication (external PLC).

IndraMotion MLD as stand-alone "Motion Logic Control"

If "IndraMotion MLD" is to be used as a motion system, control functions can be transferred to the PLC integrated in the drive (MLD).

- The drive takes over control functionality and commands the axis/axes, i.e., a sequence of motion commands is programmed as desired.
 - Single-axis "Motion Logic Control" ("IndraMotion MLD-S", MPB18VRS and above)
 - Multi-axis "Motion Logic Control" ("IndraMotion MLD-M", MPC18VRS and above)

Features Observe the following aspects when using "IndraMotion MLD" as a "stand-alone "Motion Logic Control"":

- The drive can work without a higher-level control unit (PLC).
- The integrated control commands the axis/axes and can set the cyclic command values, as well as the operation mode.
- Independent motion programs can be created.
- Apart from the standard function blocks, there are corresponding manufacturer function blocks available for programming. The manufacturer function blocks are compiled into libraries.

Using IndraMotion MLD

Freely programmable PLC in accordance with IEC 61131

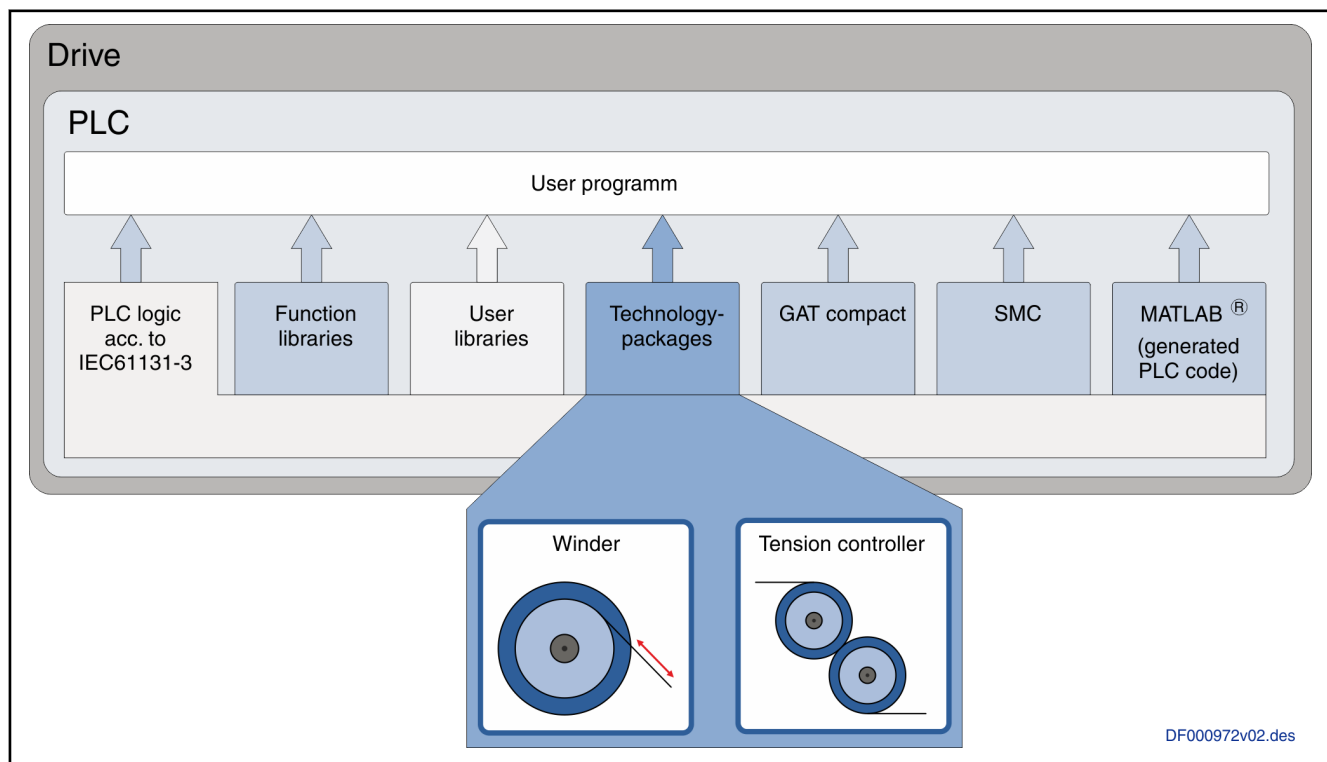


Fig. 1-4: Options for creating a user program and using "IndraMotion MLD"

Bosch Rexroth provides the following support when creating user programs:

- **Function libraries**
 - Basic IEC function blocks
 - PLCopen function blocks
 - AxisInterface
- **Technology libraries**

Bosch Rexroth develops sector- and application-based technology function blocks and provides them in the form of technology libraries (e.g., winders, tension controllers):

 - **Motion functions** (MX_TechMotion)
 - **Extended drive functions** (MX_TechWinder)
- **User libraries**

The user can also program function blocks or packages and compile them in their own user libraries.

The **user program** can consist of any combination of function libraries, technology packages and user libraries.

Using technology packages and other programming solutions

In addition to the option of freely programming "IndraMotion MLD", the integrated PLC can be used as a basic function when using ready-made, self-contained solutions (e.g., technology packages).

The following characteristics are available:

- **Technology package** (encapsulated technology function)
Compiled PLC project which is loaded to the drive as a self-contained "spj" file (parameter file). The technology function is operated like a "normal firmware function" via P-parameters and does not require any programming knowledge. There are the following types:
 - **Motion functions** (e.g., process controllers, winders, cross cutters, etc.)
 - Extended **drive functions** (e.g., event functions, programmable error reactions, quick stop, etc.)
 - Extended **control functions** (for example. "Process controller", "Vibration damping and avoiding", "Kinetic buffering")
- **Turnkey solutions** ("all-inclusive package")
Self-contained (ready-made) system solutions with system documentation that can be ordered individually. No programming knowledge required. Any existing dependencies are taken into account or avoided.



Encapsulated technology functions or "turnkey solutions" cannot use the extended PLC memory.



Technology packages have been programmed in IEC and can be loaded via the commissioning software as projects in the form of parameter sets or integrated in a project as function blocks from libraries.

During operation, only the runtime of the PLC is required as a platform for the technology functions and extended drive functions.

- **GAT compact sample project**
Ready-to-run programming frame in which the application code is entered at the predefined points. Professional and well-structured application programs with prepared elements using the dialog-based code generation of the GAT wizard are created. GATcompact is available from IndraWorks 14V14 with MPx20 and above (see "[Automated program generation with GAT compact](#)").
- **Sequential Motion Control (SMC)**
Sequential programming and functions for axis coupling. Preconfigured system solution for all sequential motion tasks; provides comprehensive functions for automatic and manual modes.

2 Important directions for use

2.1 Appropriate use

2.1.1 Introduction

Rexroth products are developed and manufactured to the state-of-the-art. The products are tested prior to delivery to ensure operational safety and reliability.

WARNING

Personal injury and property damage by using products incorrectly!

The products have been designed for use in an industrial environment and may only be used as intended. Failure to use them in the appropriate way may cause situations resulting in property damage and personal injury.



Rexroth as the manufacturer shall not honor any warranty, liability or compensatory claims for damages resulting from inappropriate use of the products. The user alone shall bear the risks of inappropriate use of the products.

Before using Rexroth products, make sure that all the prerequisites for an appropriate use of the products are satisfied:

- Personnel that in any way, shape or form uses our products must first read and understand the relevant safety instructions and be familiar with their appropriate use.
- Leave hardware products in their original state, i.e., do not make any structural modifications. It is not permitted to decompile software products or alter their source codes.
- Do not install damaged or faulty products or put them into operation.
- Make sure that the products have been installed as described in the relevant documentation.

2.1.2 Areas of use and application

Drive controllers by Rexroth are designed to control electric motors and monitor their operation.

Controlling and monitoring the Drive controllers may require additional sensors and actuators.



The drive controllers may only be used with the accessories and attachments specified in this documentation. Components that are not expressly mentioned may neither be attached nor connected. The same applies to cables and lines.

Operation is only allowed in the specified configurations and combinations of the components using the software and firmware as specified in the relevant functional descriptions.

Drive controllers have to be programmed before commissioning to ensure that the motor executes the functions specific to the application.

Drive controllers of the IndraDrive series have been developed for use in single- and multi-axis drive and control tasks.

Device types with different drive power and interfaces are available for using the Drive controllers in specific applications.

Typical applications include, for example:

- Handling and mounting systems
- Packaging and food machines
- Printing and paper converting machines
- Machine tools

Drive controllers may only be operated under the assembly and installation conditions specified in this documentation, in the specified position of normal use and under the specified ambient conditions (temperature, degree of protection, humidity, EMC, etc.).

2.2 Inappropriate use

"Inappropriate use" refers to using the Drive controllers outside of the operating conditions, technical data and specifications described in this documentation.

Drive controllers must not be used, if ...

- they are exposed to operating conditions that do not meet the specified ambient conditions. This includes, for example, operation under water, under extreme temperature fluctuations or extreme maximum temperatures.
- Furthermore, Drive controllers may not be used in applications that have not been expressly authorized by Rexroth. Therefore, please carefully follow the specifications outlined in the general safety instructions!



Components of the IndraDrive system are **products of Category C3** (with restricted distribution) in accordance with IEC 61800-3. This Category comprises EMC limit values for line-based and radiated noise emission. Compliance with this Category (limit values) requires the appropriate measures of interference suppression to be used in the drive system (e.g., mains filters, shielding measures).

These components are not provided for use in a public low-voltage mains supplying residential areas. If these components are used in such a mains, high-frequency interference is to be expected. This can require additional measures of interference suppression.

NOTICE

Danger of malfunction or control failure due to unauthorized access via a direct, unsecured Internet connection!

The Ethernet interface of the Drive controllers is not secured against unauthorized access. For this reason, only use the Drive controllers in secure networks (security).

3 Safety instructions for electric drives and controls

3.1 Definitions of terms

Application documentation	Application documentation comprises the entire documentation used to inform the user of the product about the use and safety-relevant features for configuring, integrating, installing, mounting, commissioning, operating, maintaining, repairing and decommissioning the product. The following terms are also used for this kind of documentation: Operating Instructions, Commissioning Manual, Instruction Manual, Project Planning Manual, Application Description, etc.
Component	A component is a combination of elements with a specified function, which are part of a piece of equipment, device or system. Components of the electric drive and control system are, for example, supply units, drive controllers, mains choke, mains filter, motors, cables, etc.
Control system	A control system comprises several interconnected control components placed on the market as a single functional unit.
Device	A device is a finished product with a defined function, intended for users and placed on the market as an individual piece of merchandise.
Electrical equipment	Electrical equipment encompasses all devices used to generate, convert, transmit, distribute or apply electrical energy, such as electric motors, transformers, switching devices, cables, lines, power-consuming devices, circuit board assemblies, plug-in units, control cabinets, etc.
Electric drive system	An electric drive system comprises all components from mains supply to motor shaft; this includes, for example, electric motor(s), motor encoder(s), supply units and drive controllers, as well as auxiliary and additional components, such as mains filter, mains choke and the corresponding lines and cables.
Installation	An installation consists of several devices or systems interconnected for a defined purpose and on a defined site which, however, are not intended to be placed on the market as a single functional unit.
Machine	A machine is the entirety of interconnected parts or units at least one of which is movable. Thus, a machine consists of the appropriate machine drive elements, as well as control and power circuits, which have been assembled for a specific application. A machine is, for example, intended for processing, treatment, movement or packaging of a material. The term "machine" also covers a combination of machines which are arranged and controlled in such a way that they function as a unified whole.
Manufacturer	The manufacturer is an individual or legal entity bearing responsibility for the design and manufacture of a product which is placed on the market in the individual's or legal entity's name. The manufacturer can use finished products, finished parts or finished elements, or contract out work to subcontractors. However, the manufacturer must always have overall control and possess the required authority to take responsibility for the product.
Product	Examples of a product: Device, component, part, system, software, firmware, among other things.
Project Planning Manual	A Project Planning Manual is part of the application documentation used to support the sizing and planning of systems, machines or installations.
Qualified persons	In terms of this application documentation, qualified persons are those persons who are familiar with the installation, mounting, commissioning and operation of the components of the electric drive and control system, as well as with the hazards this implies, and who possess the qualifications their work

requires. To comply with these qualifications, it is necessary, among other things,

- to be trained, instructed or authorized to switch electric circuits and devices safely on and off, to ground them and to mark them.
- to be trained or instructed to maintain and use adequate safety equipment.
- to attend a course of instruction in first aid.

Qualified personnel for handling functionally safe products

Individuals configuring, commissioning and operating functionally safe products must have the knowledge specified under "[Qualified persons](#)". Additionally, these individuals must be familiar with technical safety concepts as well as prevailing standards and regulations in the field of functional safety.

User

A user is a person installing, commissioning or using a product which has been placed on the market.

3.2 General information

3.2.1 Using the Safety instructions and passing them on to others

Do not attempt to install and operate the components of the electric drive and control system without first reading all documentation provided with the product. Read and understand these safety instructions and all user documentation prior to working with these components. If you do not have the user documentation for the components, contact your responsible Rexroth sales partner. Ask for these documents to be sent immediately to the person or persons responsible for the safe operation of the components.

If the component is resold, rented and/or passed on to others in any other form, these safety instructions must be delivered with the component in the official language of the user's country.

Improper use of these components, failure to follow the safety instructions in this document or tampering with the product, including disabling of safety devices, could result in property damage, injury, electric shock or even death.

3.2.2 Requirements for safe use

Read the following instructions before initial commissioning of the components of the electric drive and control system in order to eliminate the risk of injury and/or property damage. You must follow these safety instructions.

- Rexroth is not liable for damages resulting from failure to observe the safety instructions.
- Read the operating, maintenance and safety instructions in your language before commissioning. If you find that you cannot completely understand the application documentation in the available language, please ask your supplier to clarify.
- Proper and correct transport, storage, mounting and installation, as well as care in operation and maintenance, are prerequisites for optimal and safe operation of the component.
- Only qualified persons may work with components of the electric drive and control system or within its proximity.
- Only use accessories and spare parts approved by Rexroth.
- Follow the safety regulations and requirements of the country in which the components of the electric drive and control system are operated.

- Only use the components of the electric drive and control system in the manner that is defined as appropriate. See chapter "Appropriate Use".
- The ambient and operating conditions given in the available application documentation must be observed.
- Applications for functional safety are only allowed if clearly and explicitly specified in the application documentation "Integrated Safety Technology". If this is not the case, they are excluded. Functional safety is a safety concept in which measures of risk reduction for personal safety depend on electrical, electronic or programmable control systems.
- The information given in the application documentation with regard to the use of the delivered components contains only examples of applications and suggestions.

The machine and installation manufacturers must

- make sure that the delivered components are suited for their individual application and check the information given in this application documentation with regard to the use of the components,
- make sure that their individual application complies with the applicable safety regulations and standards and carry out the required measures, modifications and complements.
- Commissioning of the delivered components is only allowed once it is sure that the machine or installation in which the components are installed complies with the national regulations, safety specifications and standards of the application.
- Operation is only allowed if the national EMC regulations for the application are met.
- The instructions for installation in accordance with EMC requirements can be found in the section on EMC in the respective application documentation.

The machine or installation manufacturer is responsible for compliance with the limit values as prescribed in the national regulations.

- The technical data, connection and installation conditions of the components are specified in the respective application documentations and must be followed at all times.

National regulations which the user has to comply with

- European countries: In accordance with European EN standards
- United States of America (USA):
 - National Electrical Code (NEC)
 - National Electrical Manufacturers Association (NEMA), as well as local engineering regulations
 - Regulations of the National Fire Protection Association (NFPA)
- Canada: Canadian Standards Association (CSA)
- Other countries:
 - International Organization for Standardization (ISO)
 - International Electrotechnical Commission (IEC)

3.2.3 Hazards by improper use

- High electrical voltage and high working current! Danger to life or serious injury by electric shock!

- High electrical voltage by incorrect connection! Danger to life or injury by electric shock!
- Dangerous movements! Danger to life, serious injury or property damage by unintended motor movements!
- Health hazard for persons with heart pacemakers, metal implants and hearing aids in proximity to electric drive systems!
- Risk of burns by hot housing surfaces!
- Risk of injury by improper handling! Injury by crushing, shearing, cutting, hitting!
- Risk of injury by improper handling of batteries!
- Risk of injury by improper handling of pressurized lines!

3.3 Instructions with regard to specific dangers

3.3.1 Protection against contact with electrical parts and housings



This section concerns components of the electric drive and control system with voltages of **more than 50 volts**.

Contact with parts conducting voltages above 50 volts can cause personal danger and electric shock. When operating components of the electric drive and control system, it is unavoidable that some parts of these components conduct dangerous voltage.

High electrical voltage! Danger to life, risk of injury by electric shock or serious injury!

- Only qualified persons are allowed to operate, maintain and/or repair the components of the electric drive and control system.
- Follow the general installation and safety regulations when working on power installations.
- Before switching on, the equipment grounding conductor must have been permanently connected to all electric components in accordance with the connection diagram.
- Even for brief measurements or tests, operation is only allowed if the equipment grounding conductor has been permanently connected to the points of the components provided for this purpose.
- Before accessing electrical parts with voltage potentials higher than 50 V, you must disconnect electric components from the mains or from the power supply unit. Secure the electric component from reconnection.
- With electric components, observe the following aspects:
 Always wait **30 minutes** after switching off power to allow live capacitors to discharge before accessing an electric component. Measure the electrical voltage of live parts before beginning to work to make sure that the equipment is safe to touch.
- Install the covers and guards provided for this purpose before switching on.
- Never touch any electrical connection points of the components while power is turned on.

- Do not remove or plug in connectors when the component has been powered.
- Under specific conditions, electric drive systems can be operated at mains protected by residual-current-operated circuit-breakers sensitive to universal current (RCDs/RCMs).
- Secure built-in devices from penetrating foreign objects and water, as well as from direct contact, by providing an external housing, for example a control cabinet.

High housing voltage and high leakage current! Danger to life, risk of injury by electric shock!

- Before switching on and before commissioning, ground or connect the components of the electric drive and control system to the equipment grounding conductor at the grounding points.
- Connect the equipment grounding conductor of the components of the electric drive and control system permanently to the main power supply at all times. The leakage current is greater than 3.5 mA.
- Establish an equipment grounding connection with a minimum cross section according to the table below. With an outer conductor cross section smaller than 10 mm² (8 AWG), the alternative connection of two equipment grounding conductors is allowed, each having the same cross section as the outer conductors.

Cross section outer conductor	Minimum cross section equipment grounding conductor Leakage current ≥ 3.5 mA	
	1 equipment grounding conductor	2 equipment grounding conductors
1.5 mm ² (16 AWG)	10 mm ² (8 AWG)	2 × 1.5 mm ² (16 AWG)
2.5 mm ² (14 AWG)		2 × 2.5 mm ² (14 AWG)
4 mm ² (12 AWG)		2 × 4 mm ² (12 AWG)
6 mm ² (10 AWG)		2 × 6 mm ² (10 AWG)
10 mm ² (8 AWG)		-
16 mm ² (6 AWG)	16 mm ² (6 AWG)	-
25 mm ² (4 AWG)		-
35 mm ² (2 AWG)		-
50 mm ² (1/0 AWG)	25 mm ² (4 AWG)	-
70 mm ² (2/0 AWG)	35 mm ² (2 AWG)	-
...	...	...

Tab. 3-1: Minimum cross section of the equipment grounding connection

3.3.2 Protective extra-low voltage as protection against electric shock

Protective extra-low voltage is used to allow connecting devices with basic insulation to extra-low voltage circuits.

On components of an electric drive and control system provided by Rexroth, all connections and terminals with voltages up to 50 volts are PELV ("Protec-

tive Extra-Low Voltage") systems. It is allowed to connect devices equipped with basic insulation (such as programming devices, PCs, notebooks, display units) to these connections.

Danger to life, risk of injury by electric shock! High electrical voltage by incorrect connection!

If extra-low voltage circuits of devices containing voltages and circuits of more than 50 volts (e.g., the mains connection) are connected to Rexroth products, the connected extra-low voltage circuits must comply with the requirements for PELV ("Protective Extra-Low Voltage").

3.3.3 Protection against dangerous movements

Dangerous movements can be caused by faulty control of connected motors. Some common examples are:

- Improper or wrong wiring or cable connection
- Operator errors
- Wrong input of parameters before commissioning
- Malfunction of sensors and encoders
- Defective components
- Software or firmware errors

These errors can occur immediately after equipment is switched on or even after an unspecified time of trouble-free operation.

The monitoring functions in the components of the electric drive and control system will normally be sufficient to avoid malfunction in the connected drives. Regarding personal safety, especially the danger of injury and/or property damage, this alone cannot be relied upon to ensure complete safety. Until the integrated monitoring functions become effective, it must be assumed in any case that faulty drive movements will occur. The extent of faulty drive movements depends upon the type of control and the state of operation.

Dangerous movements! Danger to life, risk of injury, serious injury or property damage!

A **risk assessment** must be prepared for the installation or machine, with its specific conditions, in which the components of the electric drive and control system are installed.

As a result of the risk assessment, the user must provide for monitoring functions and higher-level measures on the installation side for personal safety. The safety regulations applicable to the installation or machine must be taken into consideration. Unintended machine movements or other malfunctions are possible if safety devices are disabled, bypassed or not activated.

To avoid accidents, injury and/or property damage:

- Keep free and clear of the machine's range of motion and moving machine parts. Prevent personnel from accidentally entering the machine's range of motion by using, for example:
 - Safety fences
 - Safety guards
 - Protective coverings
 - Light barriers

- Make sure the safety fences and protective coverings are strong enough to resist maximum possible kinetic energy.
- Mount emergency stopping switches in the immediate reach of the operator. Before commissioning, verify that the emergency stopping equipment works. Do not operate the machine if the emergency stopping switch is not working.
- Prevent unintended start-up. Isolate the drive power connection by means of OFF switches/OFF buttons or use a safe starting lockout.
- Make sure that the drives are brought to safe standstill before accessing or entering the danger zone.
- Additionally secure vertical axes against falling or dropping after switching off the motor power by, for example,
 - mechanically securing the vertical axes,
 - adding an external braking/arrester/clamping mechanism or
 - ensuring sufficient counterbalancing of the vertical axes.
- The standard equipment **motor holding brake** or an external holding brake controlled by the drive controller is **not sufficient to guarantee personal safety!**
- Disconnect electrical power to the components of the electric drive and control system using the master switch and secure them from reconnection ("lock out") for:
 - Maintenance and repair work
 - Cleaning of equipment
 - Long periods of discontinued equipment use
- Prevent the operation of high-frequency, remote control and radio equipment near components of the electric drive and control system and their supply leads. If the use of these devices cannot be avoided, check the machine or installation, at initial commissioning of the electric drive and control system, for possible malfunctions when operating such high-frequency, remote control and radio equipment in its possible positions of normal use. It might possibly be necessary to perform a special electromagnetic compatibility (EMC) test.

3.3.4 Protection against electromagnetic and magnetic fields during operation and mounting

Electromagnetic and magnetic fields!

Health hazard for persons with active implantable medical devices (AIMD) such as pacemakers or passive metallic implants.

- Hazards for the above-mentioned groups of persons by electromagnetic and magnetic fields in the immediate vicinity of drive controllers and the associated current-carrying conductors.
- Entering these areas can pose an increased risk to the above-mentioned groups of persons. They should seek advice from their physician.
- If overcome by possible effects on above-mentioned persons during operation of drive controllers and accessories, remove the exposed persons from the vicinity of conductors and devices.

3.3.5 Protection against contact with hot parts

Hot surfaces of components of the electric drive and control system. Risk of burns!

- Do not touch hot surfaces of, for example, braking resistors, heat sinks, supply units and drive controllers, motors, windings and laminated cores!
- According to the operating conditions, temperatures of the surfaces can be **higher than 60 °C (140 °F)** during or after operation.
- Before touching motors after having switched them off, let them cool down for a sufficient period of time. Cooling down can require **up to 140 minutes!** The time required for cooling down is approximately five times the thermal time constant specified in the technical data.
- After switching chokes, supply units and drive controllers off, wait **15 minutes** to allow them to cool down before touching them.
- Wear safety gloves or do not work at hot surfaces.
- For certain applications, and in accordance with the respective safety regulations, the manufacturer of the machine or installation must take measures to avoid injuries caused by burns in the final application. These measures can be, for example: Warnings at the machine or installation, guards (shieldings or barriers) or safety instructions in the application documentation.

3.3.6 Protection during handling and mounting

Risk of injury by improper handling! Injury by crushing, shearing, cutting, hitting!

- Observe the relevant statutory regulations of accident prevention.
- Use suitable equipment for mounting and transport.
- Avoid jamming and crushing by appropriate measures.
- Always use suitable tools. Use special tools if specified.
- Use lifting equipment and tools in the correct manner.
- Use suitable protective equipment (hard hat, safety goggles, safety shoes, safety gloves, for example).
- Do not stand under hanging loads.
- Immediately clean up any spilled liquids from the floor due to the risk of falling!

3.3.7 Battery safety

Batteries consist of active chemicals in a solid housing. Therefore, improper handling can cause injury or property damage.

Risk of injury by improper handling!

- Do not attempt to reactivate low batteries by heating or other methods (risk of explosion and cauterization).
- Do not attempt to recharge the batteries as this may cause leakage or explosion.
- Do not throw batteries into open flames.
- Do not dismantle batteries.

- When replacing the battery/batteries, do not damage the electrical parts installed in the devices.
- Only use the battery types specified for the product.



Environmental protection and disposal! The batteries contained in the product are considered dangerous goods during land, air, and sea transport (risk of explosion) in the sense of the legal regulations. Dispose of used batteries separately from other waste. Observe the national regulations of your country.

3.3.8 Protection against pressurized systems

According to the information given in the Project Planning Manuals, motors and components cooled with liquids and compressed air can be partially supplied with externally fed, pressurized media, such as compressed air, hydraulics oil, cooling liquids and cooling lubricants. Improper handling of the connected supply systems, supply lines or connections can cause injuries or property damage.

Risk of injury by improper handling of pressurized lines!

- Do not attempt to disconnect, open or cut pressurized lines (risk of explosion).
- Observe the respective manufacturer's operating instructions.
- Before dismounting lines, relieve pressure and empty medium.
- Use suitable protective equipment (safety goggles, safety shoes, safety gloves, for example).
- Immediately clean up any spilled liquids from the floor due to the risk of falling!



Environmental protection and disposal! The agents (e.g., fluids) used to operate the product might not be environmentally friendly. Dispose of agents harmful to the environment separately from other waste. Observe the national regulations of your country.

3.4 Explanation of signal words and the Safety alert symbol

The Safety Instructions in the available application documentation contain specific signal words (DANGER, WARNING, CAUTION or NOTICE) and, where required, a safety alert symbol (in accordance with ANSI Z535.6-2011).

The signal word is meant to draw the reader's attention to the safety instruction and identifies the hazard severity.

The safety alert symbol (a triangle with an exclamation point), which precedes the signal words DANGER, WARNING and CAUTION, is used to alert the reader to personal injury hazards.

DANGER

In case of non-compliance with this safety instruction, death or serious injury **will** occur.

WARNING

In case of non-compliance with this safety instruction, death or serious injury **could** occur.

CAUTION

In case of non-compliance with this safety instruction, minor or moderate injury could occur.

NOTICE

In case of non-compliance with this safety instruction, property damage could occur.

4 Basic functions of Rexroth IndraMotion MLD

4.1 Overview

The runtime system of the drive-integrated PLC in its MLD-S and MLD-M designs is embedded in the drive runtime system as a module with its functions and interfaces.

The figure below illustrates the basic functions of a drive with integrated PLC and how they interact (MLD-S design in this case):

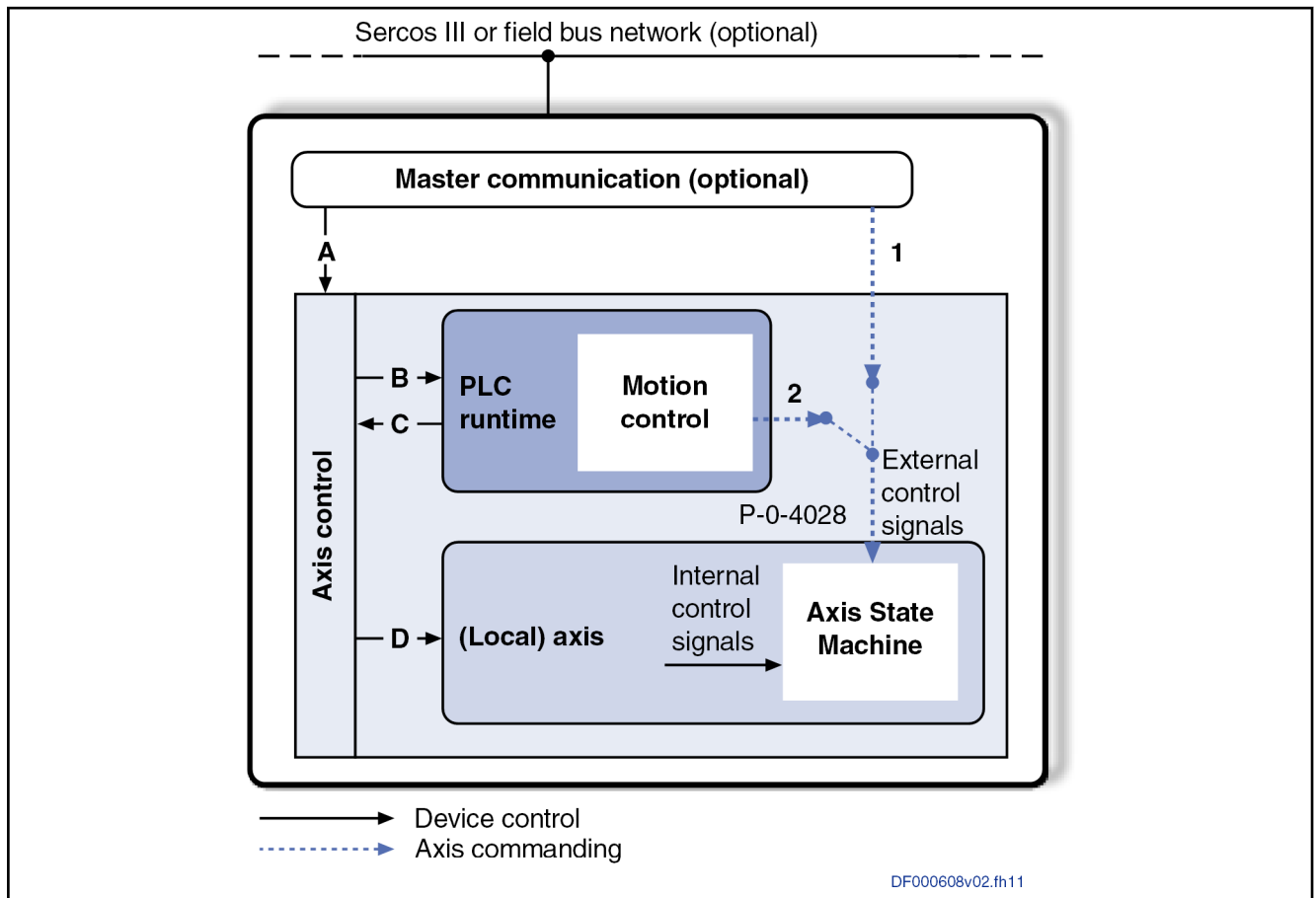


Fig. 4-1: Overview of device control and axis control, MLD-S

4.2 Explanation of terms

MLD-S, MLD-M, MLD The "Rexroth IndraMotion MLD" drive-integrated PLC is available in different variants.

The variant for single-axis applications is called "MLD-S". In this case, the control options of the PLC in the device are limited to the "local axis" and the virtual master axis generator.

If drives with Sercos III slave interface are connected via the "CC" option [Sercos III master (cross communication)], the PLC can control these axes. This variant is called "MLD-M".

The general short form "MLD" is used when describing independent functionalities.

Motion control	Motion control is the umbrella term for position, velocity or acceleration control or a combination thereof. • Technology function blocks Function blocks for extending the drive functionality • PLCopen function blocks Motion control via the PLC program is carried out using PLCopen-based function blocks. For this purpose, PLCopen has defined a number of IEC 61131 function blocks that are used to control the axes. Apart from the function blocks already defined by PLCopen, there are additional function blocks based on the standard. • AxisInterface The AxisInterface pools and extends PLCopen motion function blocks and provides an easy-to-operate interface for the drive functionality. The axis interface contains control signals and parameters for the various operation modes of the master axis and slave axis, as well as options for setting selected process values. The AxisInterface consists of the PLC function blocks running in the user program. • GAT compact GAT compact is a sample project based on IEC 61131-3. It is amongst others made available by the IndraWorks Engineering framework for IndraMotion MLD, and a wizard is used for loading and commissioning it.
CCD	"CCD" is the abbreviation of C ross C ommunication D rives, the interface for cross communication based on Sercos III. Devices in the "Rexroth IndraDrive" range can be configured with the "CCD" option to allow electronic (digital) coupling of drives and I/O devices.
CCD master / CCD slave	The "CCD master" is a drive with "CCD" option (Sercos III master interface) which acts like an external control unit for the "CCD slaves" [drive with "CCD" option (Sercos III slave interface)] of a CCD group.
Local axis	The "local axis" for MLD-S is the axis of the drive, and for MLD-M the axis in the drive with the "CCD" option (Sercos III master interface).
Remote axes	For MLD-M, "remote axes" are the axes in the CCD slaves.
Virtual master axis	The drive contains a master axis generator. It can be controlled like a real axis and then simulates axis motion. The position of this master axis can be used as input for the local and remote axis. MLD can control this master axis via the motion function blocks.
AT	"AT" is the abbreviation of " A ntriebs t elegramm" (drive telegram). The drive telegram is transmitted from the slave to the master via the real-time data channel.
MDT	"MDT" is the abbreviation of M aster D ata T elegram. The master data telegram is transmitted from the master to the slave via the real-time data channel.
PII	"PII" is the abbreviation of " P rocess I nterface I mage". In the task cycle, the switch states are read at the inputs before the program code is called and stored in the PII. This information is then transmitted to the control program and processed.
POI	"POI" is the abbreviation of " P rocess O utput I mage". The states in the POI are transmitted to the physical outputs at the end of the task after the program code has been processed.
Run-up mode	For drives with Sercos III interface as master communication, the run-up mode allows the connection between communication phase and device state

to be decoupled. For drives without a master communication interface or for drives with a master communication interface other than the Sercos III interface, run-up mode defines the device state of the drive after boot-up (operating mode or parameter mode).

4.3 Device control

4.3.1 General information

The device control defines how the basic operating states of the primary system blocks interact.

4.3.2 Master communication

The drive (in this case with embedded MLD) can be linked to a master control unit via the master communication (Sercos slave interface or field bus slave interface). The bus-specific state machine controls the initialization of the communication system (until cyclic data exchange has been established, e.g., Sercos communication phases 0..4). This state machine is controlled by the bus master or the state of the bus itself.

A bus-compatible master communication interface is optional.

See also Functional description of firmware: "Communication phases of master communication"

4.3.3 Axis control

Axis control is the primary management framework for all drive functions, such as axis or integrated PLC. It comes with initialization, parameter mode and operating mode states.

With Sercos master communication, the axis control state machine is linked to this master communication, i.e., initializing the Sercos ring (switching to communication phase 4) automatically switches the axis control to the operating mode [see (A) in [chapter 4.1 "Overview" on page 41](#)]. With the Sercos III interface, the run-up mode allows the linking between communication phase and device state machine to be removed.

If master communication is not available or for all master communication interfaces except Sercos III, the axis can be controlled independently of the bus system. Run-up mode allows the device status after boot-up (operating mode or parameter mode) to be defined for the axis. In firmware version MPx17 and above, this is also possible for Sercos with the commands "S-0-0420, C0400 Activate parameterization level 1 procedure command" and "S-0-0422, C0200 Exit parameterization level procedure command".

See also Functional description of firmware "Device-internal state machine".

4.3.4 Axis

The (local) axis is responsible for generating or processing the axis command values and for drive control. Implementation is in accordance with Sercos specification. The primary external control information for commanding the axis is as follows:

- "Drive on"
- "Drive enable"
- "Drive Halt"
- Command operation mode input

These signals control the state machine of the axis.

The control information only takes effect when axis control is in the "operating mode" state. This means the axis control state machine superimposes the axis state machine [see (D) in [chapter 4.1 "Overview" on page 41](#)].



The axis cannot be commanded when axis control is in parameter mode. The axis then is also in the parameter mode!

4.3.5 PLC runtime

The runtime system of the drive-integrated PLC has the primary states "STOP" and "RUN". Switching from "STOP" to "RUN" during system initialization takes place depending on the device control [see (B) in [chapter 4.1 "Overview" on page 41](#)]. Several options can be set via "P-0-1367, PLC configuration" (bits 1, 0) or the IndraWorks dialog:

- "00" / "Default start behavior": The PLC starts upon axis control transition from initialization mode to parameter mode, provided it was in RUN when the drive was powered off. Otherwise it remains in STOP (default start behavior).
- "01" / "Start when booting": Same as "00", but independent of the last state (goes to RUN).
- "10" / "Stop": The PLC remains in the STOP state.
- "11" / "Start in operating mode": The PLC starts upon axis control transition from parameter mode to operating mode.

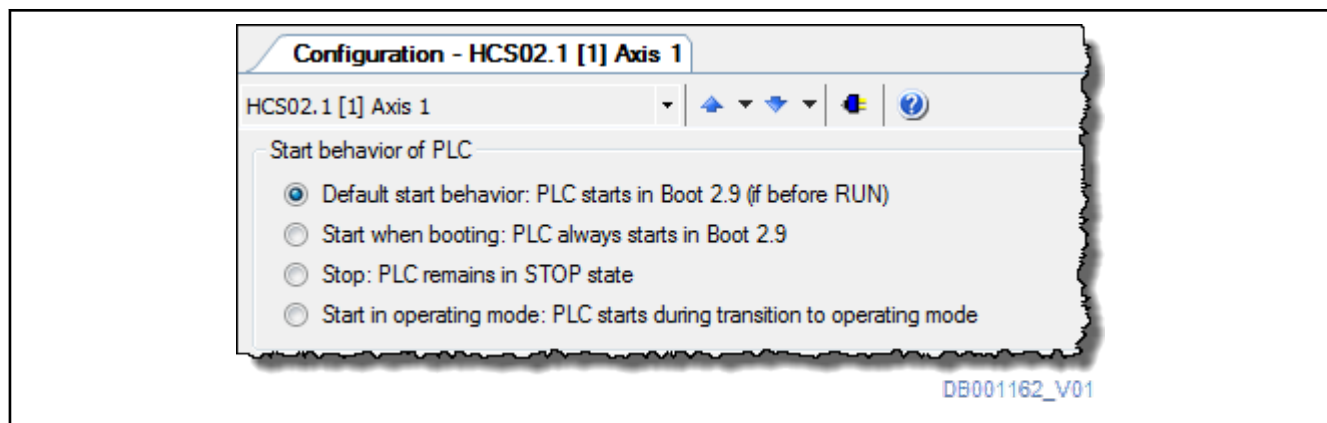


Fig. 4-2: Setting the start behavior of the PLC runtime

4.3.6 MLD options for influencing axis control

General information

The initialization state of the local axis control and with it the local axis itself can be influenced in the PLC user program by using the "MX_SetDevice-Mode" firmware function block [see (C) in [chapter "Device control with MLD-M" on page 44](#)]. This is necessary, e.g., when basic changes in configuration, i.e. parameter modifications, are to be carried out that can only be made in the parameter mode. This is also possible if Sercos is used for master communication.

Device control with MLD-M

The device control of the CCD group in the MLD-M design is based on the basic functions of MLD-S.

Please note the following additions to the MLD-S design (see also figure below):

- The Sercos III master interface is an additional system block in MLD-M (in "CCD master" device). This interface uses its state machine to define the CCD group (Sercos III network) and with it the run-up of the connected CCD slave devices.
- The coupling property between the state machines of the CCD group and the axis control in the CCD master can be configured (P-0-1800.0.1, bit 5). If they are not linked, the axis control parameter mode in the CCD master does not switch the CCD group to communication phase 2.

- In the linked mode (P-0-1800.0.1, bit 5="1"), the state machine of the Sercos III master interface is specified by the axis control in the CCD master (**E**), i.e., axis control parameter mode in the CCD master translates to communication phase 2 in the CCD group, or axis control operating mode in the CCD master translates to communication phase 4 in the CCD group.

In the linked mode, MLD-M can use the firmware function block "MX_SetDeviceMode" to control the state of axis control and with it the state of the CCD group. "MX_SetDeviceMode" can only take effect on the local axis (and with it, possibly, on the group as a whole). Calling "MX_SetDeviceMode" (local axis in CCD master) with the input "OperationMode"=FALSE switches the CCD master to parameter mode and the CCD group and CCD slaves connected to it to communication phase 2. As a result, the CCD slaves are also switched to parameter mode, i.e., the entire CCD group is set to parameter mode.

- In the unlinked mode (P-0-1800.0.1, bit 5="0"), the command "P-0-1802.0.1, C7400 CCD: Switching to phase 2" must be used to switch the CCD communication to phase 2 in order to write parameters in the CCD group.

Basic functions of Rexroth IndraMotion MLD

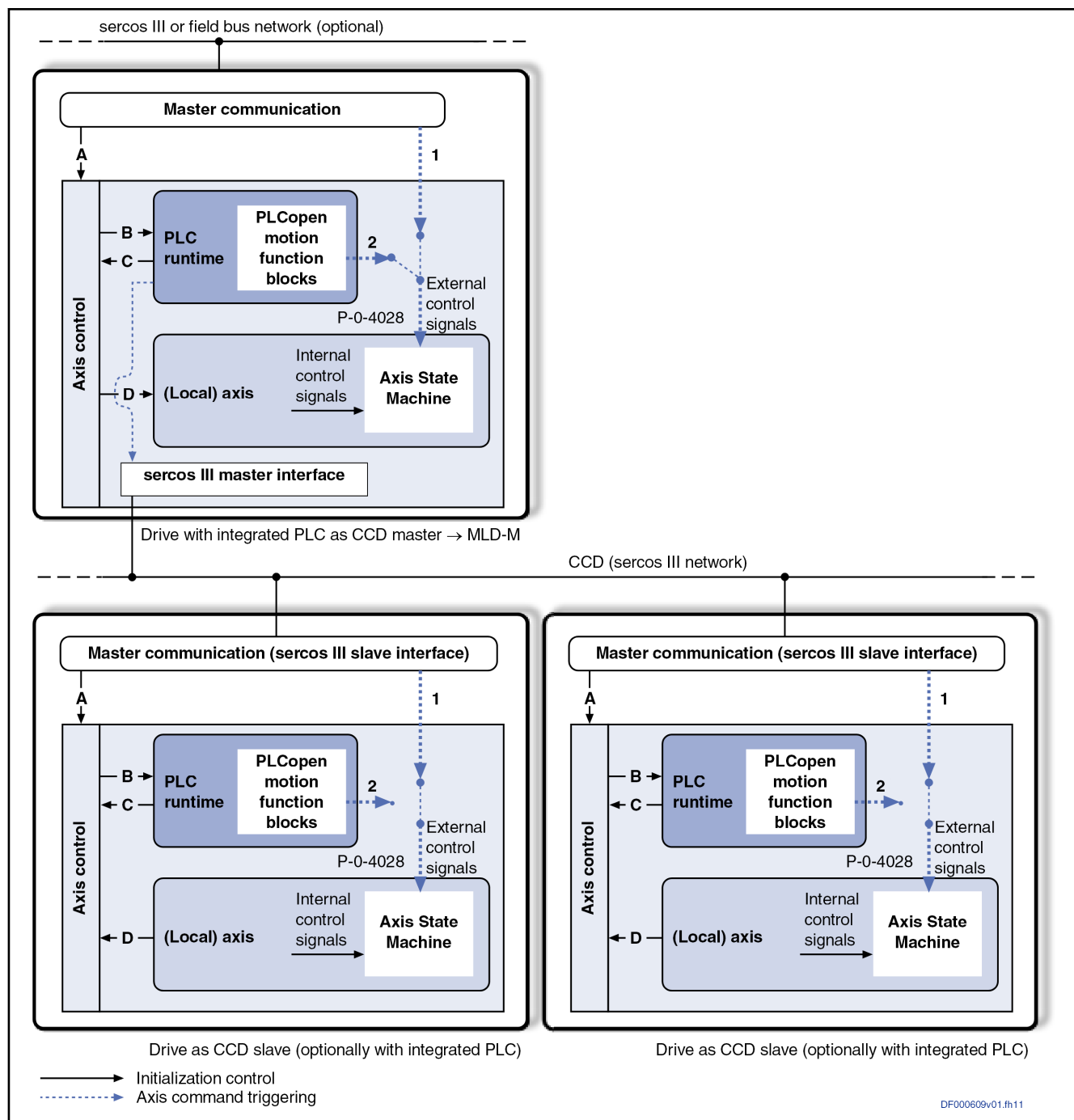


Fig. 4-3: Overview of MLD-M device control and axis commanding

MLD function block and system-wide variable for axis control

Function block:

- "MX_SetDeviceMode": Firmware function block for axis control, uses S-0-0420, S-0-0422, S-0-0424

System-wide variable:

- MX_AXISDATA AxisData[<Axis>].dwDriveStatus_i, bits 15,14: Displays the primary operating state of axis and axis control (see marginal note "Structure of AxisData")

Device control parameters and diagnostic messages

Pertinent parameters	• S-0-0128, C5200 Communication phase 4 transition check • S-0-0420, C0400 Activate parameterization level 1 procedure command • S-0-0422, C0200 Exit parameterization level procedure command • S-0-0423, IDN-list of invalid data for parameterization levels • S-0-0424, Status parameterization level • P-0-1350, PLC control word • P-0-1367, PLC configuration • P-0-1802.0.1, C7400 CCD: Switching to phase 2 • P-0-1802.0.2, C7500 CCD: Switching to phase 4 • P-0-4086, Master communication status
Pertinent diagnostic messages	• Diagnostic messages for Sercos master communication operating states: P-1, P0, .., P3 • Diagnostic message for axis control operating states: OM, PM • Diagnostic message for axis operating states: bb, Ab, AF, .. • Commands for device control / master communication state switching, axis control and axis: C0100, C0200, C0400, C5200, C7400, C7500 • Device control command errors: C01xx, C02xx, C04xx, C52xx, C74xx, C75xx

4.4 Axis commanding

4.4.1 Basics on axis control

General information

The operating state of the axis is defined by external and internal control signals (error, under power, etc.). The primary external control information for commanding the axis is as follows:

- "Drive on"
- "Drive enable"
- "Drive Halt"
- Command operation mode input

There are also drive control commands for activating complex, pre-configured commands, such as "drive-controlled homing procedure", "auto-adjustment functions", etc.

Depending on the application, the external control signals are preset by an external control unit via master communication or by the integrated PLC (MLD). The table below contains these applications with the necessary configurations:

Basic functions of Rexroth IndraMotion MLD

Use case	Axis control configuration in MLD-device ("P-0-1367, PLC configuration", bit 4)	Configuration (no MLD active) or axis control in CCD slaves ("P-0-1367, PLC configuration", bit 4)	Additional notes
MLD-S as "intelligent servo axis"	0: No permanent control	Not relevant	Only temporary axis control by MLD is possible
MLD-S as "stand-alone single-axis "Motion Logic Control""	1: Permanent control	Not relevant	--
MLD-M as "stand-alone multi-axis "Motion Logic Control""	1: Permanent control	(No MLD active) or 0: No permanent control	P-0-1800.0.1, CCD: Configuration, Bits 3, 4 = "01": MLD-M system mode required!

Tab. 4-1: Specifying the external control signals

Configuring MLD with IndraWorks MLD is configured via IndraWorks in the following dialog:

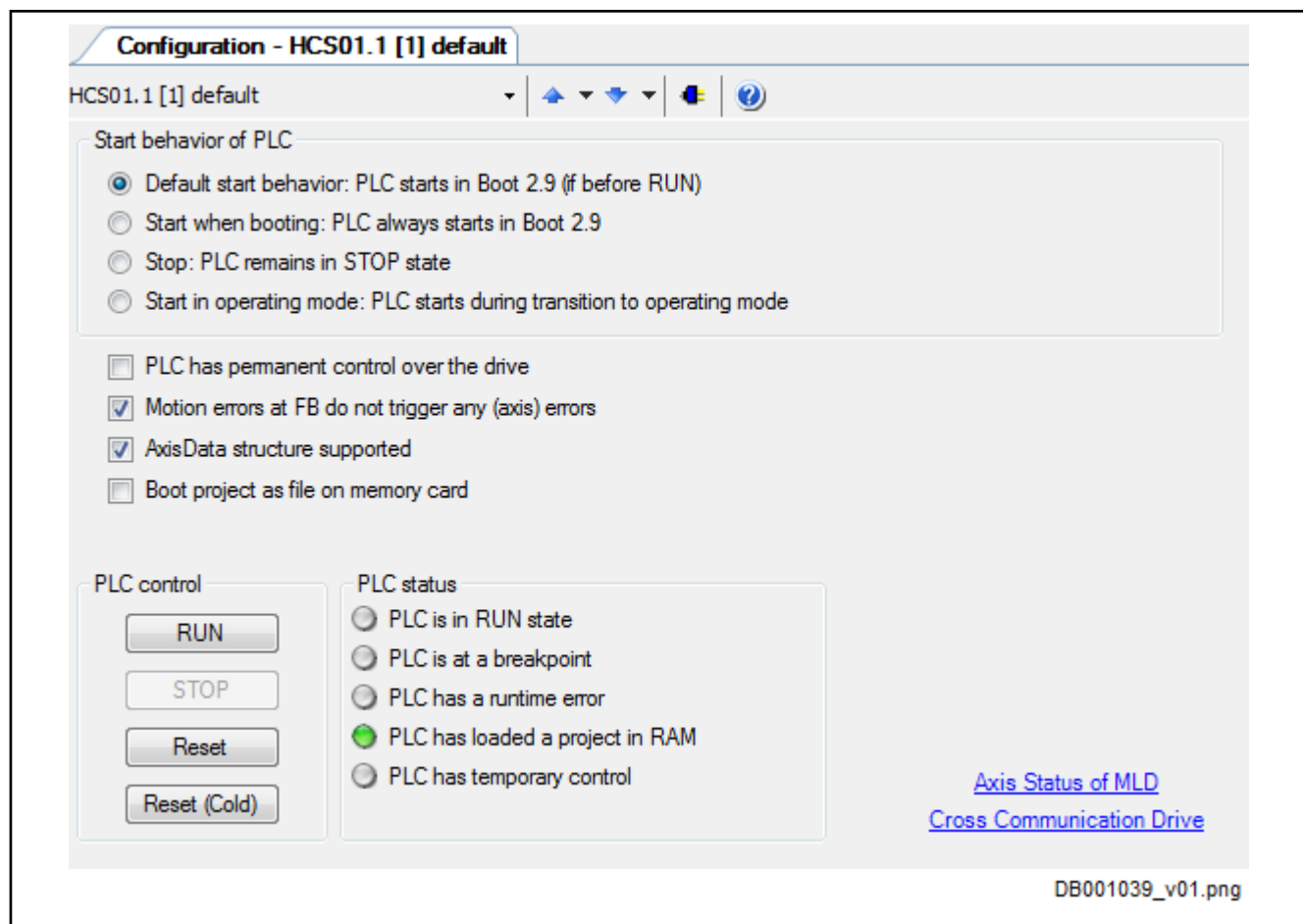


Fig. 4-4: Configuring MLD with IndraWorks

When MLD-M is used for multi-axis control, the MLD-M system mode is activated so that the connected axes follow the inputs of MLD-M.

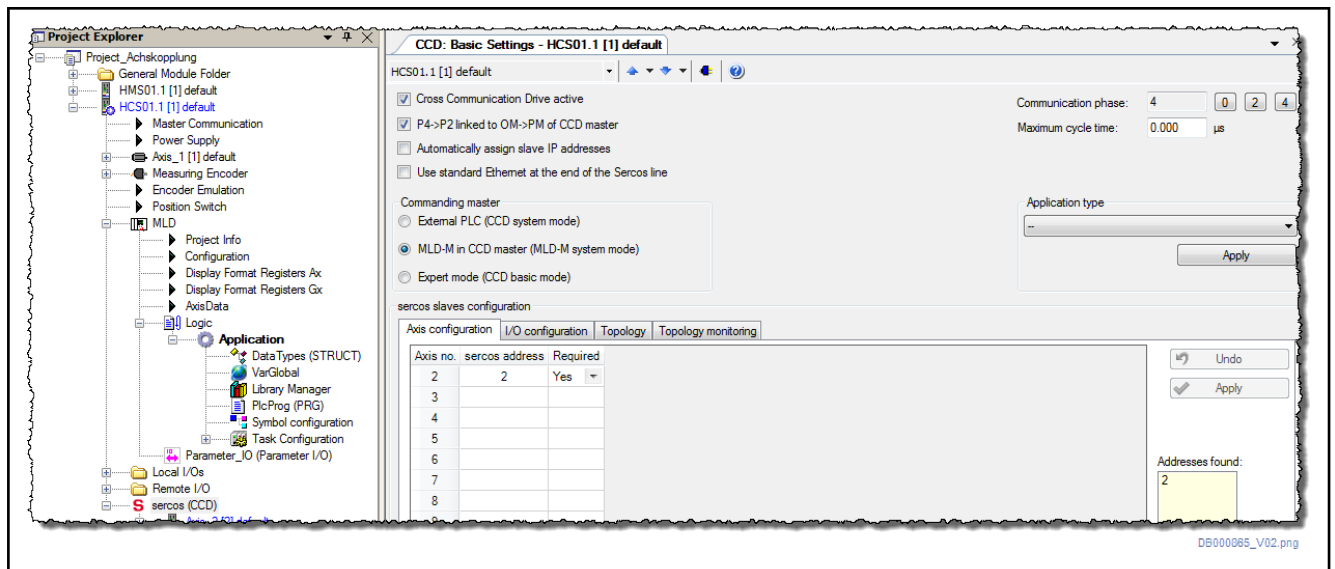


Fig. 4-5: Configuring MLD with IndraWorks, selecting CCD setting "MLD-M system mode"

MLD-S as "intelligent servo axis"

When MLD-S is used as an "intelligent servo axis", the external control unit has control over the (local) axis **(6)** by default. The external control signals are specified via the specific control word of each master communication interface.

If required, MLD can temporarily give itself axis control using the firmware function block "MX_SetControl" **(7)**. This can be necessary, e.g., for an intelligent, decentralized error reaction. This means MLD is able to temporarily control the axis via motion function blocks in conformity with PLCopen.

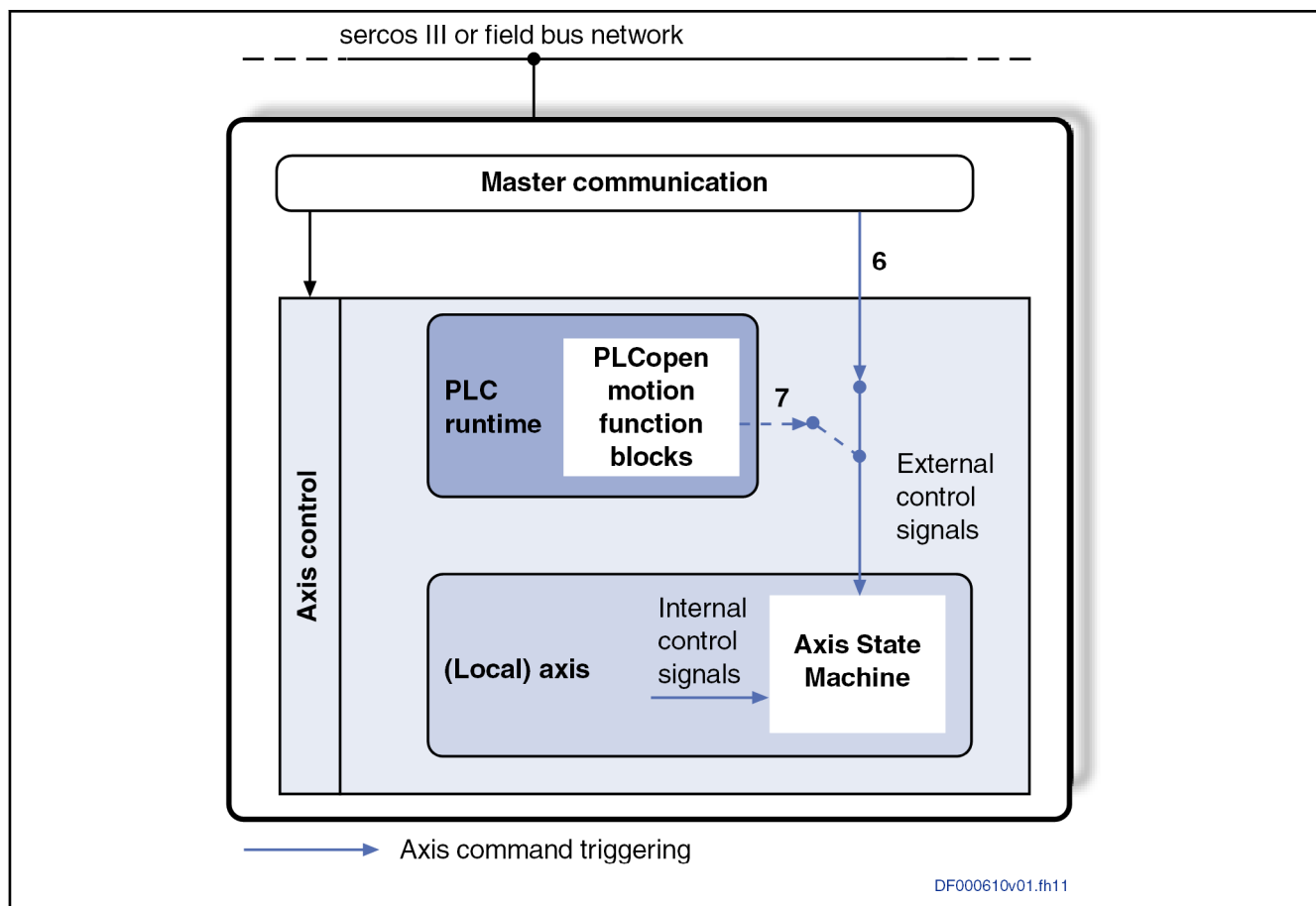


Fig. 4-6: Axis commanding, MLD-S as "intelligent servo axis"

MLD-S as "stand-alone single-axis "Motion Logic Control""

When MLD-S is used as "stand-alone single-axis "Motion Logic Control"", MLD has permanent axis control and thereby can completely replace external motion commanding (7). If available, master communication is usually only used to exchange process data with the drive or MLD, and a control word for commanding the axis is not transmitted. If, however, a control word has been applied, only the "drive enable" signal will take effect, provided it supports the bus-specific control word (only with Sercos).

The external control signals "Drive on", "Drive Halt" and the command operation mode input are indirectly controlled by the PLCopen motion function blocks called in the PLC user program.

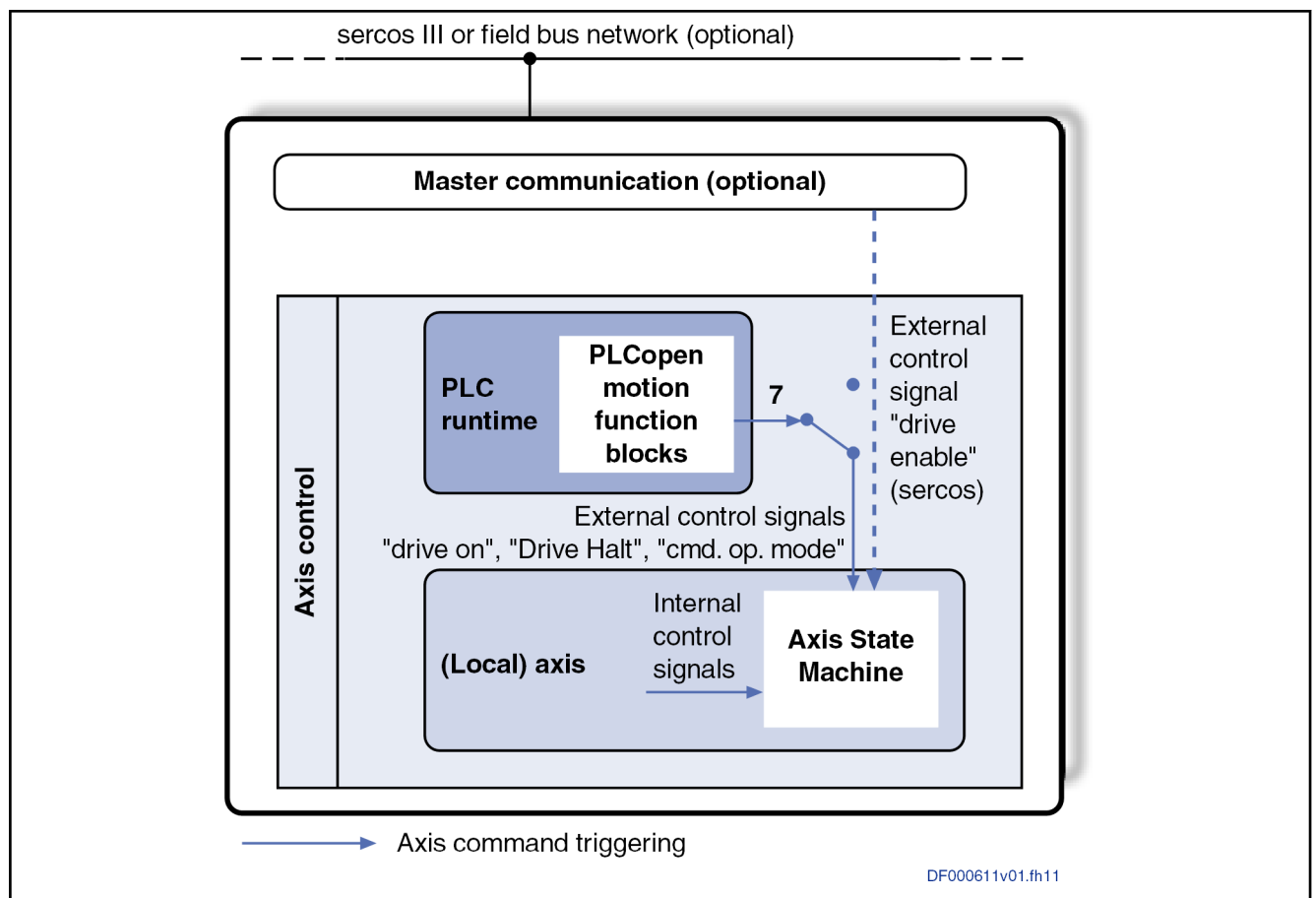


Fig. 4-7: Axis commanding, MLD-S as "stand-alone Motion Logic Control"

MLD-M as "stand-alone multi-axis Motion Logic Control"

When MLD-M is used as "stand-alone multi-axis Motion Logic Control", the integrated PLC in the CCD master (MLD-M) commands up to 9 remote axes in the CCD slaves in addition to the local axis. For these CCD slaves, MLD-M in the CCD master is an external control unit that controls their axes via Sercos III master communication.

This makes this structure a mixture of MLD-S as "intelligent servo axis" and MLD-S as "stand-alone single-axis Motion Logic Control". While the local axis in the CCD master is under the permanent control of MLD-M available in the same device, the programmable logic controllers integrated in the CCD slaves are only optional and, at most, are used for implementing an "intelligent servo axis". Commanding is effected by MLD-M in the CCD master by default.

Basic functions of Rexroth IndraMotion MLD

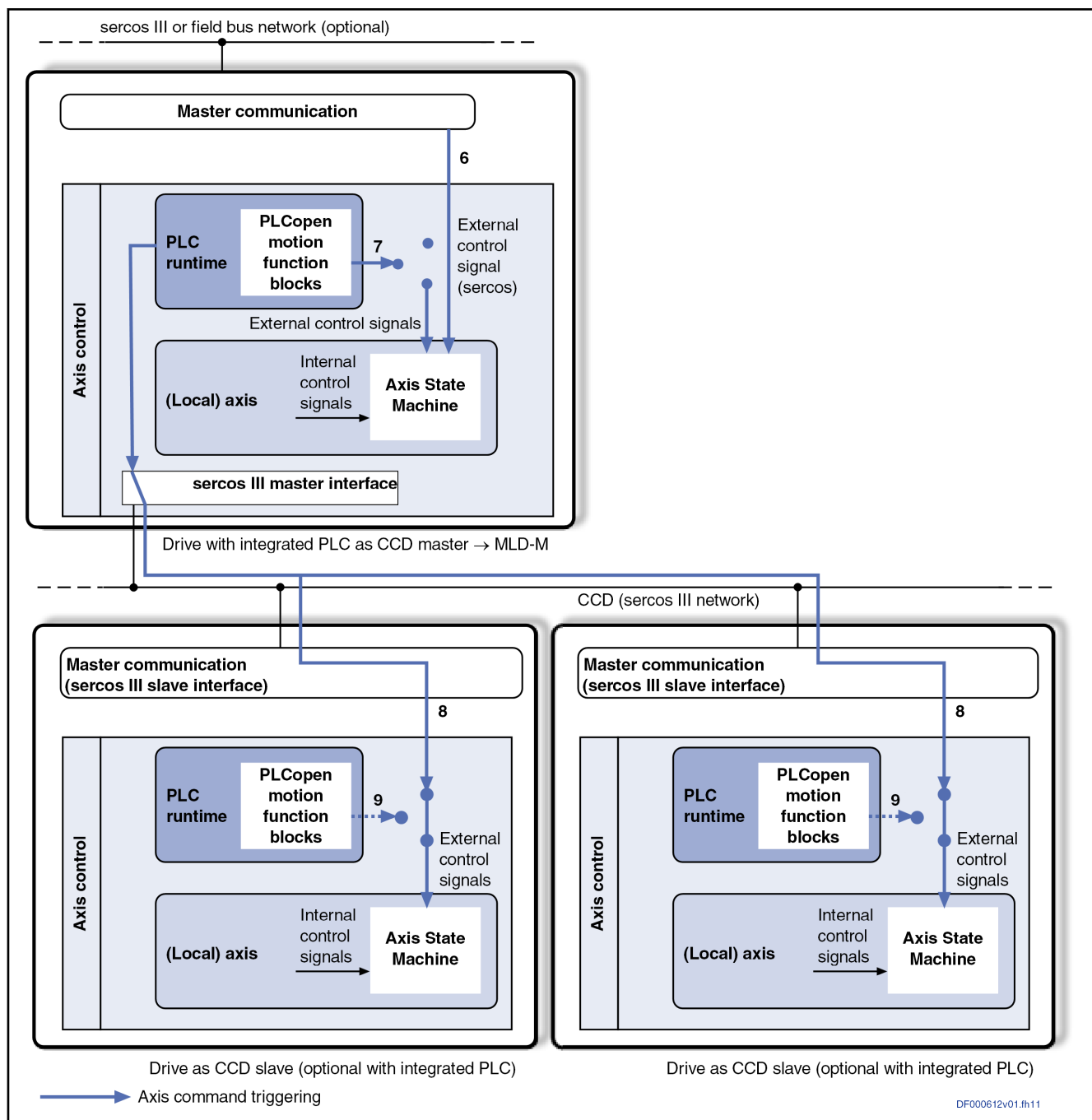


Fig. 4-8: Axis commanding, MLD-M as "stand-alone multi-axis "Motion Logic Control"

4.4.2 Implementing motion commanding in the axis

The axis is commanded by MLD by calling PLCopen-based motion function blocks in the PLC user program. By specifying control signals and command value parameters, the motion function blocks implement the activation of Sercos-based operation modes in the axis.

The applications are as follows:

Use case	Operation modes for implementing motion function blocks in conformity with PLCopen
MLD-S as "intelligent servo axis"	Uses internal operation modes "torque control", "velocity control" and "drive-controlled positioning" as independent instances of the equivalent Sercos modes with parameters P-0-1450 to P-0-1465 This ensures that with temporary control, the operation mode settings of the external control are not illegally manipulated.
MLD-S as "stand-alone single-axis "Motion Logic Control"" or MLD-M as "stand-alone multi-axis "Motion Logic Control""	Uses primary operation mode and secondary operation modes 1 through 7 (S-0-0032 to S-0-0035, S-0-284) with the preset operation modes "velocity control", "torque control", "position synchronization", "drive-controlled positioning" and "velocity synchronization", and with it the parameters used in these modes (for details, see " Motion command channel ")

4.4.3 Basic functional principle of motion function blocks in conformity with PLCopen

Timing

The timing diagram illustrates how the inputs and outputs work. "Done" is used as "Execute" of the next function block. In the second case, the signal is in addition connected logically.

Basic functions of Rexroth IndraMotion MLD

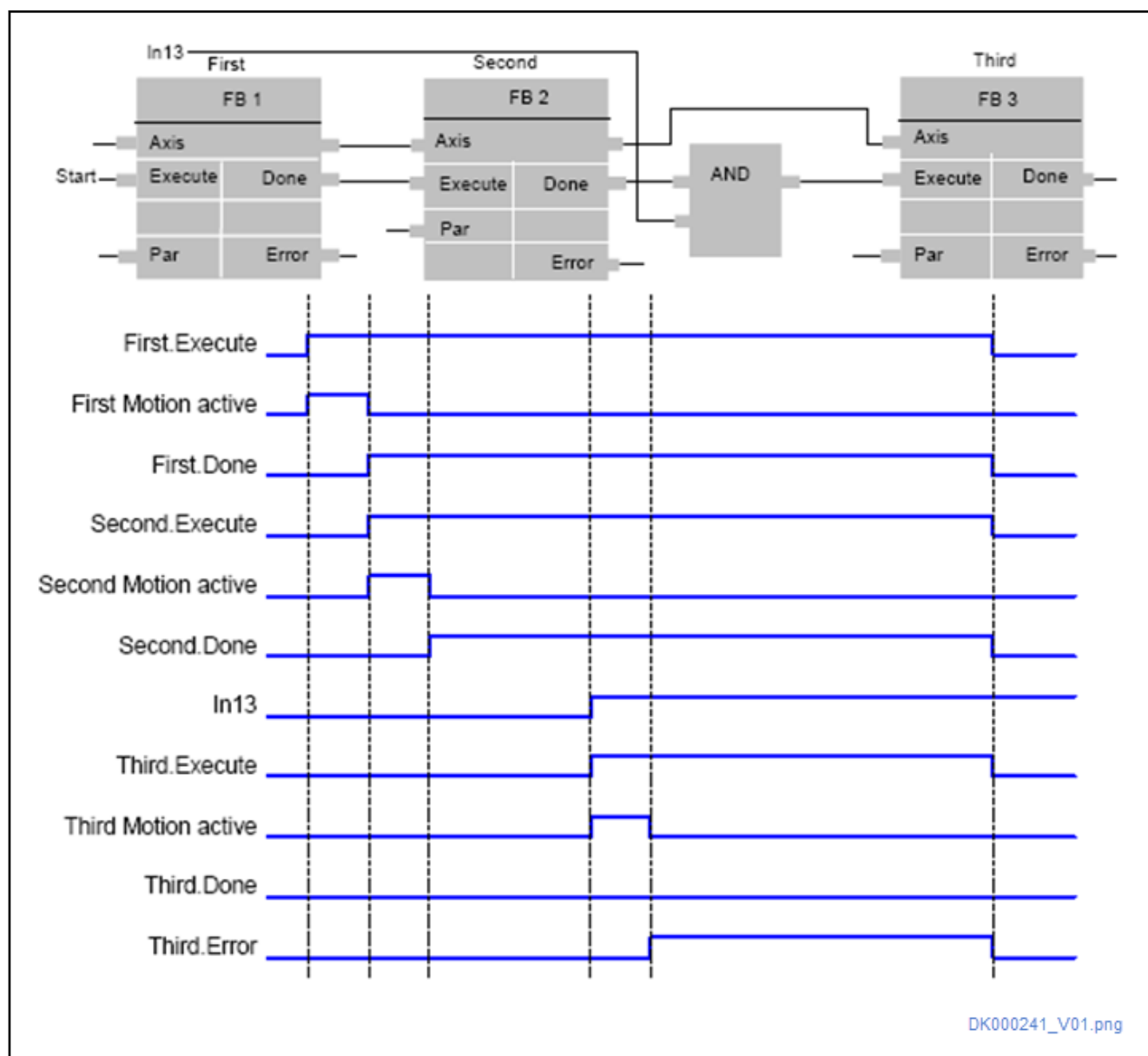


Fig. 4-9: Function block timing behavior (edge control)

Instantiation

"Execute" edges retrigger the function block with different input values in the same instance.

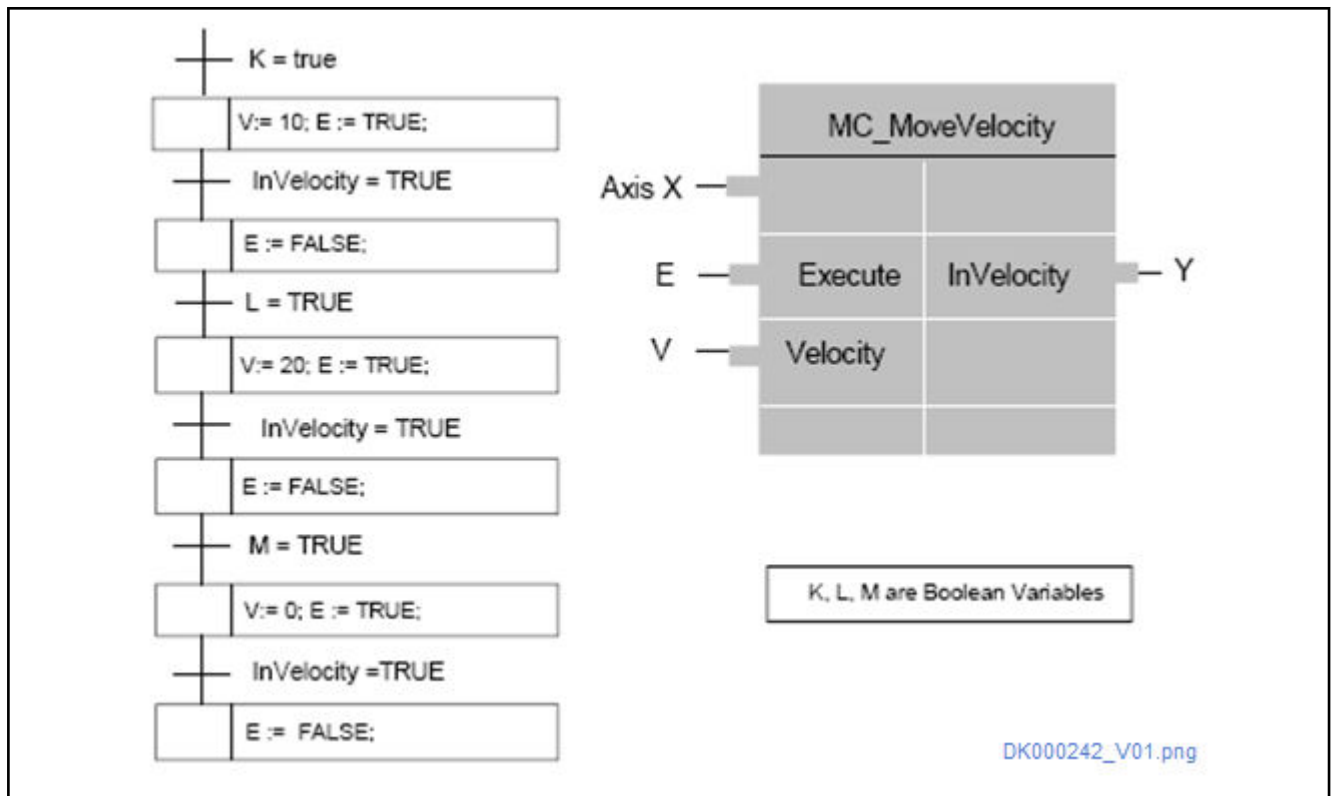


Fig. 4-10: Function block instantiation (same instance)

The figure below illustrates how different instances are instantiated. Each instance has its own status that is visible via the "InVelocity" outputs, etc.

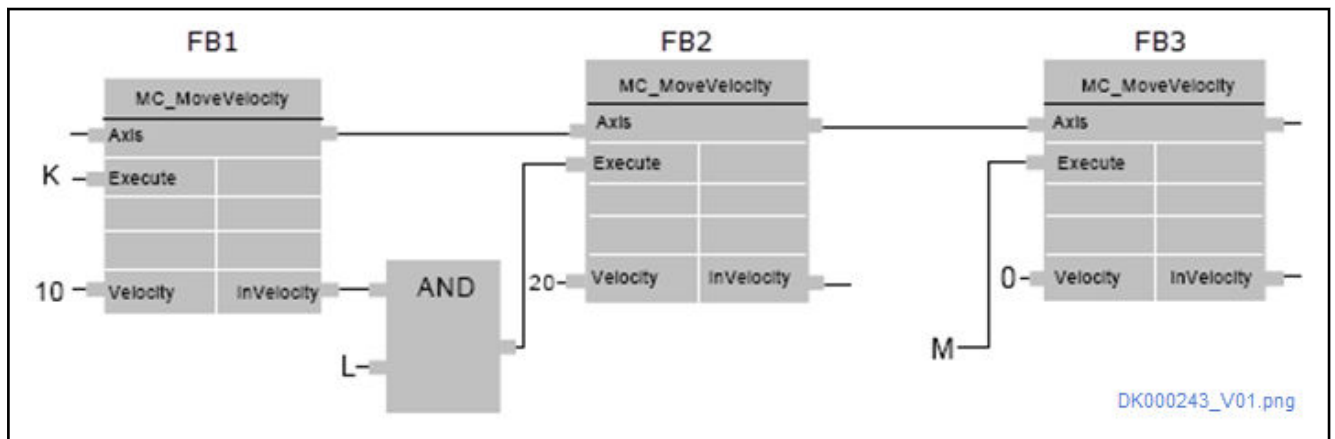


Fig. 4-11: Function block instantiation (different instances)

4.4.4 Axis addressing

The function blocks for motion commanding, parameter access and generating diagnostic messages are mostly axis-related functions. This is why the target axis has to be transmitted to the function block as information. For this purpose, PLCopen has defined an input/output called "Axis" ("AXIS_REF" data type). In addition to function blocks for discrete positioning and infinite travel, there are motion function blocks for synchronous running (synchronous motion, cam control, motion profile) of a slave axis with a master axis. In addition to the slave axis, which in this case is the target axis of these motion

function blocks, the master axis is transmitted to the function block as another axis reference.

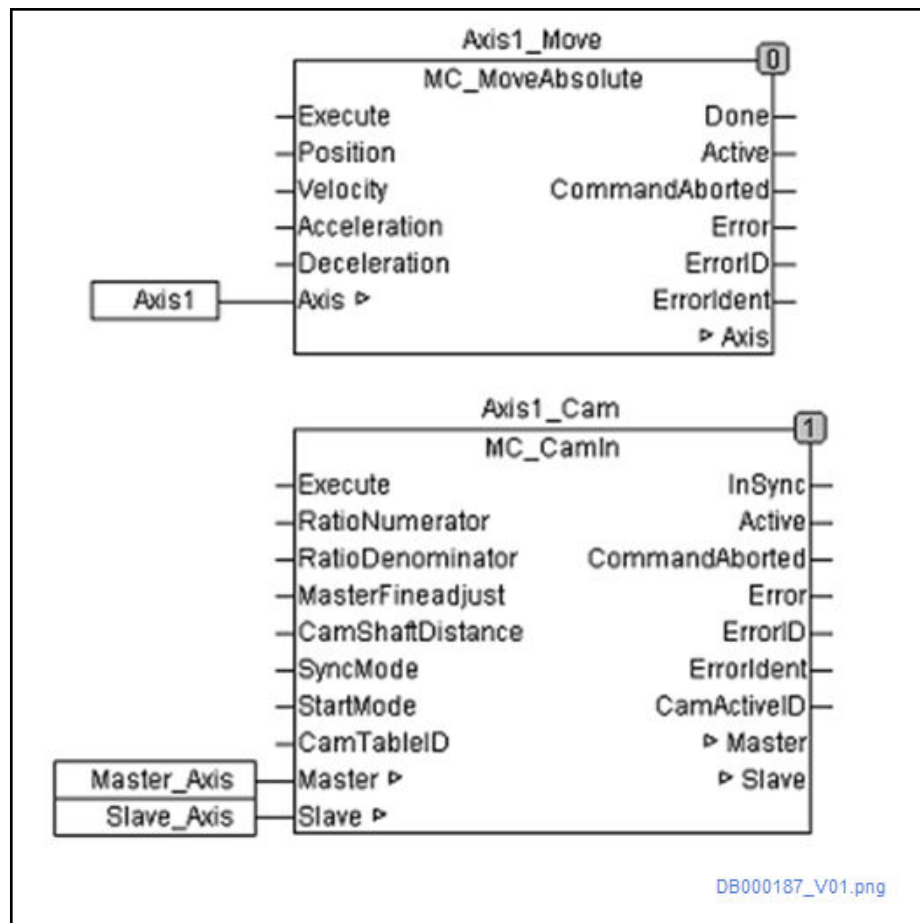


Fig. 4-12: Examples of axis reference at axis-related function blocks

According to PLCopen, variables of the structure data type "AXIS_REF" have to be used as "VAR_IN_OUT", which for all IndraMotion systems consists of the "control unit number" and "axis number" elements. The only control unit number that can be used is the constant `LOCAL_CNTRL := 0`. The corresponding constants for real axes, virtual master axes and real master axes are defined as axis numbers in the "MX_CommonTypes" library.

The MLD- system defines "AXIS_REF" type variables for all MLD-relevant axes. The variables can be directly used in the user program.

Real axes for motion commanding / parameter access / diagnostic message generation

Variable name	AXIS_REF Structure elements	Meaning	Available MLD-S / MLD-M	Commanding as: Target / slave axis	AxisData available
Axis1	CntrlNo := LOCAL_CNTRL AxisNo := AXIS_1	Local real axis of MLD-S or in CCD master of MLD-M Example: The real axis "Axis1" can be moved to an absolute position by means of the "MC_MoveAbsolute" block. "Axis1" must be transmitted at the "Axis" input of the "MC_MoveAbsolute" block.	X / X	X	X
Axis2	CntrlNo := LOCAL_CNTRL AxisNo := AXIS_2	Real axis in a CCD slave of MLD-M Example: In the MLD-M system mode, the PLC of the CCD master can command the CCD slave axes. For example, to move the "Axis2" axis in the Velocity control operating mode, the "MC_MoveVelocity" functional block must be called in the PLC of the CCD master and "Axis2" has to be transmitted at the "Axis" input.	-- / X	X	X
...	...	...	...	...	...
Axis10	CntrlNo := LO- CAL_CNTRL AxisNo := AXIS_10	Real axis in a CCD slave of MLD-M	-- / X	X	X

Axes for synchronous operating modes

Commanding the local virtual master axis generator

The local virtual master axis generator "VmAxis1" serves for generating a master axis position which can be used as input variable for the position synchronization modes and the "velocity synchronization" mode. The local virtual master axis generator is available once on every real axis.



The chapter "[Axis availability for functions and function blocks](#)" specifies for which functions and function blocks the local virtual master axis generator is available.

Variable name	AXIS_REF Structure elements	Meaning	Available MLD-S / M	AxisData available
VmAxis1	CntrlNo := LOCAL_CNTRL AxisNo := VMA_1	Axis for commanding the local virtual master axis generator	X / X	--

The local virtual master axis generator can be used with both, MLD-S and MLD-M. It can only be controlled on the local axis; i.e. it is not possible with MLD-M either to command the master axis generator on a remote CCD slave axis. In order to allow for controlling of the virtual master axis generator, the following requirements have to be satisfied:

1. At the "Axis" input of the relevant functional block, "VmAxis1" must be transmitted as axis variable.

Example:

```
// Positionieren = MC_MoveAbsolute
fbMbAbs( Execute      := bLGExecute,
          Position     := rCmdPosition,
          Velocity     := rCmdVelocity,
          Acceleration := rCmdAcceleration,
          Deceleration := rCmdAcceleration,
          Axis         := VmAxis1);
```

2. The master axis generator must be activated (P-0-0917, bit 0="1"):

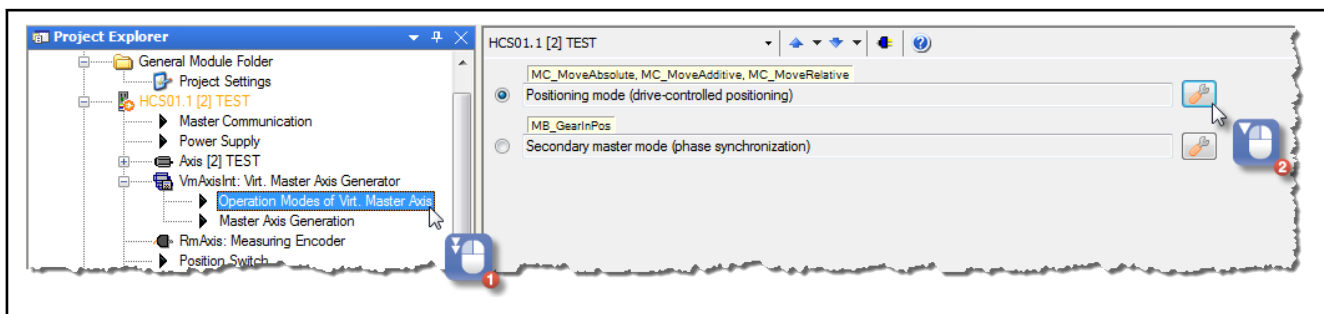


Fig. 4-13: Activating the master axis generator (dialog call)

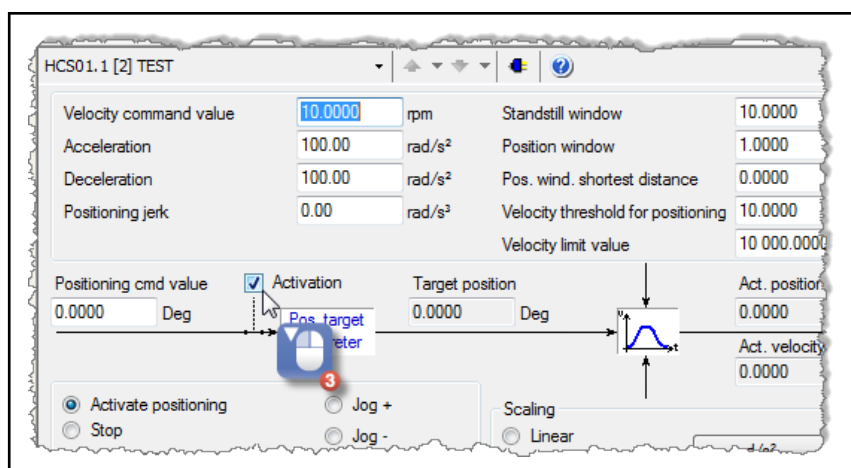


Fig. 4-14: Activating the master axis generator

3. "P-0-0758, Virtual master axis, actual position value" must be entered in "P-0-0916, Master axis format converter signal selection":

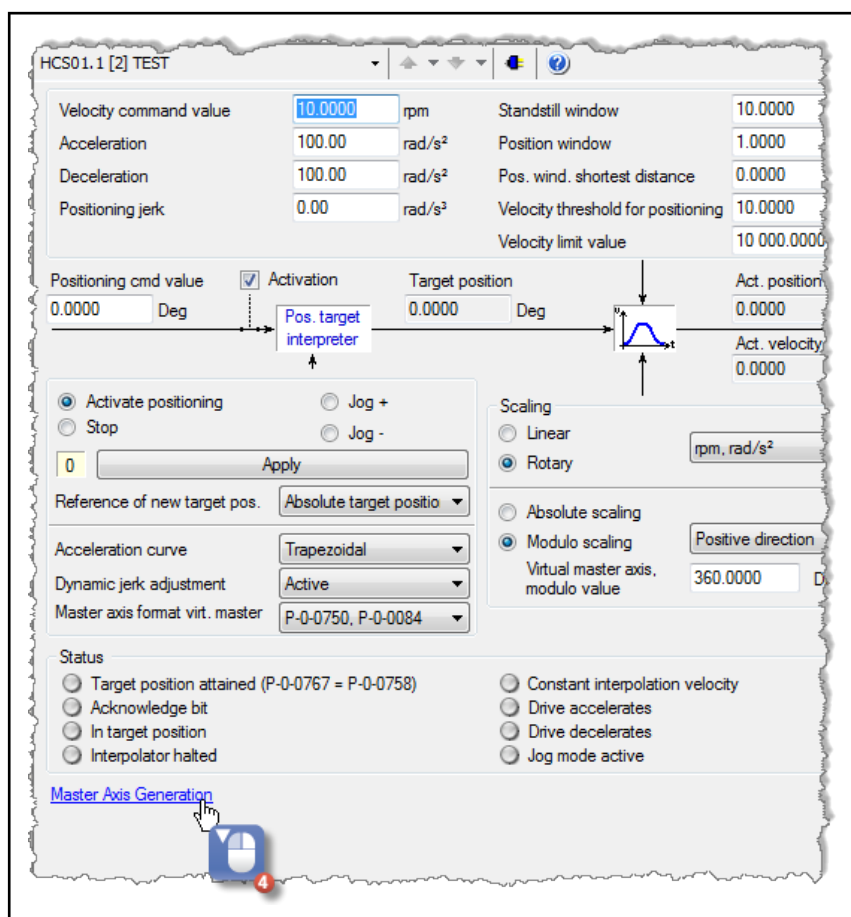


Fig. 4-15: Calling the master axis generation

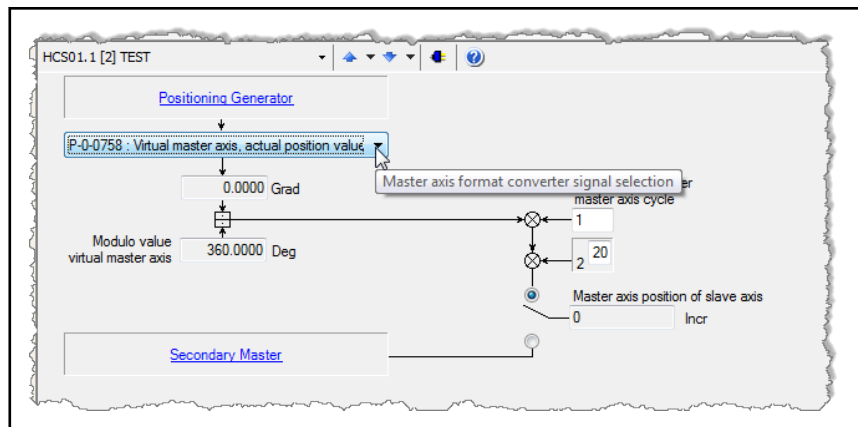


Fig. 4-16: Entering the actual position value of the virtual master axis as signal for the master axis format converter

Axis variables in synchronous operating modes

A number of variables of the "AXIS_REF" type pre-defined in the MLD system are intended for use with motion function blocks which affect synchronous running of one master and one slave axis. The naming of the axis variables already provides some information regarding the use of the axis variables:

- Axes ending with "Int" can only be used as master axes.
- When using axes ending with "Int", the master axis position is always the parameter locally available on the axis. When using "RmAxisInt" for example, the locally available P-0-0052 is the master axis position and not P-0-0052 on a remote axis in the CCD group.
- Axes ending with "1" to "n" (e.g. "RmAxis1", "Axis3", ...):
 - MLD-M: When using these axes as master axis in the MLD-M system mode, these axes have to be configured as master axes in "P-0-1820.0.1, CCD: Master axes configuration list".

Configuration via IndraWorks (context menu **Sercos (CCD)** ► **CCD: Configuration application type**)

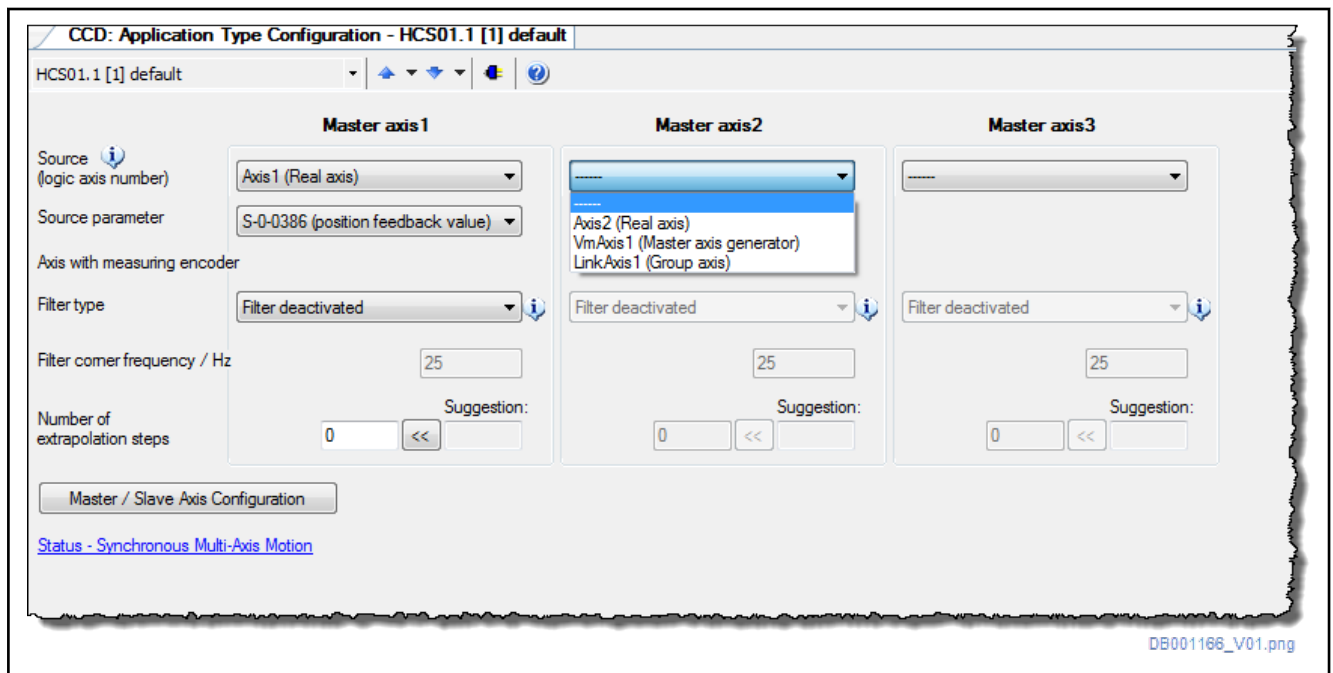


Fig. 4-17: Configuring the master axis (axes)

- MLD-S to MPx-20: With MLD-S, these axes cannot be used as master axis.
- MLD-S as of MPx-20: The axes "VmAxis1", "RmAxis1" and "LinkAxis1" can also be used as master axis with MLD-S (synonymous with the corresponding axis ending with "Int").

Basic functions of Rexroth IndraMotion MLD

Master axes

Variable name	AXIS_REF Structure elements	Parameters (comprises the master axis position)	Meaning / example application	Possible as master axis MLD-S / MLD-M	AxisData available
VmAxis1	CntrlNo := LOCAL_CNTRL AxisNo := VMA_1	"P-0-0761, Master axis position for slave axis"	VmAxis1 is the virtual master axis position in "P-0-0761, Master axis position for slave axis". <i>Examples:</i> - MLD-M A real CCD slave axis is to follow the virtual master axis in the CCD master in a synchronous operating mode. In this case, the VmAxis1 axis has to be transmitted as master and Axisn (with n=2...10) as slave at the function blocks for the synchronous operating modes (e.g. MB_GearInPos). In parameter "P-0-1820.0.1, CCD: Master axes configuration list", VMA_1 must moreover be configured as master axis in order to allow for the transmission of "P-0-0761, Master axis position for slave axis" via CCD to the slave axis. - MLD-S - up to MPx-20: VmAxis1 cannot be used on an MLD-S as master axis for a synchronization block. - MPx-20 and above: If the real axis of the MLD-S is to follow its local master axis generator, the VmAxis1 axis can be transmitted as master and Axis1 as slave at the function blocks for the synchronous operating modes (e.g. MB_GearInPos).	up to MPx-20: -- / X MPx-20 and above: X / X	--
VmAxisInt	CntrlNo := LOCAL_CNTRL AxisNo := VMA_INT	"P-0-0761, Master axis position for slave axis"	VmAxisInt is the internal virtual master axis in every drive, exclusively for local use. <i>Examples:</i> - MLD-M If a real CCD slave axis is to follow its local master axis generator, the VmAxis1 axis must be transmitted as master and Axisn (n = 1...10) as slave at the function blocks for the synchronous operating modes (e.g. MB_GearInPos). - MLD-S to MPx-20 If the real axis of the MLD-S is to follow its local master axis generator, the VmAxisInt axis must be transmitted as master and Axis1 as slave at the function blocks for the synchronous operating modes (e.g. MB_GearInPos). (In MPx-20 and above, VmAxis1 can be used as master.)	X / X	--

Variable name	AXIS_REF Structure elements	Parameters (comprises the master axis position)	Meaning / example application	Possible as master axis MLD-S / MLD-M	AxisData available
RmAxis1	CntrlNo := LO-CAL_CNTRL AxisNo := RMA_1	"P-0-1820.0.3, CCD: Actual position value of measuring encoder"	RmAxis1 is a prepared master axis position (position of a measuring encoder). The source of the prepared master axis is "P-0-0052, Actual position value of measuring encoder" of a measuring encoder that is connected to a real axis. <i>Examples:</i> - MLD-M The real CCD master axis is to follow the measuring encoder position in the CCD slave Axis2 in a synchronous operating mode. In this case, the RmAxis1 axis has to be transmitted as master and Axis1 as slave at the function blocks for the synchronous operating modes (e.g. MB_GearInPos). In parameter "P-0-1820.0.1, CCD: Master axes configuration list", RMA_1 (= 23) must moreover be configured as master axis and in "P-0-1820.0.2, CCD: Source of the actual position value of measuring encoder", the logic axis number of the CCD slave axis Axis2 has to be configured, in order to provide for the transmission of "P-0-0052, Actual position value of measuring encoder" of CCD slave axis 2 via CCD to the CCD master axis. - MLD-S - up to MPx-20: RmAxis1 cannot be used on an MLD-S as master axis for a synchronization block. - MPx-20 and above: If the real axis of the MLD-S is to follow its local measuring encoder, the RmAxis1 axis can be transmitted as master and Axis1 as slave at the function blocks for the synchronous operating modes (e.g. MB_GearInPos).	up to MPx20: -- / X MPx20 and above: X / X	--
RmAxisInt	CntrlNo := LO-CAL_CNTRL AxisNo := RMA_INT	"P-0-0052, Actual position value of measuring encoder"	RmAxisInt is the local measuring encoder position in every drive, exclusively for local use. <i>Examples:</i> - MLD-M If a real CCD slave axis is to follow its local measuring encoder, the RmAxisInt axis must be transmitted as master and Axisn (n = 1...10) as slave at the function blocks for the synchronous operating modes (e.g. MB_GearInPos). - MLD-S to MPx-20 If the real axis of the MLD-S is to follow its local measuring encoder, the RmAxisInt axis must be transmitted as master and Axis1 as slave at the function blocks for the synchronous operating modes (e.g. MB_GearInPos). (In MPx-20 and above, RmAxis1 can be used as master.)	X / X	--

Basic functions of Rexroth IndraMotion MLD

Variable name	AXIS_REF Structure elements	Parameters (comprises the master axis position)	Meaning / example application	Possible as master axis MLD-S / MLD-M	AxisData available
LinkAxis1	CntrlNo := LOCAL_CNTRL AxisNo := MA_LINK_1	"P-0-0053, Master axis position "	LinkAxis1 is a prepared master axis position (position of the group axis). The source of the prepared master axis is "P-0-0053, Master axis position". <i>Examples:</i> - MLD-M The real CCD slave axis Axisn is to follow the group axis ("P-0-0053, Master axis position" of the CCD master axis) in a synchronous operating mode. In this case, the LinkAxis1 axis has to be transmitted as master and Axisn (n=2...10) as slave at the function blocks for the synchronous operating modes (e.g. MB_GearInPos). In parameter "P-0-1820.0.1, CCD: Master axes configuration list", MA_LINK_1 (= 26) must moreover be configured as master axis in order to provide for the transmission of "P-0-0053, Master axis position" of the CCD master axis via the CCD group into "P-0-0053, Master axis position" of the CCD slave axis. - MLD-S - up to MPx-20: LinkAxis1 cannot be used on an MLD-S as master axis for a synchronization block. - MPx-20 and above: If the real axis of the MLD-S is to follow its local group axis position ("P-0-0053, Master axis position"), the LinkAxis1 axis can be transmitted as master and Axis1 as slave at the function blocks for the synchronous operating modes (e.g. MB_GearInPos).	up to MPx-20: -- / X MPx-20 and above: X / X	--

Variable name	AXIS_REF Structure elements	Parameters (comprises the master axis position)	Meaning / example application	Possible as master axis MLD-S / MLD-M	AxisData available
LinkAxisInt	CntrlNo := LO-CAL_CNTRL AxisNo := MA_LINK_INT	"P-0-0053, Master axis position" with slave axis Axis1, "P-0-0787, Group axis 1 position" of the slave axis with Axis2...10 as slave axis	LinkAxisInt is the local group axis position. The group axis position is cyclically written by the higher-level control (e.g. MLD) via the master communication. Depending on the slave axis, the higher-level control must write the group axis position into different parameters: - If Axis1 is transmitted as slave axis to a synchronization block (and LinkAxisInt as master axis), Axis 1 follows the master axis position in "P-0-0053, Master axis position". I.e. the higher-level control must write the master axis position into "P-0-0053, Master axis position" of Axis1. - If Axis2 to Axis10 is transmitted as slave axis to a synchronization block, these axes follow the master axis position in their relevant "P-0-0787, Group axis 1 position". I.e. the higher-level control must write the master axis position into the relevant "P-0-0787, Group axis 1 position" of Axis2 to Axis10. <i>Examples:</i> - MLD-M If a real CCD slave axis is to follow its group axis position, the LinkAxisInt axis must be transmitted as master and Axisn (n=1...10) as slave at the function blocks for the synchronous operating modes (e.g. MB_GearInPos). - MLD-S to MPx-20 If the real axis of the MLD-S is to follow its local group axis position, the LinkAxisInt axis must be transmitted as master and Axis1 as slave at the function blocks for the synchronous operating modes (e.g. MB_GearInPos). (In MPx-20 and above, LinkAxis1 can be used as master.)	X / X	--
Axisn (n=1...10)	CntrlNo := LO-CAL_CNTRL AxisNo := AX-IS_2 ... AxisNo := AX-IS_10	"P-0-1808.x.10, CCD: Active position feedback value" or "P-0-1821.x.1, CCD: Position command value for master axis n" (selection in "P-0-1820.0.4, CCD: Axis position selection") Note: P-0-1808.x.10 comprises "S-0-0386, Active position feedback value" of axis x, P-0-1821.x.1 comprises "P-0-0434, Position command value of controller" of axis x.	"Axisn" are the real axes. "Axisn" can only be used as master axis for a synchronization block with MLD-M. With MLD-S, "Axisn" is not supported as master axis for a synchronization block. <i>Examples:</i> - MLD-M The real CCD slave axis Axis3 is to follow the real CCD slave axis Axis2 in a synchronous operating mode. In this case, the Axis2 axis has to be transmitted as master and Axis3 as slave at the function blocks for the synchronous operating modes (e.g. MB_GearInPos). In "P-0-1820.0.1, CCD: Master axes configuration list", AXIS_2 must moreover be configured as master axis in order to allow for the transmission of "P-0-0386, Active position feedback value" or optionally also P-0-0434, Position command value of controller via CCD to the slave axis. - MLD-S Axisn cannot be used on an MLD-S as master axis for a synchronization block.	-- / X	X

Tab. 4-2: Supported axis variables at the functional block input "Master"

Basic functions of Rexroth IndraMotion MLD

Slave axes

Variable name	Meaning / example application
VmAxis1	VmAxis1 is the virtual master axis position in "P-0-0761, Master axis position for slave axis". If VmAxis1 is transmitted as slave axis at a synchronization block (e.g. "MB_GearInPos"), the local virtual master axis is in the secondary master mode. In the secondary master mode, the local virtual master axis may follow two different master axes (real master axis RmAxisInt="P-0-0052, Actual position value of measuring encoder") or external virtual master axis LinkAxisInt="P-0-0053, Master axis position"). In secondary master mode, only the phase synchronization mode is supported. Consequently, VmAxis1 can only be transmitted as slave axis at the MB_GearInPos function block (not at the MC_GearIn, MB_MotionProfile and MC_CamIn function blocks). The secondary master mode can be used with both, MLD-S and MLD-M. It can only be activated on the local axis; i.e. it is not possible with MLD-M either to activate the secondary master mode on a remote CCD slave axis. Example: Secondary master mode with MLD-S/MLD-M: The local virtual master axis is to follow the real axis (measuring encoder). In this case, the measuring encoder of the MLD-S axis and/or CCD master axis has to be activated and the RmAxisInt axis variable has to be transmitted at the MB_GearInPos functional block as master axis and VmAxis1 as slave axis. Then, the local virtual master axis ("P-0-0761, Master axis position for slave axis") follows the local measuring encoder in the phase synchronization operating mode.
Axis1	Axis1 is the real local axis. Axis1 is always available, with MLD-S and with MLD-M. <i>Examples:</i> 1. MLD-M The CCD master axis (Axis1) is to follow a master axis in a synchronous operating mode. The following axes may serve as master axis: • Local virtual master axis generator ("P-0-0761, Master axis position for slave axis" of Axis1 = VmAxisInt or VmAxis1, if VMA_1 is configured as master axis in "P-0-1820.0.1, CCD: Master axes configuration list".) • Measuring encoder of the local axis ("P-0-0052, Actual position value of measuring encoder" = RmAxisInt) / a remote axis (P-0-0052 of the remove axis = RmAxis1), if RMA_1 is configured as master axis in "P-0-1820.0.1, CCD: Master axes configuration list". (Parameterization of the measuring encoder source axis using parameter "P-0-1820.0.2, CCD: Source of measuring encoder position feedback value". P-0-1820.0.3 comprises the actual position value of the measuring encoder.) • Group axis ("P-0-0053, Master axis position" = LinkAxisInt oder LinkAxis1, if MA_LINK_1 is configured as master axis in "P-0-1820.0.1, CCD: Master axes configuration list".) • Real axis (CCD slave axis, e.g. Axis3, if AXIS_3 is configured as master axis in "P-0-1820.0.1, CCD: Master axes configuration list") 2. MLD-S The local axis (Axis1) is to follow a master axis in a synchronous operating mode. The following axes may serve as master axis: • Virtual master axis generator ("P-0-0761, Master axis position for slave axis" = VmAxisInt up to MPx20, VmAxis1 MPx20 and above) • Group axis ("P-0-0053, Master axis position" = LinkAxisInt up to MPx20, LinkAxis1 MPx20 and above) • Measuring encoder ("P-0-0052, Actual position value of measuring encoder" = RmAxisInt up to MPx20, RmAxis1 MPx20 and above)

Variable name	Meaning / example application
Axis2...Axis10	Axis2 to Axis10 are the real axes in a CCD group. Axis2 to Axis10 are only available with CCD and MLD-M, not with MLD-S. The remote axes contained in the list "P-0-1801.0.10, CCD: Addresses of projected drives" correspond to the order of Axis2 to Axis10 (Axis2 acts on the drive of list element 0 of P-0-1801.0.10,...) Example: The CCD slave axis Axis2 is to follow a master axis in a synchronous operating mode. The following axes may serve as master axis: - Virtual master axis generator of Axis1 ("P-0-0761, Master axis position for slave axis" of the CCD master axis = VmAxis1). VMA_1 must be configured as master axis in P-0-1820.0.1. - Virtual local master axis generator ("P-0-0761, Master axis position for slave axis" of Axis2 = VmAxisInt) - Measuring encoder of the local axis ("P-0-0052, Actual position value of measuring encoder" of Axis2 = RmAxisInt) / a remote axis (P-0-0052 of the remove axis = RmAxis1), if RMA_1 is configured as master axis in "P-0-1820.0.1, CCD: Master axes configuration list". (Parameterization of the measuring encoder source axis using parameter "P-0-1820.0.2, CCD: Source of measuring encoder position feedback value". P-0-1820.0.3 comprises the actual position value of the measuring encoder.) - Group axis ("P-0-0053, Master axis position" = LinkAxis1: Master axis position is transmitted by P-0-0053 of the CCD master axis if MA_LINK_1 is configured as master axis in "P-0-1820.0.1, CCD: Master axes configuration list" / "P-0-0787, Group axis 1 position" = LinkAxisInt: P-0-0787 has to be written by the higher-level control unit) - Real axis (CCD slave axis, e.g. Axis3, if AXIS_3 is configured as master axis in "P-0-1820.0.1, CCD: Master axes configuration list")

Tab. 4-3: Supported axis variables at the functional block input "Slave"

Axis availability for functions and function blocks

The following table provides an overview of all available function blocks with axis reference inputs and their availability for each axis:

Designation	Availability for axis							
	Axis1 (local axis)	"Axis2" to "Axis10" (remote axes)	VmAxis1	VmAxisInt	RmAxis1	RmAxisInt	LinkAxis1	LinkAxisInt
MB_Command	X	X	--	--	--	--	--	--
MB_ChangeProfile-Set	X	X	--	--	--	--	--	--
MB_ChangeProfileStep	X	X	--	--	--	--	--	--
MB_Home	X	X	--	--	--	--	--	--
MB_MotionProfile	MasterAxis SlaveAxis	MasterAxis SlaveAxis	MasterAxis*1	MasterAxis	MasterAxis*1	MasterAxis	MasterAxis*1	MasterAxis
MB_SetPositionControlMode	X	X	--	--	--	--	--	--
MB_GearInPos	MasterAxis SlaveAxis	MasterAxis SlaveAxis	MasterAxis*1 SlaveAxis	MasterAxis	MasterAxis*1	MasterAxis	MasterAxis*1	MasterAxis
MB_PhasingSlave	X	X	X ²	--	--	--	--	--
MB_PreSetMode	X	X	--	--	--	--	--	--
MB_Stop	X	X	X	--	--	--	--	--

Basic functions of Rexroth IndraMotion MLD

Designation	Availability for axis							
	Axis1 (local axis)	"Axis2" to "Axis10" (remote axes)	VmAxis1	VmAxisInt	RmAxis1	RmAxisInt	LinkAxis1	LinkAxisInt
MC_CamIn	MasterAxis SlaveAxis	MasterAxis SlaveAxis	MasterAxis* ¹	MasterAxis	MasterAxis* ¹	MasterAxis	MasterAxis* ¹	MasterAxis
MC_CamOut	X	X	X	--	--	--	--	--
MC_GearIn	MasterAxis SlaveAxis	MasterAxis SlaveAxis	MasterAxis* ¹	MasterAxis	MasterAxis* ¹	MasterAxis	MasterAxis* ¹	MasterAxis
MC_GearOut	X	X	--	--	--	--	--	--
MC_Reset	X	X	--	--	--	--	--	--
MC_Jog	X	X	X	--	--	--	--	--
MC_Phasing	X	X	--	--	--	--	--	--
MC_Power	X	X	--	--	--	--	--	--
MC_MoveAbsolute	X	X	X	--	--	--	--	--
MC_MoveAdditive	X	X	X	--	--	--	--	--
MC_MoveRelative	X	X	X	--	--	--	--	--
MC_MoveVelocity	X	X	X	--	--	--	--	--
MC_Stop	X	X	X	--	--	--	--	--
MC_TorqueControl	X	X	--	--	--	--	--	--
MX_MoveAbsolute	X	X	X	--	--	--	--	--
MX_MoveAdditive	X	X	X	--	--	--	--	--
MX_MoveRelative	X	X	X	--	--	--	--	--
MX_SetOpMode	X	X	--	--	--	--	--	--

*¹

With MLD-S only possible as master axis from MPx-20

*²

Only possible from MPx-20

Tab. 4-4:

Available motion function blocks

Designation	Availability for axis							
	Axis1 (local axis)	"Axis2" to "Axis10" (remote axes)	VmAxis1	VmAxisInt	RmAxis1	RmAxisInt	LinkAxis1	LinkAxisInt
MX_Command	X	X	--	--	--	--	--	--
MX_SetControl	X	--	--	--	--	--	--	--
MX_SetDeviceMode	X	--	--	--	--	--	--	--
MX_SynchronControl	X	--	--	--	--	--	--	--
MX_fGetDriveWarning	X	--	--	--	--	--	--	--
MX_fSetDriveError	X	--	--	--	--	--	--	--
MX_fSetDriveWarning	X	--	--	--	--	--	--	--

Tab. 4-5:

General function blocks/functions

Assignment of logic MLD-M axis number to Sercos address of CCD slaves

Axis numbers 2 to 10 (AxisNo=AXIS_2 to AXIS_10) of an MLD-M system are real axes in the CCD slaves. Axis number 1 represents the local axis in the CCD master. The AXIS_REF variables and their "AxisNo" are used for pro-

programming in the PLC user program. This gives a maximum of 10 real axes with the logical numbers 1 to 10 in an MLD-M system.

The Sercos addresses are assigned to logical axis numbers 1 to 10 in MLD-M through the Sercos address setting for the CCD master and for the CCD slaves via the order of Sercos addresses of the remote drives in parameter "P-0-1601, CCD: Addresses of projected drives" or in the IndraWorks dialog "Cross Communication Drive : Settings".

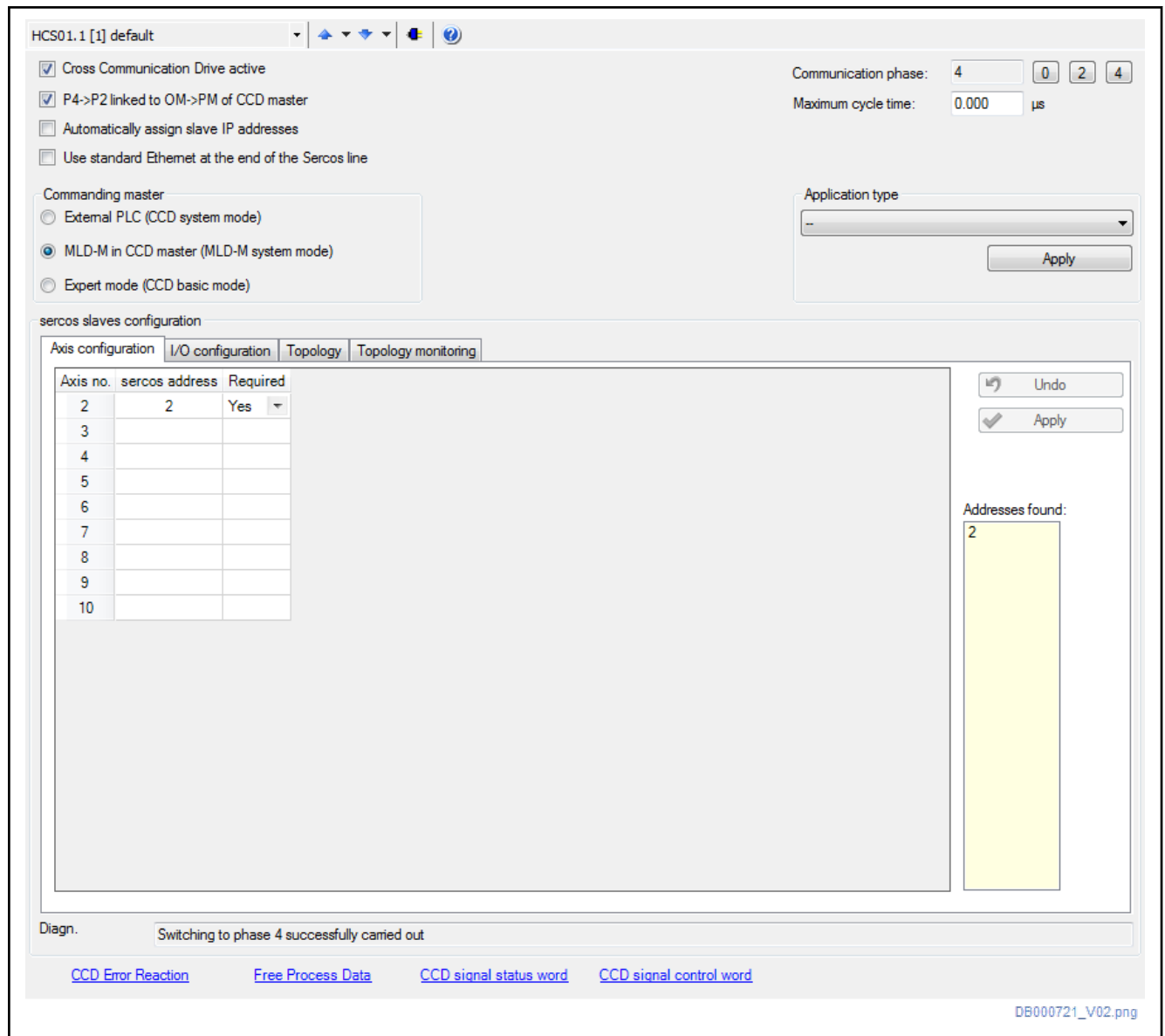


Fig. 4-18: "Cross Communication Drive: Settings" dialog box

Parameter "P-0-4031, Overview of device addresses" shows the overall configuration of all available real MLD-M axes and their Sercos addresses.

Element no. of P-0-4031	High word content = Sercos address of the axis	Meaning
0	1	Sercos address of Axis1 (1st logical MLD- axis, of the local axis [CCD master])
1	3	Sercos address of Axis2 (2nd logical MLD- axis, of an axis in the CCD slave)

Tab. 4-6: Content of "P-0-4031, Overview of device addresses" for example above

See also Functional description of firmware "Cross communication (CCD)"

4.4.5 Notes on application and programming

General information

The following paragraphs use examples to show how to use axis variables in regard to PLCopen and program porting:

Example A (project using axis variables from the "MX_Base" library)

```
instMX_MoveAbsolute1(Axis:= Axis3, ...);
instMX_MoveAbsolute2(Axis:= VmAxisInt, ...);
```

Example B (project with user-defined axis variables)

```
VAR
Porta1X : AXIS_REF:= ( CntrlNo := LOCAL_CNTRL, AxisNo :=
AXIS_1);
Porta1Y : AXIS_REF:= ( CntrlNo := LOCAL_CNTRL, AxisNo :=
AXIS_4 );
AuxiliaryAxis : AXIS_REF:= ( CntrlNo := LOCAL_CNTRL, AxisNo :=
AXIS_2 );
Tool : AXIS_REF:= ( CntrlNo := LOCAL_CNTRL, AxisNo :=
AXIS_3 );
MasterAxis : AXIS_REF:= ( CntrlNo := LOCAL_CNTRL, AxisNo :=
VMA_EXT);
END_VAR
```

Addressing is therefore as follows:

```
instMX_MoveAbsolute(Axis:= AuxiliaryAxis, ...);
- or -
instMX_MoveAbsolute(Axis:= MasterAxis, ...);
```

MLD function blocks and parameters for axis commanding

Functions and function blocks:

- "MX_SetControl": Controls temporary control over local axis when controlling local axis via master communication
- "MX_SetOpMode": Directly activates a Sercos operation mode
- "MC_Power", "MB_PresetMode": Activate axis controller enable and possibly preselect operation mode
- "MC_Stop", "MB_Stop": Stop the drive
- "MX_Command"/"MB_Command": Activate a drive command
- "MC_Home": Activates "drive-controlled homing procedure"
- "MC_MoveVelocity": "Preset velocity" operation mode
- "MC_MoveAbsolute"/"MX_MoveAbsolute", "MC_MoveRelative"/"MX_MoveRelative", "MC_MoveAdditive"/"MX_MoveAdditive": "Positioning" mode

- "MC_Jog": "Positioning (jog +/-)" operation mode
- "MC_GearIn", "MC_GearOut": "Velocity synchronization" operation mode
- "MB_GearInPos", "MC_CamIn", "MC_CamOut": "Position synchronization" operation mode
- "MB_Phasing", "MB_PhasingSlave": Additive position offset for position synchronization
- "MB_MotionProfile": "Position synchronization" operation mode with motion profiles

Parameters and diagnostic messages for axis commanding

- "P-0-1367, PLC configuration", bit 4: Activates "permanent control" of the local axis for MLD
- "P-0-1800.0.1, CCD: Configuration", bits 4 and 3: Activate MLD-M system mode
- Parameters of the used "velocity control", "torque control", "position synchronization", "drive-controlled positioning" and "velocity synchronization" modes
- Parameters of the "internal operation modes" with temporary control for the operating mode as intelligent servo axis:
 - P-0-1450, PLC/setting-up mode Positioning command value
 - P-0-1451, PLC/setting-up mode, positioning velocity
 - P-0-1452, PLC/setting-up mode, positioning acceleration
 - P-0-1453, PLC/setting-up mode, positioning deceleration
 - P-0-1454, PLC/setting-up mode, positioning command value acceptance
 - P-0-1455, PLC/setting-up mode, positioning command value acknowledge
 - P-0-1460, PLC/setting-up mode, velocity command value
 - P-0-1461, PLC/setting-up mode Ramp pitch
 - P-0-1463, PLC/setting-up mode Deceleration ramp
 - P-0-1465, PLC/setting-up mode Torque/force command value

Working with retain variables

Retain variables are designated by the keyword "RETAIN". These variables keep their value after uncontrolled termination and after the control unit is switched off and back on normally (according to the "Online" "Reset" command). Upon program restart, the saved values are used for further processing. One application example is a piece counter in a production line which is expected to continue to count after a power failure. All other variables are re-initialized in this case, either with their initialized values or with the standard initializations. Retain variables are reinitialized after "reset cold", "reset origin" and - in contrast to persistent variables - after the program is downloaded again.

Depending on the hardware (standard or Advanced control panel with memory card), either 472 bytes or 31704 bytes of memory are available for **retain variables** and **persistent variables**.

The retain data can be read or written in one step using P-0-1359. The list length (MaxLen) of P-0-1359 depends on the size of the retain memory (126 or 7934 elements of 4 bytes each).



Writing is not allowed while the program is running. If P-0-1359 is written, the data have to come from the same program (i.e., there can be no changes to the retain data or persistent data).

During boot-up, the retain memory is transmitted to the RAM and in the case of "Power-Fail IRQ" the data are copied from the RAM to the retain memory.

4.5 Task system

4.5.1 Definition of terms

- **Task:** A task is a self-contained, elementary control flow within a process. Each task has its own state with program counter, memory stack, etc.
- **Multitasking** (multiple processor operation): Multitasking is the ability to perform several tasks simultaneously. The different processes are activated alternately in such short intervals that this creates the impression of simultaneity.
- **Scheduler:** A scheduler is an important element of a multitasking operating system. According to a given strategy, it determines the order in which ready tasks are processed. The term "scheduling" refers to the corresponding strategies and methods with which the processing order of the task is implemented.

4.5.2 Brief description

- Supported task types** Rexroth IndraMotion MLD supports up to four IEC61131 user tasks of the following task types:
- Periodic task ("cyclic" starting from 1 ms)
 - Freewheeling task
 - Event task (triggering at 1-ms intervals)
 - Task triggered by external event (external event task); two system events were provided for this task type:
 - "FKM_SYNCHRONIZED_TASK" event: Motion task in synchronism with master communication
 - "CCD_SYNCHRONIZED_TASK" event: Motion task in synchronism with CCD (only with MPC)

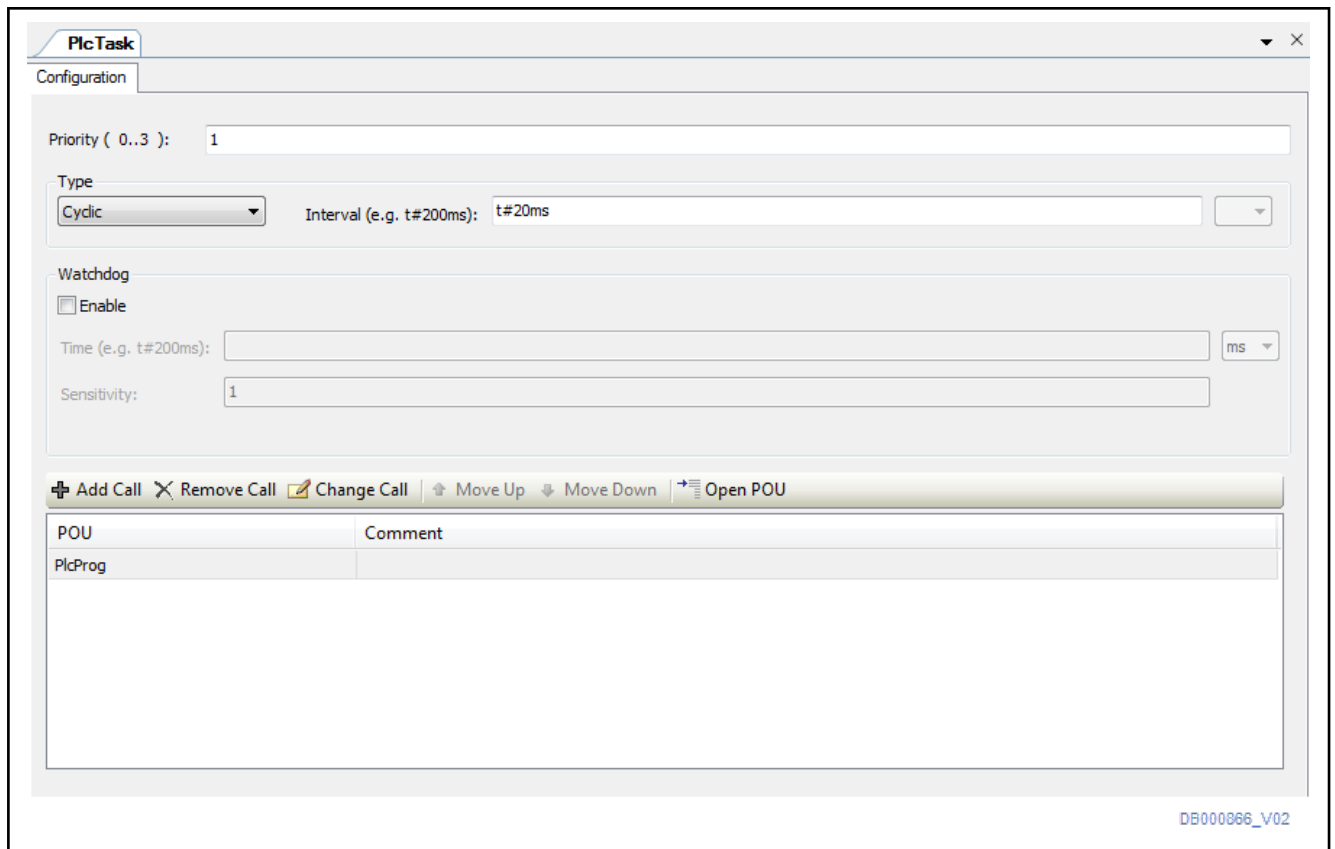


Fig. 4-19: (Task) configuration



Rexroth IndraMotion MLD supports system events.

Task system and priorities

The internal timing and the sequence of the processing of the individual tasks is managed via what is called the task system:

- Preemptive multitasking is supported. This allows low priority tasks to be interrupted by high priority tasks at any time.
- A maximum of four tasks can be "active" at the same time.
- There are four levels of priority [priority 0...3 (0=high priority)].
 - The tasks used have to have different priority so that defined processing is obvious.
 - High priority PLC tasks interrupt low priority PLC tasks during processing (exceptions: tasks cannot be interrupted for a few µs for transmitting the process image or commanding a motion).
 - If a low priority task is started, it may start running with a delay.
- The IEC tasks are not permanently assigned to any axis or task. This means there is no dedicated motion task. The existing configuration options can be used for the various tasks.
- An error message is output when a task interval value is set too high in the task configuration.
- Generally, as few tasks as possible should be created to keep the program sequences clearly structured.



It is possible to create up to four different tasks. (Creating tasks: see documentation "IndraLogic 2G, PLC Programming System (R911336876)".)

4.5.3 Task properties

Periodic (cyclic) tasks	Periodic ("cyclic") tasks have the following features: • Constant cycle time • The cyclic task is started exactly once per specified period, i.e., the code is run exactly once in every time interval, provided the previous run was completed (in case of time overflow and activated watchdog, the PLC is stopped with an "Exception" error status).
	If a periodic task uses up the allocated computing time and is still active when it should already have restarted, it is not restarted before the next period. This means the system does not attempt to "make up" for the lost cycle.
Freewheeling task	A freewheeling task has the following features: • A freewheeling task runs constantly. • There can only be one freewheeling task and it has to be created with the lowest priority. • A freewheeling task restarts immediately after the end of the task.
Event tasks	Event tasks have the following features: • Event tasks are started by setting a global Boolean PLC variable. The start conditions are periodically checked, whereby the reaction times are defined by the minimum possible PLC cycle time (see "Performance data"). • An event task is processed exactly once after the event (edge) occurred.
External event task	In principle, an external event task works like an event task. The difference lies in how the task is activated. An external event task is activated via a system event. The user cannot influence system events. The following two system events are supported: • FKM_SYNCHRONIZED_TASK • CCD_SYNCHRONIZED_TASK The external event task allows a motion task synchronized to the NC cycle or a motion task synchronized to the CCD cycle to be implemented.

4.5.4 Motion task

General information

A motion task is a task synchronous to the NC or CCD cycle. The motion function blocks and technology functions are normally processed in this task.

The real-time data of AxisData are calculated at the beginning and end of the motion task.

Motion task in synchronism with master communication

The motion task in synchronism with master communication is a task synchronized with the NC cycle.

It is created by an "external event" task with the "FKM_SYNCHRONIZED TASK" system event.

Motion task cycle time	The motion task in synchronism with master communication runs synchronously with the NC cycle. The resulting motion task cycle time corresponds to the NC cycle time. The NC cycle time can be taken from parameter "S-0-0001, NC cycle time (TNcyc)" (except for Sercos III master communication). With Sercos III master communication, the NC cycle time corresponds to the "producer cycle time" (S-0-1050.x.10) of the connection configured by the master in the MDT telegram.
	 The NC cycle time is specified by the higher-level master communication master only if the Sercos III master communications is used. For all other master communications, the NC cycle time can be freely set via parameter "S-0-0001, NC cycle time (TNcyc)". Consider the following aspects when setting the NC cycle time in conjunction with the motion task in synchronism with master communication: • The minimum allowed NC cycle time is 1 ms. All integral multiples of the minimum allowed NC cycle time are also allowed.
Application	The time MLD-S is called should be synchronized for physically separate drives so that each MLD-S can react to command values and actual values at the same time. For this purpose, the task is synchronized with the NC cycle.
Example	A higher-level control unit cyclically writes master axis positions to a global PLC register. MLD evaluates the master axis positions and adds a master axis offset to them. The result is written to parameter "P-0-0053, Master axis position". When the next MDT (Master Data Telegram) is received, the drive applies the value of P-0-0053. This means the command value from the NC is subject to an offset and takes effect one NC cycle later.
Timing	With MLD, the motion task in synchronism with master communication starts after and ends before the processing of the MDT and the AT (MDT: Master Data Telegram ; AT: Antriebs-Telegramm [drive telegram]).

Motion task in synchronism with CCD

The motion task in synchronism with CCD is a task synchronized with the CCD cycle (MS channel).

It is created by an "external event" task with the "CCD_SYNCHRONIZED_TASK" system event.



If the axis is a CCD master axis and CCD is active, the motion task in synchronism with CCD is only allowed in the MLD-M system mode.

If CCD is not active or the axis is not a CCD master axis, a motion task in synchronism with CCD configured in the PLC project is handled like a motion task in synchronism with master communication. In this case, the description of the master communication-synchronous motion task applies.

Motion task cycle time (CCD cycle time)	If Sercos III master communication is used, the CCD cycle time (and therefore the motion task cycle time) corresponds to the "producer cycle time" (S-0-1050.x.10) of the connection configured by the master in the MDT, if this time is shorter than the value of "P-0-1800.0.10, CCD: Cycle time". With all other master communications, the motion task cycle time, as well as the CCD cycle time, correspond to the NC cycle time [S-0-0001, NC cycle time (TNcyc)], provided the NC cycle time is shorter than the value of "P-0-1800.0.10, CCD: Cycle time". If not, they correspond to the value of

"P-0-1800.0.10, CCD: cycle time". The active CCD cycle time can be taken from parameter "P-0-1810.0.3, CCD: Timing settings".

Consider the following aspects when setting the CCD cycle time in conjunction with the motion task in synchronism with CCD:

- The minimum allowed CCD cycle time is 1 ms. All integral multiples of the minimum allowed CCD cycle time are also allowed for the motion task in synchronism with CCD.



CCD in MLD-M system mode currently supports CCD cycle times of 1 ms, 2 ms and 4 ms.

Application

The time at which a periodic PLC task (motion task) is called in the CCD master axis should be linked to the CCD cycle (MS channel). This is why the MLD-M system mode and "permanent MLD control" have to be activated in the CCD master axis.

The motion task in synchronism with CCD is only used in conjunction with the MLD-M system mode. In MLD-M system mode, the MLD-M of the CCD master axis controls the master axis (local slave) itself, as well as the remote axes. This often requires processing actual values of all axes latched at the same time, as well as writing command values that take effect in all axes at the same time. This consistent data exchange is achieved with the motion task in synchronism with CCD in conjunction with the "AxisData" axis structure.

Example

A milling head is to be moved in three dimensions with MLD-M. To achieve proper milling results, the control unit (MLD-M) has to know the actual position values of all axes involved in the motion relative to the same time (T4) so that it can calculate the corresponding command values. These calculated command values have to be transmitted to all axes at the same time so they take effect at the same time. A task synchronized with the CCD clock is required to ensure this occurs. The real-time data are exchanged via the "AxisData" axis structure.

Timing

With MLD, the motion task in synchronism with CCD runs between CCD AT processing and CCD MDT processing.

The real-time data of the "AxisData" axis structure are updated before the start of the task and after the end of the task. The actual values are read after CCD AT processing and the command values are written before CCD MDT processing. This way consistent data can always be accessed or written using the "AxisData" axis structure within the motion task in synchronism with CCD. The values written via the axis data structure take effect in the axes at the same time.



Only one "external event" motion task can be used at a time.



Both types of motion tasks are always running. However, they are only synchronized once cyclic data are exchanged via the master communication or via CCD.

Diagnostics and error messages in relation to a motion task

- **PLC error**
If the boot project was loaded incorrectly, this is signaled in P-0-1365 and IndraWorks ("Log" tab page of the device editor).
- **Drive error: F6010 PLC runtime error**
Error F6010 is triggered, e.g., if an error occurs while the boot project is loading. Possible errors in conjunction with a motion task are the PLC

errors mentioned above (see "F6010" in the "Troubleshooting Guide" documentation).

- **Transition command error**

- C0241 Incorrect parameterization of motion task
- C0266 Incorrect CCD phase switch

Please refer to the "Troubleshooting Guide" documentation for possible causes and remedies for the C0266 transition command error.

In conjunction with a motion task in synchronism with CCD, the error occurs when the drive detects that the motion task in synchronism with CCD cannot be operated with the configured CCD cycle time (P-0-1810.0.3, CCD: Timing settings). The parameter "P-0-1810.0.10, CCD: Diagnostic message text" contains the detailed error message in case the transition command error C0266 occurs.

Freewheeling task

Another motion task variant is the freewheeling task:

- A freewheeling task runs constantly.
- There can only be one freewheeling task and it has to be created with the lowest priority.
- A freewheeling task restarts immediately after the end of the task.



Bosch Rexroth provides sample configurations for the different characteristics so the user can create suitable task concepts.

4.5.5 Task monitoring (watchdog)

Monitoring for each task can generally be set individually. This makes it possible, e.g., to set the watchdog of a task to a multiple of the periodic time. This is useful especially for motion tasks, because they have to react quickly, but sometimes also have to perform computing-intensive tasks.

Using network variables

If network variables are used, the task is loaded with additional runtime depending on the entire communication via the Ethernet interface. Depending on the number of network channels and other communication, the runtime can increase by a few milliseconds.

⇒Rapidly running tasks with a sensitive watchdog are not suitable for transmitting network variables. We recommend using network variables in slow or separate tasks with a tolerant watchdog.

Watchdog function

Rexroth IndraDrive supports the following watchdog function:

- Time
- Sensitivity (number of overtimes until watchdog triggers)
- *Reaction:*
 - A PLC exception is generated.
 - All outputs are set to "0" and the PLC goes to STOP.
 - Error "F6010" is generated on the drive controller; this causes the best possible deceleration for the axes.



After the watchdog triggers, the PLC has to be reset with RESET (cf. P-0-1350).

Configuring the watchdog For each task, a watchdog can be configured.

The watchdog is active when the option has been activated. The task then is terminated with an error status ("exception") as soon as the configured watchdog time is exceeded, whereby the specified sensitivity is taken into account for the calculation. If the **Update IO while in stop** option has been activated in the "PLC settings", the outputs are reset to the defined standard values. The following instances are possible:

Several successive timeouts; the following applies:

Sensitivity	Exception in cycle...
0, 1, 2	1
3	2
...	...
n	n-1

One-time timeout:

Exception when the cycle time of the current cycle is longer than "time × sensitivity".

Example:

Time="t#10ms", sensitivity="5"

⇒ Exception as soon as the task (one-time) runs longer than 50 ms.

Please note that a watchdog can be switched off for certain PLC cycles using the "CmpIecTask.library" functions, which is useful for cycles that can take more time due to initializations.

Switch a watchdog off or on as follows:

1. Declare a suitable variable for the task handle (RTS_IEC_HANDLE type):

Program:

```
hIecTask : RTS_IEC_HANDLE;
```

2. Use the interface functions to switch off a watchdog (and switch it on again afterwards):

Program:

```
hIecTask := IecTaskGetCurrent(0);
IecTaskDisableWatchdog(hIecTask);
... // Code that is protected against watchdog
IecTaskEnableWatchdog(hIecTask);
```

4.5.6 Task stack verification

With MPx-19 and above, the memory requirement is determined in a task stack verification. With MPB, 8 kB of stack can be used; 10 kB can be used with MPC. This means that a fixed stack size is available in each task for the (nested) function calls within the IEC PLC program (refers to "Function" calls).

4 kB memory has been reserved for calling operating system functions (firmware functions and function blocks) from a function.

If the available stack memory is exceeded, an error is generated when the PLC application is compiled.

If dynamic constructions are used (object-oriented programming), it is not possible to calculate the memory requirement of the task. In this case, a warning will be displayed.



In functions, no large data arrays should not be used as parameters, variables or return values.

4.5.7 Runtime measurements

Measuring task runtimes in milliseconds

There are various ways to determine the required time in MLD. The system automatically measures and statistically evaluates the task runtimes. In addition, the remaining time is measured in every PLC time slot.

The task runtimes can be evaluated using a function in the PLC program. For this purpose, the "SysTaskGetInfo" function with the "SYS_TASK_INFO" structure is made available in the "SysTask" library. This provides the runtime of the desired task as a structure with statistics.

Include the maximum and average runtime when evaluating:

- If a one-time timeout already causes problems and is to be monitored, the maximum runtime "dwCycleTimeMax" has to be used for evaluation (corresponds to watchdog sensitivity "1"). The ratio of maximum time to watchdog time indicates the load of the corresponding task. Make sure that the load is not too high (e.g., not above 50%) for subsequent firmware updates that may require slightly more processing power, and primarily for future program enhancements.
- If a one-time timeout is tolerated, higher maximum runtimes can be allowed for the task. In this case, the ratio of average computing time to watchdog time is the one that is decisive. The maximum runtimes of tasks mostly occur in one cycle or in a few cycles.
- This time measurement is relatively inaccurate due to the resolution in milliseconds for high demands with a tolerated maximum time of only one or two milliseconds. This is why there is another way to measure the available remaining time.

Example of a multitasking PLC application

Basic functions of Rexroth IndraMotion MLD

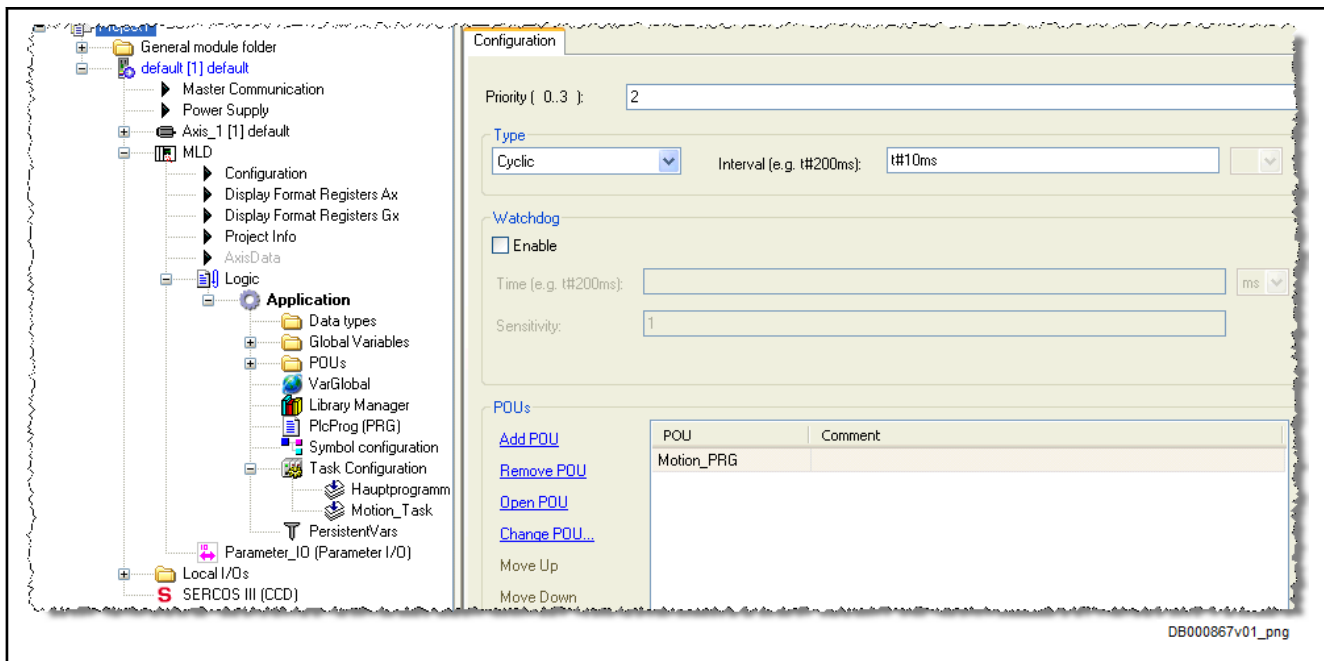


Fig. 4-20: Example of a multitasking PLC application

The above example shows the runtimes of three tasks with cycle times of 1 ms, 10 ms and 50 ms. The 1 ms task had a maximum runtime of 1 ms. In case of a watchdog of e.g. 2 ms, the reserve is $\geq 50\%$. The load of the 1ms task is not obvious in this case.

The load of the 10ms task is a maximum of 2 ms. In case of a watchdog of e.g. 10 ms, the reserve is 80%.

The load of the 50ms task is a maximum of 27 ms. In case of a watchdog of e.g. 50 ms, the reserve is 45%.

Measuring the idle time (P-0-1364)

The PLC function "MX_fGetFreeTicks" or parameter P-0-1364 is used to read the PLC idle time. The PLC idle time displays the unused computing time in the PLC time slot.

In a typical constellation of fast and slow time-controlled tasks (without free-wheeling task), the cyclic display of the idle time (with the oscilloscope, for example) provides a rough overview of the load of all tasks.

Extended runtime measurement

If necessary, the runtime of the PLC tasks can be measured more precisely. This is particularly interesting for fast tasks (in a range of one to a few milliseconds) which must always be processed in their periodic time.

The time slot method is taken into account when measuring. The runtime of the task is compared to its maximum possible runtime (according to the periodic time). The result is the load of each task in percent. When the computing time of a high priority task is increased, both its load and the load of the low priority tasks are increased, since their available computing time is reduced. Interrupts, such as current or velocity controllers, are not taken into account.

The measurement does not always take place, but can be controlled in the PLC project by using the corresponding function block "MX_IECTaskGetLoad" from the "MX_PLCOpen" library. This function block provides the values of the specified task. Basically, the function block can be called in any task. (It is useful to call it in a low-priority task, unless you want to watch every cycle.) Thus, the measurements are activated after the project containing the function block has been loaded. When another project is loaded that does not contain the function block, the function is deactivated.

In the figures below, the runtimes and maximum runtimes of three tasks (as above) were measured using the function block "MX_IECTaskGetLoad" and shown as bars (yellow: current load, red below: maximum load).

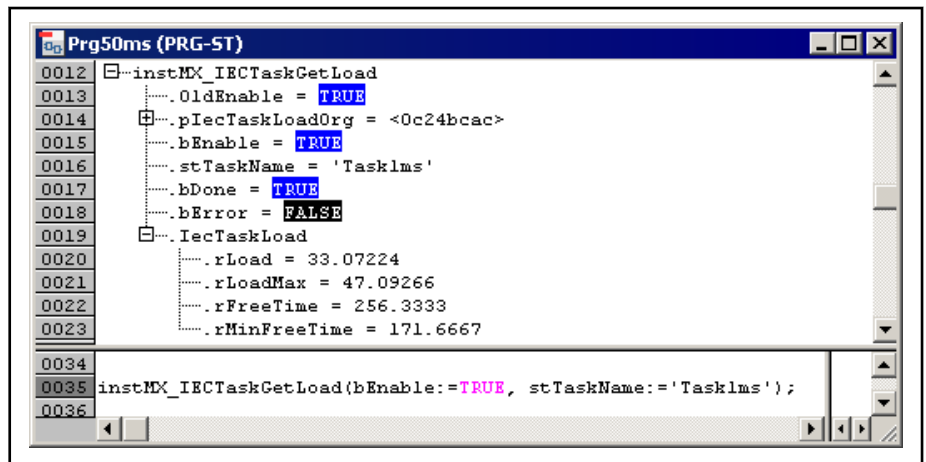


Figure name DB000346

Fig. 4-21: Example of a runtime measurement

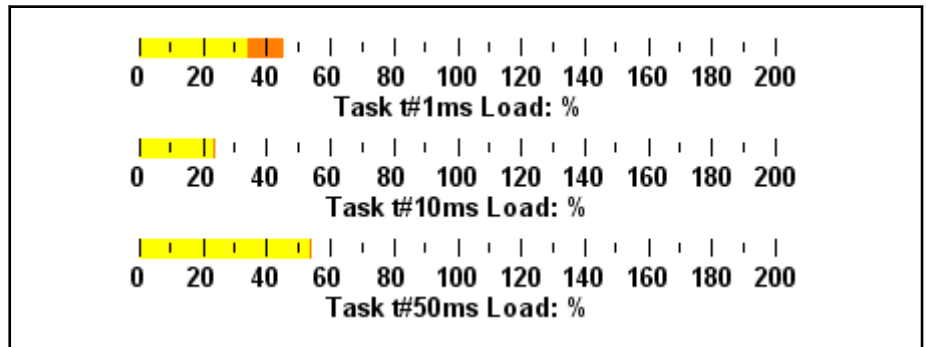
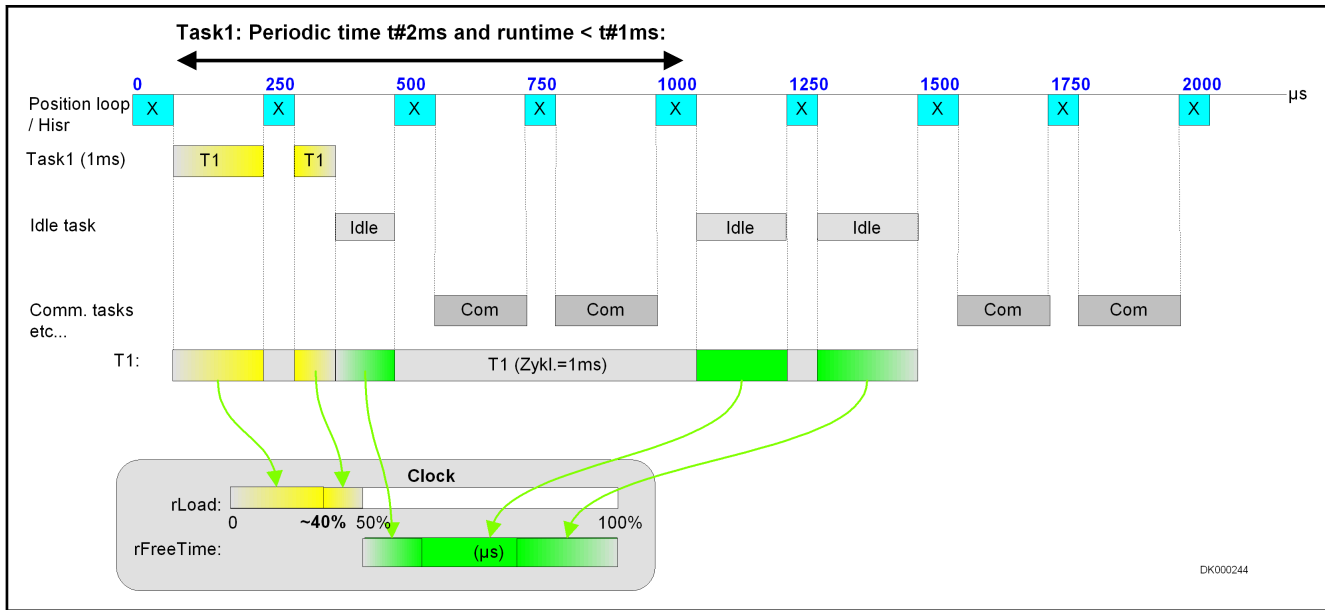


Figure name DB000347

Fig. 4-22: Bar chart of runtime measurement

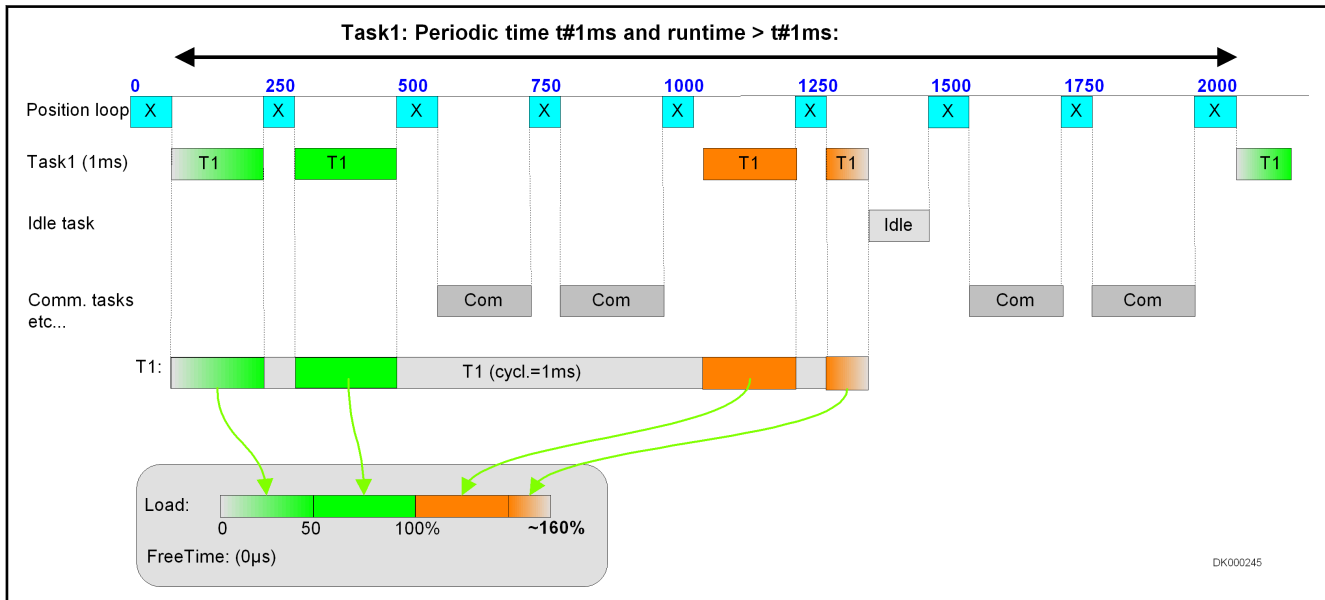
The following two examples show how the load display is determined. Drive control in this case works in advanced performance with a position controller clock rate (cycle time) of 250µs.

Basic functions of Rexroth IndraMotion MLD



X Position controller
Com PLC communication task
rLoad Used computing time
rFreeTime Remaining unused computing time
Fig. 4-23: Task runtime measurement, "underload"

The above figure shows a measurement of a task with a periodic time of $t\#2\text{ms}$. The task only consumes a small part of the calculating time. Note that the 100% mark corresponds to the total computing time available for this task. The computing time actually used is compared to the 100% time and displayed as percentage value at the "rLoad" output. The remaining unused computing time is displayed as a time value at the "rFreeTime" output in μs .



X Position controller
Com PLC communication task
rLoad Used computing time
rFreeTime Remaining unused computing time
Fig. 4-24: Task runtime measurement, "overload"

The above figure shows a measurement of a task with a periodic time of $t_{\#1\text{ms}}$. In this case, the task consumes more than the given interval time. The 100% mark again corresponds to the net time intended for the task within one millisecond. Here the task requires more time, so the measured percentage value at the "rLoad" output displays more than 100%. The "rFreeTime" output displays 0 (μs).

4.5.8 Task configuration

Creating a cyclic or freewheeling task is required to allow for an I/O update. If no such task is defined, an error message is displayed during compilation of the MLD project and download to the control unit is not possible. In a created task, a corresponding program has to be called; otherwise, a warning will be displayed during compilation of the MLD project.

4.5.9 Task cycle times and timing

PLC cycle time The PLC cycle time T_{PLC} of MLD defines the periodic time for cyclic tasks. The T_{PLC} consists of one or several PLC time slices.

PLC time slice The integrated PLC (IndraMotion MLD) works cyclically in time slices, regardless of the task type used. Since the integrated PLC is calculated by the drive processor, it must share the resources with the rest of the system [position controller, velocity controller, current controller, different background tasks (such as command task), etc.]. Depending on the system load, a certain computing time is available to the MLD for each PLC time slice. All PLC IEC tasks run in the context of PLC time slices.

Minimum PLC cycle time The minimum PLC cycle time ($T_{\text{PLC,min}}$) is one PLC time slice; it also defines the reaction time of the event tasks. The minimum cycle time ($T_{\text{PLC,min}}$) does not depend on the control unit and is 1 ms.



The maximum configurable cycle time is 2000 s (or 33 minutes)!

Event task reaction times See also Functional description of firmware "Performance data"
Event tasks are also started within the PLC time slices, i.e., their reaction times also depend on the PLC cycle time.

Timing The attainable program cycle time or timing depends on the integration of the PLC in the task system of the drive.

The performance of devices with MPB-18 and MPC-18 firmware can be modified by the setting in P-0-0556:

- P-0-0556, bit 2="0" $\Rightarrow$ Basic performance
- P-0-0556, bit 2="1" $\Rightarrow$ Advanced performance



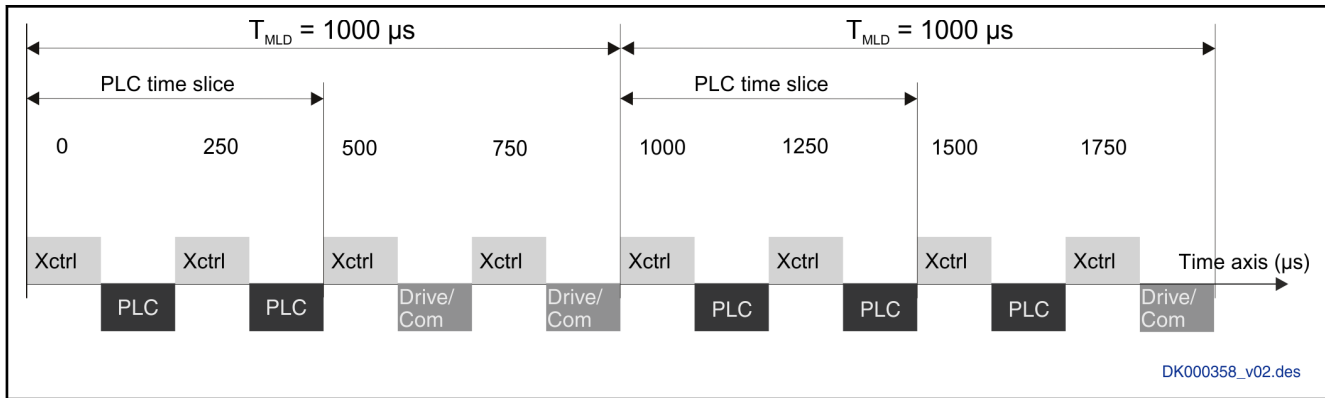
If the "MLD-M system mode" has been configured in the MPC firmware (P-0-1800.0.1), the "Advanced" performance level is unavailable.

Performance [μs]	Current controller clock ($T_{A\_current}$)	Velocity controller clock ($T_{A\_velocity}$)	Position controller clock ($T_{A\_position}$)	PLC cycle time (IndraMotion MLD) (T_{MLD})	Master communication cycle time (T_{MastCom})
Basic	125	250	500	1000	500
Advanced	62.5	125	250	1000	250

Tab. 4-7: Performance settings (performance level)

The figures below illustrate the functional principle (interruptions by velocity controller, current controller, etc., are not shown):

Basic functions of Rexroth IndraMotion MLD



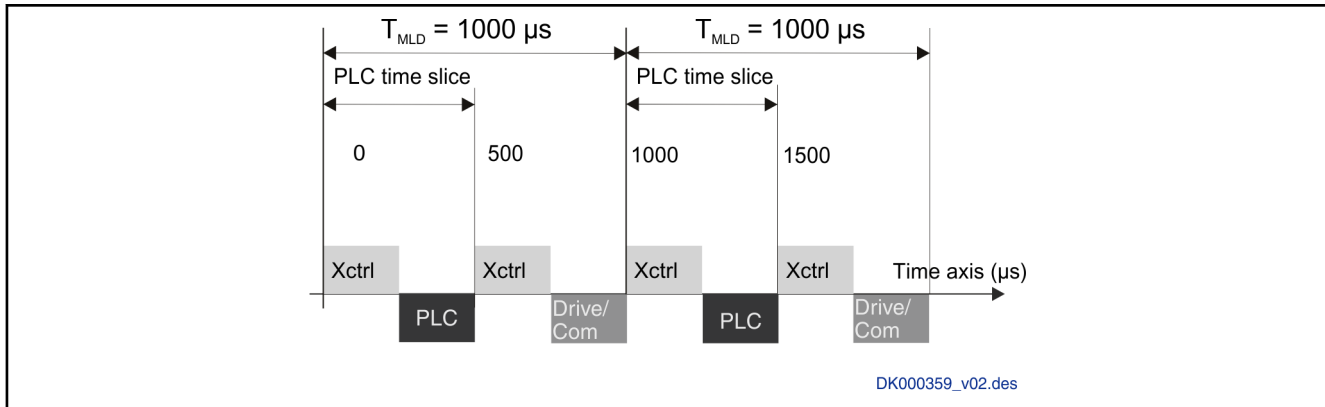
XREG Position controller
PLC All PLC tasks (IEC tasks) (freewheeling tasks, periodic tasks, event tasks)
Drive/Com Communication and management task
Fig. 4-25: Advanced performance (position controller every 250 µs)



When the drive is operated in advanced control, the PLC time slice is also interrupted by the position controller and not only by the velocity and current controller.



Employing the "Advanced performance" label (P-0-0556, bit 2) is not possible with the firmware MPB-18V08 and above, if "IndraMotion MLD" is active [expansion packages "ML", "MA" or "TF" (with MPx-20 and above)].



XREG Position controller
PLC All PLC tasks (IEC tasks) (freewheeling tasks, periodic tasks, event tasks)
Drive/Com Communication and management task
Fig. 4-26: Basic performance (position controller every 500 µs)

4.6 IndraMotion MLD error handling

4.6.1 IndraMotion MLD operating behavior

Behavior when program is modified or during downloads

MLD(-S) behavior A PLC program can be loaded, even if the axis is running. For this purpose, the axis is switched to "AH" (for STOP) or "AB" without drive enable (for re-

set) in addition to the outputs being set to a safe state (FALSE) when a program is modified or downloaded in MLD(-S).

MLD(-M) behavior

When a program is modified or downloaded, the local axis is switched to "AH" or "AB" when using MLD-M, as long as the PLC has control. Remote axes are also switched to "AH" or "AB" when in control (via MLD-M system mode). In addition, a message box is displayed on the screen before the download dialog if an axis is under MLD control. This message indicates that the axis (axes) is (are) stopped during the following download (not Online Change).

Messages when logging in

If a program was modified, a dialog displays during login, providing you with several options:

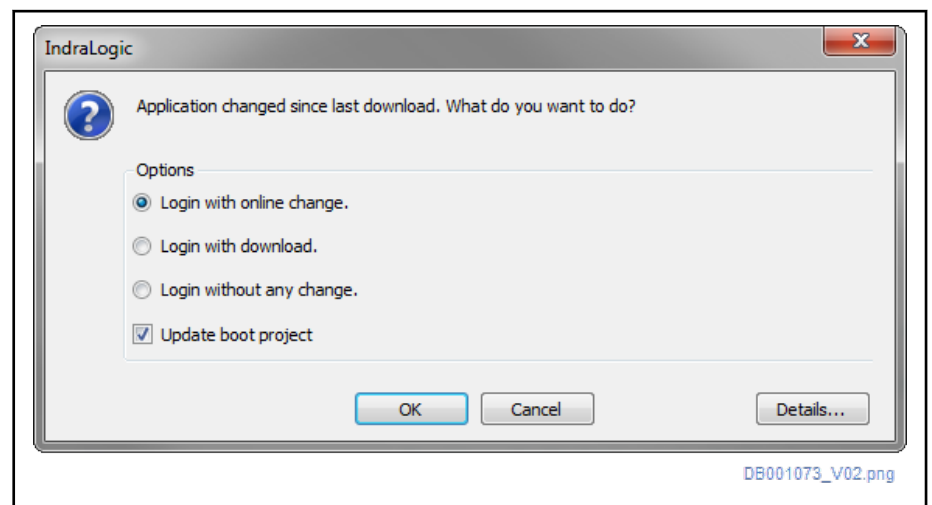


Fig. 4-27: Messages when logging in

4.6.2 General (for MLD-S and MLD-M)

General information

This section describes all general functions regarding how MLD-S and MLD-M react to errors. MLD-S and MLD-M specifically are described in separate sections.

Brief description and overview



An overview of all possible PLC function block errors is provided in the "Error reference lists" section in the library description of MLD-2G.

Functional features

MLD has been integrated in the error handling of the drive firmware. Fatal system errors are handled by the drive firmware. In this case, the drive implements a defined handling method. Other system errors caused by PLC programming are handled in the PLC.

Errors handled in the drive

- The drive reacts to fatal exception errors, such as incorrect address access or floating point exception. There are defined reactions to these errors (see "Troubleshooting Guide" documentation, chapter "Behavior in the case of fatal system errors").
- Drive errors and drive warnings can be recognized at any time via the diagnostic drive message (cf. S-0-0390, S-0-0095, etc.).

	• A warning or an error is generated from a PLC program that can be triggered using simple function calls: – 7 freely definable PLC warnings (E2011 .. E2017) – 7 freely definable PLC errors (F2011 .. F2017)
Pertinent function blocks/functions	The following function blocks are used in conjunction with error handling: • MX_fSetDriveWarning • MX_fSetDriveWarningText • MX_fGetDriveWarning • MX_fSetDriveError • MX_fSetDriveErrorText
Pertinent diagnostic messages	The following diagnostic messages are used in conjunction with error handling: • E2011 PLC - Warning no. 1 • E2012 PLC - Warning no. 2 • E2013 PLC - Warning no. 3 • E2014 PLC - Warning no. 4 • E2015 PLC - Warning no. 5 • E2016 PLC - Warning no. 6 • E2017 PLC - Warning no. 7 • F2011 PLC - Error no. 1 • F2012 PLC - Error no. 2 • F2013 PLC - Error no. 3 • F2014 PLC - Error no. 4 • F2015 PLC - Error no. 5 • F2016 PLC - Error no. 6 • F2017 PLC - Error no. 7

Errors handled in the PLC

Errors that can be caused by faulty programming

Errors that can be caused by faulty programming are so-called PLC exception errors ("Exceptions"). The following errors are handled:

- Field access outside of the limits for arrays
- Incorrect access to subrange types
- Division by zero
- Watchdog (see "Task configuration" in IndraLogic)
- Incorrect pointer access

The reaction to these errors is defined:

- Drive error F6010 is generated, shutting down the axis according to parameterization.
- Display by an error bit in the PLC status (P-0-1351). A higher-level control unit can thereby react to an MLD-S error in a specific way.
- All outputs go to the safe state and the PLC goes to STOP when so defined (default setting) in **PLC settings ► Behavior for outputs in Stop**.



Alternatively, "Keep current values" or "Execute program" can be configured.

- The affected task is immediately aborted.
- All PLC variables in the affected task remain frozen to facilitate debugging.
- Other tasks run to their end and are not called any more (like "PLC STOP"). If a task does not reach its natural end in double its cycle time, it is aborted.
- When the PLC has control over the drive, the drive is switched off (internal system control ON).
- The error is also displayed on next login.
- The PLC cannot be set to RUN any more -> an error is signaled on the first attempt. The RUN state of the PLC then is only possible after a warm restart of the PLC.

Troubleshooting compilation errors

The easiest way to troubleshoot compilation errors is with a connected (even subsequently) programming system. Errors (as well as warnings and notifications) are displayed in the "Messages" window with a short description. Double-clicking a message leads to the point in the source text which caused the message.

Troubleshooting runtime errors

Runtime errors can be read in plain text in parameter "P-0-1365, PLC error message".

If no programming system is available, the error can be located offline by reading the text of "P-0-1365, PLC error message".

See also "[Tracing PLC exceptions](#)"

Errors detected by functions or function blocks

The individual function blocks have outputs which can be used for error handling:

- "Error": An error bit (BOOL)
- "ErrorID": An error number (ENUM type, few values)
- "ErrorIdent": An error code with extended error information

"Error"

The "Error" output is BOOL type. The rising edge of "Error" signals that an error has occurred while processing the function block.

"ErrorID"

The "ErrorID" output is ERROR_CODE type. When an error has occurred at the function block, the error classification can be seen at the "ErrorID" output.



The ERROR_CODE type is defined in the "CommonTypes" library of each device.

The table below shows the possible error codes (ErrorID) and their causes:

ErrorID	Error cause
NONE	No error
INPUT_INVALID	Value outside of programmable range of values, e.g., axis number 30000
INPUT_RANGE	Value outside of currently possible range of values, e.g., axis 5 not configured
RESOURCE	Function not enabled, "MoveRelative" with "Open Loop" or hardware not found
CALCULATION	Calculation error, e.g., overflow in PID controller

Basic functions of Rexroth IndraMotion MLD

ErrorID	Error cause
COMMUNICATION	Communication error, e.g., failure in communication to external devices
STATE_MACHINE	Current state of state machine prevents or prohibits further processing
ACCESS	Access not allowed: Writing to specific parameters in phase 4 or incorrect phase, no enable signal, etc.
OTHER	Error output is TRUE, but error number not set
SYSTEM	System error, e.g., operating system: memory management. Firmware error, e.g., specific block instantiated several times or too many real-time variables used

Tab. 4-8: Standard function block error codes and causes

"ErrorIdent" The "ErrorIdent" output facilitates detailed error diagnostics. The values of the output are ERROR_STRUCT type, which consists of the following elements:

- **"Error.Table"**: Enumerator, 15 bits with reservations per system and user range; characterizes the "error table" from which error numbers are entered in "ErrorAdditional"
- **"Error.Additional1"**: DWORD, with different assignment according to "ErrorTable", e.g., Sercos errors
- **"Error.Additional2"**: DWORD, according to "ErrorTable" as additional error information, if available

Element	Element data type	Description
Table	ERROR_TABLE	The value of the "Table" element contains the information with which error table (IndraDrive, Sercos or MLD-S) the error code of the "Additional1" element has to be interpreted
Additional1	DWORD	The "Additional1" element contains the error code for precisely specifying the error that has occurred
Additional2	DWORD	"Additional2" contains additional information (if available)

Tab. 4-9: Output structure "ErrorIdent" of ERROR_STRUCT type



The structure and its elements have been defined in the "CommonTypes" library and have the default value of "0".

Notes on application and programming

Resetting errors in MLD

Errors can be reset in MLD as follows:

- Via the function block "MC_Reset" at all axes, individually for each axis.
- Via the function block "MB_ClearAllError" at all axes and I/O modules
- or -
- via PLC reset (cf. P-0-1350); all axis errors are reset, either only the error flag or by C05 command.



This behavior is compatible with IndraMotion MLC.

Displaying errors

If errors are detected when slaves are commanded with motion function blocks, they are displayed by device in the commanding device, i.e., the MLD-M master. If an error occurs in the slave, it is primarily displayed in the slave.

Suppressing PLC error messages when booting up

Proceed as follows to prevent a PLC boot project from starting automatically in order to avoid unwanted PLC error messages:

- Before "Boot 2.9" appears on the control panel, press and hold the <ESC> and <ENTER> keys at the control panel at the same time.
- If the functional package "Motion Logic" (drive PLC and technology functions) has been enabled, the display reads "PLC ?".
- Pressing the arrow keys on the control panel (up arrow/down arrow) switches the display between "Run PLC" and "Stop PLC".
- Pressing <ENTER> on the control panel when "Stop PLC" is displayed prevents the PLC boot project from starting.

Triggering PLC warnings

The following PLC warnings can be triggered and reset in the drive by calling the "MX_fSetDriveWarning" function:

- E2011 PLC - Warning no. 1
- E2012 PLC - Warning no. 2
- E2013 PLC - Warning no. 3
- E2014 PLC - Warning no. 4
- E2015 PLC - Warning no. 5
- E2016 PLC - Warning no. 6
- E2017 PLC - Warning no. 7

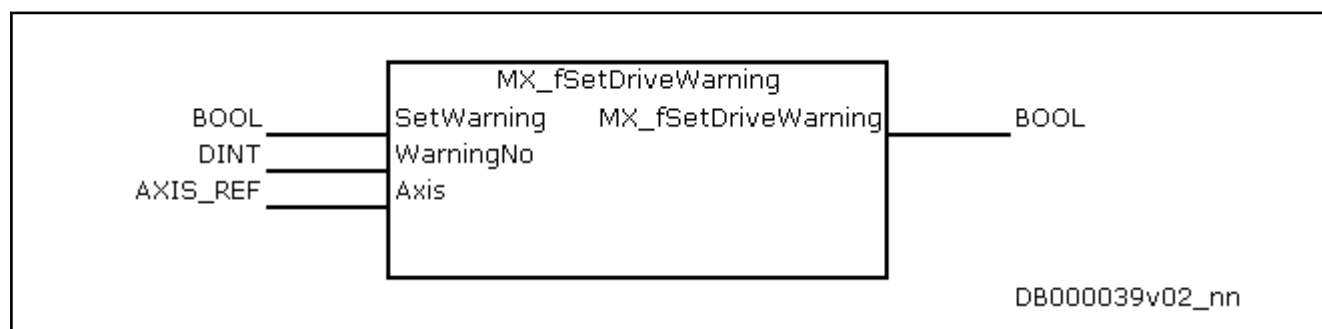


Fig. 4-28: Structure of "MX_fSetDriveWarning"

The active PLC warnings can be read by calling the "MX_fGetDriveWarning" function.

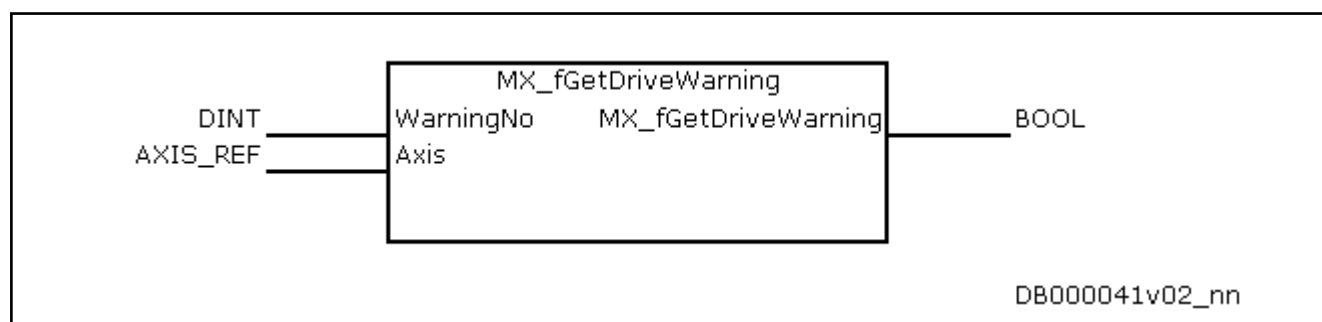


Fig. 4-29: Structure of "MX_fGetDriveWarning"

Defining a text for PLC warnings

A freely selectable text can be defined for the transmitted PLC warnings by calling the "MX_fSetDriveWarningText" function. Switching the drive off sets the diagnostic texts to their default values again.

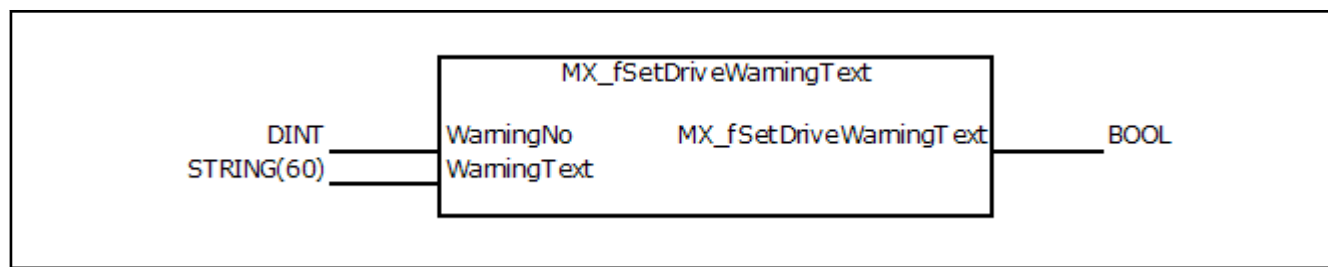


Fig. 4-30: Structure of "MX_fSetDriveWarningText"

Triggering PLC errors

The following drive errors can be triggered by calling the "MX_fSetDriveError" function:

- F2011 PLC - Error no. 1
- F2012 PLC - Error no. 2
- F2013 PLC - Error no. 3
- F2014 PLC - Error no. 4
- F2015 PLC - Error no. 5
- F2016 PLC - Error no. 6
- F2017 PLC - Error no. 7

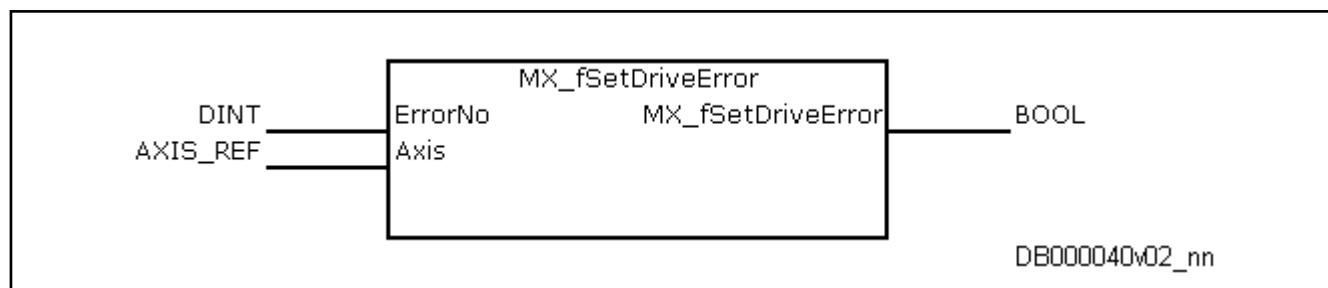


Fig. 4-31: Structure of "MX_fSetDriveError"

Defining a text for PLC errors

A freely selectable text can be defined for the transmitted PLC error by calling the "MX_fSetDriveErrorText" function. Switching the drive off sets the diagnostic texts to their default values again.

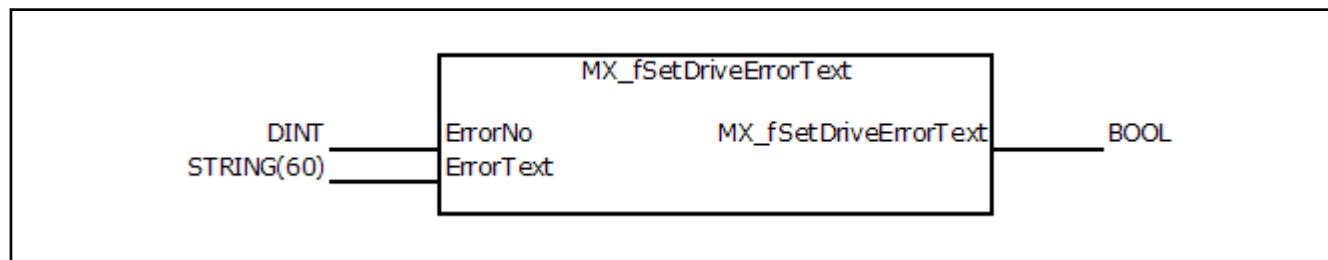


Fig. 4-32: Structure of "MX_fSetDriveErrorText"

4.6.3 Configuring the error reaction in MLD-S

Brief description

An automatic error reaction can be activated for errors detected by motion function blocks.

Configuring the error reaction for MLD-S

The automatic reaction to "function block errors" can be activated/deactivated in the PLC configuration:

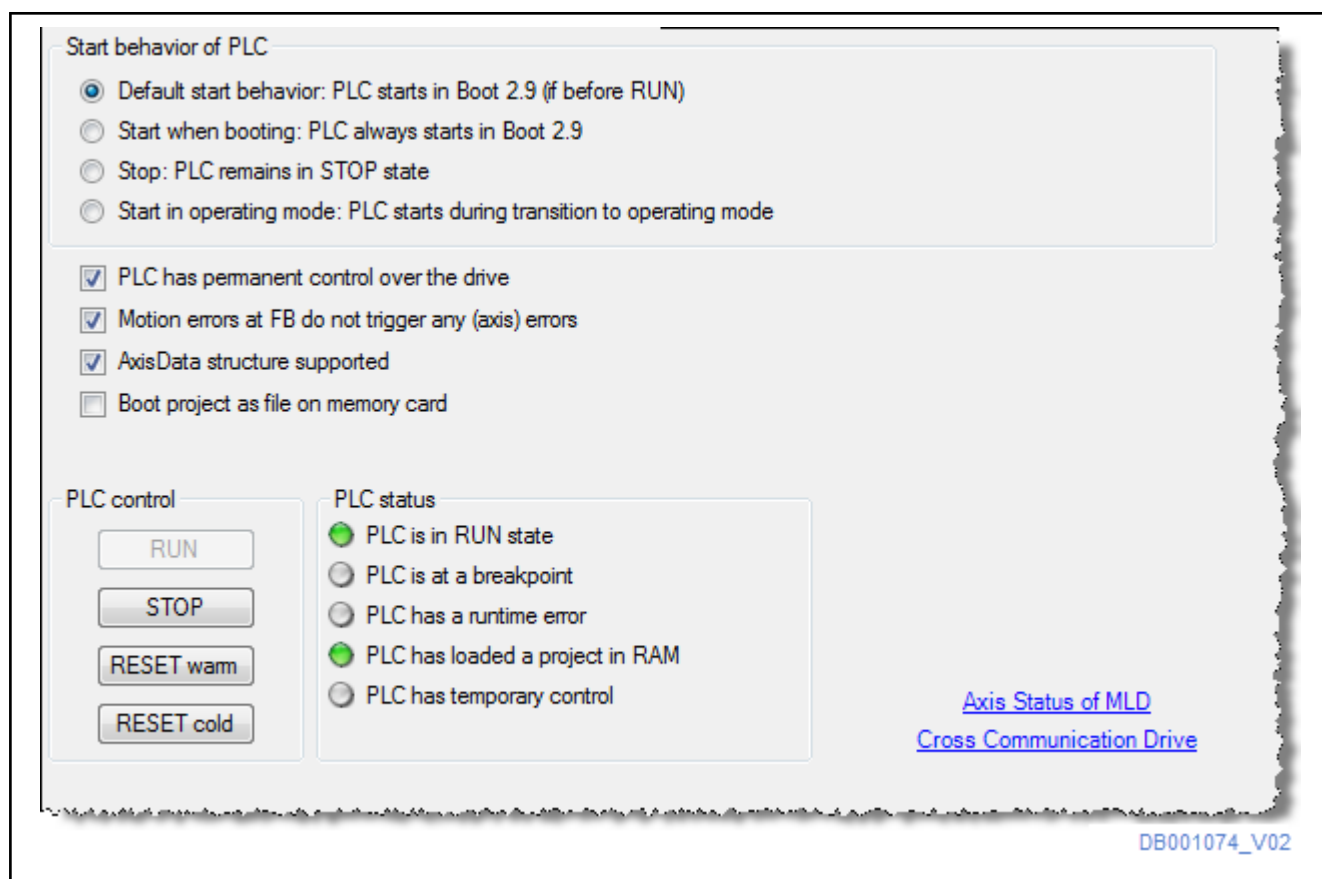


Fig. 4-33: Activating/deactivating automatic reaction to "function block errors"

Error	Reaction (P-0-1367, bit 7="0")
Function block error	F2150

Tab. 4-10: MLD-S error reaction

If the error reaction has been activated, error F2150 is generated when a faulty function block call is detected. The PLC keeps running and the function block signals an error at the Error output.

4.6.4 MLD-M error handling and reaction

Brief description

Note with the error reaction of an MLD-M drive system that there is a digital connection of the axes (cross communication via Sercos III) in addition to any existing module bus connection. This results in various options for the group reacting to an error that can be specifically selected and coordinated with one another. The following instances generally occur:

- With the **stand-alone axis-related error reaction**, all axes in the group react to the error independently if no package reaction or CCD error reaction has been activated.
- **Axis group package reaction** (DC linking): The axes operated in the axis group at a DC bus are interconnected via the module bus and execute a coordinated error reaction in the event of an error (package reaction).
- **CCD/MLD-M error reaction**: For certain applications (e.g., Gantry axes), it can be useful to shut down the entire CCD group in a controlled manner or at least evenly in all axes when an error occurs in a CCD slave or

CCD/MLD-M error reaction features	in the CCD master. This makes it possible to select a CCD error reaction in the CCD master, if necessary. The CCD/MLD-M error handling and reaction have the following features: • Configurable global error reaction for multiple or all axes • Optional automatic group reaction to shut down all axes in case of an error, even without PLC program • Extensive and transparent group reaction using centralized and comprehensive parameterization of the error reaction
Parameters used	The following parameters are used in conjunction with the MLD-M/CCD error reaction: • "P-0-1367, PLC configuration" • "P-0-1800.0.1, CCD: Configuration" • "P-0-1810.0.16, CCD: Axis error" [bit list with one error bit (class 1 diagnostics) for each axis (master and slaves)]
Pertinent diagnostic messages	The following diagnostic messages can occur in conjunction with the MLD-M/CCD error reaction: • "F2150 MLD motion function block error": This error is generated on the master (or with MLD-S) if an error is detected when the master axis is commanded. • "E2140 CCD node error": This warning is generated in the master when a function block error is detected for an axis or when an axis is in error state.

Configuring the error reaction for MLD-M

The following causes result in a function block commanding error:

- **Motion interpreter / motion handler**
 - Axis is in "Stopping" and is to be commanded
 - "MX_SetOpMode" attempts to switch to an invalid operation mode index (free profile)
 - Commanding without enabling of functional package
 - Commanding without torque
 - Commanding without control
- **Motion function block**
 - Concurrent task access to an instance
 - Checking of input values (acceleration, etc.)

Configuring the MLD-M/CCD error reaction

P-0-1800.0.1, bit 8, 7, allows the behavior of the entire CCD axis group (local and remote axis) to be configured in the case of an error. Here, it must be determined whether an error in a CCD slave or in the master triggers the group reaction. The following overview shows the possible cases:

- **P-0-1800.0.1, Bit8="0", Bit7="0": Error reaction in case in case of function block errors**
 - There is no reaction to a function block error in the affected or any other axis.
 - An error in the slave is not displayed in the CCD master.



If required, a specific error reaction can be programmed in MLD using the bit list of "P-0-1810.0.16, CCD: Axis error" or "AxisData".

- P-0-1800.0.1, Bit8="0", Bit7="1": **Error reaction in case of function block errors**
 - When an axis or function block commanding error occurs, only the affected axis carries out its selective error reaction, while all other axes remain in control.
 - Warning E2140 is displayed in the CCD master.
 - Function block commanding errors cause the affected slave axis to be decelerated with "RfAus¹⁾"!
- P-0-1800.0.1, bit8="1", bit7="0": **Master-controlled synchronous error reaction of the entire CCD/MLD group**
 - When an axis or function block commanding error occurs in an axis, a synchronous and coordinated error reaction for the entire group is carried out in the master (NC reaction).
 - Error F2140 is displayed in the CCD master, and error Fxxxx in each slave axis.
 - All axes are decelerated with the ramp parameterized in the master (cf. P-0-0119).



When the master is not in control (e.g., "Ab"), then warning E2140 is generated in the master instead of error F2140!

- P-0-1800.0.1, Bit8="1", Bit7="1": **Immediate error reaction of the CCD/MLD group**
 - When an axis or function block commanding error occurs in an axis, the best possible deceleration (cf. P-0-0119) is initiated in the master for all axes (local or remote) by deleting bit 15 in the control word (cf. S-0-0134).
 - Error F2140 is displayed in the CCD master, and error Fxxxx in each slave axis.
 - All axes are decelerated with the **ramp parameterized locally** (cf. P-0-0119).



When the master is not in control (e.g., "Ab"), warning E2140 is generated in the master instead of error F2140!

Notes on application and programming



MLD- and CCD error reactions are interconnected, see also Functional Description of firmware "Cross communication (CCD)".

CCD error reaction

Note the following points in regard to the CCD error reaction:

- The error reaction to function block errors is only activated if both of the following conditions are met:
 - The error reaction to function block errors is configured in P-0-1367 (bit 7="0")
 - and**
 - The error reaction to function block errors is configured in P-0-1800.0.1 (bit 8="0", bit 7="1")

¹⁾ "RfAus": Drive enable is switched off

Basic functions of Rexroth IndraMotion MLD

- In the MLD-M system mode, the diagnostic message number (cf. S-0-0390) is automatically configured in the AT (drive telegram) of all CCD slaves, meaning the axes that have triggered an error cannot be selectively chosen.
- In CCD basic mode or CCD system mode, the error number can be selectively configured in the AT (drive telegram) in order to exclude specific axes from automatic error generation.



With MLD-M, the remote axes have to be taken into account in addition to how the local axis handles errors.

Configuring in IndraWorks

The error reaction is configured via the following IndraWorks dialog:

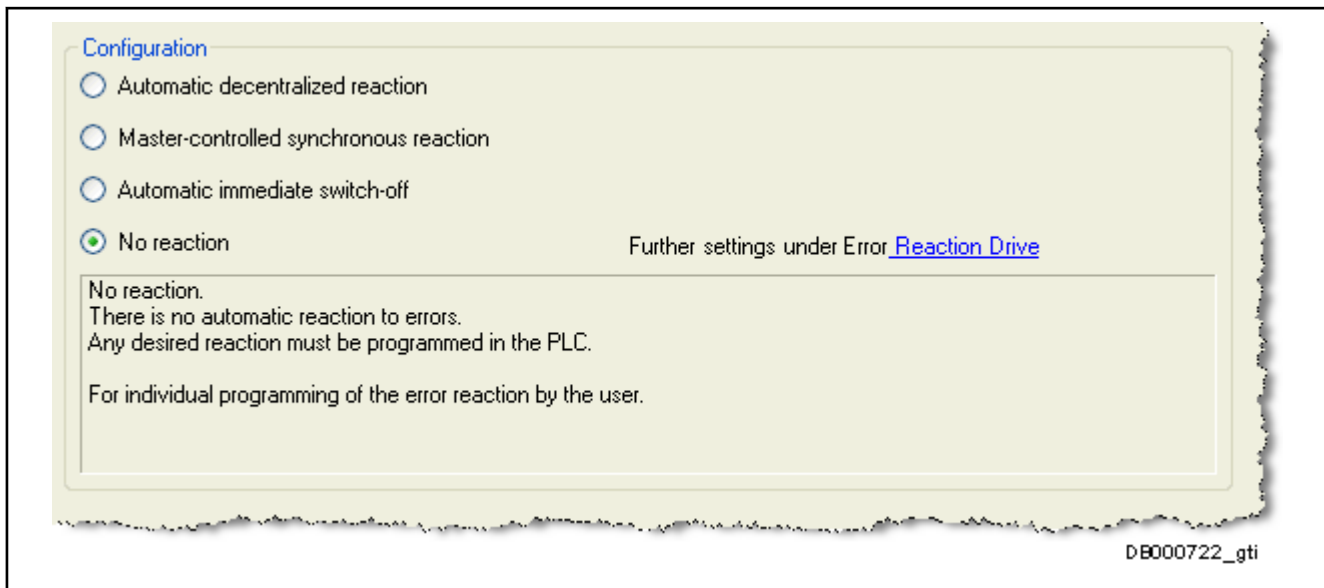


Fig. 4-34: IndraWorks dialog: CCD/MLD- error reaction

Configuring via direct parameter access

Independent of the dialog, it is also possible to directly edit parameter "P-0-1367, PLC configuration" and use bit 7 "deactivate function block error reaction" to select the reaction to a commanding error:

- Bit7="1": Function block commanding errors are ignored
- Bit7="0": Function block commanding errors are handled according to the table below

P-0-1800.0.1	P-0-1367, bit7="0"	Error location	Error event	Master	Slave1	Slave2	
Bit7="0" Bit8="0"	only MLD	Master	FB error	AF	AF	AF	No reaction No automatic reaction to errors. Any desired reaction must be programmed in the PLC. For individual programming of the error reaction by the user.
		Slave1	FB error	AF	AF	AF	
		Slave2	FB error	AF	AF	AF	
	with MLD or CCD	Master	FnXXX	FnXXX	AF	AF	
		Slave1	FnXXX	AF	FnXXX	AF	
		Slave2	FnXXX	AF	AF	FnXXX	
Bit7="1" Bit8="0"	only MLD	Master	FB error	F2150	AF	AF	Automatic decentralized reaction A warning is displayed in the master for F errors in the axis. The axis is decelerated with "RfAus" ¹⁾ and a warning displayed in the master for function block commanding errors. For simple applications in which only the affected axis is to be decelerated.
		Slave1	FB error	E2140	RfAus	AF	
		Slave2	FB error	E2140	AF	RfAus	
	with MLD or CCD	Master	FnXXX	FnXXX, (E2140)	AF	AF	
		Slave1	FnXXX	E2140	FnXXX	AF	
		Slave2	FnXXX	E2140	AF	FnXXX	
Bit7="0" Bit8="1"	only MLD	Master	FB error	F2150, (E2140)	AF	AF	Master-controlled synchronous reaction The master reacts to all errors (also function block errors) with an error and the configured error reaction. All slaves (without F errors) remain in control to continue following the master. For synchronous group with master-controlled synchronous error reaction.
		Slave1	FB error	F2140, (E2140)	AF	AF	
		Slave2	FB error	F2140, (E2140)	AF	AF	
	with MLD or CCD	Master	FnXXX	FnXXX, (E2140)	AF	AF	
		Slave1	FnXXX	Fn140, (E2140)	FnXXX	AF	
		Slave2	FnXXX	Fn140, (E2140)	AF	FnXXX	
Bit7="1" Bit8="1"	only MLD	Master	FB error	F2150, (E2140)	RfAus	RfAus	Automatic immediate shutdown All axes are decelerated with "RfAus" ¹⁾ in the case of any errors (except for F axis errors). This means that each axis directly carries out its error reaction. The error is displayed by a warning in the master. For non-synchronous multi-axis applications with automatic complete error reaction.
		Slave1	FB error	F2140, (E2140)	RfAus	RfAus	
		Slave2	FB error	F2140, (E2140)	RfAus	RfAus	
	with MLD or CCD	Master	FnXXX	FnXXX, (E2140)	RfAus	RfAus	
		Slave1	FnXXX	F2140, (E2140)	FnXXX	RfAus	
		Slave2	FnXXX	F2140, (E2140)	RfAus	FnXXX	

1) "RfAus": Drive enable is switched off

Tab. 4-11: MLD-M reaction to function block commanding errors

Special cases of MLD-M error reaction

Note the following special cases in regard to MLD-M error reaction:

- When an axis, in the case of a function block error, does not carry out any error reaction, the status in the PLCopen State Machine remains unaffected.
P-0-1810.0.16 can be used to detect a function block error.
- Detecting function block errors can be completely deactivated via "P-0-1367, PLC configuration", bit 7, so they are only visible at the function block output.
- E2140 is always generated in the master when an axis is in error (P-0-1810.0.16 != "0") and the CCD configuration is not "No reaction".

Basic functions of Rexroth IndraMotion MLD

Warning E2140 may be covered by an error F2140, but the information can always be seen in the PLC status register.

- When the master is not in control ("Ab"), warning E2140 is generated in the master instead of error F2140.

Note the following points when selecting the error reaction (cf. P-0-1800.0.1, bit8,7):

P-0-1800.0.1 , bit8,7	Function	Instructions for use	Notes on parameterization
Bit7="0" Bit8="0"	No reaction to function block errors	Useful for error reaction programmed by motion control (e.g., all axes decelerate synchronously controlled by virtual master axis generator, which is stopped by control unit in the event of an error).	If required, a specific error reaction can be programmed in MLD using the bit list of "P-0-1810.0.16, CCD: Axis error" or "AxisData".
Bit7="0" Bit8="1"	Error reaction to function block errors	When an axis or function block commanding error occurs, only the affected axis carries out its selective error reaction, while all other axes remain in control.	
Bit7="1" Bit8="0"	Master-controlled synchronous error reaction of the entire CCD/MLD group	Useful for automated error reaction with synchronous group of CCD slaves to the master motion of the CCD master (e.g., Gantry axes with CCD slave follow actual position value of master, etc.)	1. When this error reaction is active, parameter "P-0-170x, CCD: Diagnostic message number, slave" is cyclically evaluated. For this purpose, P-0-170x has to be configured in P-0-1805.x.2 for each slave. S-0-0390 has to be entered at the appropriate location in P-0-1805.x.4. 2. Bit 10 can be used to select whether or not P-0-179x and S-0-0390 are configured automatically for all slaves. 3. The NC reaction has to be configured in all slaves (cf. P-0-0117) so they also remain in control in the event of an error and can follow the master command value in order to ensure synchronous reaction when an error occurs in the slave.
Bit7="1" Bit8="1"	Immediate error reaction of the CCD/MLD group	Useful for automated error reaction without complex programming in the motion control (default state).	The class 1 diagnostics error bits (cf. S-0-1135) of the axes in the CCD group are cyclically monitored in the CCD master.

Tab. 4-12: Notes on selecting the error reaction

5 MLD communication interfaces and data channels

5.1 Introduction and overview

- MLD communication interfaces** The PLC integrated in the drive (Rexroth IndraMotion MLD) has communication interfaces to ...
- ... the device-internal (local) axis (drives connected via Sercos III) and its inputs/outputs and parameters.
 - ... the "Rexroth Inline" I/O modules (Sercos III **block** I/O modules, Sercos III **modular** I/O modules).
 - ... an external control unit (e.g., field bus PLC) via a master communication interface (Sercos, field bus, etc.).
 - ... an external operator panel (HMI).
 - ... a third-party device via Ethernet by programming socket communication.

MLD communication interfaces and data channels

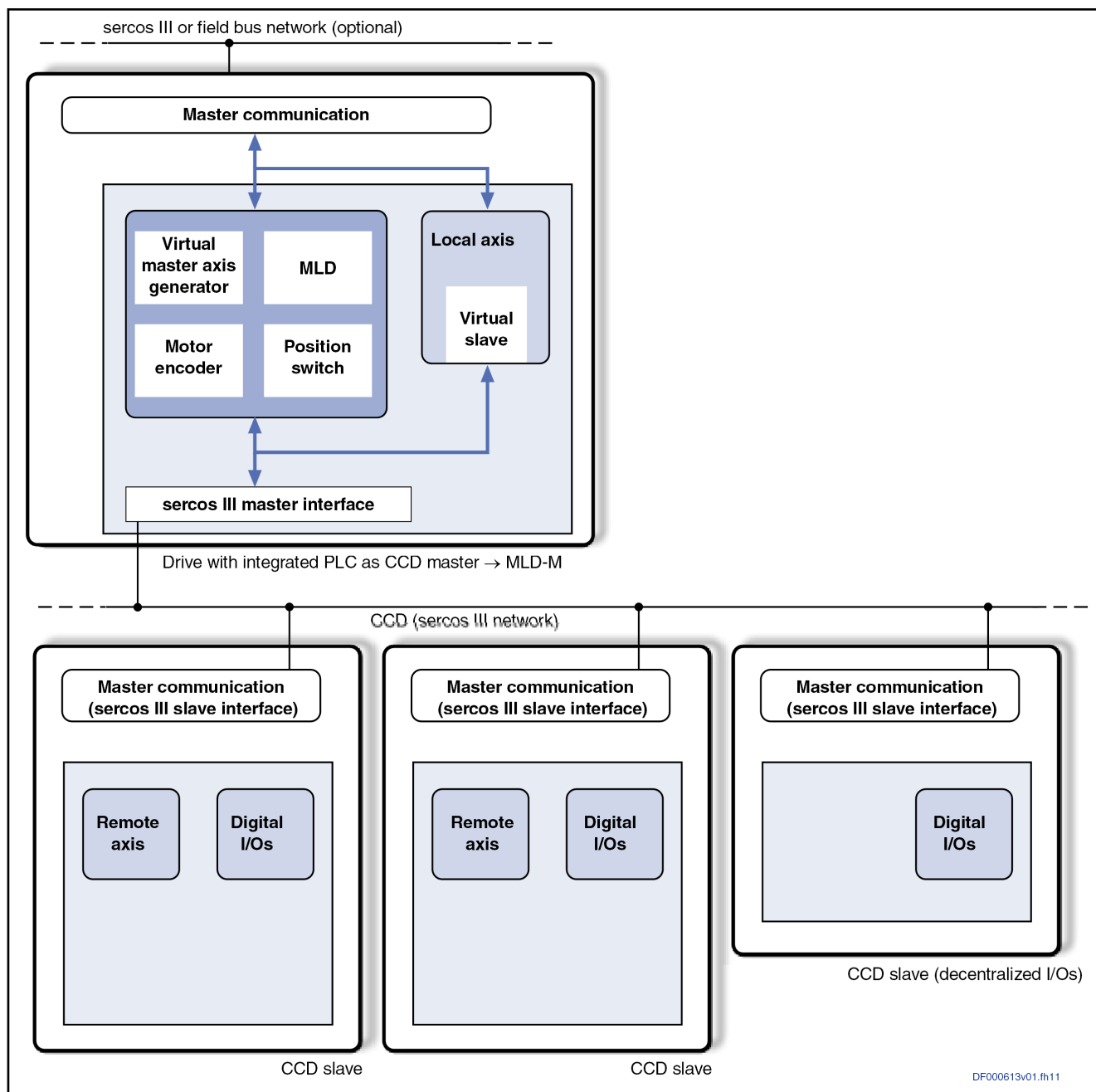


Fig. 5-1: MLD-M logic structure and communication interfaces

IndraMotion MLD data channels

IndraMotion MLD comes with a variety of data channels:

- For **cyclic data exchange** between IndraMotion MLD and axes
- For **acyclic parameter communication** between IndraMotion MLD and axes
- For **connecting** IndraMotion MLD to a **third-party device** via a **defined communications protocol**, **external control panels (HMI)** or to **external control units**

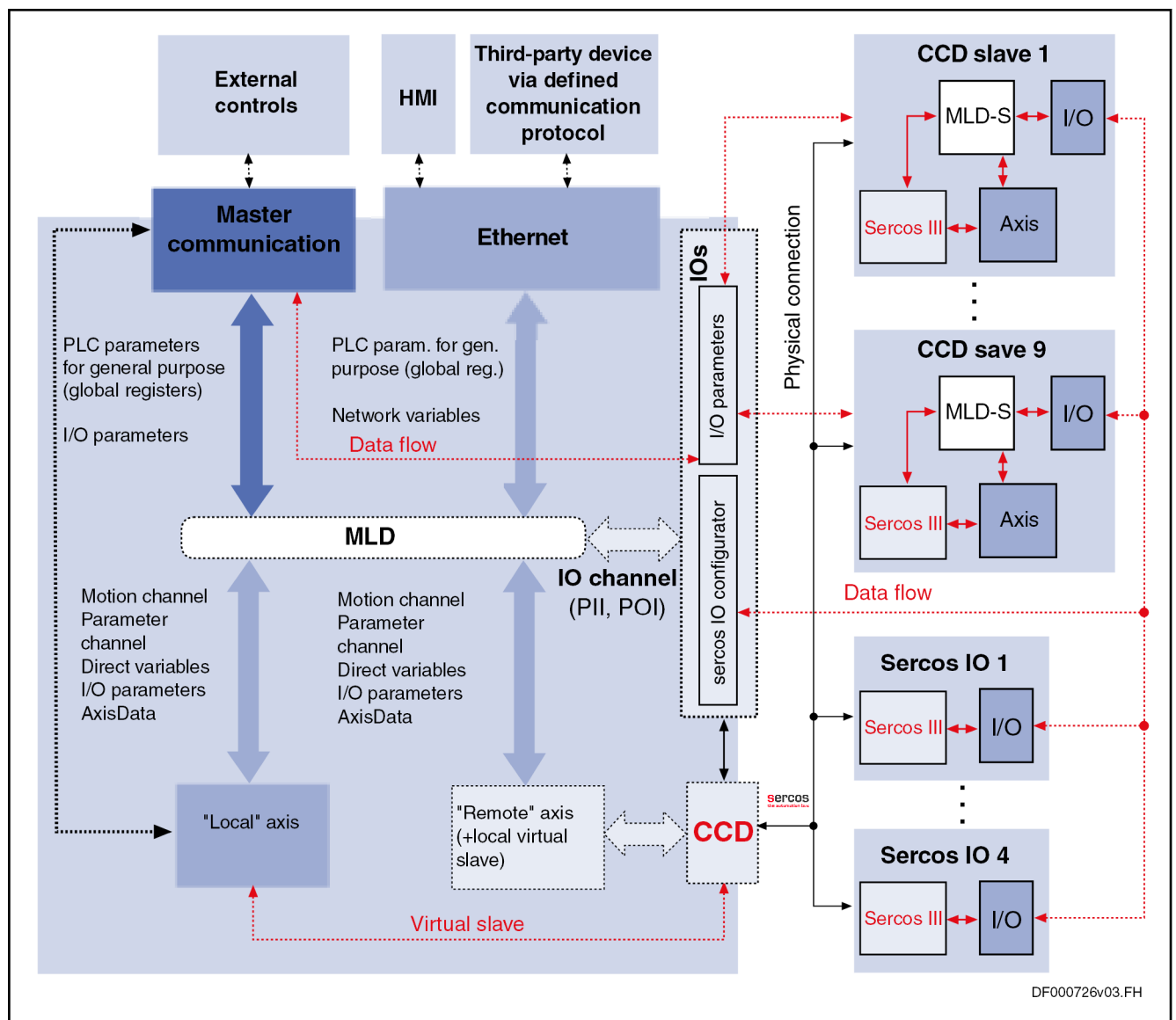


Fig. 5-2: IndraMotion MLD data channels for accessing local and remote axes

The illustration above shows the data channels from the MLD (PLC) using a simplified device model.

5.2 IndraMotion MLD data channels

5.2.1 Cyclic data channels

Introduction

A great number of data channels provides access from the PLC program to drive-internal variables and parameters, or inputs/outputs and sensors evaluated by the drive, or other drive interfaces. Rexroth IndraMotion MLD supports the following data channels:

- **I/O channel (PII, POI)**
Access to analog and digital inputs/outputs of the drive using a process image
- **"AxisData" cyclic axis data**

Cyclic actual and command values are provided using a data structure. This structure is particularly suited when used with single-axis and multi-axis control.



The local real-time channel is no longer available in MLD-2G. Instead, the AxisData interface and synchronized tasks can be used in applications (see also documentation "Rexroth IndraMotion MLD (2G), commissioning as of MPx-18", index entry "Real-time channel").

I/O channel (PII, POI)

The I/O channel is the contact of IndraMotion MLD to external devices, since it allows digital and analog inputs/outputs to be evaluated and addressed.

The process images of the inputs/outputs can be accessed from the program of MLD via variables indicating the corresponding address.

- **Accessing the inputs**
 - The digital and analog inputs are cyclically evaluated by the respective drive function in the position controller clock (see also Functional Description of firmware "Digital inputs/outputs" and "Analog inputs").
 - Before the start of the task, the task system copies the image from the "PLC input" parameters (P-0-1390, P-0-1391, ...) to the process input image (PII).
 - The process image of the inputs (PII) is read in every task cycle at the beginning, i.e., the PII is read to MLD. See also "[Amount of resources \(memory\)](#)" / "[Performance data](#)"
- **Accessing the outputs**
 - The process image of the outputs (POI) is written in every task cycle at the end, i.e., the "PLC output" parameters (P-0-1410, P-0-1411, ...) are changed by the PLC.
 - Writing data to the parameters of the process images is not allowed or rather does not make sense.
 - The digital and analog outputs are cyclically written by the respective drive function in the position controller clock (see also Functional Description of firmware "Digital inputs/outputs" and "Analog inputs").

The following applies to the logic circuit:

- Any change in the logic circuit becomes active when switching to the operating mode.
- After booting, the change in the logic circuit already takes effect at the start of the PLC.



When the PLC is in "STOP", the inputs and outputs are nevertheless operated!

Inputs and outputs which are not used in a task (in the code range) remain unchanged and are not operated.

Input variables which are not used in the program code are not updated.

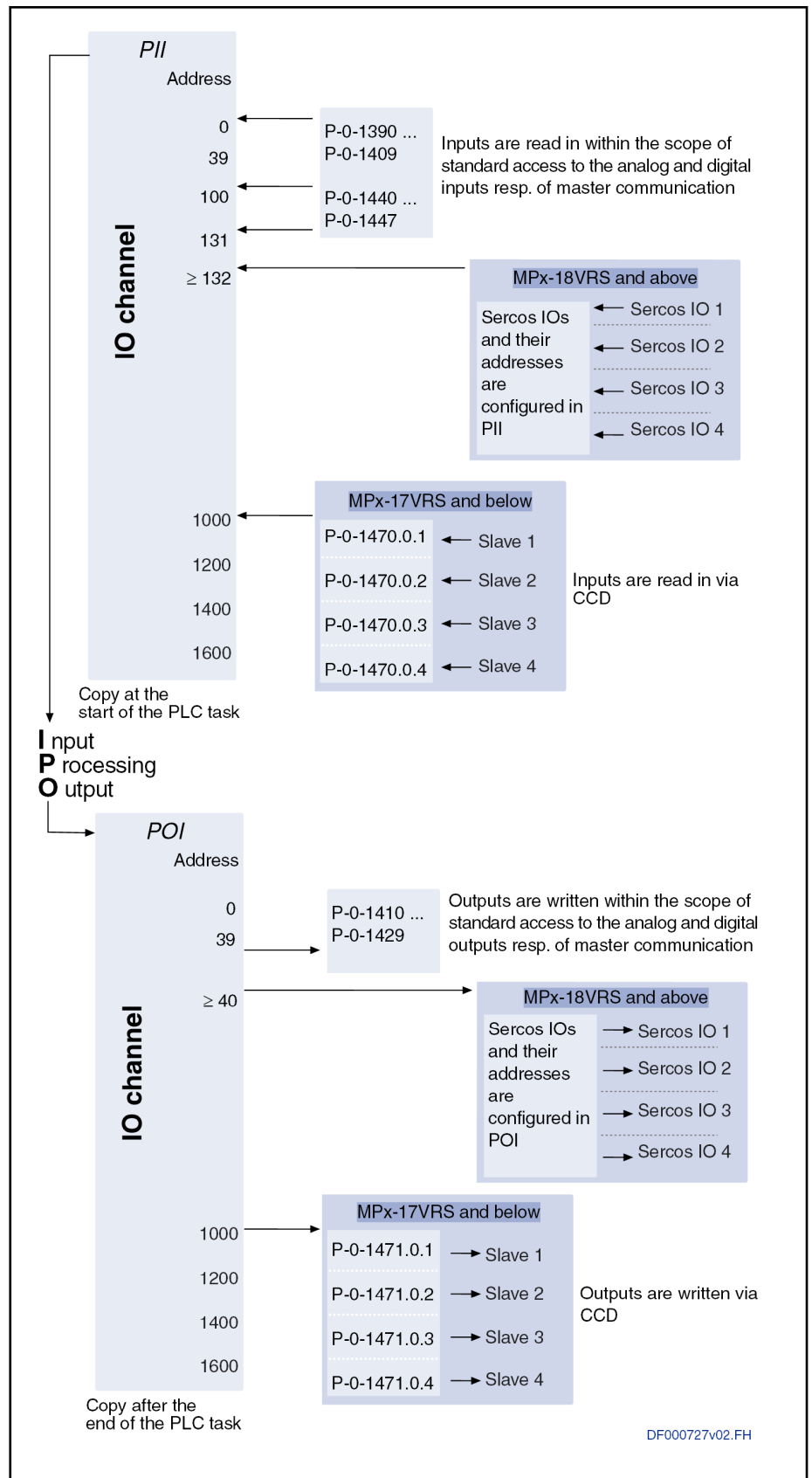


Fig. 5-3: I/O channel (PII, POI)

MLD communication interfaces and data channels

IndraMotion MLD allows accessing all local inputs and outputs of the drive controller.

With IndraMotion MLD, cyclic command values and actual values of a master control unit can be used as inputs and outputs.



"Rexroth Inline" I/Os can also be used in single-axis operation in MLD-M system mode (without other axes). Up to four "Rexroth Inline" I/O modules can be operated.

Features

The I/O channel has the following features:

- Inputs and outputs are in the respective process image.
- There is a collective process image for all tasks.
- The PLC works in byte-oriented form according to IEC 61131.
- Distribution of physical inputs/outputs between drive and PLC is possible via IDNs (parameters).
- Changes in the I/O configuration take effect after switching from parameter mode to operating mode.
- The I/O channel is configured by means of IndraWorks dialogs.
- Inputs and outputs are only updated when they are used in the program.
- Safe values in case the connection is interrupted or in "STOP"
 - Reset of all outputs in "STOP", "RESET" or, if applicable, "Watch-dog" states
 - Reset of all inputs in case the communication is interrupted
- **Updating the process input images**
(PII) max. $T_{PLC}/2$ (500 μ s for ADVANCED) before the PLC task. For inputs read via the master communication or via CCD, the update of the process input images is additionally delayed by the corresponding bus cycle time.
- **Updating the process output images**
(POI) max. $T_{PLC}/2$ (500 μ s for ADVANCED) after the PLC task. For outputs written via the master communication or via CCD, the update of the process output images is additionally delayed by the corresponding bus cycle time.

Parameters involved

The following parameters are used in conjunction with the I/O channel (process image):

- Process input image (PII):
P-0-1390 to P-0-1409, P-0-1440 to P-0-1447 (please observe the availability specified in the parameter description)
- Process output image (POI):
P-0-1410 to P-0-1429 (please observe the availability specified in the parameter description)



For users switching from MLD-1G to MLD-2G:

In MLD-2G, the list parameters P-0-1470.0.x / P-0-1471.0.x are no longer contained in the process images.

When Sercos I/O modules are assigned to a drive which supports MLD-2G (IndraWorks Project Explorer, "Sercos IO" branch), the Sercos I/O modules and their addresses are automatically created in the process images of MLD. This allows adjusting the Sercos I/O modules to the requirements of the installation.

Memory ranges and addressing

The parameters of the process images are word-oriented (inputs additionally long-word-oriented). This allows assigning all inputs and outputs available in the drive for use in the PLC.

The following addresses are reserved for the I/O channel:

- Process input image (PII):
 - The address range %IB0 to %IB131 is reserved for the PII of the local inputs/outputs. The address range cannot be changed and is always existing.
[Parameters P-0-1390 to P-0-1409, P-0-1440 to P-0-1447 (please observe the availability specified in the parameter description)]
 - The address range starting with %IB132 for "Rexroth Inline" I/Os (Sercos IO)
- Process output image (POI):
 - The address range %QB0 to %QB38 is reserved for the POI of the local inputs/outputs. The address range cannot be changed and is always existing.
[Parameters P-0-1410 to P-0-1429 (please observe the availability specified in the parameter description)]
 - The address range starting with %QB40 for "Rexroth Inline" I/Os

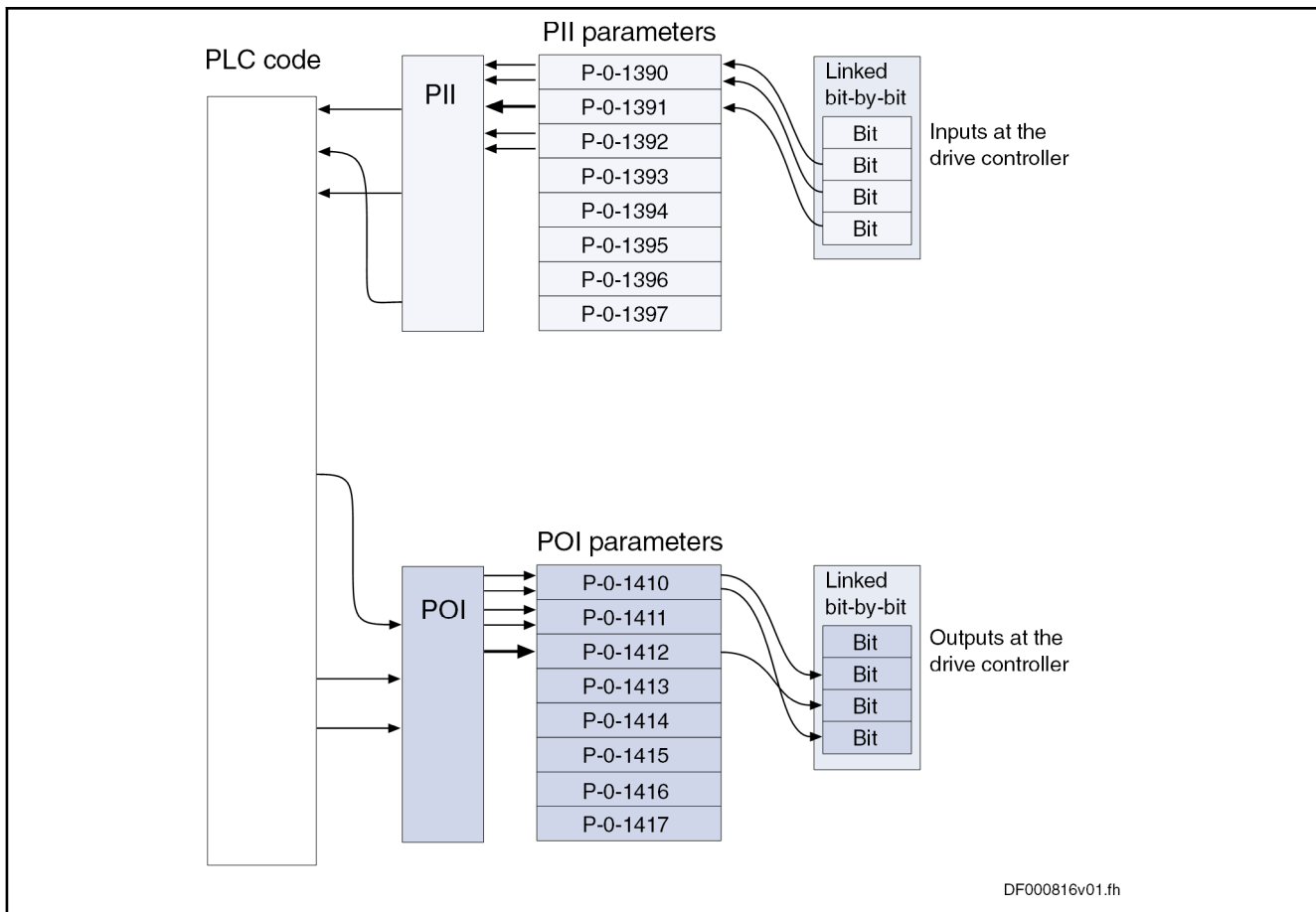
A total of 16 kbytes each.

The examples below explain how the inputs and outputs are addressed in the process image:

Declaration	Format	Process image	Parameters
Var AT %IX0.2 : BOOL	Bit	Bit#2 of word to address 0	P-0-1390, bit 2
Var AT %IX1.9 : BOOL	Bit	Bit#9 of word to address 0	P-0-1391, bit 9
Var AT %IW3 : WORD	2 bytes	Word to address 6	P-0-1393
Var AT %QB2 : BYTE	1 byte	Byte to address 2	P-0-1411 (low byte)
Var AT %QD2 : WORD	2 bytes	Word to address 8	P-0-1414
Var AT %QB8 : DWORD	4 bytes	Long word to address 8	P-0-1414 and P-0-1415

Tab. 5-1: Examples of addressing inputs and outputs in the process image

MLD communication interfaces and data channels



PII: Process input image
POI: Process output image

Fig. 5-4: Example of using the device inputs and outputs

Available device inputs/outputs

"IndraDrive Cs" drive controllers provide the following inputs/outputs:

Analog inputs

- 1 analog input (12 bits, +/-10 V, $T_A = T_{\text{position}}$)

Digital inputs/outputs

- 7 digital inputs (24 V, $T_A = T_{\text{position}}$)
- 1 switchable digital input/output (24 V, $T_A = T_{\text{position}}$)

The PLC can access the analog and digital inputs/outputs of the drive with the process input image (PII) and the process output image (POI). To do this, the inputs and outputs have to be configured accordingly.

See Functional description of firmware "Digital inputs/outputs", "Analog inputs" and "Analog outputs"



If access takes place via MLD, the physical inputs and outputs are connected to the process images (PII and POI) via the logic circuit of the drive.

Collective use of input/output variables

There is a process image for the inputs and outputs respectively (PII and POI). For variables of collective use, this results in tasks interrupting each other.



Effect in case of multitasking: "read - modify - write".

If an input is used in a task and this task is then interrupted by a higher-priority task using the same input, the process image at this point of time is updated for the high-priority task. The low-priority task then continues to run and reads the possibly new value. This should be taken into account when several tasks of different priority are used.



Writing to the same outputs in a variety of tasks should be avoided!

Start values

The value of the start values depends on the declaration of the variables:

- Without pre-initialization all variables are set to "0"
- With pre-initialization all variables are preset to the corresponding initialization value. Example: "MyOutput AT %QX2.4 : BOOL := TRUE;"

Safe values in "STOP"

In the "STOP", "RESET" or, if applicable, "Watchdog" states, all outputs are "0", the output variables in the process image of the outputs remain unchanged.

Safe values in case connection is interrupted

- In case the connection to the **master control unit** is interrupted, the assigned inputs are set to zero.
- In case the connection to the **slave axes** is interrupted, the assigned inputs are set to zero.
- In case the connection to the **Sercos III I/O devices** is interrupted, the assigned inputs are set to zero.

Processing digital values

For processing digital values it is also possible to use individual bits instead of whole Boolean variables. However, this has an unfavorable effect on the processing rate.

Bit-by-bit processing is possible with all digital inputs/outputs, byte-by-byte processing is only possible with the digital inputs/outputs at the control section and the digital I/O extensions.

Multitasking

If bits belonging to a byte are accessed, it is possible to safely access individual bits in spite of preemptive multitasking (read - modify - write).



Setting and clearing of bits by the PLC cannot be interrupted and therefore are "task-proof"! This however does not apply to external access (e.g., via the programming system)!

"AxisData" cyclic axis data

Brief description

Predefined axis data structures ("AxisData") with several command values and actual values are available for cyclically accessing the local axis and remote axes.

Features

- "AxisData" is an IEC61131 data structure definition which contains some important actual values of the axis. In addition, it contains configurable actual values and command values with which the user can read and, if applicable, write their own cyclic data. This allows easy access to the most important axis data.
- The "AxisData" structure is already declared as an "Array of Struct" array across all real axes in the "MX_Base" library. `AxisData : ARRAY[1..10] OF MX_AXISDATA.`



The "AxisData" structure is not accessed via function blocks, but via direct access to the data structures.

Structure of AxisData

The "MX_AXISDATA" structure type is defined in "MX_Base".

Element name	Data type	Direction	Parameter in MLD master	Parameter in axis	Significance
Configurable command value bits					
wUserCmdDataBitA_q	WORD	Command value	P-0-172x.12	(configurable)	Command value bit for free use
wUserCmdDataBitB_q	WORD	Command value	P-0-172x.13	(configurable)	Command value bit for free use
wUserCmdDataBitC_q	WORD	Command value	P-0-172x.14	(configurable)	Command value bit for free use
wUserCmdDataBitD_q	WORD	Command value	P-0-172x.15	(configurable)	Command value bit for free use
Configurable command values					
dwUserCmdDataA_q	SM_TYPES	Command value	32-bit target parameter: P-0-173x 16-bit target parameter: P-0-182x	(configurable)	Command value for free use
dwUserCmdDataB_q	SM_TYPES	Command value	32-bit target parameter: P-0-174x 16-bit target parameter: P-0-183x	(configurable)	Command value for free use
dwUserCmdDataC_q	SM_TYPES	Command value	32-bit target parameter: P-0-175x 16-bit target parameter: P-0-184x	(configurable)	Command value for free use
dwUserCmdDataD_q	SM_TYPES	Command value	32-bit target parameter: P-0-176x 16-bit target parameter: P-0-185x	(configurable)	Command value for free use
Configurable actual value bits					
wUserActualDataBitA_i	WORD	Actual value	P-0-171x.12	(configurable)	Actual value bit for free use
wUserActualDataBitB_i	WORD	Actual value	P-0-171x.13	(configurable)	Actual value bit for free use
wUserActualDataBitC_i	WORD	Actual value	P-0-171x.14	(configurable)	Actual value bit for free use
wUserActualDataBitD_i	WORD	Actual value	P-0-171x.15	(configurable)	Actual value bit for free use
Configurable actual values					
dwUserActualDataA_i	SM_TYPES	Actual value	32-bit source parameter: P-0-177x 16-bit source parameter: P-0-186x	(configurable)	Actual value for free use
dwUserActualDataB_i	SM_TYPES	Actual value	32-bit source parameter: P-0-178x 16-bit source parameter: P-0-187x	(configurable)	Actual value for free use

MLD communication interfaces and data channels

Element name	Data type	Direction	Parameter in MLD master	Parameter in axis	Significance
dwUserActualDataC_i	SM_TYPES	Actual value	32-bit source parameter: P-0-179x 16-bit source parameter: P-0-188x	(configurable)	Actual value for free use
dwUserActualDataD_i	SM_TYPES	Actual value	32-bit source parameter: P-0-180x 16-bit source parameter: P-0-189x	(configurable)	Actual value for free use
Axis status information (normally accessed via state bits or axis status [PLCopen state machine])					
wDriveStatus_i	WORD	Actual value	P-0-166x	S-0-0135	Axis status - the bits are individually available
wDriveExtStatus_i	WORD	Actual value	-	-	Axis status - the bits are individually available
wSignalStatus_i	WORD	Actual value	P-0-171x	S-0-0144	Signal status - the bits are individually available
wSyncStatus_i	WORD	Actual value	P-0-181x	P-0-0089	Synch status - the bits are individually available
dwDeviceStatus_i	DWORD	Actual value	P-0-1630.x.2	S-0-1045	Device status - the bits are individually available
dwPLCopenStatus_i	DWORD	Actual value	-	-	PLCopen-based status - the bits are individually available
dwDiagNumber_i	DWORD	Actual value	P-0-170x	S-0-0390	Diagn. message no.
Fixed actual values					
rActualPosition_i	REAL	Actual value	P-0-167x	S-0-0386	Active position feedback value
rActualVelocity_i	REAL	Actual value	P-0-168x	S-0-0040	Actual velocity value
rActualTorqueForce_i	REAL	Actual value	P-0-169x	S-0-0084	Torque/force feedback value
Fixed state bits					
Axis_CamTab_0	BIT	Actual value	P-0-181x.0	P-0-0089.0	Active cam bit#0
Axis_CamTab_1	BIT	Actual value	P-0-181x.1	P-0-0089.1	Active cam bit#1
Axis_CamTab_2	BIT	Actual value	P-0-181x.2	P-0-0089.2	Active cam bit#2
Axis_CamSwitching	BIT	Actual value	P-0-181x.5	P-0-0089.5	Status distance switching
Axis_CmdValueReached	BIT	Actual value	P-0-171x.0	P-0-0115.12	Command value attained
Axis_Standstill	BIT	Actual value	P-0-171x.1	S-0-0331.0	Standstill message
Axis_InSynchron	BIT	Actual value	P-0-181x.8	P-0-0089.8	Slave axis has been synchronized
Axis_Homed	BIT	Actual value	P-0-171x.2	S-0-0403.0	Status of actual position value reference encoder
Axis_Inbb	BIT	Actual value	P-0-166x, bit (!15 & 14)	S-0-0135 (bit 15 bit 14)	Control section ready for operation
Axis_InAb	BIT	Actual value	P-0-166x, bit (15 & !14)	S-0-0135 (bit15=1)	Control and power sections ready for operation
Axis_Power	BIT	Actual value	P-0-166x, bit (15 & 14)	S-0-0135 (bit15=1 & bit14=1)	Drive with torque

Element name	Data type	Direction	Parameter in MLD master	Parameter in axis	Significance
Axis_FollowingCommand	BIT	Actual value	P-0-166x, bit 3=1	S-0-0135 (bit 3=1)	Drive follows command value input by MLD
Axis_Interrupted	BIT	Actual value	P-0-166x (bit15=1 & bit14=1 & bit13=0 & bit3=0) and P-0-165x (bit13=1)	S-0-0135 (bit15=1 & bit14=1 & bit13=0 & bit3=0) and S-0-0134 (bit 13 = 1)	Motion commanding was interrupted by higher-level drive function
Axis_Warning	BIT	Actual value	P-0-166x.12	S-0-0135.12	Class 2 diagnostics change bit - warning
Axis_Error	BIT	Actual value	P-0-166x.13	S-0-0135 (bit13=1)	Drive error, error in class 1 diagnostics
Axis status (PLCopen state machine)					
Errorstop	BIT	Actual value	-	-	PLCopen status: Error-stop
Stopping	BIT	Actual value	-	-	PLCopen status: Stopping
Homing	BIT	Actual value	-	-	PLCopen status: Homing
DiscreteMotion	BIT	Actual value	-	-	PLCopen status: DiscreteMotion
ContinuousMotion	BIT	Actual value	-	-	PLCopen status: ContinuousMotion
SynchronizedMotion	BIT	Actual value	-	-	PLCopen status: SynchronizedMotion
StandStill	BIT	Actual value	-	-	PLCopen status: Stand-Still
Disabled	BIT	Actual value	-	-	PLCopen status: Disabled
PreSetMode	BIT	Actual value	-	-	PLCopen status: PreSet-Mode

Element name

IEC name of the data element. This is the name used to address the structure element in the program

Data type

IEC data type of the data element

Direction

Command values are written by the user and transmitted by the system

Parameter in MLD master

The master contains a diagnostic parameter for this element. This parameter displays the value but is not writable

Parameter in axis

Source or target parameter on the respective axis. A fixed parameter can be contained here. The parameter can be set for the configurable elements. For the other elements, there is no parameter on the axis.

Tab. 5-2: Structure of AxisData

Configurable contents

The following elements can be assigned to different axis parameters by configuring them accordingly. The AxisData dialog is used for configuration (see also "5-2: Structure of AxisData").

- Configurable command value bits**

It is possible to command up to four freely configurable parameter bits for each real axis with the elements "wUserCmdDataBitA_q" to "wUserCmdDataBitD_q".

- Configurable command values**

It is possible to command up to four freely configurable parameter bits for each real axis with the elements "dwUserCmdDataA_q" to "dwUserCmdDataD_q".

- **Configurable actual value bits**

It is possible to read in up to four freely configurable parameter bits from each real axis with the elements "wUserActualDataBitA_i" to "wUserActualDataBitD_i".

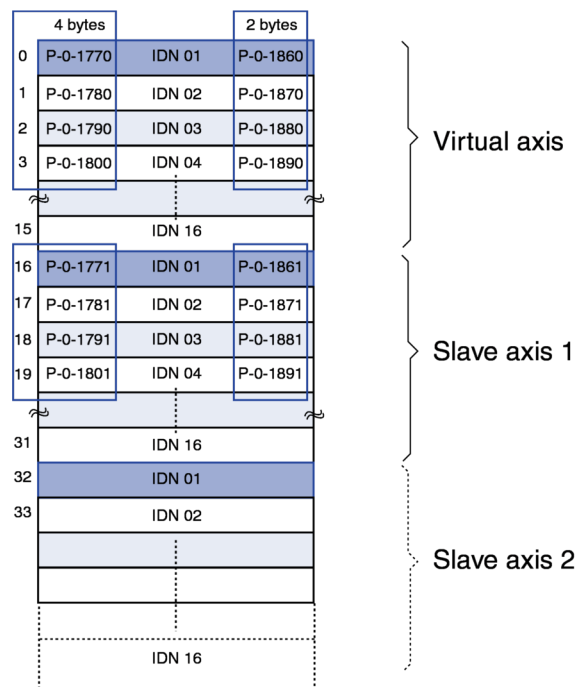
- **Configurable actual values**

It is possible to read in up to four freely configurable parameters from each real axis with the elements "dwUserActualDataA_i" to "dwUserActualDataD_i".



If the configurable actual values are configured "manually", i.e. not with IndraWorks, be sure to note the following (here using the example of the parameters P-0-1624 and P-0-1626):

- For parameters P-0-1624 and P-0-1626, the first four ID numbers (IDNs) of each axis have to be used for the configurable actual values. Be sure to observe the order (IDN01=dwUserActualDataA_i, IDN02=dwUserActualDataB_i, etc.; see figure).
- The mapping for the configurable actual values is parameterized in parameter P-0-1624. The parameters belonging to data containers A-D are listed in "5-2: Structure of AxisData".
- The desired/interesting axis information is parameterized in the same order in parameter P-0-1626.



DF000556v02

SM_TYPES The "SM_TYPES" structure ("MX_Base" library) is used for the configurable actual values and command values. The corresponding element from "SM_TYPES" is used depending on the format of the configured parameter.

There are different components for the corresponding data type of the parameter:

- Parameters in fixed point format and decimal places $\neq 0$ are contained in the "REAL_" component in scaled format.
- Unsigned 16-bit or 32-bit parameters are contained in the "UINT_" or "UDINT_" component.
- Signed 16-bit or 32-bit parameters are contained in "INT_" or "DINT_".

"SM_TYPES" structure:

Program:

```
TYPE SM_TYPES :
STRUCT
  REAL_ :REAL;      (* REAL (32-Bit) *)
  DINT_ :DINT;      (* DINT (32-Bit) *)
  UDINT_ :UDINT;    (* UDINT (32-Bit) *)
  UINT_ :UINT;      (* USINT (16-Bit) *)
  INT_ :INT;        (* SINT (16-Bit) *)
END_STRUCT
END_TYPE
```

Fixed contents

Fixed actual values

The "AxisData" structure contains some important axis data as REAL values in scaled form.

Element name	Data type	Direction	Parameter in MLD master	Parameter in axis	Significance
rActualPosition_i	REAL	Actual value	P-0-167x	S-0-0386	Active position feedback value
rActualVelocity_i	REAL	Actual value	P-0-168x	S-0-0040	Actual velocity value
rActualTorqueForce_i	REAL	Actual value	P-0-169x	S-0-0084	Torque/force feedback value

Element name	IEC name of the data element. This is the name used to address the structure element in the program
Data type	IEC data type of the data element
Direction	Command values are written by the user and transmitted by the system
Parameter in MLD master	The master contains a diagnostic parameter for this element. This parameter displays the value but is not writable
Parameter in axis	Source or target parameter on the respective axis. A fixed parameter can be contained here. The parameter can be set for the configurable elements. For the other elements, there is no parameter on the axis.

Axis status information

- wDriveStatus_i**

The actual value "wDriveStatus_i" contains the essential information from parameter "S-0-0135, Drive status word" or parameter "P-0-0115, Device control: Status word".

The following table shows the available bits.

Bit	wDriveStatus_i	S-0-0135 / P-0-0115
2-0	not used	Control information for service channel
3	Status of command value processing	Status of command value processing
4	not used	--

MLD communication interfaces and data channels

Bit	wDriveStatus_i	S-0-0135 /P-0-0115
5	Sercos III Status of control encoder	Sercos III Status of control encoder
5	not Sercos III master communication Command status change bit	not Sercos III master communication Command status change bit
7/6	not used	Real-time status bits 1 and 2
10-8	Operation mode	Actual operation mode 000: Primary operation mode active 001: Secondary operation mode 1 active 010: Secondary operation mode 2 active, etc.
11	not used	--
12	not used	--
13	Drive error	Drive error, error in class 1 diagnostics
15/14	Ready for operation 00: Drive not ready for power on, since internal checks incomplete 01: Ready for power on 10: Control and power sections ready for operation, torque-free 11: In operation, with torque	Ready for operation 00: Drive not ready for power on, since internal checks incomplete 01: Ready for power on 10: Control and power sections ready for operation, torque-free 11: In operation, with torque

Tab. 5-3: Structure of "wDriveStatus_i"

- **wDriveExtStatus_i**

The actual value "wDriveExtStatus_i" contains bit combinations from parameters "S-0-0135, Drive status word" and "S-0-0134, Master control word" or from parameters "P-0-0115, Device control: Status word" and "P-0-0116, Device control: Control word".

The following table shows the available bits.

Bit	dwDriveExtStatus_i	
10-0	not used	-
11	Motion commanding interrupted ["Axis_Interrupted"] ^{*1)}	P-0-166x (bit 15 = 1 & bit 14 = 1 & bit13 = 0 & bit 3 = 0) and P-0-165x (bit 13 = 1)
12	Drive follows command value input by MLD ["Axis_FollowingCommand"]	P-0-166x: Permanent control, bit 3 (status of external command value input) Temporary control (local axis only); bit 6 of P-0-0115 (status of internal command value input)
13	Control section ready for operation ("Axis_Inbb")	P-0-166x: Bit 15 bit 14: Control section ready for operation
14	Control and power sections ready for operation ("Axis_InAb")	P-0-166x: Bit 15=1: Control and power sections ready for operation
15	Drive with torque ("Axis_Power")	P-0-166x: Bit 15 = 1 and bit 14 = 1: Drive with torque

Tab. 5-4: Structure of "dwDriveExtStatus_i"

***1) Bit access "Axis_Interrupted"**

The IndraDrive firmware provides higher-level drive functions which can interrupt motion commanding by a higher-level PLC, but normally cannot stop the command from being issued.

The bit access "Axis_Interrupted" is intended to help detect interruptions in motion commanding so that the desired reaction can be executed afterwards. This does not affect function blocks "MC_Stop" and "MB_Stop".

Example:

Examples of higher-level drive functions

- Quick stop with probe detection
 - SBH (Safe operating stop)
 - E8029 Positive position limit exceeded
 - E8030 Negative position limit exceeded
 - E8043 Positive travel range limit switch activated
 - E8044 Negative travel range limit switch activated
 - Drive-controlled positioning
 - E2053 Target position out of travel range
 - Drive control commands
 - ...
-

If the active motion function block (DiscreteMotion, ContinuousMotion or SynchronizedMotion) on the MLD target detects that motion commanding was interrupted by a higher-level drive function, the function block aborts its processing with "CommandAborted"=TRUE.

- Once the higher-level drive function has been terminated, the motion function block remains in the "CommandAborted" state
- Once the higher-level drive function has been terminated, the last commanded operation mode becomes active again, unless a new operation mode was activated in the meantime:
 - "DiscreteMotion" operation modes no longer move to the last target position
 - "ContinuousMotion" and "SynchronizedMotion" operation modes follow the command value input again

- **wSignalStatus_i**

The actual value "wSignalStatus_i" corresponds to parameter "S-0-0144, Signal status word" of the specified axis.

- **wSyncStatus_i**

The actual value "wSyncStatus" corresponds to parameter "P-0-0089, Status word synchronization modes" of the specified axis. The parameter provides important status information regarding the execution of the synchronization modes (synchronous motion function blocks, such as "MC_GearIn", "MC_CamIn", "MB_GearInPos", "MB_MotionProfile", etc.).

- **dwPLCOpenStatus**

The bits of the axis status information "dwPLCOpenStatus" represent the status of the "state machine" according to PLCOpen (see figure "5-11: Motion state diagram"). Only one bit displaying the current status can be active at a time. The individual bits are available as fixed state bits in the "AxisData" structure (see "5-2: Structure of AxisData").

- **dwDiagNumber_i**

The actual value "dwDiagNumber_i" corresponds to parameter "S-0-0390, Diagnostic message number" of the specified axis. This parameter provides the current diagnostic drive message.

- **dwDeviveStatus_i**

The actual value "dwDeviveStatus_i" contains the essential information from parameter "S-0-1045, Sercos III: Device Status (S-Dev)"

Fixed state bits

Some important axis states, as well as all PLCopen states, are available as BOOL values (see "5-2: Structure of AxisData").

Data in "AxisData" structure**Information on using the "AxisData" structure**

If an error reaction programmed by the user was triggered by an F4 communication error, the "AxisData" structure may not be used.



Motion task in synchronism with master communication: The command values of the "AxisData" structure are written, even if the motion task in synchronism with master communication has not been completely processed within the NC cycle. In this case, the command values can be inconsistent.

Motion task in synchronism with CCD: The command values of the "AxisData" structure are written, even if the motion task in synchronism with CCD has not been completely processed within the CCD cycle. In this case, the command values can be inconsistent.



The data of the "AxisData" structure are only fully available if the drive is in phase 4. Otherwise, "Axis_Error" is the only element that is set for error diagnostics, provided an error is present at the corresponding axis.



Note the following when using a motion task in synchronism with master communication in conjunction with the MLD-M system mode: Consistent data access to the "AxisData" structure is impossible, since the motion task in synchronism with master communication runs synchronously with the NC cycle and, in this instance, not synchronously with the CCD cycle, although the "AxisData" structure is processed synchronously with the CCD cycle.

Validity and consistency of "AxisData" structure data

The data (command values and actual values) are centrally contained in a global array.

Motion task in synchronism with master communication without MLD-M system mode

When using a **motion task in synchronism with master communication** (see chapter "Basic functions of Rexroth IndraMotion MLD", section "[Task system](#)"), the "AxisData" structure is processed synchronously to the NC clock and thereby synchronously to the motion task in synchronism with master communication. That is to say the actual values are updated after the time T4 (T4 is the time when the actual values are latched for the drive telegram [AT]). This makes the actual values available at the start of the motion task.

Once the task has ended, the command values of the "AxisData" structure are written directly before the master data telegram is evaluated, i.e., the command values from the "AxisData" structure become valid at the same time as the command values from the MDT from the higher-level, potentially synchronizable Sercos III master. If the drive is connected to a master communication other than Sercos III, the drive synchronizes itself in the NC cycle interval. The NC cycle time is set via the parameter S-0-0001. This does not change the generation of the axis data structure. Consistent data from the "AxisData" structure are available within the motion task in synchronism with master communication.

If other tasks access the axis data structure, the actual values and command values should be read or written at the beginning of the task, if possible. The consistency of the data of the "AxisData" structure cannot be guaranteed in this case.

No motion task in synchronism with master communication

If no motion task in synchronism with master communication is used, the "AxisData" structure is processed at the beginning and end of a PLC time slice (see chapter "Basic functions of Rexroth IndraMotion MLD", section "[Task system](#)"). That is to say the actual values are updated directly before the start of a new time slice and the command values are written directly after the end of the PLC time slice. Accessing data consistently with the "AxisData" structure is only possible if the "AxisData" structure is used in a cyclic task with the smallest periodic time; in all other cases this cannot be achieved.

Motion task in synchronism with master communication with MLD-M system mode

When using a **motion task in synchronism with master communication** (see chapter "Basic functions of Rexroth IndraMotion MLD", section "[Task system](#)") **when MLD-M system mode is active**, the "AxisData" structure is synchronized to the CCD cycle. That is to say the actual values of the "AxisData" structure are updated after a CCD AT is processed. This means the actual values of the "AxisData" structure are updated when all actual values of the CCD slave axes, which were transmitted via the corresponding drive telegram, are known to the CCD master axis.

The command values of the "AxisData" structure are written directly before the master data telegram (MDT¹⁾) is transmitted to all CCD slave axes and therefore are part of the MDT.

Motion task in synchronism with CCD

The motion task in synchronism with CCD (see "Basic functions of Rexroth IndraMotion MLD", section "[Task system](#)") runs synchronously to the CCD cycle. That is to say the motion task starts after the CCD AT is processed and ends before the next CCD MDT is processed. It is possible to consistently access all data of the "AxisData" structure within the motion task in synchronism with CCD. If other tasks access the axis data structure, the actual values and command values should be read or written at the beginning of the task, if possible. The consistency of the data of the "AxisData" structure cannot be guaranteed in this case.

No motion task in synchronism with CCD

If no motion task in synchronism with CCD is used, consistent access to the data in the "AxisData" structure cannot be guaranteed. In this case, it is recommended to read or write the actual values and command values at the beginning of the task.



The consistency of individual structure elements is achieved by the hardware architecture (max. 32 bits).

Accessing elements in "AxisData"

These are some examples of code for accessing the elements of "AxisData". Declaration in "MX_Base":

¹⁾ Command values for the individual CCD slaves

MLD communication interfaces and data channels

Program:

```
AxisData : ARRAY[1..10] OF
MX_AXISDATA;
```

Examples of use in the PLC program:

Program:

```
bMyStandstill :=
AxisData[MyAxis.AxisNo].Axis_Standstill;
AxisData[MyAxis.AxisNo].dwUserCmdDataA_q.REAL :=
rMyValue;
rTorque :=
AxisData[Axis3.AxisNo].rActualTorqueForce_i;
```

Activating "AxisData"

The "AxisData" structure can be activated via the menu in IndraWorks (see figure below). To do this, select the "AxisData structure supported" option from the "PLC configuration" dialog (corresponds to "P-0-1367, PLC configuration", bit6=1).

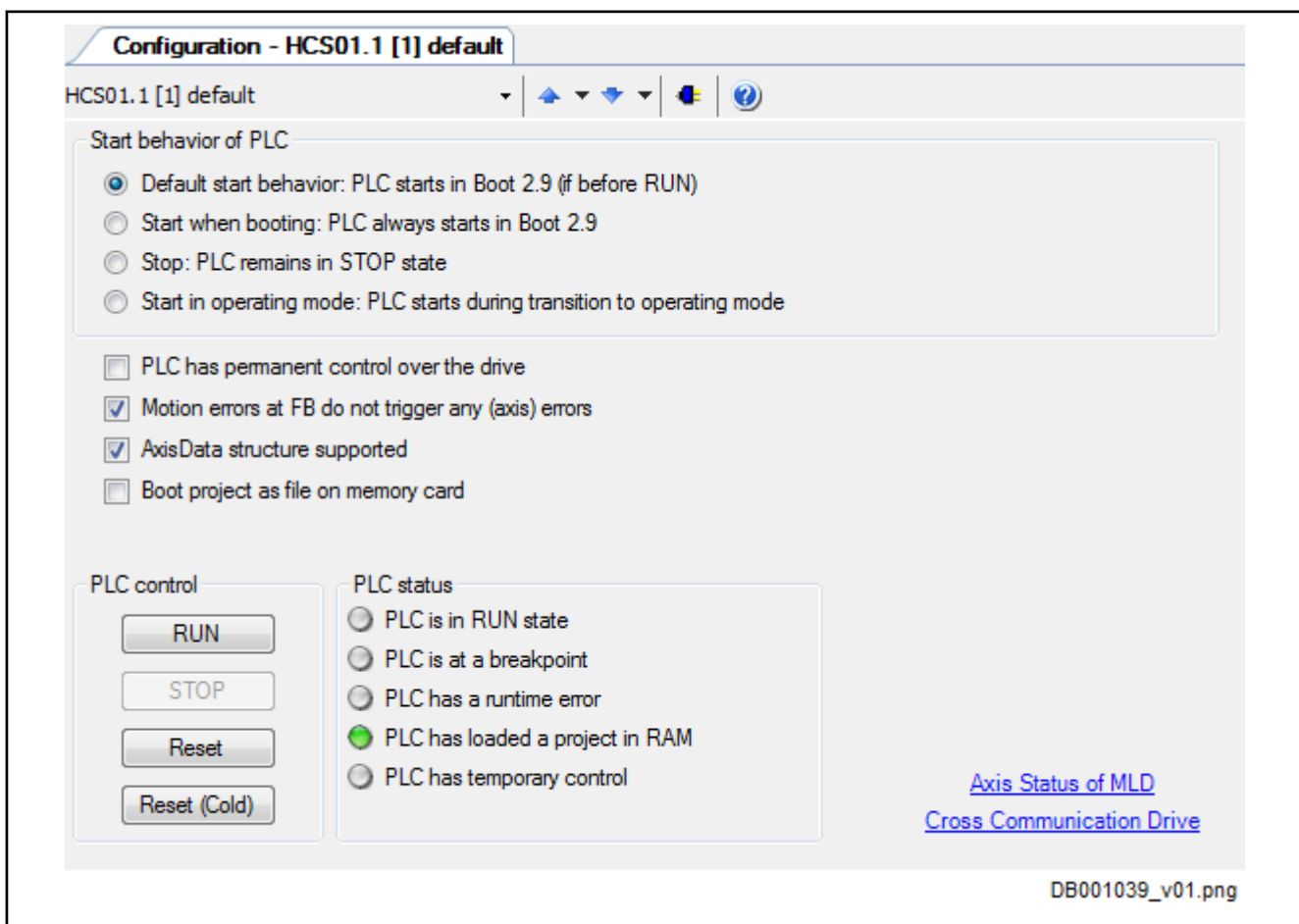


Fig. 5-6: Activating "AxisData"



Please note the following for activating "AxisData":

- "Permanent control" of the local axis has to be activated if the "AxisData" structure is used.
- For real axes, values are available for all elements of the "AxisData" structure, but are only available for some elements for the internal virtual axis.

Configuring user data in "AxisData"

If user-specific command values and actual values are to be transmitted in addition to the preset actual values, "AxisData" has to be configured in the following dialog:

Fig. 5-7: Dialog for setting the configurable AxisData elements

The parameters input in the dialog box are automatically entered in the CCD configuration. The entire list of CCD cyclic values produced can be displayed in the CCD configuration dialogs.

5.2.2 Acyclic data channels / interfaces

Introduction

IndraMotion MLD has the following alternatives for acyclic access to axis parameters:

- **Direct variable channel**
Simple and quick functional access from the PLC program to cyclically configurable S- and P-parameters in the drive. (The only axis that can be accessed is the local axis. For cyclic access to parameters of remote axes, see "[Cyclic data in MLD-M system mode](#)".)
- **Parameter channel** for acyclic access (read and write) to axis parameters of the local and remote axes using ready-made functions or function blocks. This allows all S- and P-parameters in the drive to be accessed, including the "PLC register" parameters, via function blocks and functions.
- **PLC parameters for general purpose (PLC registers)**

Drive parameters that can be used as desired in the PLC program, e.g., for data management and/or communication between IndraMotion MLD and external devices, inputs/outputs, sensors, etc. This actually is not a data channel, but drive parameters that can be used for communication or data management.

- **Motion command channel**

Internal data channel for transmitting consistent input from motion function blocks. This channel is not visible to the user and cannot be directly accessed, but is used by the motion control of IndraMotion MLD.

- **Socket communication**

User-programmed data channel for transmitting any information via Ethernet (TCP or UDP).

Options

- For connecting a third-party device via a documented legacy communication protocol in the third-party device and addressing the protocol from MLD.
- Defining/programming a communication protocol as needed in the third-party device and in the drive controller.
- or –
- Addressing the drive controller via Sercos Internet Protocol (S/IP) for accessing drive parameters by implementing the interface in the third-party device.

Accessing local parameters via direct variables

Brief description

Another way to process parameters easily and quickly is accessing them with direct variables. This allows parameters in the PLC program to be accessed for reading and writing, with a simple syntax, without function call or function blocks and without having to take the long way via the process image. The PLC source code therefore is very simple and clearly structured.



As with processing cyclic master communication data, the data in this case are neither stored nor are the limit values checked nor are errors processed.

Features

Access via direct variables allows all cyclically configurable parameters in the drive to be directly accessed (cf. "S-0-0187, List of configurable data in the cycl. actual value data channel" and "S-0-0188, List of configurable Data in the cycl. command value data channel").

The parameters of the inputs and outputs (P-0-1390 to P-0-1429, P-0-1440 to P-0-1447, P-0-1450 to P-0-1455 and P-0-1460 to P-0-1466) cannot be used as direct variables, because their values are accessed via I/O access.



Direct variables are declared as a `DV_Axis[ ]` global array of the "MX_DirectVarAxis" function block. This is part of the "GVL_Base" global variable list of the "MX_Base" library.

Examples of access:

- IDN:
DV_Axis[AXIS_1].S_0_0092
- EIDN:

DV_Axis[AXIS_1].S_0_1101_000_001

Accessing parameters via direct variables provides the following **advantages**:

- rapid parameter access
- simple and clearly structured programming in PLC

Accessing parameters via direct variables also has the following **restrictions**:

- access to cyclically configurable parameters only
- no access to parameters of remote axes
- direct variables cannot be triggers for a PLC event task
- bit access impossible (e.g., DV_Axis[AXIS_1].P_0_0115.3)

Parameters involved

The following parameters are used in conjunction with access via direct variables:

- S-0-0187, List of configurable data in the cycl. actual value data channel
- S-0-0188, List of configurable data in the cycl. command value data channel

Function

Available direct values

All cyclically configurable parameters (cf. "S-0-0187, List of configurable data in the cycl. actual value data channel" and "S-0-0188, List of configurable data in the cycl. command value data channel") are already available as global direct variables in the "Base" library.

Important notes

Note the following when accessing via direct variables:

- You cannot define your own direct variables
- List parameters cannot be addressed with direct variables
- Errors are not processed when accessing via direct variables, so this access requires a certain knowledge of the associated parameters (e.g., limit values, etc.)

Parameter channel

Brief description

The parameter channel allows all S- and P-parameters of the drive to be acyclically accessed for reading and writing.



The parameters are directly accessed, i.e., the duration of a read or write process is very short for the local axis.

Access to parameters of remote axes is possible and uses the CCD service channel in Sercos III. As a result, the function blocks have to be called several times in the service channel and signal "Done" when the transmission is over.

Accessing via function or function block

There are several ways to access parameters.

- Features of accessing parameters via a function:
 - faster than a function block
 - no instances
 - no error check
 - no access to remote axes
- Features of accessing parameters via a function block:
 - transmission with error check

- instance required
- action takes place at rising "Execute" edge
- accessing "Rexroth Inline" I/O modules is possible (Sercos III)

Parameter channel features

The parameter channel has the following features:

- The function blocks and functions for processing the parameters are contained in the "MX_PLCOpen" library.
- For a single-axis system, the axis number "Axis1" always has to be specified.



The "RIL_SercosIII" library is also available as an interface between the PLC programming environment and the Sercos III nodes. For example, "Rexroth Inline" I/O modules can be diagnosed or Sercos parameters can be acyclically written or read.

Here, the Sercos III nodes have to be addressed via their Sercos address.

Functional principle

This section describes in detail how accessing drive parameters from a PLC program works.

Addressing

The parameters are addressed via constants from the Base library (FP_....). The constants used in this library contain the Sercos-compatible address for P-/S-parameters. For the function blocks of the Base library, the "Ident" or "ParameterNumber" input is provided with these constants.

The screenshot displays the 'Library repository' window with a table of system libraries and a detailed view of the 'MX_Base, 20.12.0.0 (System)' library.

Name	Namespace	Effective version
BASELIB = MX_Base, 20.12.0.0 (System)	MX_Base	20.12.0.0
CHECKLIB = MX_CheckRtv, 20.1.2.0 (System)	MX_CheckRTV	20.1.2.0
IecVarAccess = IecVarAccess, 3.5.5.0 (System)	IecVarAccessLibrary	3.5.5.0
IoStandard = IoStandard, 3.5.7.0 (System)	IoStandard	3.5.7.0
PLCOPENLIB = MX_PLCopen, 20.8.0.0 (System)	MX_PLCopen	20.8.0.0
RIL_COMMONTYPES = MX_CommonTypes, 20.10.0.0 (System)	MX_CommonTypes	20.10.0.0
RIL_FIELDBUSTYPES = RIL_FieldbusTypes, 14.6.0.0 (System)	RIL_FieldbusTypes	14.6.0.0
RIL_SERCOSIII = RIL_SercosIII, 14.16.0.0 (System)	RIL_SercosIII	14.16.0.0
RIL_UTILITIES = RIL_Utility, 14.16.0.0 (System)	RIL_Utility	14.16.0.0
Standard = Standard, 3.5.5.0 (System)	Standard	3.5.5.0

The 'MX_Base, 20.12.0.0 (System)' library details are shown below:

VAR_GLOBAL CONSTANT GVL_P_Param_ID					
Name	Type	Inherited from	Address	Initial	Comment
FP_P_0_0001	MB_IDN			32769	Switching frequency of
FP_P_0_0002	MB_IDN			32770	SPI flash aging counter
FP_P_0_0003	MB_IDN			32771	Status of parameter block
FP_P_0_0004	MB_IDN			32772	Velocity loop smoothing
FP_P_0_0005	MB_IDN			32773	SPI flash aging counter
FP_P_0_0006	MB_IDN			32774	Diagnostic message code
FP_P_0_0008	MB_IDN			32776	Activation E-Stop function
FP_P_0_0010	MB_IDN			32778	Excessive position correction
FP_P_0_0011	MB_IDN			32779	Last valid position correction

DB001022_V02

Fig. 5-8: Addressing parameters via constants



Due to its width of 16 bits, the input does not accept any other parameter types (such as "A" or "Y"). This means the range of values is automatically protected from incorrect values.

Parameterization and programming information

Read/write parameters in DINT

With function blocks "MX_ReadParamDINT"/"MX_WriteParamDINT", 16-bit and 32-bit parameters are transferred in Sercos scaling format. (Example: With standard scaling, 1 degree corresponds to a value of 10000.) This means the values are provided as signed 32-bit "DINT" values when read.



If a PLC register parameter (e.g., P-0-1370) has been set to the data format FLOAT, function block "MB_ReadRealParameter"/"MB_WriteRealParameter" has to be used to read/write the value (data) of the parameter.



For the function blocks that access parameters, the "Parameter-Number" input was changed from "DINT" to "MB_IDN" as of MPx06. "MB_IDN" is available for all IndraMotion variants. With "IndraMotion MLD", "MB_IDN" corresponds to the Sercos III definition, giving it a data width of 32 bits. Both "DINT" type and "MB_IDN" type constants can be used at the "ParameterNumber" input. PLC programs created before MPx06 are still operable.

MLD communication interfaces and data channels

Read/write text parameters The functions "MX_fReadStringParam"/"MX_fWriteStringParam" can be used for processing text parameters.

The "WriteBuffered" input can be used to buffer each parameter during writing.

Parameter channel libraries The figure below shows an example from the "PLCopen" library:

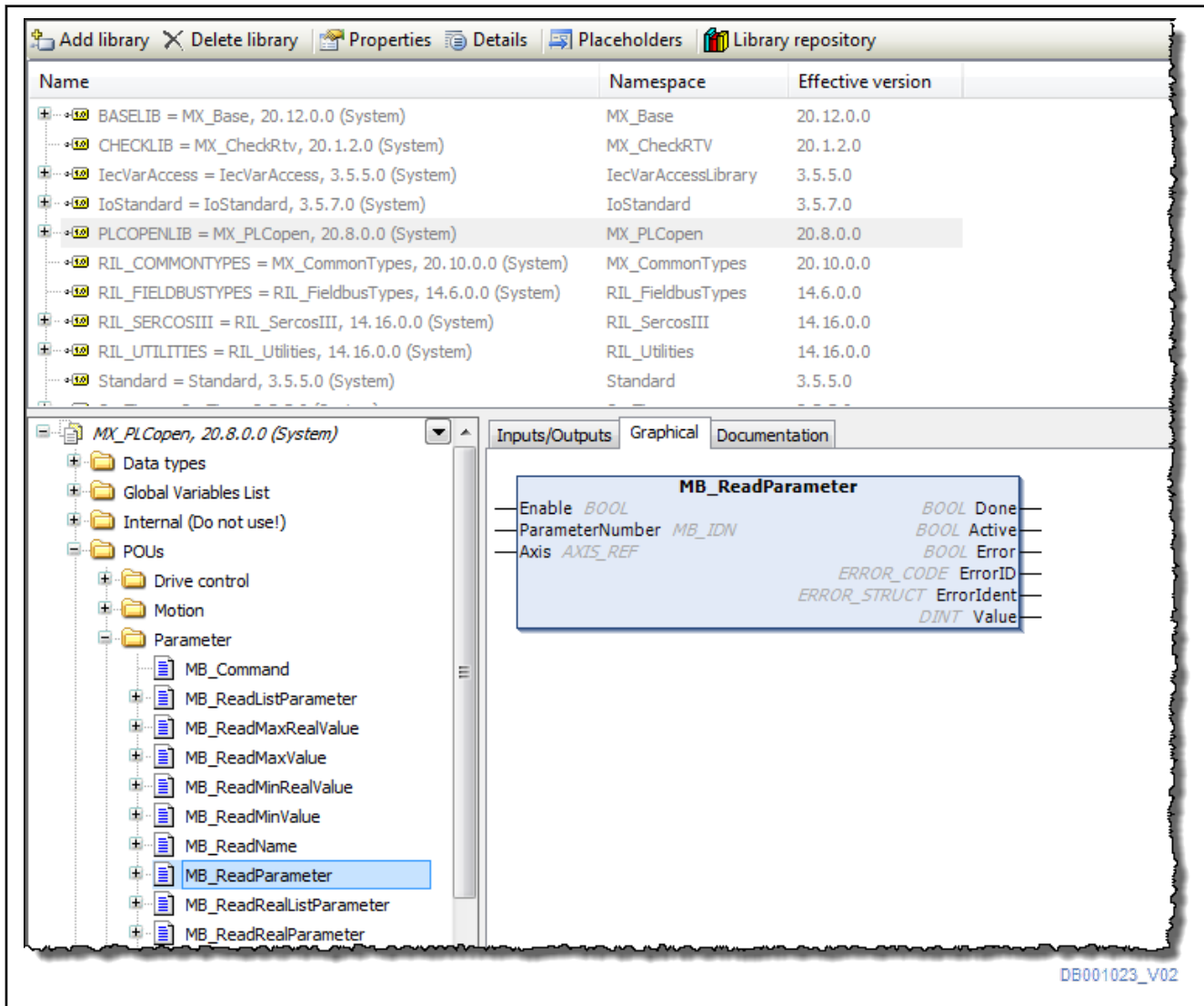


Fig. 5-9: Functions and function blocks for accessing parameters

PLC parameters for general purpose (global registers)

Brief description

Access via "parameters for general purpose" (global registers) can be used to exchange data between MLD and a neighboring drive (in MP*04VRS and above via CCD) or a higher-level control unit or an external operator terminal (e.g., BTV).



The global registers do not have any direct influence on the drive, but only take effect in conjunction with MLD.

Features The display format of the global registers can be defined as desired. Buffered and unbuffered registers are available. Registers with list structure are available for transmitting large data volumes.

- 32 unbuffered global registers for parameterizing PLC functions or function blocks
- 32 buffered global registers for voltage-failure-safe configuration of PLC functions or function blocks
- 2 global text registers as text parameters that can be freely used to display diagnostic message texts
- 3 buffered list registers with 1024 4-byte values
- 1 unbuffered list register with 8192 4-byte values

Pertinent parameters The registers are represented by the following parameters:

- **Global PLC registers, unbuffered:**
 - P-0-1270, PLC Global Register A0
 - P-0-1271, PLC Global Register A1
 - P-0-1272, PLC Global Register A2
 - ...
 - P-0-1301, PLC Global Register A31
- **Global text registers, unbuffered:**
 - P-0-1387, PLC Global text register AT0
 - P-0-1388, PLC Global text register AT1
- **Global list register, unbuffered:**
 - P-0-1368, PLC Global Register AL0
- **Global PLC registers, buffered:**
 - P-0-1370, PLC Global Register G0
 - P-0-1371, PLC Global Register G1
 - P-0-1372, PLC Global Register G2
 - ...
 - P-0-1385, PLC Global Register G15
 - P-0-1316, PLC Global Register G16
 - P-0-1317, PLC Global Register G17
 - P-0-1318, PLC Global Register G18
 - ...
 - P-0-1331, PLC Global Register G31
- **Global list registers, buffered:**
 - P-0-1389, PLC Global Register GL0
 - P-0-1311, PLC Global Register GL1
 - P-0-1312, PLC Global Register GL2
- **To configure the display format of the registers:**
 - P-0-1386, PLC display format Global Register

Function

Adjusting global register format The display formats of the global registers can be individually adjusted with "P-0-1386, PLC display format Global Register". It is possible to define the display format (e.g., BIN, Dec, Hex, etc.) and the number of decimal places.

Applications for Gxx global registers

The content of the global registers "Gxx" and "GLx" is buffered in case control voltage fails, i.e., the register contents are stored in non-volatile form so that the parameter contents do not get lost in case voltage fails.

The global registers G0 .. G31 and GL0 to GL2 can be used for the following applications:

- Configuring PLC functions or function blocks
- Communicating with the external control unit via the master communication interface



When the global registers are used as command values (command values from the higher-level control unit), note that the values will be set to "0" if communication fails (as is typical for inputs).

Applications for Axx global registers

- Use as non-volatile (permanent) memory for MLD-S, because the contents are retained in case voltage fails

The global registers A0 .. A31 and AL0 can be used for the following applications:

- Setting parameters online for PLC functions or function blocks
- Communicating with the external control unit via the master communication interface

Applications for ATx global registers

Global text registers "P-0-1387, PLC Global Register AT0" and "P-0-1387, PLC Global Register AT1" are available as freely usable text parameters with a maximum of 255 characters plus closing "0" character.

The global registers (AT0 and AT1) can be used for the following applications:

- Communicating with higher-level control unit or HMI
- Defining freely definable diagnostic message texts

Parameterization and commissioning information

The display format for the parameters for general purpose in the parameter channel can be defined with "P-0-1386, PLC display format Global Register". P-0-1386 is a list parameter with 68 elements and the following assignment:

Element no.	Affected parameter
1..16	P-0-1370 to P-0-1385 (global register G0 to G15)
17	P-0-1389, global register GL0
18	P-0-1368, global register AL0
19..34	P-0-1316 to P-0-1331 (global register G16 to G31)
35..66	P-0-1270 to P-0-1301 (global register A0 to A31)
67..68	P-0-1311 to P-0-1312 (global register GL1 to GL2)

Tab. 5-5: P-0-1386, PLC display format Global Register

Limits, names and units of the parameters for general purpose can be defined as desired with the PLC program. This can be done with the following functions from the "MX_Base" library:

- MX_fSetParamLimits
- MX_fSetParamName
- MX_fSetParamUnit

Motion command channel

Brief description

The motion command channel is an internal data channel for transmitting consistent inputs of ready-made motion function blocks. These ready-made motion function blocks can be called directly from the PLC program. Internally, the motion command channel is mainly used for implementing "motion control" tasks or for technology functions. The available motion function blocks conform to PLCopen, i.e., programs created with the motion function blocks can be transferred to other targets.

- Features**
- The motion function blocks have a logical axis address for selecting the axis. In the MLD-S single-axis system, this allows the local axis and the master axis generator to be controlled.
 - The most important values during motion control are preset via the corresponding inputs at the motion function blocks. Other specific settings, such as a scaling, jerk, etc., have to be parameterized. This is normally done when commissioning the axes with the IndraWorks commissioning software. However, other settings (e.g., jerk) can also be modified by accessing the parameters.

Activation "Permanent control" of the local axis has to be activated for MLD motion control. The local axis will then follow the input of MLD and can be controlled with the motion function blocks.

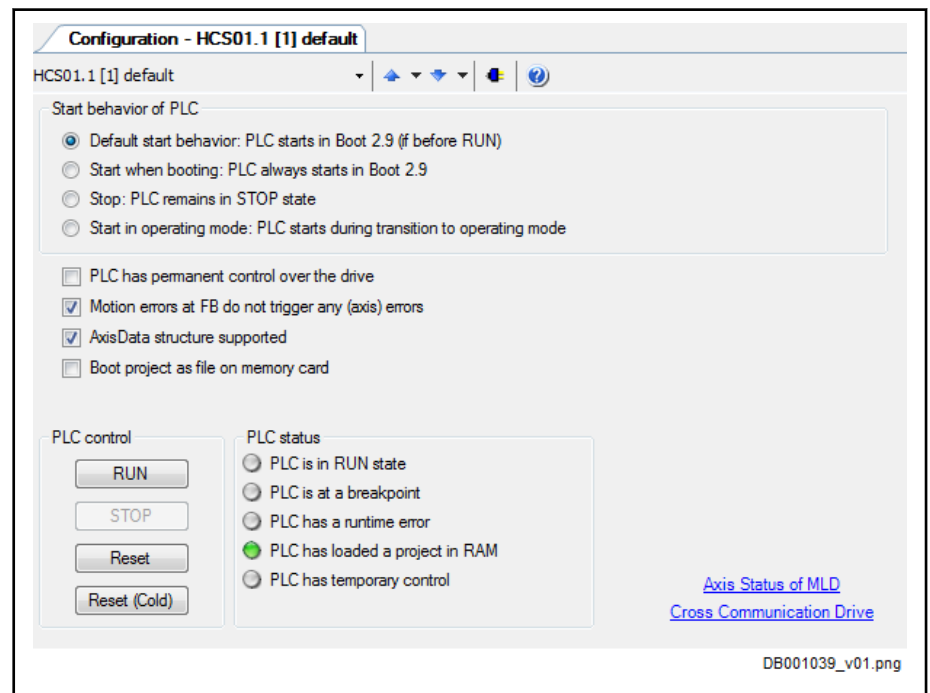


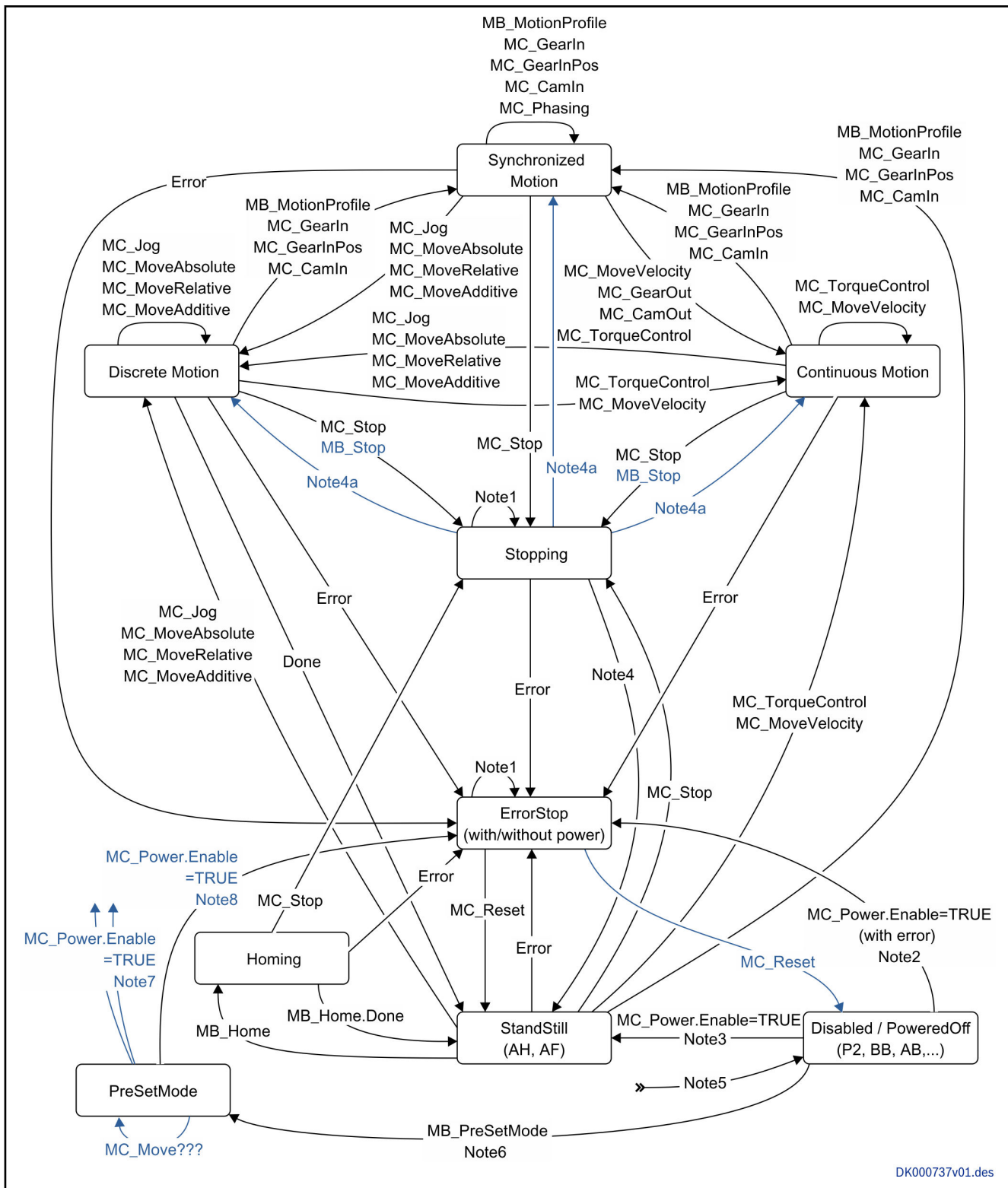
Fig. 5-10: PLC configuration; "Permanent control"

PLCopen Motion control via the PLC program is carried out using PLCopen-based function blocks. For this purpose, PLCopen has defined several IEC 61131 function blocks with which the axes can be controlled. Aside from the function blocks already defined by PLCopen, there are other function blocks based on this standard that provide additional functionality to our drives.

According to PLCopen, all axes behave according to a defined status model (PLCopen state machine). Rexroth supports this status model; additional statuses (e.g., "PreSetMode") were added for advanced functionalities.

The current PLCopen-based status can be seen in "AxisData" via status bits.

MLD communication interfaces and data channels



Note x See table below for explanations on the notes
 Fig. 5-11: Motion state diagram

Note 1	In this status ("ErrorStop" or "Stopping"), all functions blocks can be called, although they are not executed; except for "MC_Reset" and "MB_ClearAllError" which respectively trigger the transition to "StandStill" or "ErrorStop"
Note 2	Power.Enable = TRUE and there is an error in the axis
Note 3	Power.Enable = TRUE and there is no error in the axis (standard without MB_PresetMode)
Note 4	MC_Stop.Done AND NOT MC_Stop.Execute
Note 4a	New motion command while NOT MB_Stop.Execute (without MC_Stop)
Note 5	When "Enable" is removed from "MC_Power" in any status, the drive goes to "PowerOff"
Note 6	"MB_PreSetMode" is used to reach the "PreSetMode" state. Commanding is possible in this mode without power. Subsequently switching on with "Power" causes the preselected mode to start
Note 7	A rising edge at "Enable" of "MC_Power" starts a preselectable mode from the "PreSetMode" state
Note 8	An edge at "MC_Power" without previous commanding in status "PreSetMode" will cause an error and thereby "PowerOff"

Tab. 5-6: Legend of "Motion state diagram" figure

Programming	Programming takes place with the function blocks of the "MX_PLCOpen" library. The function blocks have been designed in such a way that they are cyclically called. They are normally activated by an input edge and provide information on status outputs. The exact functional principles are described in the respective library documentation.
Libraries	The "MX_PLCOpen" library contains function blocks for motion control, such as "MC_MoveAbsolute" or "MC_Stop", as well as function blocks for axis control, such as "MC_Power" and "MC_Reset", which are used to bring the axis in control or clear an error.
Settings	After the local axis has activated permanent control, the required operation modes of the axes are parameterized automatically. The operation modes thereby are automatically selected. Normally it is not necessary to make further settings.
Scaling	During commissioning, the axes have to be scaled in the corresponding dialogs of the IndraWorks commissioning software. At the motion function blocks, the preset values are set via the function block inputs as physical values.
Other settings	Other settings, such as the jerk, cannot be parameterized at the function blocks, but have to be set in the IndraWorks commissioning software. If such values are to be changed, they can be set at runtime via the parameter channel (e.g., "MB_WriteParameter").

Master axis generator

As far as commanding is concerned, the master axis generator is an independent axis for the user. The axis can be moved with some motion function blocks and stopped with "MB_Stop" / "MC_Stop".

To use the master axis generator, it has to be activated once.

MLD communication interfaces and data channels

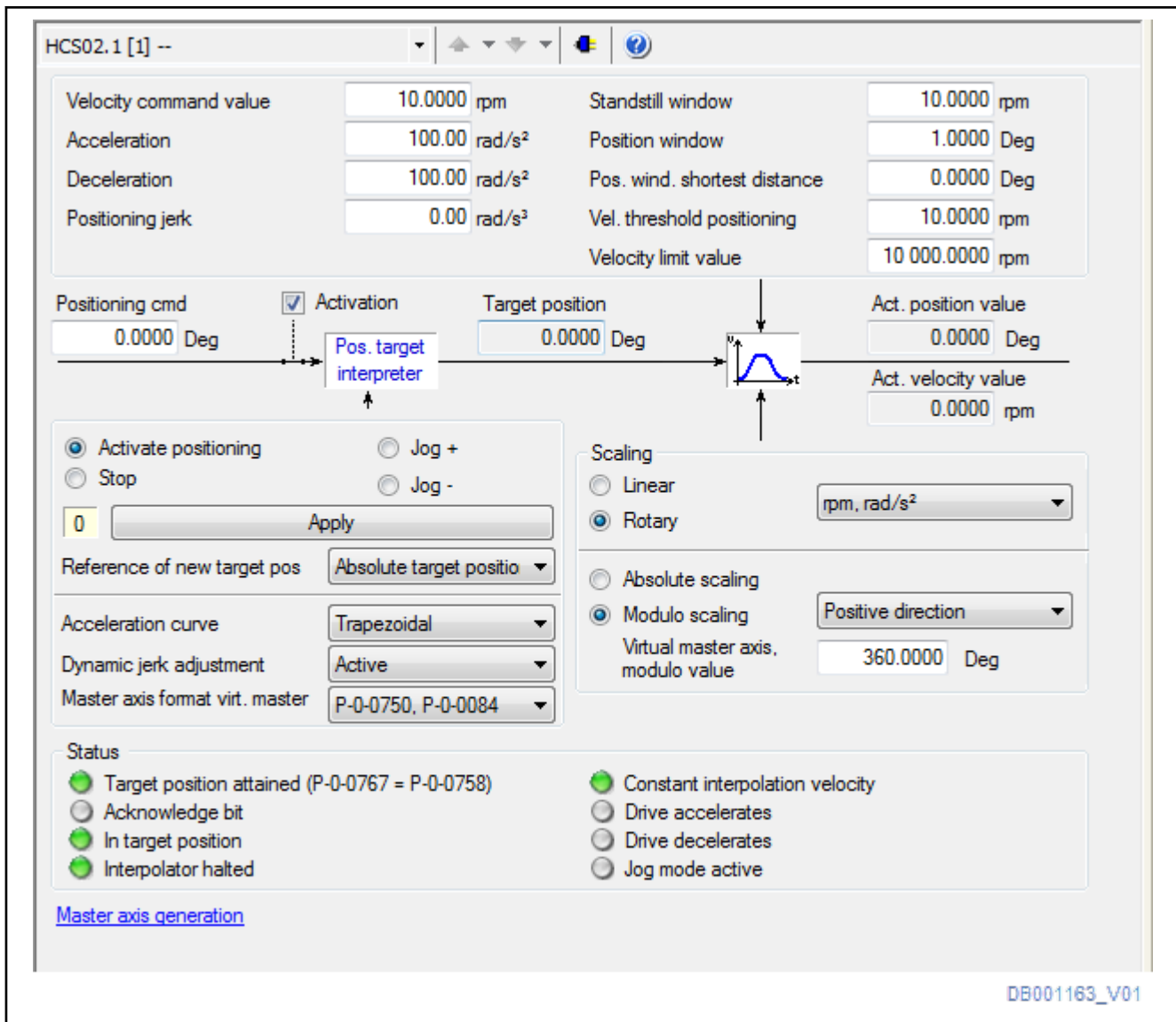


Fig. 5-12: Activating the master axis generator in IndraWorks



The virtual master axis can be controlled via the axis input "VmAxisint".

Possible function blocks

The following function blocks are available for using the master axis generator:

- "MC_MoveRelative" or "MX_MoveRelative"
- "MC_MoveAbsolute" or "MX_MoveAbsolute"
- "MC_MoveAdditive" or "MX_MoveAdditive"
- "MC_MoveVelocity"
- "MC_Stop" or "MB_Stop"

Internal motion channel implementation



The following information on the internal implementation of the motion channel generally is not required for using the function blocks. In fact, it is additional information on how the function blocks use the drive with its operation modes and parameters.

With this information, it is possible to diagnose the input of the motion function blocks on the drive side.

Operation modes used

The following operation modes are set when the motion channel is activated:

Operation mode	Setting
Primary operation mode	Velocity control
Secondary operation mode 1	Torque/force control ⁽¹⁾
Secondary operation mode 2	Position synchronization ⁽²⁾
Secondary operation mode 3	Drive-controlled positioning ⁽¹⁾
Secondary operation mode 4	Velocity synchronization ⁽²⁾

- 1 only in base package with "CL" (Closed-Loop) characteristic; besides requires any functional package to be enabled
- 2 requires the functional package "Synchronization" to be enabled

Tab. 5-7: Operation modes for "Motion Control" command interface



If the packages for the operation modes cannot be enabled, the relevant operation mode is set to "velocity control". The concerned motion function blocks cannot be operated and signal an error.

Processing in IndraMotion MLD-S

With IndraMotion MLD-S, the parameter inputs for the operation modes are directly written by the PLC motion system. The parameters concerned are listed in the library description under the function blocks. When a motion command is activated, the corresponding operation mode is selected and all corresponding input parameters are synchronously transmitted to the drive.

General use

Observe the following points when using the motion channel:

- Axes can be commanded from several preemptive tasks.
- In a motion cycle (e.g., in 1 ms), all axes can be simultaneously commanded.
- In a motion cycle, parallel motion inputs (primary and secondary command values) per axis can be simultaneously commanded (up to 3 channels).
- Even in the case of commands transmitted in quick succession, it is ensured that the last command is processed.
 - Several commands transmitted in the same cycle for the same drive are initially stored in a FIFO memory and then processed by the command interpreter. It is possible to buffer 3 commands. When too many commands are transmitted and an overflow occurs, an error is signaled.
 - When a task transmits a command, it overwrites a command which the interpreter possibly has not fetched yet. The command thus overwritten signals "CommandAborted" in its instance.

MLD communication interfaces and data channels

- If the command was triggered by the same instance, the message cannot be generated.

- A new command with a primary command value aborts running commands with secondary command values.
- Secondary command values can only be commanded in the correct drive status: A new command with a secondary command value is only accepted when the corresponding port was commanded. The corresponding operation mode is checked.
- As far as commanding is concerned, the virtual master axis behaves like a real axis, i.e., it has its own axis number!

The paragraphs below contain information on how to use some specific motion function blocks:

Homing The "MB_Home" function block used for homing an axis can only be started if no other function block has been started. This has to be ensured by the PLC program.

Absolute position function block The absolute position can also be set in the "AH" state. Thus the command for setting absolute position can be directly transmitted. When axis motion is running, the command does not work. Only when the drive goes to "AH" or is without power can the command be started.

"MB_Stop" Observe the following points when using function block "MB_Stop":

- The control input is called "ExecuteLock".
- As long as "ExecuteLock"="TRUE", no other function block can set the drive in motion. In the PLC program it should also be excluded for any other function block to set the drive in motion.
- If the state is "Active" and "ExecuteLock" = FALSE, "MB_Stop" can be aborted using a motion function block.
- The user should only use one instance of "MB_Stop" to make the program clearly structured and safe.
- If there are multiple instances of "MB_Stop", the last instance automatically dominates. The last instance has higher priority.

"MC_Stop" Observe the following points when using "MC_Stop":

- As long as "Execute"="TRUE", no other function block can set the drive in motion. In the PLC program it should also be excluded for any other function block to set the drive in motion.
- The user should only use one instance of "MC_Stop" to make the program clearly structured and safe.
- If there are multiple instances of "MC_Stop", the last instance automatically dominates. The last instance has higher priority.

"MC_Power" Observe the following points when using "MC_Power":

- When switching on, the axis goes to "AH" unless motion was precommanded before (with "MB_PreSetMode" and motion function block).
- In case motion was precommanded, the axis goes directly to the preselected operation mode.

Encoder switching, lag error switching For encoder switching and switching with/without lag error, the function blocks have separate inputs.

Notes on commissioning

⚠ DANGER

Lethal injury and/or property damage due to axis moving unintentionally!

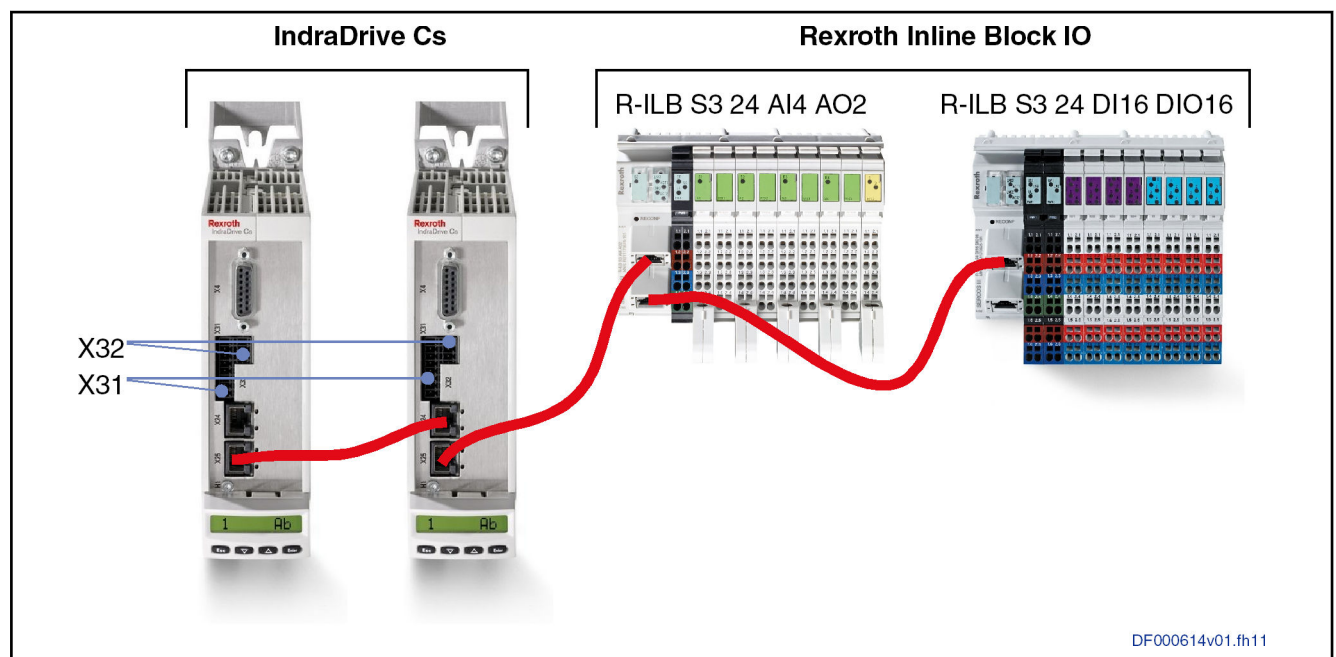
⇒ In case of stand-alone operation (internal PLC has control over the drive), the drive can be freely moved via the MC command interface. It is therefore absolutely necessary to ensure that accidental axis motion cannot be triggered.

5.2.3 Configuring the inputs and outputs (I/O configuration of MLD)

Introduction

IndraMotion MLD theoretically allows all inputs and outputs with a Sercos interface to be accessed, e.g.:

- all inputs and outputs of the control section (including remote axes)
- all inputs and outputs of the optional module "DA"
- inputs and outputs of "Rexroth Inline Block IO"
- inputs and outputs of "Rexroth Inline Modular IO"



X31

digital inputs, digital output

X32

analog input

R-ILB S3 24 AI4 AO2

Inline block IO with 4 analog inputs and 2 analog outputs

R-ILB S3 24 DI16 DIO16

Inline block IO with 16 digital inputs and 16 configurable digital inputs or outputs

Fig. 5-13: Overview of the inputs/outputs (block I/Os as examples of "Rexroth Inline")

Configuring the digital and analog inputs/outputs

IndraWorks features the dialogs required for assigning local inputs and outputs to the PLC. The configuration parameters are automatically filled by the corresponding dialog. Module-specific dialogs allow the local inputs and outputs to be configured.

The tables below list all the inputs and outputs.

MLD communication interfaces and data channels

Function	Hardware	Number
Digital inputs	Rexroth Inline Block IO, digital	min. 16 (max. 32) ^{1), 2)}
	Rexroth Inline Modular IO, digital	8 (on-board) ³⁾ extendable
Digital outputs	Rexroth Inline Block IO, digital	min. 0 (max. 16) ^{1), 2)}
	Rexroth Inline Modular IO, digital	4 (on-board) ³⁾ extendable
Analog inputs	Rexroth Inline Block IO, analog	4 ²⁾
	Rexroth Inline Modular IO, analog	depending on extension ³⁾
Analog outputs	Rexroth Inline Block IO, analog	2 ²⁾
	Rexroth Inline Modular IO, analog	depending on extension ³⁾

- 1 16 configurable digital inputs or outputs
2 For further information, see index entry "Rexroth Inline Block IO"
3 For further information, see index entry "Rexroth Inline Modular IO"

Tab. 5-8: Overview of inputs/outputs of "Rexroth Inline Block IO" / "Rexroth Inline Modular IO"

Drive system / control section		Basic device (on board)		Optional module "DA"		
		Standard inputs (probe inputs thereof)	Switchable inputs/outputs	Inputs	Outputs	Switchable inputs/outputs
HCS01.1		7 (2)	1	6	6	2
HCQ02.1, HCT02.1		16 (-)	-	-	-	-
KSM02.1, KMS02.1		-	4	-	-	-
CSE02.1A, CSB02.1A		7 (2)	1	-	-	-
CDB02.1B		14 (4)	8	6	6	2
CSB02.1B, CSH02.1B		11 (2)	5	6	6	2
Digital inputs	Configuration parameters	P-0-0300, P-0-0301, P-0-0306, P-0-0307				
	Input image	P-0-0303				
Digital outputs	Configuration parameters	P-0-0310, P-0-0311, P-0-0312, P-0-0313, P-0-0316				
	Output image	P-0-0304				

Tab. 5-9: Overview of the digital inputs/outputs of the drive system/control section

Function	Hardware	Number	Configuration parameters	Register ²⁾
Analog inputs	Drive system / control section	HCS01.1: 1 CSE02.1A, CSB02.1A: 1 CDB02.1B: 2 CSB02.1B, CSH02.1B: 3	P-0-0213 .. P-0-0220 P-0-0231 / P-0-0232 P-0-0255	P-0-0210 P-0-0211 P-0-0228
	Optional module "DA"	2 ¹⁾	P-0-0233 P-0-0234 P-0-0255	P-0-0229 P-0-0208
Analog outputs	Drive system / control section	CDB02.1B, CSB02.1B, CSH02.1B: 2	P-0-0419 / P-0-0423 / P-0-0425 P-0-0418 / P-0-0420 / P-0-0422 P-0-0427	P-0-0139 P-0-0140
	Optional module "DA"	HCS01.1: 2 ¹⁾ CDB02.1B, CSB02.1B, CSH02.1B: 2	P-0-0428 / P-0-0459 / P-0-0463 P-0-0429 / P-0-0462 / P-0-0464	P-0-0414 P-0-0415

1 not with HCS01 Economy
2 Parameters for displaying/reading the analog value

Tab. 5-10: Overview of analog inputs/outputs

Local inputs/outputs (Master/Axis1)

Based on IndraWorks dialogs, the paragraphs below describe examples of configuration for assigning process input/output images (PLC parameters) to digital and analog inputs-/outputs.

Configuring digital inputs-/outputs of the control section

Each input and output of the control section can be individually used in the drive or assigned to the PLC. Digital inputs/outputs of the control section are assigned to PLC parameters by assigning the desired bits of a PLC parameter to a digital input or output. IndraWorks features a dialog for this purpose. The dialog can be called in the device tree under "Local I/O - X31/X32".

See also Functional description of firmware "Digital inputs/outputs"

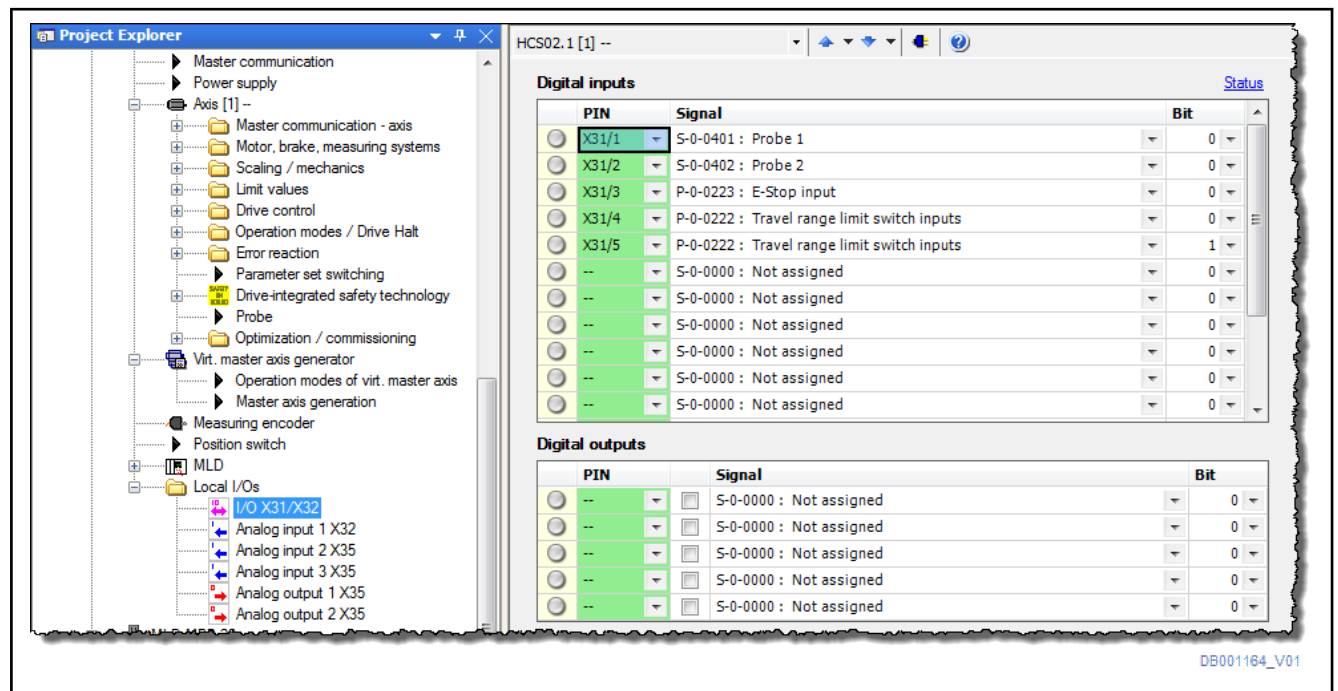


Fig. 5-14: Configuring the digital inputs/outputs in IndraWorks

MLD communication interfaces and data channels

Configuring the analog input of the control section

Reading in analog voltage values requires configuring the respective PLC parameters (P-0-1390, etc.) for the analog input. The example below illustrates the assignment of the PLC parameter "P-0-1391, PLC input WORD1 AT %IB2" to analog input 1 (see also Functional description of firmware "Analog inputs").

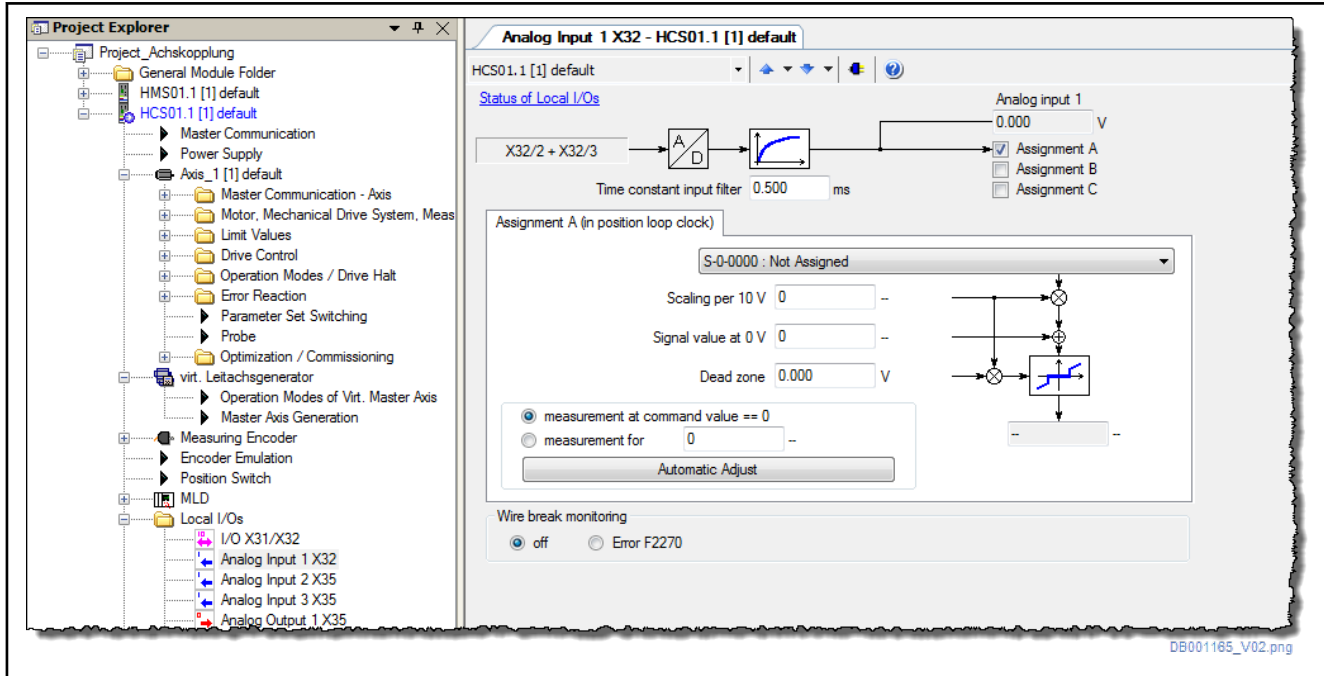


Fig. 5-15: Configuring an analog input in IndraWorks

Inputs/outputs of the optional module "DA"

The digital and analog inputs/outputs are accessed via defined parameters. Each digital input/output is accordingly preset via a configuration menu.

Inputs/outputs from a master control unit

In the "Master communication settings" dialog, the MLD process image parameters can be entered under the process data in order to establish a connection from the master control unit to the I/Os of MLD.



MLD inputs coming from a master control unit are set to zero in case the connection is interrupted.

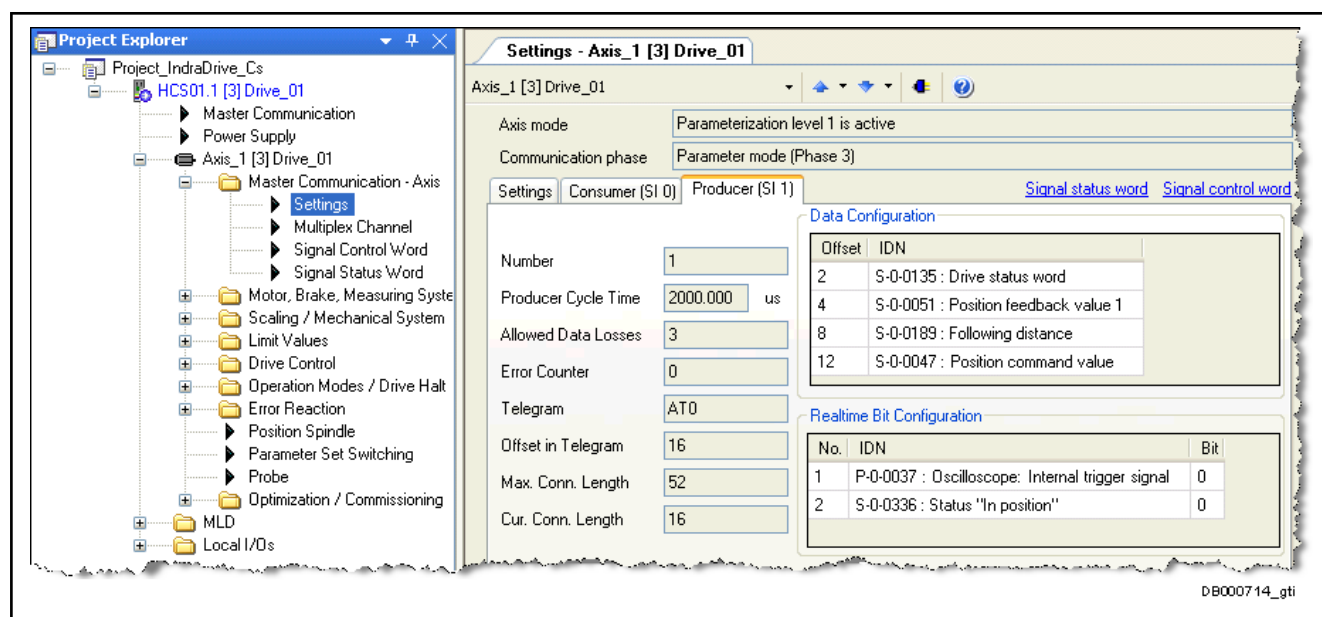


Fig. 5-16: "Master communication settings", "real-time input (AT)"

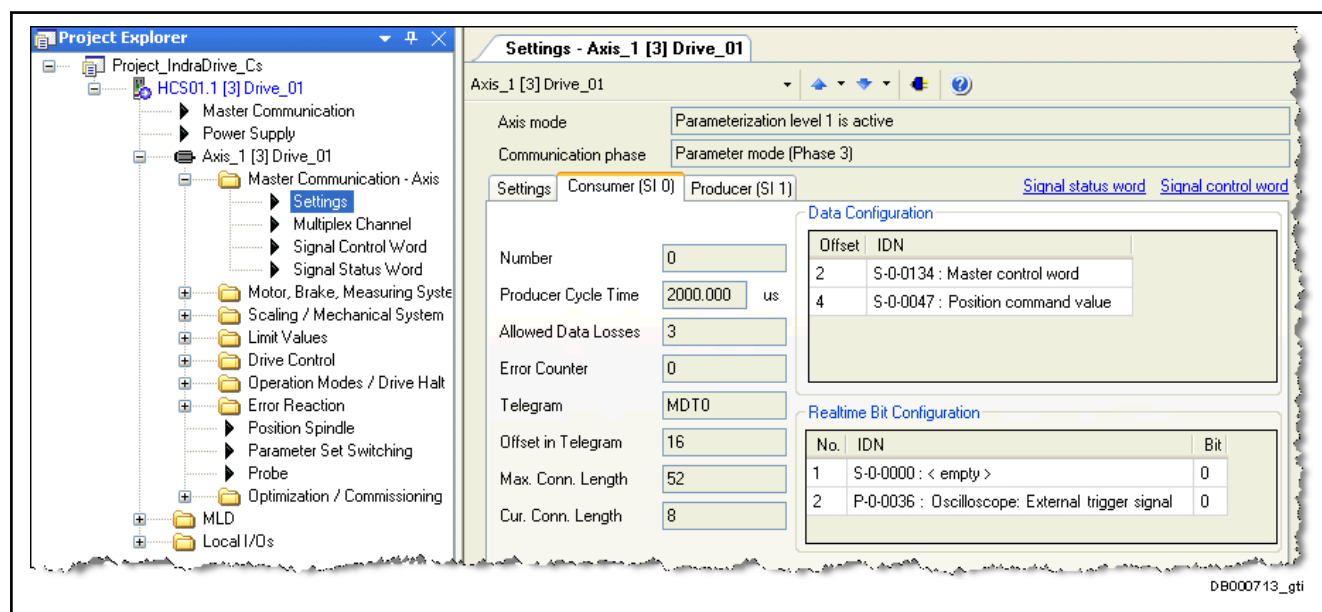


Fig. 5-17: "Master communication settings", "real-time output (MDT)"

Remote axes and "Rexroth Inline" I/O modules

Including drives or Sercos III devices in the bus

Drives or Sercos III devices (in this case "Rexroth Inline" I/O modules) can be included in the Sercos configuration "CCD: Basic settings" both in online and offline mode. To achieve this, a Sercos address and a logical drive number, or the number of the Sercos III device must be assigned.

MLD communication interfaces and data channels

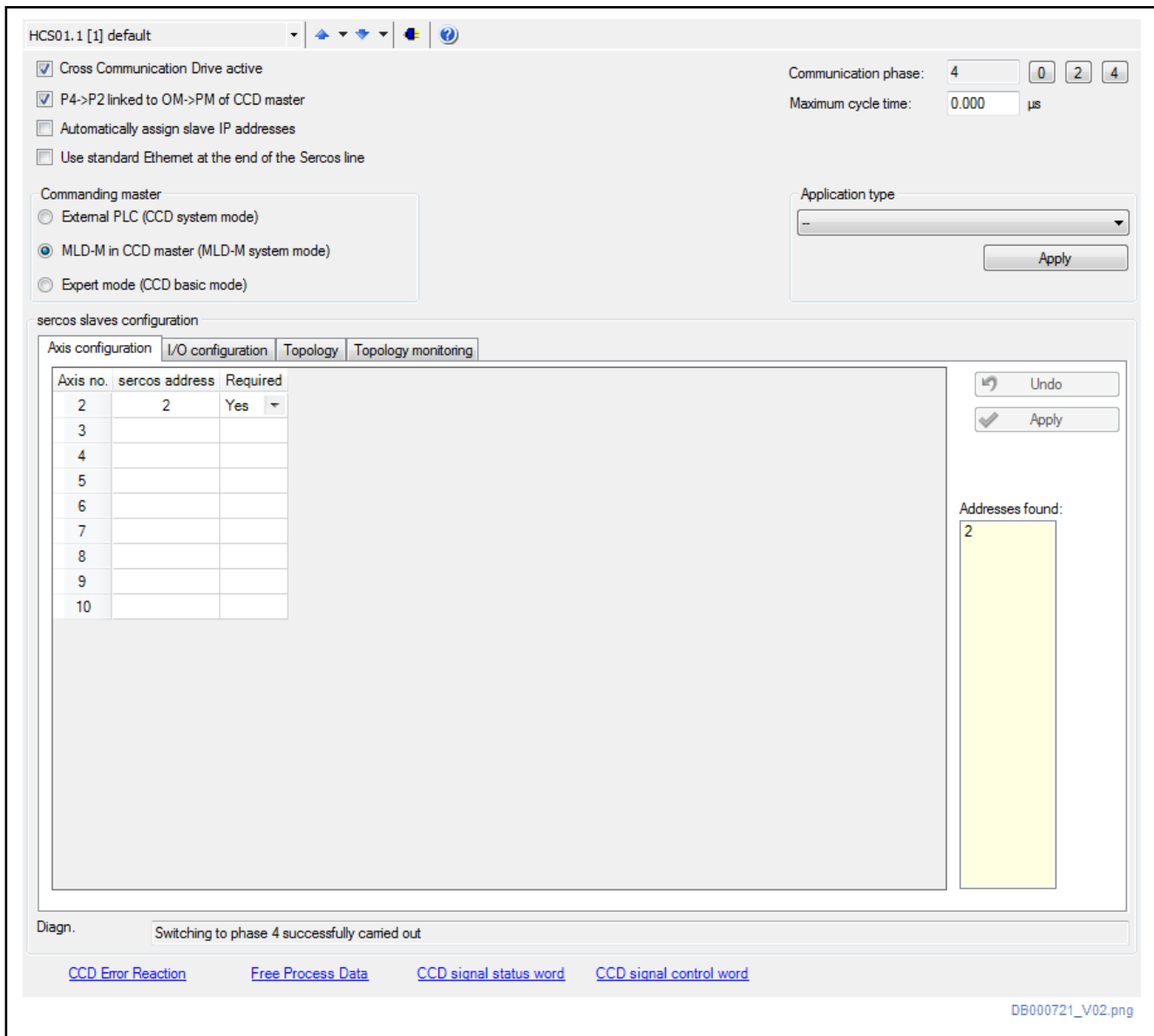


Fig. 5-18: Dialog "CCD: Basic settings"

A new Sercos address set or a Sercos address changed in the dialog (in the "Sercos slaves configuration" dialog section) is saved in the device using the "Apply" button.

The drives and "Rexroth Inline" I/O modules are then displayed in the project tree under the "SERCOS III (CCD)" branch as drives or "IO".

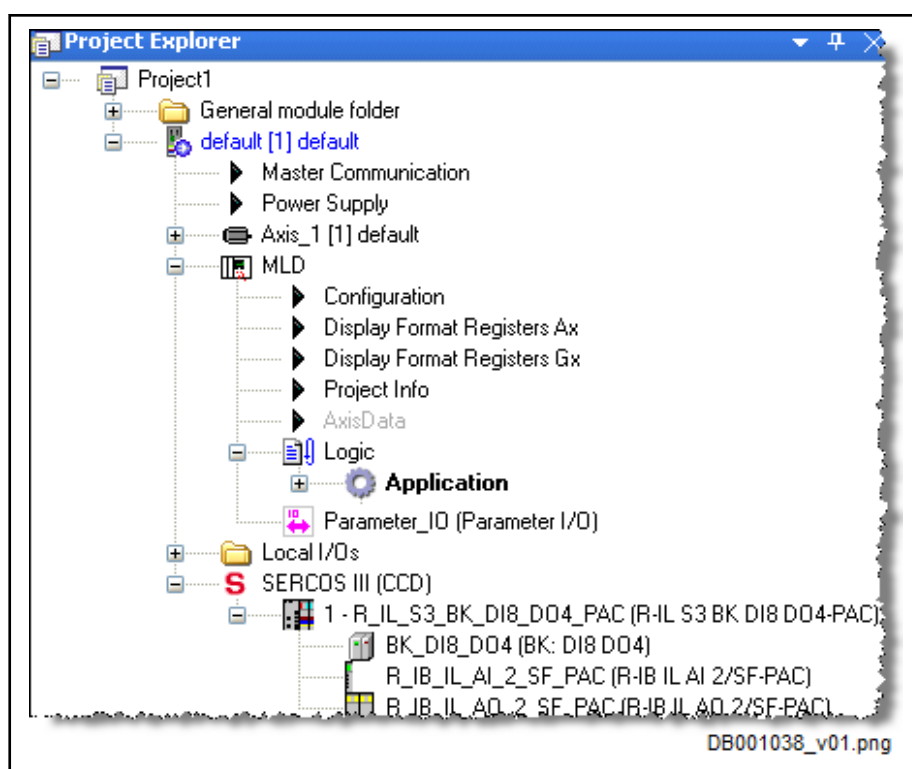


Fig. 5-19: Project explorer with Sercos III devices



MLD inputs coming from a Sercos III slave are set to zero in case the connection is interrupted.

Inputs/outputs of remote axes (Slave/Axis2 to 10)

For parameterizing inputs/outputs on remote axes, the "MLD-M I/O configuration" dialog is available in IndraWorks MLD (in the Project Explorer in the "MLD" folder within the tree structure of the MLD-M drive).

In its left half, the dialog provides some process image parameters to be selected; the desired parameter of the process image has to be entered. Its right half displays a selection of all inputs or outputs available on the slave; the corresponding register of the I/O module has to be entered.



The parameters set in this dialog are automatically added to the cyclic command values or actual values of the respective axis. A list of all cyclic parameters can be viewed in the CCD dialog.

The following examples show how to configure remote inputs/outputs for MLD-M.

Example 1

In the first example, the input double word 25 (P-0-1440) was assigned to the digital inputs of slave #2 via P-0-0303.



Please observe in which bits the corresponding terminals take effect.



P-0-0303 contains 32 bits and therefore has to be assigned to a 32-bit process image register, such as P-0-1440.

MLD communication interfaces and data channels

In addition, input word 2 (P-0-1392) was assigned to analog input 1 of slave #2 via P-0-0210.

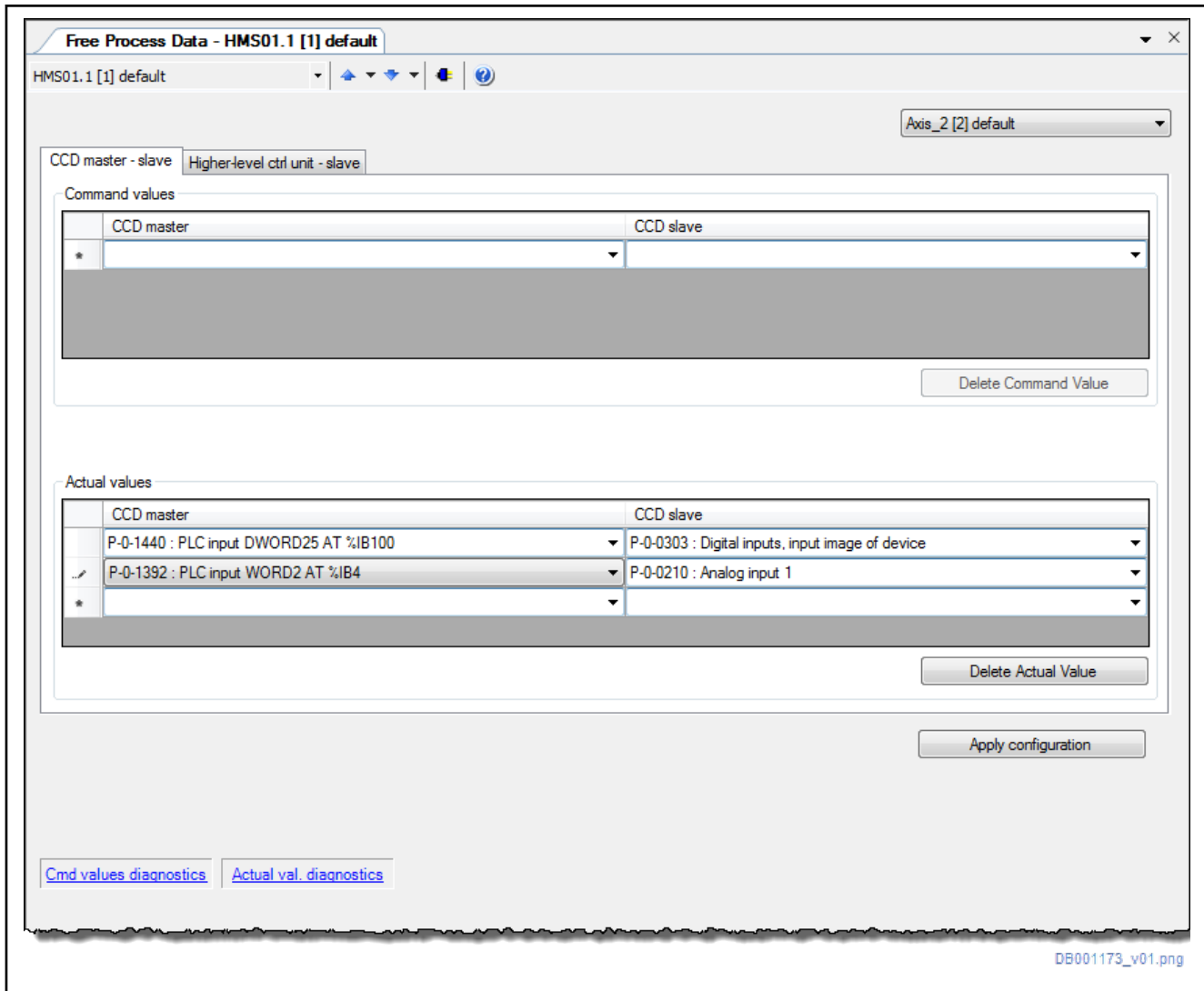


Fig. 5-20: Assigning inputs

Example 2 In the second example, output word 0 (P-0-1410) was assigned to the digital outputs of slave #2 via P-0-0304. The individual bits %QX2.0 et seq. control the outputs of the slave.



Please observe in which bits the corresponding terminals take effect.



It is possible that digital outputs of the slave control section are controlled by the PLC in the master and others by the slave drive itself.

In addition, output word 2 (P-0-1412) was assigned to analog output 1 of the slave via P-0-0139.

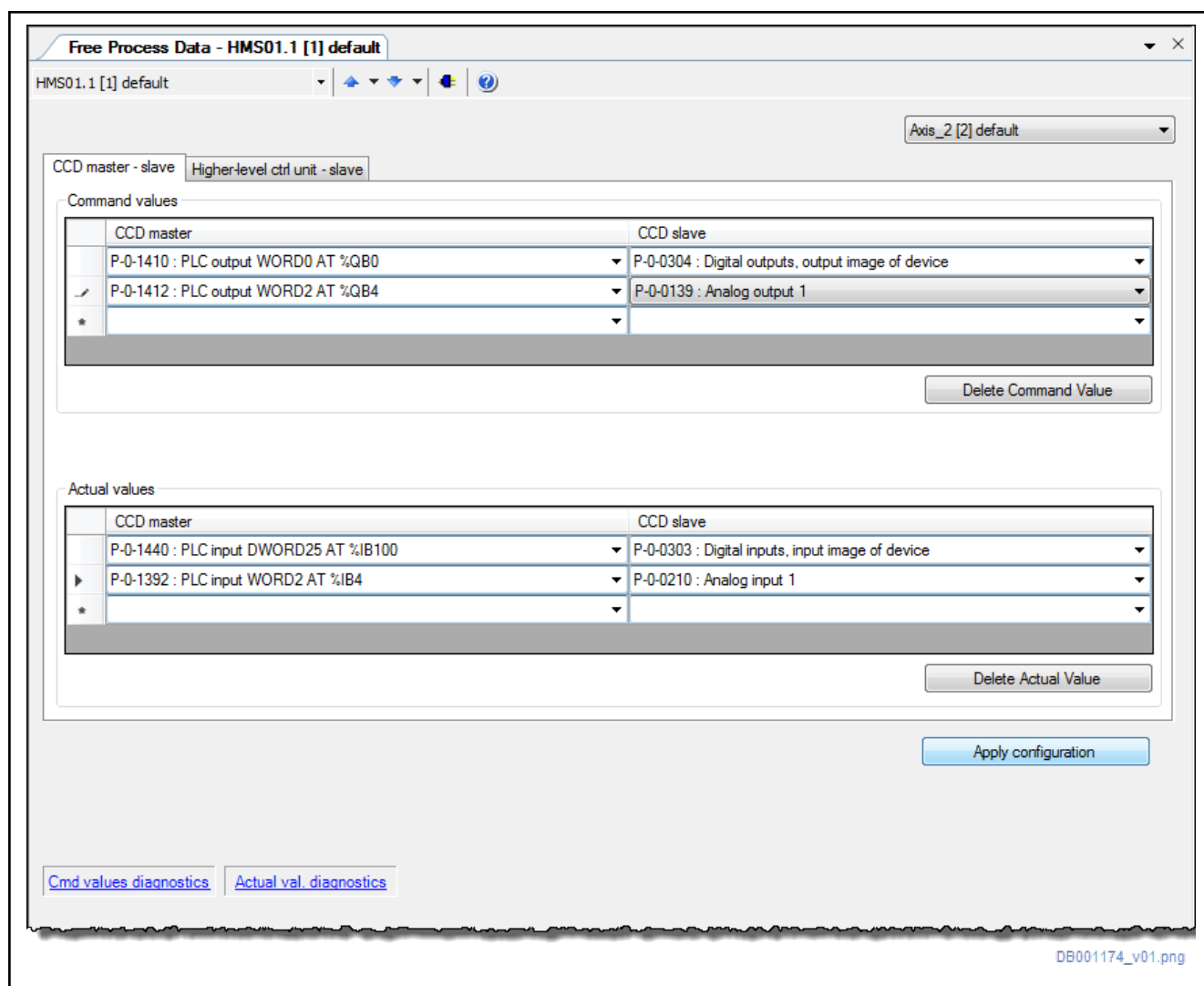


Fig. 5-21: Assigning outputs

Example 3

In the third example, the digital inputs of the local axis are read in via CCD by means of P-0-1441 and the digital outputs are read out by means of P-0-1413. This example is intended to show that using IndraMotion MLD-M also allows accessing the local I/Os via the CCD mechanism ("virtual slave"). If the CCD master is configured as a virtual slave, the local I/Os and the I/Os of remote axes are simultaneously updated; i.e., the digital inputs/outputs of the local axis are read in or out via Sercos III at the same time as the inputs/outputs of remote axes.



The same applies analogously to outputs on the local axis which have been configured via the CCD mechanism.

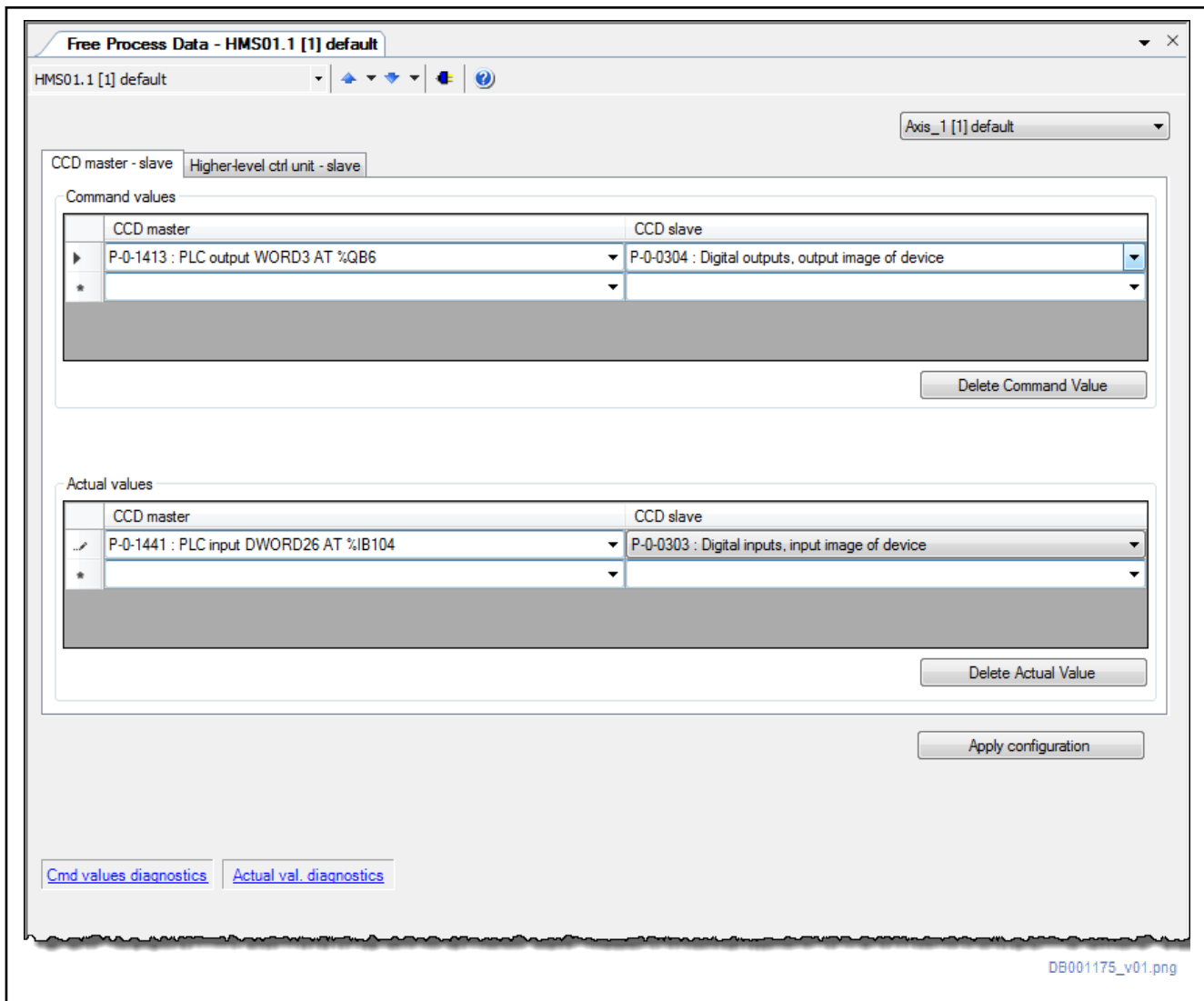


Fig. 5-22: Accessing local inputs/outputs via CCD (virtual slave)

Inputs/outputs of "Rexroth Inline"

The context menu **SERCOS (CCD master) ► SERCOS node configuration** is available in Project Explorer for parameterizing inputs/outputs on "Rexroth Inline" I/O modules ("Sercos module I/O mapping" tab in the dialog).

5.2.4 Cyclic data in MLD-M system mode

In the MLD-M system mode, the cyclic parameters of the axes are automatically configured from the fixed parameters for the motion channel and the settings for the user data in "AxisData", the I/O configuration and the free process data. The resulting cyclic parameters are only displayed but not parameterized. According to their function, there are different dialogs for parameterizing the cyclic data.

Summary of all cyclic numerical data:

- **Motion data**
The parameters of the motion channel are automatically added to the cyclic parameters. There are no inputs for this purpose.
- **I/O configuration**

The parameters required for exchanging data with remote I/Os can be entered in the "I/O configuration" dialog.

- **AxisData**

The parameters required for the user data can be defined in the "Axis-Data" dialog.

- **Free process data**

Other freely selectable cyclic parameters for specific requirements can be entered in the "Free process data" dialog.

- **Signal status bits and control bits**

The signal control or signal status word is permanently and cyclically configured in the motion data. 12 bits of the 16 bits possible for each of these are defined/reserved. The remaining 4 bits (bit 12 to bit 15) can be used as actual value or command value bits via "AxisData". This has to be set in the "AxisData" dialog. The CCD configuration includes a dialog displaying all cyclic data. All cyclic parameters are summarized in this dialog box.

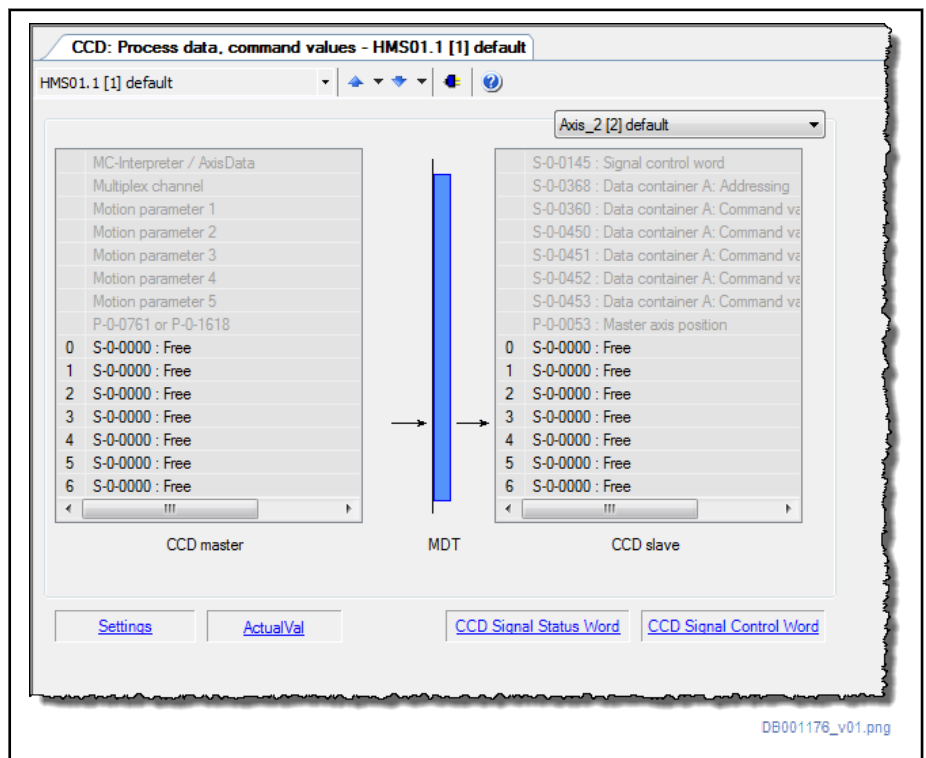


Fig. 5-23: Dialog box displaying all cyclic data

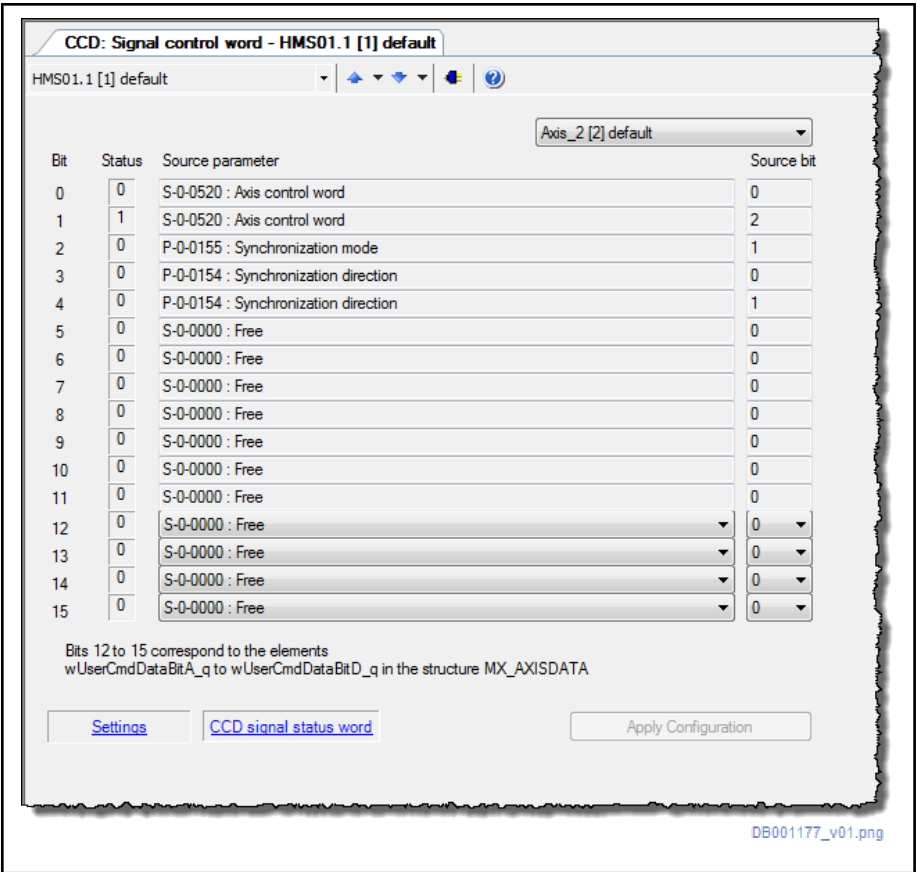


Fig. 5-24: Dialog displaying the resulting signal control word configuration

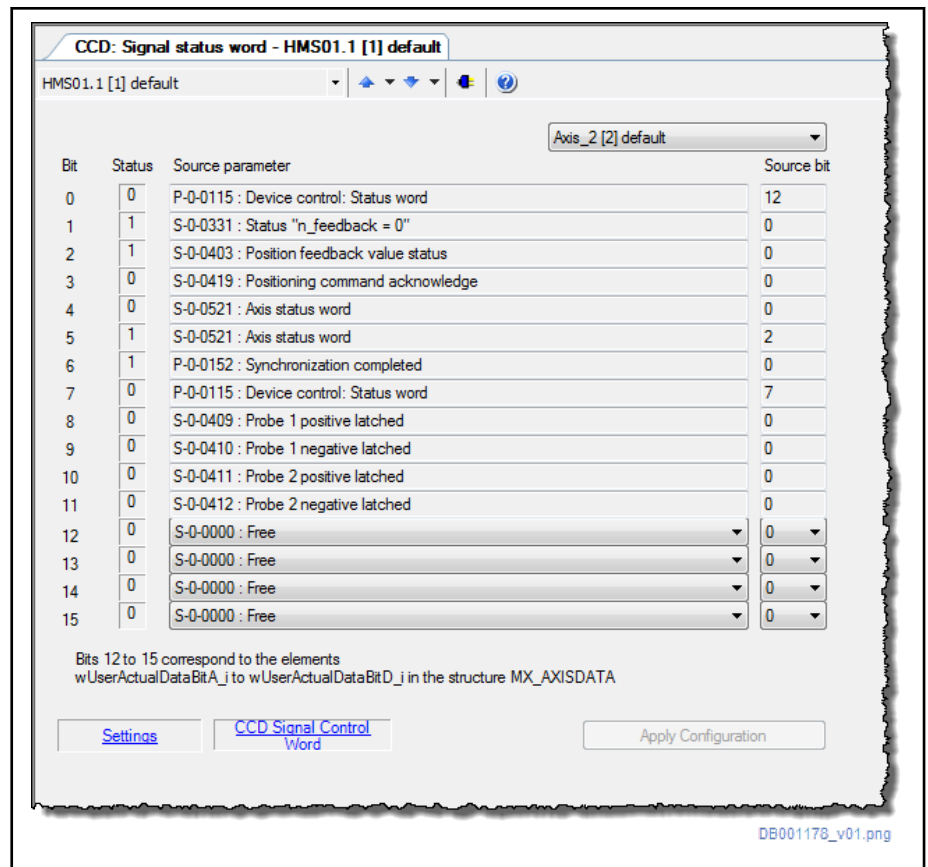


Fig. 5-25: Dialog displaying the signal status word

Pertinent parameters

The following parameters are used in conjunction with the data exchange between MLD and the drive.

For configuration, the following parameters are written via dialog:

- S-0-0187, List of configurable data in the cycl. actual value data channel
- S-0-0188, List of configurable data in the cycl. command value data channel
- P-0-1611, CCD: Configuration list signal status word
- P-0-1612, CCD: Configuration list signal control word
- P-0-1613, CCD: Assignment list signal status word
- P-0-1614, CCD: Assignment list signal control word
- P-0-1623, CCD: Configuration list master command values
- P-0-1624, CCD: Configuration list actual master values
- P-0-1625, CCD: Configuration list slave command values
- P-0-1626, CCD: Configuration list actual slave values

In the MLD-M master, the following parameters are available as data containers for "AxisData" (these parameters can be optionally used for diagnostic purposes):

- "P-0-1660, CCD: Status word compact I/Os"
- P-0-1660.x.y
 - P-0-1660.x.2, CCD: Device Control (C-Res)
 - P-0-1660.x.3, CCD: Connection-Control #1 (C-Con)

MLD communication interfaces and data channels

- P-0-1660.x.20, CCD: Resource-Control (C-Res)
- "P-0-1661, CCD: Drive status word, slave 1" to "P-0-1669, CCD: Drive status word, slave 9"
- P-0-1670, CCD: Active position feedback value, master
- "P-0-1671, CCD: Active position feedback value, slave 1" to "P-0-1679, CCD: Active position feedback value, slave 9"
- P-0-1680, CCD: Actual velocity value, master
- "P-0-1681, CCD: Actual velocity value, slave 1" to "P-0-1689, CCD: Actual velocity value, slave 9"
- P-0-1690, CCD: Torque/force feedback value, master
- "P-0-1691, CCD: Torque/force feedback value, slave 1" to "P-0-1699, CCD: Torque/force feedback value, slave 9"
- P-0-1710, CCD: Signal status word, master
- "P-0-1711, CCD: Signal status word, slave 1" to "P-0-1719, CCD: Signal status word, slave 9"
- P-0-1720, CCD: Signal control word, master
- "P-0-1721, CCD: Signal control word, slave 1" to "P-0-1729, CCD: Signal control word, slave 9"
- P-0-1810, CCD: Status word synchronization modes, master
- "P-0-1811, CCD: Status word synchronization modes, slave 1" to "P-0-1819, CCD: Status word synchronization modes, slave 9"
- "P-0-1730, CCD: Command value data container 1.0 4Byte" to "P-0-1739, CCD: Command value data container 1.9 4Byte"
- "P-0-1740, CCD: Command value data container 2.0 4Byte" to "P-0-1749, CCD: Command value data container 2.9 4Byte"
- "P-0-1750, CCD: Command value data container 3.0 4Byte" to "P-0-1759, CCD: Command value data container 3.9 4Byte"
- "P-0-1760, CCD: Command value data container 4.0 4Byte" to "P-0-1769, CCD: Command value data container 4.9 4Byte"
- "P-0-1770, CCD: Actual value data container 1.0 4Byte" to "P-0-1779, CCD: Actual value data container 1.9 4Byte"
- "P-0-1780, CCD: Actual value data container 2.0 4Byte" to "P-0-1789, CCD: Actual value data container 2.9 4Byte"
- "P-0-1790, CCD: Actual value data container 3.0 4Byte" to "P-0-1799, CCD: Actual value data container 3.9 4Byte"
- "P-0-1800, CCD: Actual value data container 4.0 4Byte" to "P-0-1809, CCD: Actual value data container 4.9 4Byte"
- "P-0-1820, CCD: Command value data container 1.0 2Byte" to "P-0-1829, CCD: Command value data container 1.9 2Byte"
- "P-0-1830, CCD: Command value data container 2.0 2Byte" to "P-0-1839, CCD: Command value data container 2.9 2Byte"
- "P-0-1840, CCD: Command value data container 3.0 2Byte" to "P-0-1849, CCD: Command value data container 3.9 2Byte"
- "P-0-1850, CCD: Command value data container 4.0 2Byte" to "P-0-1859, CCD: Command value data container 4.9 2Byte"
- "P-0-1860, CCD: Actual value data container 1.0 2Byte" to "P-0-1869, CCD: Actual value data container 1.9 2Byte"

- "P-0-1870, CCD: Actual value data container 2.0 2Byte" to "P-0-1879, CCD: Actual value data container 2.9 2Byte"
- "P-0-1880, CCD: Actual value data container 3.0 2Byte" to "P-0-1889, CCD: Actual value data container 3.9 2Byte"
- "P-0-1890, CCD: Actual value data container 4.0 2Byte" to "P-0-1899, CCD: Actual value data container 4.9 2Byte"

The following parameters are automatically read from the axes and stored in the "AxisData" structure:

- S-0-0040, Velocity feedback value
- S-0-0084, Torque/force feedback value
- S-0-0144, Signal status word
- S-0-0386, Active position feedback value
- S-0-0135/P-0-0115 (see "wDriveStatus_i")
- S-0-1045, Sercos III: Device Status (S-Dev)
- P-0-0089, Status word synchronization modes

5.3 Sercos I/O

The configuration and use of Sercos I/O modules is completely supported by a Sercos III master integrated in IndraMotion MLD-2G, i.e., the nodes (I/O modules) are both configured and automatically integrated in the process images of the inputs/outputs. This leads to fundamental changes in comparison to use with MLD-1G.

The integrated Sercos III master is a component used in all control systems, i.e., the IndraMotion MLD-2G control system generally supports all "Inline Block I/O" and "Inline Modular I/O" supported by the MLC and MTX systems. However, differences can arise due to the "Big Endian" data format.



The number of configurable I/Os currently supported by the drive has to be taken into consideration. The drive supports a maximum of 4 bus couplers with up to 63 modules. The number is accordingly restricted in IndraWorks MLD.

Configuring the Sercos I/O

The Sercos III master (MLD master) integrated in IndraMotion MLD-2G complements the CCD master implemented in the drive system. While the I/O modules are configured via the MLD master, the CCD master includes all nodes in the CCD ring.

The configuration via the CCD master only includes the addressing of the bus couplers. The Sercos I/O modules and the function of the Sercos I/O modules are configured independently of the CCD master in dialogs made available by IndraWorks, or by directly writing the corresponding parameters of the I/O modules.

Interaction of the Sercos configuration in MLD (IO) and the CCD configuration:

- IndraWorks automatically configures the MLD master with the data of the CCD master. This ensures the consistent addressing of the nodes.
- During phase switch, the CCD master checks whether or not the configuration of the Sercos I/O modules is correct. The CCD master also checks the addressing of the Sercos I/O. The MLD master checks whether or not the type and equipment of the Sercos I/O are correct. The latter can only happen when the communication via the Sercos ring is possible.

**Process images of the inputs/
outputs**

In contrast to the local inputs/outputs, the addresses in the process images of the inputs/outputs, which are assigned to the Sercos I/O, are not fixed. The address ranges, starting with %IB0 and %QB0, are reserved by the local inputs/outputs; therefore, the addresses from the unassigned range after the local I/Os are consecutively assigned when the Sercos I/O modules are created.



With the "I/O mapping" it is possible to assign, e.g., project variables used by the application, to the input, output and memory addresses of the control device. Furthermore, address values can be changed and fixed.



In the firmware, no parameters are available for the process images of the inputs/outputs of the Sercos I/O. Therefore, the Sercos I/O cannot be used by parameter links in the drive via the master communication, or for the inputs and outputs of remote and local axes.



The process images of the inputs/outputs of the Sercos I/O are only updated when the CCD ring is in communication phase 4. Otherwise, all inputs of the Sercos I/O are set to zero.

6 Visualization, engineering and connection via OCI

6.1 Overview

- Supported functions** For visualization and operation of the functions realized with the drive-integrated PLC, the following options are provided:
- Reading and writing parameters
 - Reading and writing PLC variables

Interfaces for connecting operator terminals



Apart from Bosch Rexroth operator terminals (HMI), it is possible to connect HMIs of other manufacturers to IndraDrive controllers. This documentation only describes how to connect the Bosch Rexroth HMIs.

The figure below contains an overview of the basic possibilities for connecting HMIs:

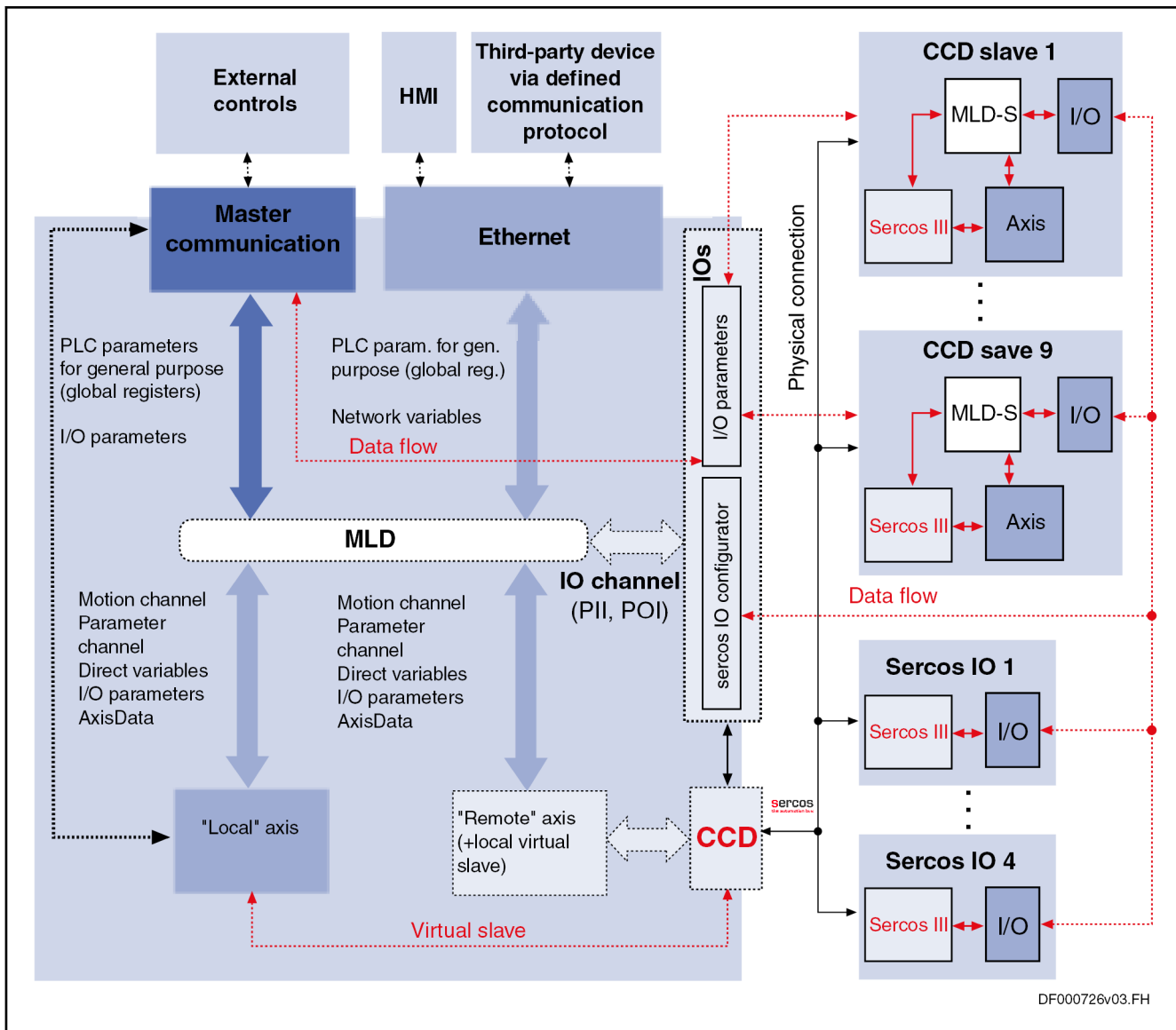


Fig. 6-1: Interfaces for HMI connection

The following communication interfaces are supported to communicate with HMIs, external inputs/outputs or sensors and a higher-level control unit, but also for communication with IndraWorks MLD (incl. WinStudio):

- **Ethernet interface** [note: TCP/IP, FTP (in preparation)]
- **Master communication interface** (PROFIBUS, DeviceNet, etc.)
Access to external inputs/outputs or sensors that are connected to the higher-level control unit via the master communication.
- **Digital and analog inputs/outputs**
Access to external inputs/outputs or sensors that are connected to the drive.

The following **hardware requirements** are necessary for connecting an operator terminal (HMI):

- Communication interface for connecting an HMI:
 - **Ethernet interface** [note: TCP/IP, FTP (in preparation)]

- **Master communication interface** (for field bus and Sercos drives)

The following **software requirements** are necessary for connecting an operator terminal (HMI): IndraWorks is required for programming an operator terminal and generating the symbol configuration (for accessing parameters and variables).

To commission the HMIs of Bosch Rexroth, the "WinStudio" component has to be installed when IndraWorks is installed. (It is possible to subsequently install a component by restarting the installation. In this case, select "Change installation" in the advanced options.

6.2 Interfaces for connecting communication partners

6.2.1 Ethernet interface (TCP/IP)

Brief description

The drive-integrated PLC and the drive parameters can be accessed via the standard Ethernet interface with TCP/IP protocol.



TCP/IP communication works according to the CSMA/CD method (Carrier Sense Multiple Access/Collision Detection). Therefore, the time behavior is not deterministic. The reaction times depend on the network structure and the data volume. For an estimation of the velocities, please contact your network administrator.

Parameters involved

The following parameters are used in conjunction with TCP/IP communication:

- P-0-1530, Engineering: MAC address
- P-0-1531, Engineering: IP address
- P-0-1532, Engineering: Network mask
- P-0-1533, Engineering: Gateway address
- ...

Setting the IP address

The relevant drive parameters have to be set according to the devices being connected in order to establish a connection between the drive controller and a PC or operator terminal. This means the IP addresses of the individual devices have to be specified according to the network masks so that the nodes belong to a single network.

The IP address can be set via

- the standard control panel at the drive controller or
- IndraWorks MLD

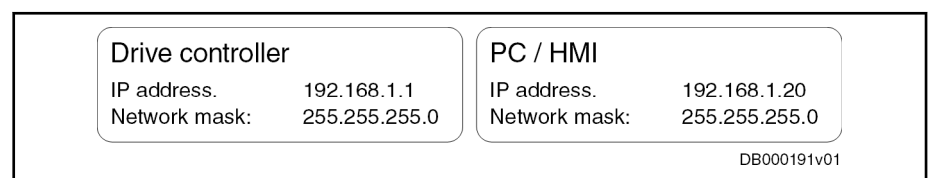


Fig. 6-2: Setting the IP addresses of the individual devices according to the network masks

Setting IP address, network mask and default gateway via standard control panel

IP address, network mask and default gateway can be parameterized via the control panel at the drive controller; this is only possible in parameter mode. It

might possibly be necessary to switch to the parameter mode via the control panel; the paragraph below describes how to do this.

1. Switch on the control voltage of the drive controller.
2. Simultaneously press the "ESC" and "Enter" keys on the control panel for approx. 8 seconds.
3. Use the "Up"/"Down" arrow keys to select "2. Comand" and confirm your selection with the "Enter" key.
4. If you have to switch to parameter mode, use the "Up"/"Down" arrow keys now to select "2.11 > PM" and start the command with the "Enter" key.
5. Now select "2.3 Kom. Ethernet" and confirm your selection with the "Enter" key.
6. The IP address can now be set under "2.3.1 P-0-1531".



The individual octets (bytes) are applied by pressing the "Enter" key. This procedure can be aborted with "Esc".

7. The network mask can be set under "2.3.2 P-0-1532".
8. The default gateway can be set under "2.3.3 P-0-1533".



In order that the parameter settings of desired IP address, network mask and default gateway take effect, the control voltage of the drive controller has to be switched off and back on.

Setting IP address, network mask and default gateway via IndraWorks

IP address, network mask and default gateway can be parameterized via IndraWorks MLD.

1. Start IndraWorks MLD; when starting, select "IndraDrive (Ethernet)" as connection type.
2. In IndraWorks MLD, open the tree in the project explorer window.
3. In the project explorer window, right-click on the drive controller and select "IP settings" in the context menu.

In the work area, the "IP settings" window opens.

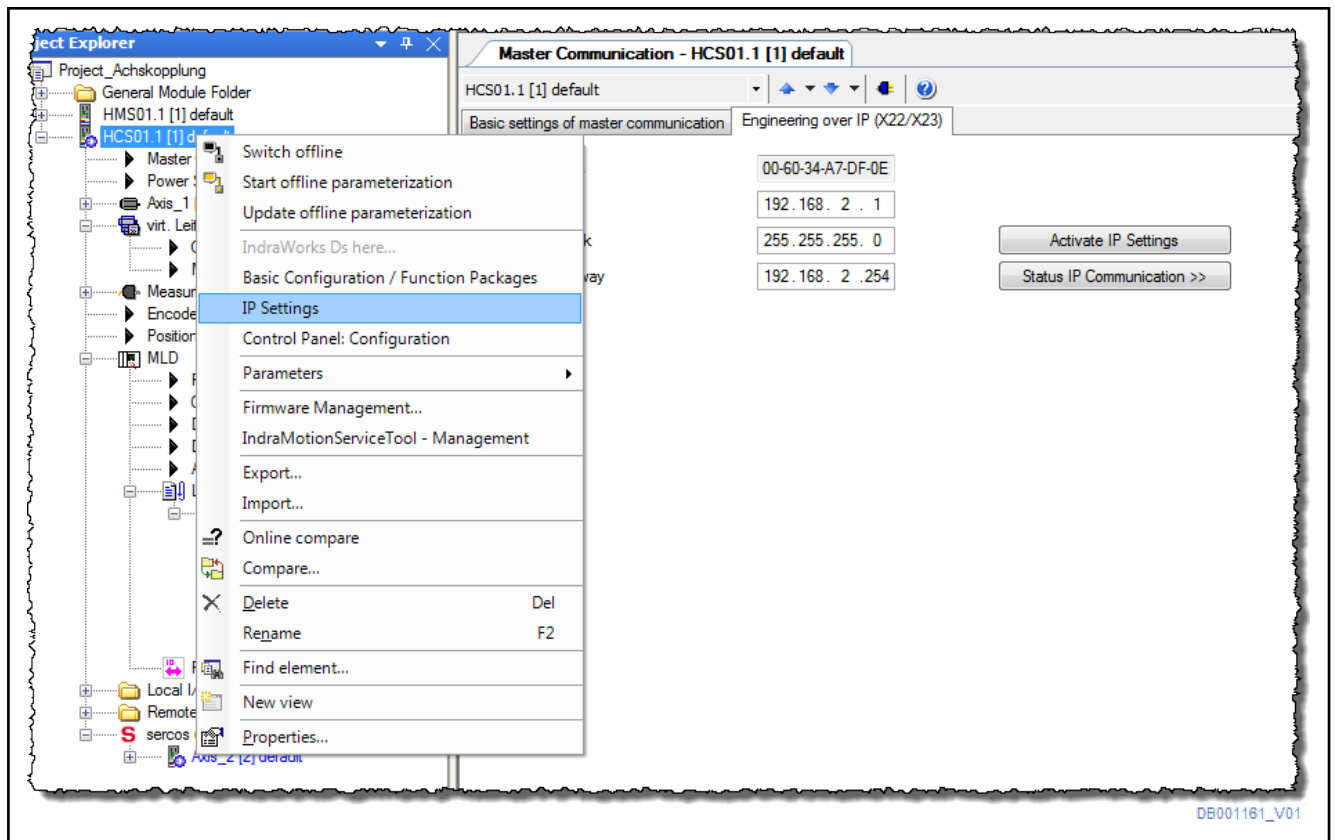


Fig. 6-3: IP settings with IndraWorks

4. In the "Engineering port" tab, now parameterize the desired IP address, network mask and the default gateway.
5. Accept the IP settings by clicking **Activate IP settings**.

6.2.2 Master communication interface

Brief description

Rexroth IndraMotion MLD uses the same master communication interfaces that are available for the drive without integrated PLC.

The basic distinction is which characteristic of the PLC is used:

- **IndraMotion MLD as intelligent servo axis** (-> external control unit has control)
- **IndraMotion MLD as stand-alone Motion Logic Control** (-> internal PLC takes over control)

Parameters involved

The following parameters are used in conjunction with the master communication:

- P-0-1368, PLC Global Register AL0
- P-0-1389, PLC Global Register GL0
- P-0-1370, PLC Global Register G0
- P-0-1371, PLC Global Register G1
- ...
- P-0-1385, PLC Global Register G15
- P-0-1390, PLC input WORD0 AT %IB0

- P-0-1391, PLC input WORD1 AT %IB2
- ...
- P-0-1397, PLC input WORD7 AT %IB14

Function

Access via the master communication interface is basically identical to access via digital inputs/outputs at the control section or access via an optional module for digital inputs/outputs. It is only the update rate that differs because the input and output data of the interface are processed in the master communication clock (cf. S-0-0001) and all other inputs/outputs are processed in the position controller clock.

Instructions for use

During commissioning, "P-0-1367, PLC configuration" has to be used to define whether the external master (e.g., PLC or NC) or the internal PLC will control the drive.

6.3 Connecting IndraControl V

6.3.1 Introduction and overview

Product presentation

VR 21 The "IndraControl VR 21" compact operator terminals are operator and visualization terminals for operating and monitoring machines. Depending on the application, actions can be triggered at the machine and/or states can be displayed with "IndraControl VR 21" by means of operating screens (write parameters / read parameters and access to PLC variables). [The operating instructions "Rexroth IndraControl VR 21 Operator Panel" (mat. no.: R911339476) contain detailed information regarding "IndraControl VR21".]

The operating screens are configured using Rexroth WinStudio (related documentation: "IndraWorks 14VRS, WinStudio 7.3+SP4", mat. no.: R911341585).

The "IndraControl VR 21" devices are available with different screen diagonals as single-touch or multi-touch screens. The devices are suitable for a number of different applications due to their compact design.

The voltage supply is effected via an external 24 V power supply unit, such as "IndraControl VAP 01.1" (related documentation: "Rexroth IndraControl VAP 01.1, Power Supply Unit", mat. no.: R911339613)

Communication with a parent control takes place via Ethernet interface. "IndraControl VR 21" moreover has two USB host interfaces and a slot for an SD memory card.

VEH / VEP The "IndraControl VEP" embedded operator terminals are **PC-based machine operator terminals** which - depending on the particular application - can trigger actions at the machine or display states (write parameters/read parameters and access to PLC variables). In addition, the "IndraControl VEP" operator terminals can assume control functionalities.

- The "IndraControl VEP"/"IndraControl VEH" embedded terminals are PC-based machine operator terminals which, depending on the application or configuration, can also fulfill control functionalities. "IndraControl VEP" features up to two PC/104 slots for plug-in cards, depending on the type.
- All "IndraControl VEP" devices are suitable for the food industry.

- The embedded terminals are available in different variants. They mainly differ in their display size.
- "IndraControl VEP 30.2", 40.2 and 50.2 devices come with a touchscreen. "IndraControl VEH 30.2" is operated via touchscreen and keys.
- Available interfaces for "IndraControl VEP" devices:
 - 1 external VGA port (15-pin, HD-Sub)
 - 2 USB ports (type A)
 - 2 Ethernet ports (RJ 45, 10/100 Base-T)
 - 1 PS/2 port (serial interface for keyboard/mouse)
 - 1 serial standard interface RS232 (9-pin; D)

For more information on "IndraControl VEH" / "IndraControl VEP", please refer to the Project Planning Manual "Rexroth IndraControl VEP / VEH", mat. no.: R911309682.

VH2110 The "IndraControl VH21" hand-held terminals are portable operator and visualization terminals for the operation, set-up, parameterization and diagnosis of a control unit connected via Ethernet. As safety functions, a 3-level enabling button as well as an EMERGENCY STOP button with two-circuit design are installed. A USB memory stick, mouse or keyboard can be connected via an USB OTG interface. [The operating instructions "IndraControl VH 2110.01 Hand-Held Terminal" (mat. no.: R911346750) contain detailed information regarding "IndraControl VH21".]

The hand-held terminal is connected via a connection module (related documentation: "IndraControl VAC 30.2, VAC 31.1, VAC 05.1 Connection Modules", mat. no.: R911338409)

All "IndraControl V" devices comprise specific operating systems. For the description of the operating systems, please refer to the Project Planning Manual "Rexroth IndraControl V-Devices Operating Systems", mat. no.: R911343901

Requirements

Drive firmware	To display PLC variables of the drive-integrated PLC (Rexroth IndraMotion MLD) and parameters, at least IndraDrive firmware FWA-INDRV*-MPx-17VRS-MS has to be used.
Device version	Use the device version that matches the drive firmware. For operating the "IndraControl VR21", "IndraControl VH21" HMI devices, the MPx-20 drive firmware is required.
Symbol configuration	If drive PLC variables should be displayed, a symbol configuration has to be created in the programming system. To display parameters (S- and P-parameters), the drive firmware provides custom-made symbol configurations. All kinds of parameters with data type have been entered in these symbol configurations. These custom-made symbol configurations provide the following options: • Displaying and editing useful data • Displaying the data statuses • Displaying maximum values • Displaying parameter names • Displaying parameter units

	 The symbol configuration for accessing drive parameters (S- and P-parameters) depends on the firmware!
Ethernet communication	Ethernet communication requires a control section with an Ethernet Engineering port. Communication via Ethernet with the TCP/IP protocol is possible with the drive firmware FWA-INDRV*-MPx-17VRS-MS and above.
Other requirements	• Installation of IndraWorks MLD with the current version of the "WinStudio" component • A device of the IndraControl V series with current firmware

6.3.2 Configuring the symbols for variable/parameter access

Brief description

Function	The symbols are required to display or edit PLC variables and drive parameters with an operator terminal (HMI). • The symbol configuration for accessing drive parameters (S- and P-parameters) depends on the firmware and is made available by Bosch Rexroth. • A symbol configuration for accessing PLC variables of MLD has to be generated by the user by means of the symbol configuration. The symbol configuration allows the variables of IndraMotion MLD and the drive parameters to be accessed using the variable name or the direct variables.

Generating the symbol configuration for accessing PLC variables of MLD

To generate a symbol configuration for accessing PLC variables of MLD, call the corresponding function in IndraWorks MLD as follows:

1. Select the **MLD ► Logic ► Application ► Symbol configuration** branch.
2. Double-click "Symbol configuration".
⇒ The "Symbol configuration" window opens.

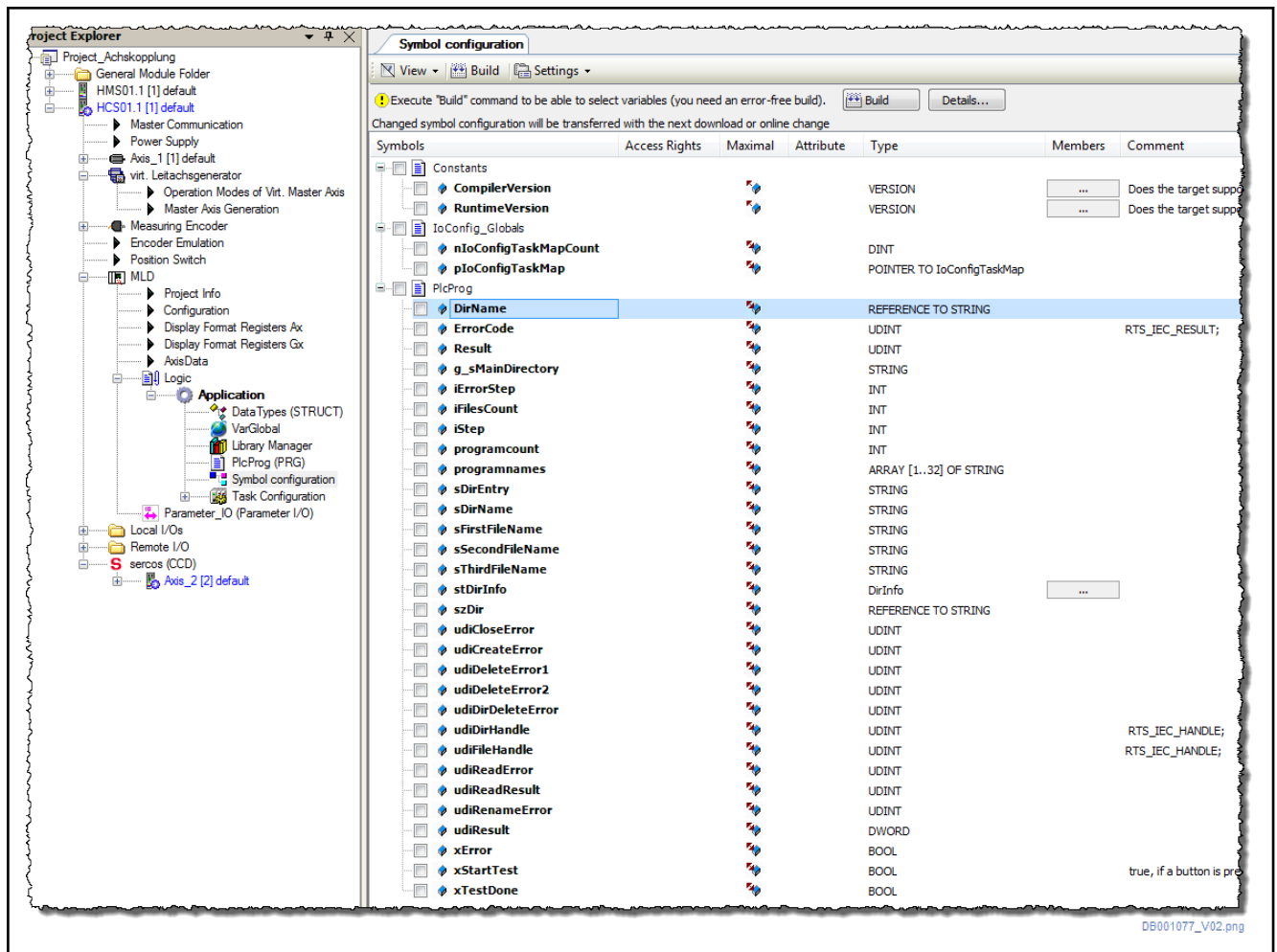


Fig. 6-4: Generating the symbol configuration for accessing PLC variables of MLD

6.3.3 Connecting IndraControl VE* via Ethernet

To establish the connection between a drive and IndraControl VE*, the following steps are required:

1. Set the IP address of the drive controller (see "Setting the IP address").
2. Add an IndraControl VE* to the project.
3. Call the context menu (as shown in the screenshot) and select **Add driver**

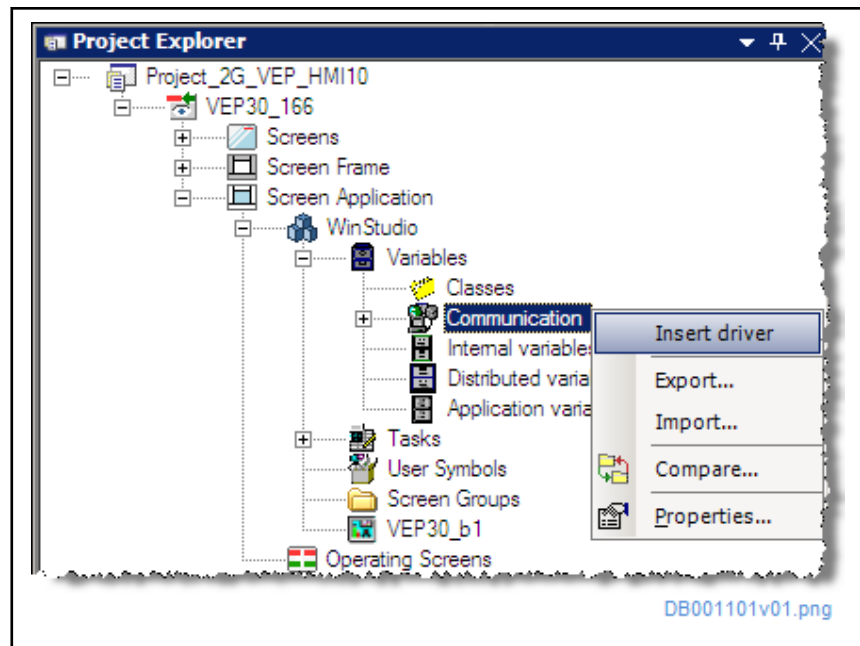


Fig. 6-5: Adding a communication driver
The "Communication driver" dialog opens.

4. Select "BR_WS" to use Com-Server services.

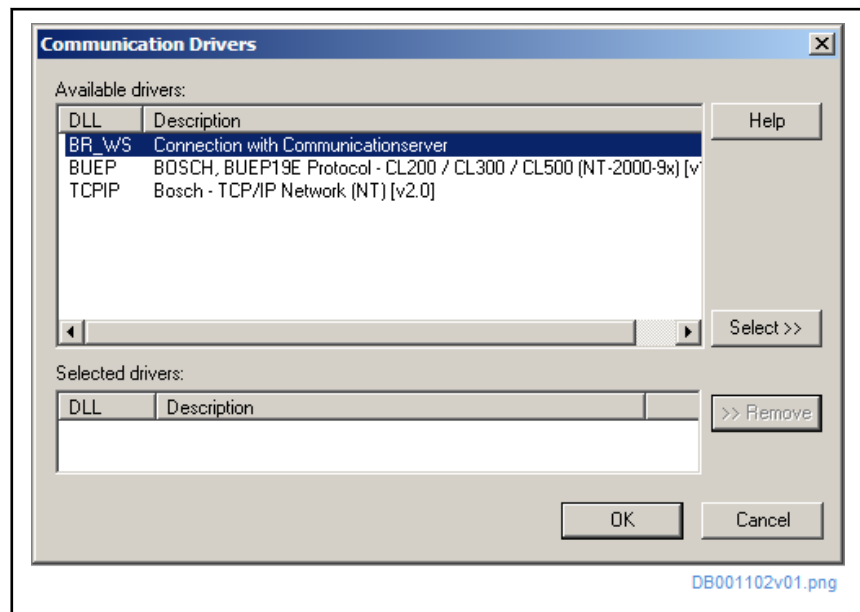


Fig. 6-6: Selecting the "BR_WS" "communication driver"

5. Use the context menu of the driver that was just added to call the "Properties" dialog.

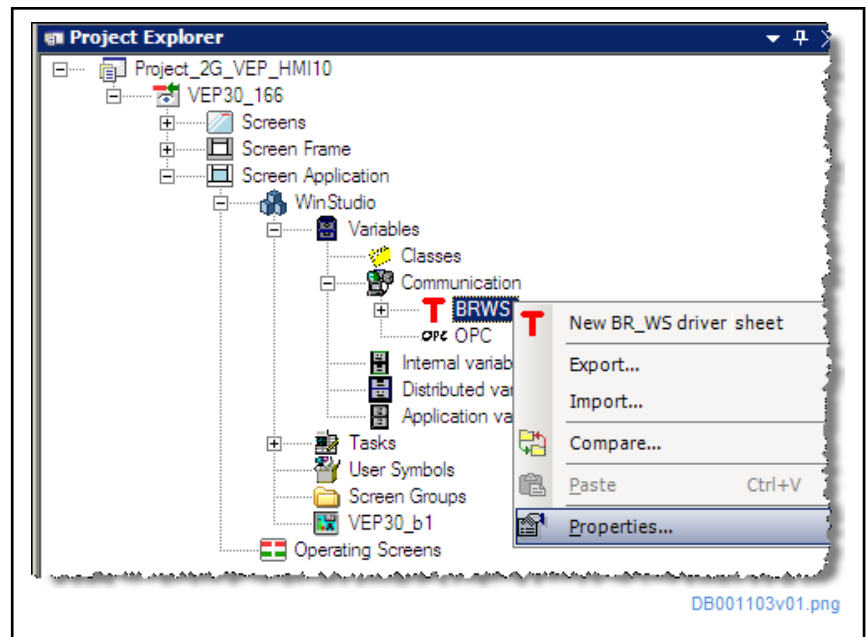


Fig. 6-7: Calling the "Properties" dialog of "BR_WS"

6. Make the following settings in the "Properties" dialog of "BR_WS":

- "Serial encapsulation": "None"
- "Timeout in ms": Enter a timer monitor for the communication connection. Recommended value for a communication connection to the drive controller: ≥ 1000 ms.
- "IP address": TCP/IP address of target control unit
- "No connection check (CE)": "0" (the communication connection should also be monitored in devices designated CE).

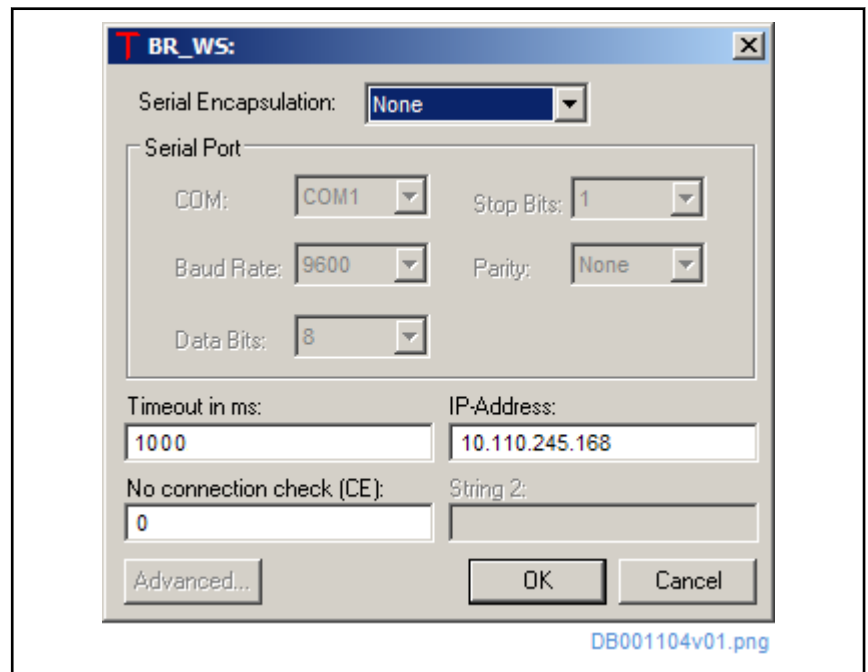


Fig. 6-8: Properties dialog of "BR_WS"



The IndraWorks online help contains more information on the communication driver "BR_WS" (Bosch Rexroth WinStudio driver) (e.g., configuring the driver sheets, supported data types, etc.).

6.4 Open Core Interface for drives

With "Open Core Interface for drives", Bosch Rexroth provides interfaces which allow for an easy connection of the machine automation to the higher-level IT automation.

For different programming environments, compatible function libraries are provided in an SDK (Software Development Kit). In these libraries, functions for direct access to all data and functions of the drive are contained.

"Open Core Interface for drives" provides the following options:

- **Smart devices**

By using the smart devices and the installed apps, new operating concepts for systems can be developed.

The apps allow for access to all hardware functions of the smart devices such as the camera or the acceleration sensors. So they offer the entire control comfort of smart devices for the commissioning, operation and generation of diagnostic messages of IndraDrive and HydraulicDrive devices.

Open Core Interface for drives currently supports the Android, iOS and Windows Phone operating systems.

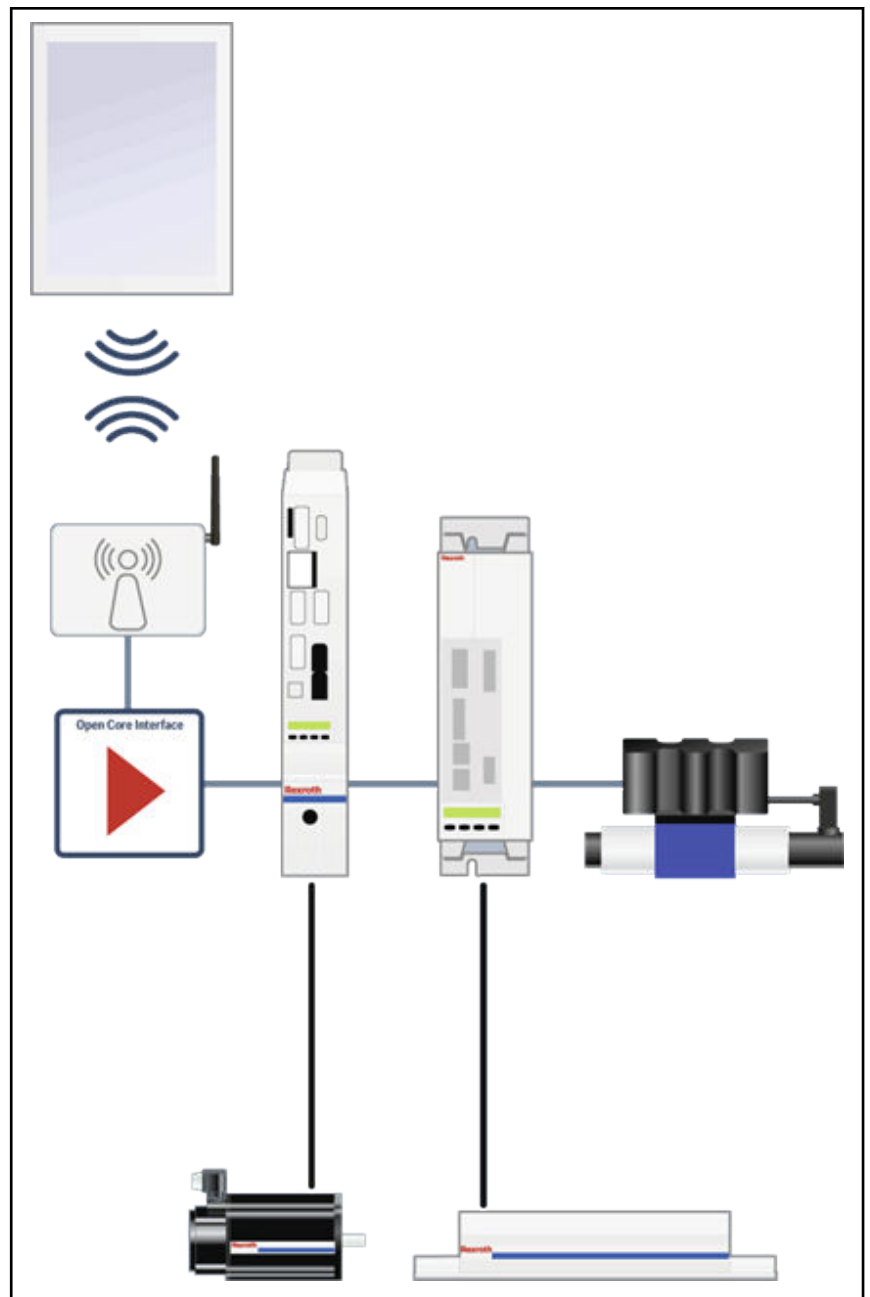


Fig. 6-9: Commissioning, operating and generating diagnostics of IndraDrive and HydraulicDrive devices via smart devices

- **IT automation**

Open Core Interface in the IT automation domain refers to the use of PC-based solutions in the automation environment of a production machine.

For fast data exchange, connections to Windows and Linux-based development environments are provided in which high-level languages such as Java, C, C++ and C# are used for programming.

- **Rapid Control Prototyping**

Open Core Interface provides all options for Rapid Control Prototyping, a design method for the closed-loop and open-loop control development. It is used for the early development of processes without a real

machine and thus considerably contributes to cost-optimized and risk-minimized development. Rapid Control Prototyping is supported by LabVIEW, Simulink™, MATLAB®.



Information and support regarding "Open Core Interface" is available in the [forum](#). Here, you can also ask members and the Bosch Rexroth experts.

6.4.1 Target platforms

With "eal_sdk" ("Easy Automation Library" "Software Development Kit"), Bosch Rexroth provides programming interfaces for the following target platforms:

- PC with Windows/PC-based control with LabVIEW
- PC with Linux as operating system
- (Mac OS upon request)
- Smart device with Android, iOS or Windows phone as operating system
- Linux-based controller (e.g., BeagleBone Black®, Raspberry pi®,...)

6.4.2 Supported devices

With "eal_sdk" you can create applications for configuring and controlling Bosch Rexroth drive controllers and frequency converters. Devices supporting the "Sercos Internet Protocol" (S/IP) (devices with Multi-Ethernet interface) are supported:

- Electric drive controllers of the IndraDrive Cs, IndraDrive C, IndraDrive M, IndraDrive Mi, IndraDrive ML ranges [with all control sections available in the product range (ECONOMY, BASIC and ADVANCED, single- and double-axis devices)]
- Hydraulic drive controllers "Integrated Axis Controller (IAC)", "Hydraulic Motion Control (HMC)"
- Frequency converters of EFC x610 type

6.4.3 Application options

One or several drives can be configured and controlled in star or line topology using the master communication interface, the Sercos master interface or the Engineering port.

With "Open Core Interface for drives", the following applications can be used:

- "Open Core Interface for drives" can replace a conventional control unit. Both the master communication and the commanding of the drive controller are performed by "Open Core Interface for drives".



The master communication has to be disabled for this use case.

- "Open Core Interface for drives" can be used in parallel with an active master communication (Sercos, PROFINET, EtherCAT with EoE, EtherNet/IP, Ethernet POWERLINK). In this case, the following applications may be implemented:
 - As HMI interface; commanding is realized via an external PLC via the master communication protocol.
 - As commissioning tool for loading a parameter file, replacing the drive firmware,...

- As diagnostics and debugging interface by using the firmware oscilloscope.
- As subsystem for drive-internal PLC (IndraMotion MLD).



If the master communication is active, it is not possible to switch the drive in control (AF) via the OCI interface.

6.4.4 SDK: Requirements and installation

To use the "eal_sdk" ("Easy Automation Library" "Software Development Kit"), please perform the following steps:

1. Make sure to use one of the following systems:
 - Electric drive controller with IndraDrive firmware MPx-18 (or higher)
 - Hydraulic drive controller with HydraulikDrive firmware HDx20 (or higher)
 - Frequency converters of EFC x610 type
 - IndraMotion MLD control system in version 13VRS or higher
2. For commissioning and engineering of the drives, you need to install IndraWorks DS/IndraWorks MLD version 13VRS or higher. Alternatively, you can also use the "Drive tool" or "Drive tool EFC" application contained in the SDK.

3. Registering for the **Engineering Network**

To download the "eal_sdk" SDK from the Bosch Rexroth Internet page, you have to register for the **Engineering Network**:

1. Go to the internet page of [Bosch Rexroth](#).
2. Register on "myRexroth". (For a description of the registration process, please refer to the [Engineering Network forum](#).)
3. Open the internet webpage for "[Open Core Engineering](#)"
4. After the registration is completed, the SDK "eal_sdk" is available in the [Download area](#) of **Engineering Network**.
4. Depending on the application, the device platform and operating system and the target device, integrate the "Easy Automation Library" contained in the SDK in your development environment.

6.4.5 SDK: Contents

The SDK "eal_sdk" is divided into different "tool boxes". Each tool box has an individual sub directory in the SDK. The functions supported by the "tool boxes" are basically identical. They differ with regard to the development environment and the operating system in which the development environment is used.



Each tool box does not only contain libraries, but also documentations and [Application examples](#).

Supported functions of the tool boxes:

- **System**
 - Establish connection to the drive controller or the frequency converter
 - System information such as reading diagnostic data and firmware version
 - Firmware upload and download
- **Parameter**
 - Read and write parameters
 - Read parameter name, attributes, unit and status
 - Stopping and executing commands

- **PLC** (firmware version 20 and above)
 - Start, stop and reset the PLC
 - Read and write PLC status
 - Browse, read and write symbol variables
- **Motion**
 - General functions (activating drive enable, bringing the drive to the STOP state, drive-controlled homing)
 - Axis motions (preset velocity, moving to absolute position, moving by a distance in addition to the target position, moving by a distance from the current actual position)
- **Oscilloscope**
 - Configuring the oscilloscope channels
 - Configuring the oscilloscope trigger
 - Reading the oscilloscope data

The SDK is continuously developed; the following chapters provide an overview of the "eal_sdk" SDK version 2.0



Since the development environments and operating systems may change, Bosch Rexroth does not assume any warranty for the proper functioning of the SDK.

EAL4Android

"EAL4Android" comprises an Android library and a "Mono for Android" app for configuring and controlling Bosch Rexroth drive controllers and frequency converters that support the "Sercos Internet Protocol" (S/IP).

System requirements: Java™ 32/64 bit, Android Studio

EAL4DotNet

The "EAL4DotNet" tool box allows functions of Bosch Rexroth drive controllers and frequency converters to be accessed by .NET libraries.

Supported operating systems of the target device: Windows XP/Vista/7/8, Android¹⁾, iOS¹⁾, Mac OS¹⁾, Linux¹⁾ [¹⁾: using Mono]

Supported development environment: Microsoft Visual Studio 2005 or newer, Visual Studio with Xamarin extension, Xamarin Studio, RAD studio®

Supported programming languages: .NET (C#, VB, F#), VBA (for MS Excel, MS Access, MS PowerPoint, MS Word), Delphi

EAL4C

Using the "EAL4C" tool box, the functions of "EAL4DotNet" are converted into structural functions with Mono.

Supported operating systems of the target device: Windows XP/Vista/7/8, Linux with mono extension

Supported development environment: Eclipse, Microsoft Visual Studio, Qt, etc.

Supported programming languages: C/C++

EAL4Java

The tool box "EAL4Java" includes the "EAL4C" functionality as object-oriented Java classes in a package by using the native Java NDK library. EAL4Java allows EAL functions to be called from the Java runtime environment.



Applications using the "EAL4Java" tool box can only run on the Java runtime environment under Windows and Linux.

Supported operating systems of the target device: Windows XP/Vista/7/8, Linux

Supported development environment: Eclipse

Supported programming languages: Java

EAL4LabVIEW

The "EAL4LabVIEW" tool box allows functions of Bosch Rexroth drive controllers and frequency converters to be accessed under LabVIEW.

Supported operating systems of the target device: Windows XP/Vista/7

Supported development environment: National Instruments LabVIEW

Supported programming languages: G (LabVIEW VI language)

6.4.6 Application examples

In the "Engineering Network" [[Download→Apps](#) (registration and login required)] and in the SDK, we provide different demo programs and apps.

For all demo programs and apps, the entire source code, the relevant project files, information about the use case, the requirements for execution and creation are provided.

In the following chapters, some examples are provided.

Windows application with Microsoft Visual Studio for controlling and monitoring "IndraDrive" controllers

In the Microsoft Visual Studio development environment, the Windows application "DriveTool" is developed.



Under `..\eal4DotNET\Samples\Sample - DriveTool`, the "eal_sdk" SDK contains the "DriveTool" Windows application and also the source files.

"DriveTool" allows the following functions of "IndraDrive" range drive controllers to be accessed:

- Browsing available drives
- Establish connection to a drive via the Ethernet IP address
- Reading the power status and switching on power
- Restarting the drive
- Initializing the drive
- Reading the firmware type code
- Switching between operating mode and parameter mode:
- Homing the drive
- Clearing drive errors
- Reading and selecting the operating mode
- Reading and writing the motor velocity, position and torque

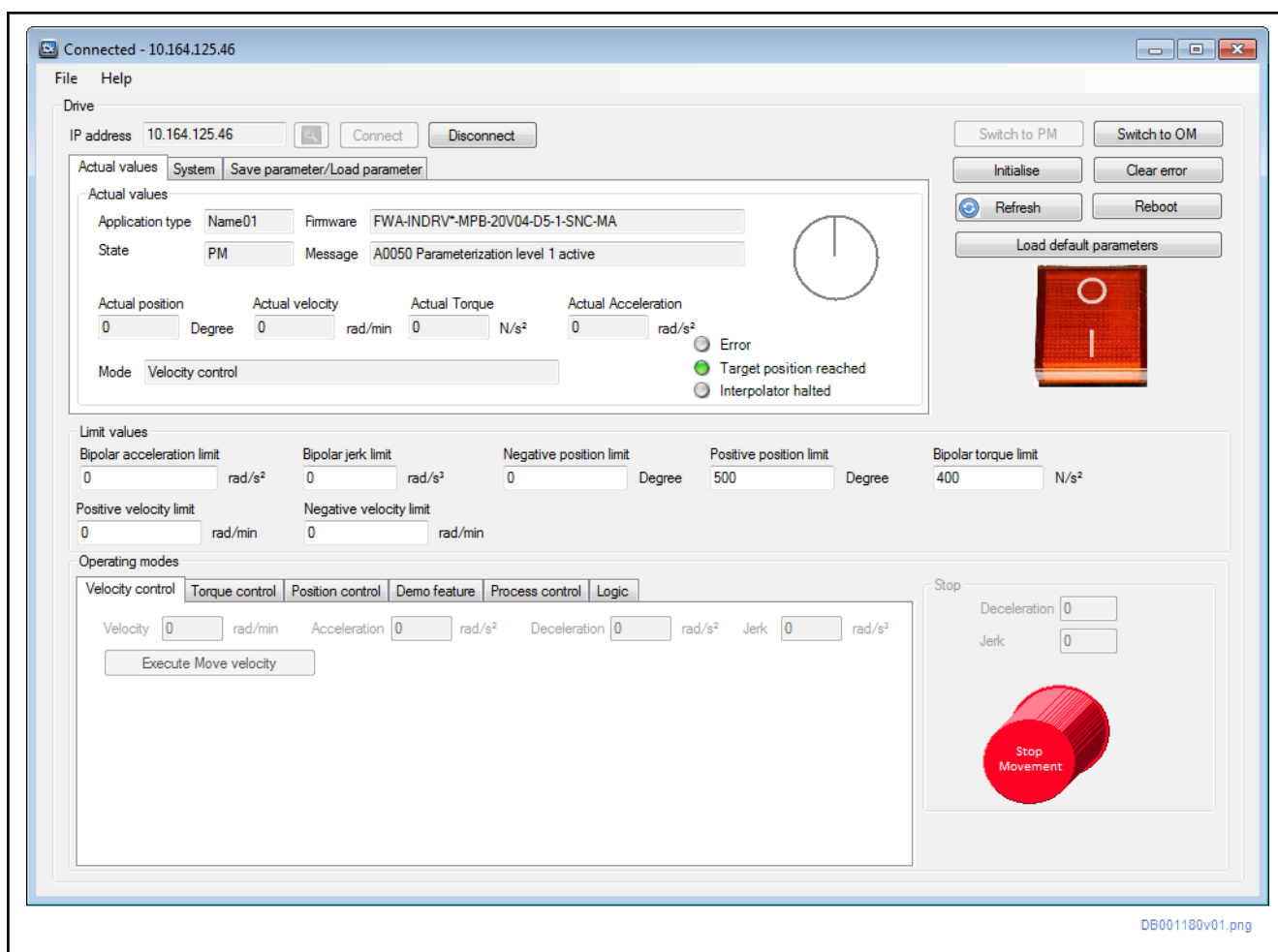


Fig. 6-10: "DriveTool"

Microsoft Excel file as user interface

The following example shows the integration of the "Open Core Interface for drives" in the Microsoft office application Excel.

The Excel user interface allows the following functions to be accessed:

- Establish connection to a drive via the Ethernet IP address
- Switch on power
- Switching between operating mode and parameter mode:
- Writing the motor velocity, acceleration, deceleration and jerk

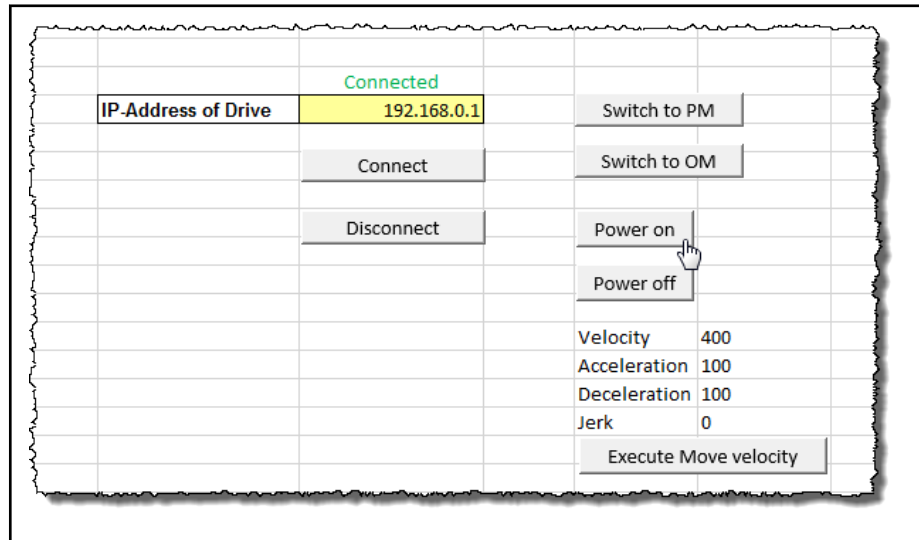


Fig. 6-11: Excel user interface

Windows application with Microsoft Visual Studio for recording device signals

In the Microsoft Visual Studio development environment, the Windows application "Oscilloscope" is developed.

In the application, you can configure, for example, the signal to be recorded, the sampling rate and the trigger to be used for commissioning, servicing and testing drive controllers.

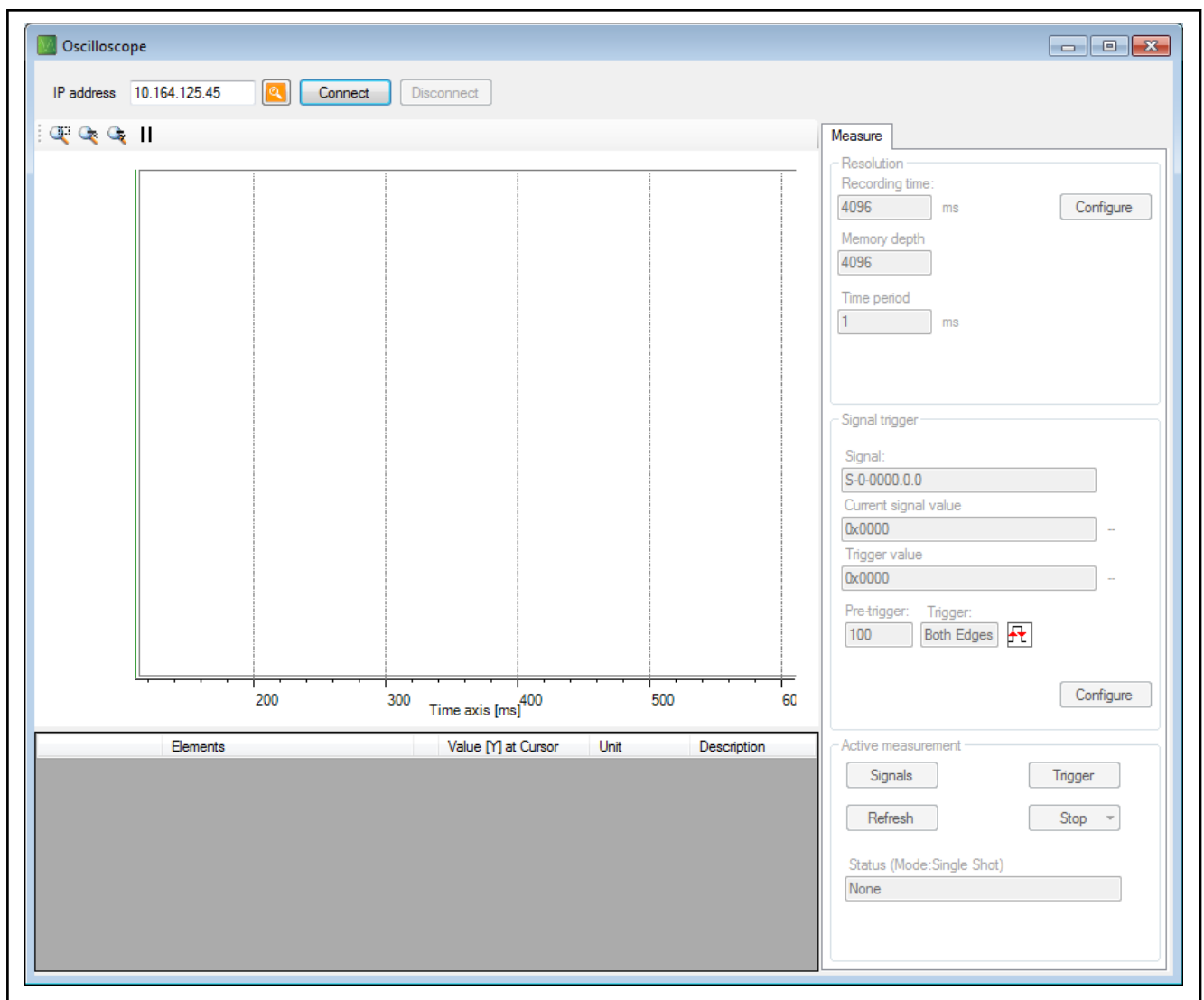


Fig. 6-12: Windows application "Oscilloscope"

Windows application with Microsoft Visual Studio for archiving and restoring firmware and parameter files

In the Microsoft Visual Studio development environment, the Windows application "Download tool" is developed.

Use the "Download tool" application to archive and restore firmware and parameter files of drive control devices of the "IndraDrive" product family.

The "download tool" can be operated in console mode and in GUI mode with graphic user interface.

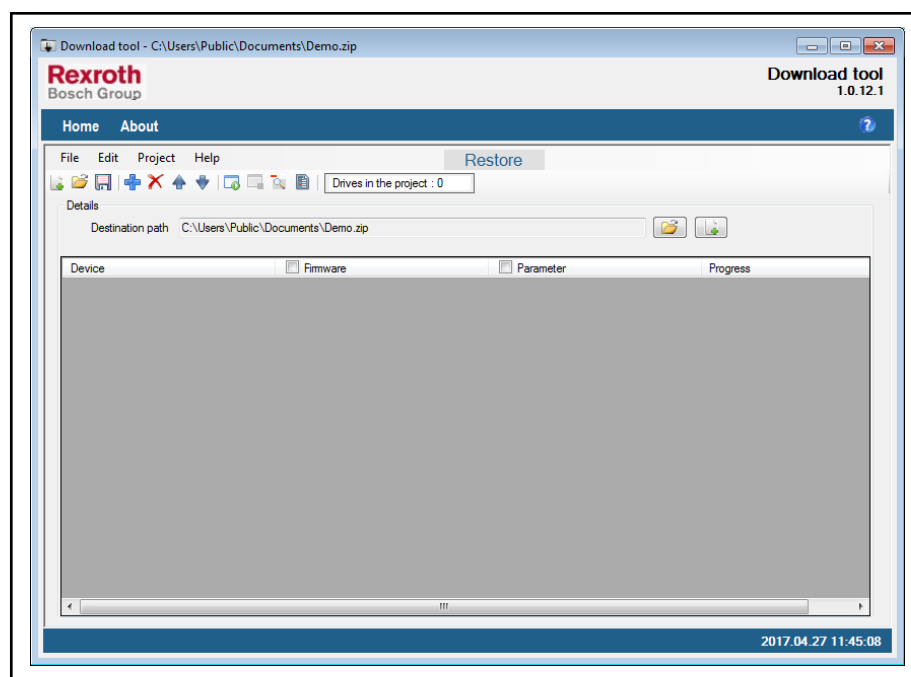


Fig. 6-13: Windows application "download tool" in GUI mode with graphic user interface

Android app

For Android, different demo apps with different functionalities are provided in the SDK:

- Travel motion
- Oscilloscope function
- Parameter backup
- Firmware update
- ...

The demo apps are provided as APK (Android Package File).

To install the demo apps, you must permit the installation of apps not coming from the Google Play Store.

1. Open the settings on your smart phone and set the check mark at **Applications ► Unknown sources** (for some devices under **Safety ► Unknown sources**).

CAUTION: The checkmark is a safety risk since apps from unknown sources may contain viruses.

⇒ Only install apply of reliable sources.

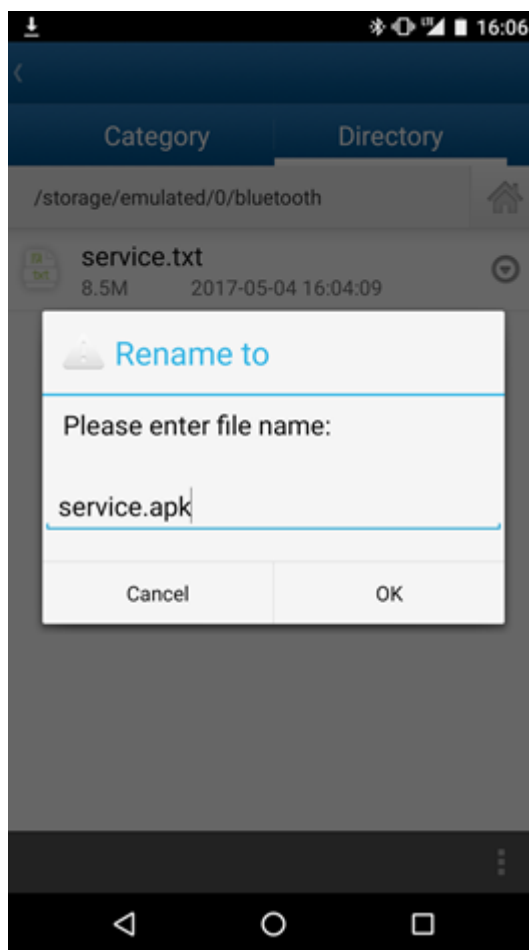
⇒ By default, disable the check mark.

2. The easiest option to transmit an APK from a computer to your smart phone is to use a USB cable.

Transmission via Bluetooth is also possible. To do so, you have to rename the *.apk file extension before the transmission (e.g., *.txt).



3. After the transmission, the file has to be renamed *.apk on the smart phone.



4. Install the APK on your smart phone.



To use "Open Core Interface for drives" on the smart phone, **you must in any case install the "service.apk"** APK for each application APK. "service.apk" is contained in the SDK in the "lib" subdirectory of the EAL4Android directory.

After the installation of "service.apk", find the app under **Settings ► Applications** as "EAL service".

Travel motion by entering command values

In the "demo app" (demoapp.apk), the travel motions are performed by entering command values (**MOVE VELOCITY/MOVE FREQUENCY**). Furthermore, limit values can be specified.

Here, the "demo app" (demoapp.apk) with the steps "Establishing connection to the drive controller" and "Calling the entry mask for velocity parameters":

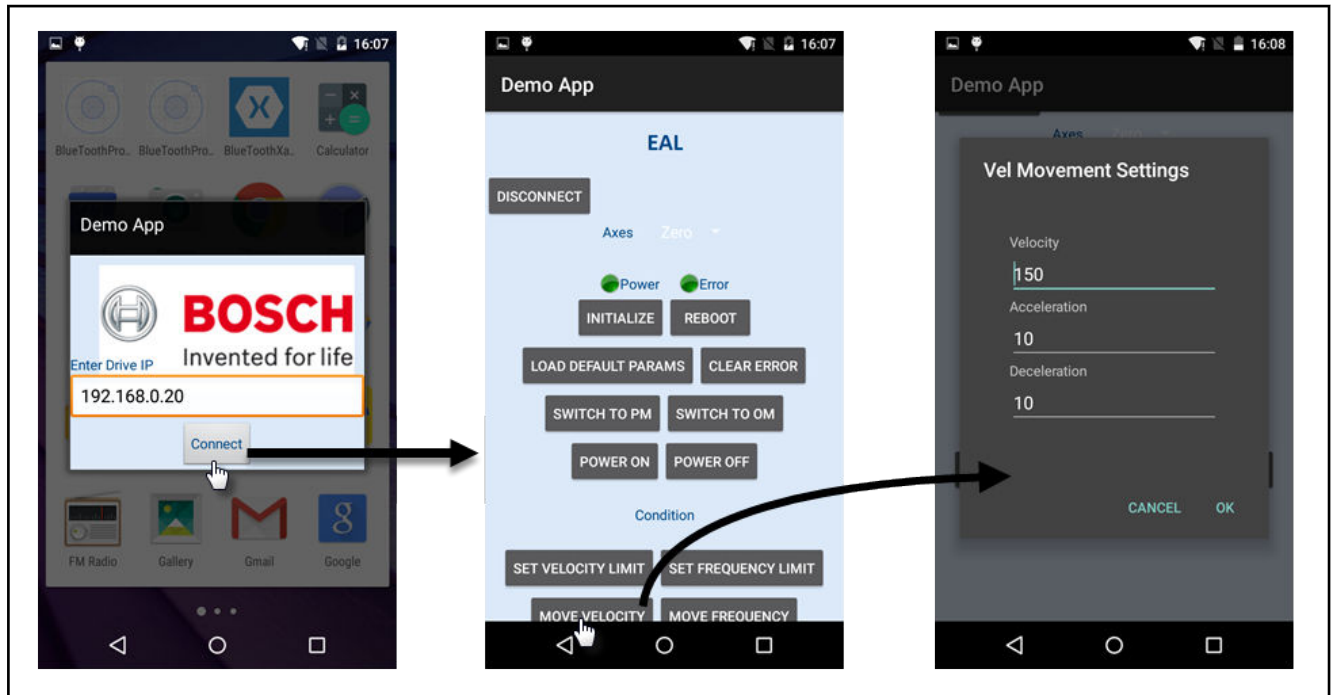


Fig. 6-14: "Establishing connection to the drive controller" and "Calling the entry mask for velocity parameters" in the "demo app"

Travel motion by rotary and tilting motion of the smart phone

To perform a travel motion of the drive, it is also possible to use the acceleration and position sensors of the smart phone. The figure below shows an app in which the drive is moved by rotary and tilting motions of the smart phone:

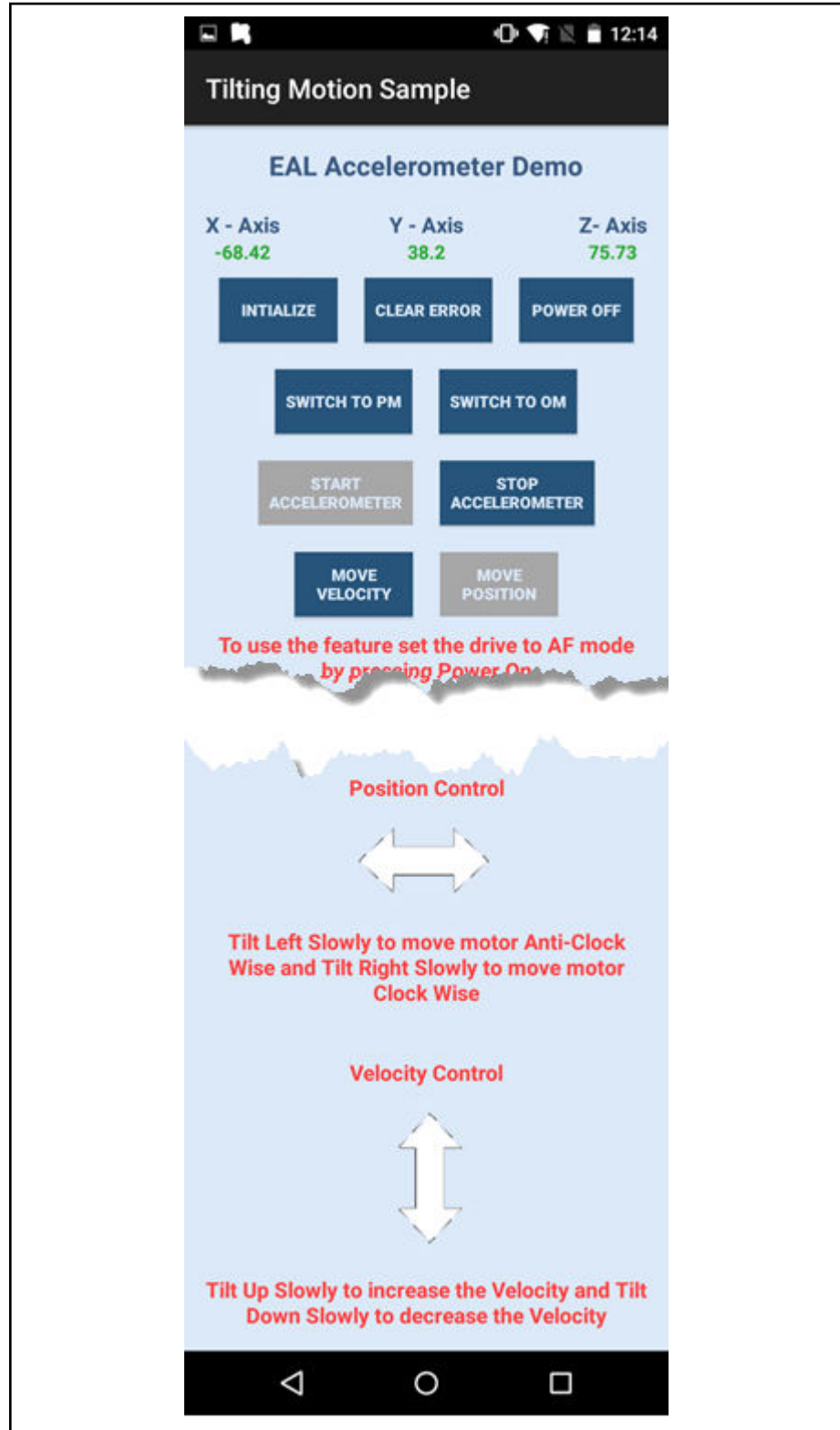


Fig. 6-15:

Demo app: Travel motions by rotary and tilting motion of the smart phone

7 Notes on commissioning and application

7.1 Requirements for using Rexroth IndraMotion MLD

7.1.1 Firmware and hardware requirements

See "System overview", "[Firmware requirements](#)" or "[Hardware Requirements](#)".

7.1.2 Software requirements

The development system for programmable logic controllers (e.g., IndraMotion MLD) is automatically made available when the IndraWorks MLD commissioning software is installed.

In detail, the development system consists of the following software components:

- Logic programming interface
The Logic programming interface contains the PLC editors and debuggers.
- IndraLogic OPC server
The OPC server is normally required by third-party visualization tools.
- IndraWorks packages
IndraWorks packages contain information files and all required libraries for the drive-integrated PLC Rexroth IndraMotion MLD. All required device supports (IW packages) including the libraries are automatically installed so that it is not necessary to select specific software components.
- IndraLogic gateway server
The "IndraLogic gateway server" provides the communication for the programming system and, if used, third-party applications.
- ENI server
The ENI server is used for communication between IndraWorks and IndraLogic and as the server for source code management.

The device supports (IW packages) can also be separately installed subsequently. This may be necessary during the introduction of new product versions in their prototype phases. The corresponding dialog can be called in the Project Explorer via the context menu **MLD ► Change IndraLogic device version**.

In the "Change IndraLogic device version" dialog, IndraWorks packages can be subsequently installed by clicking the "Install IW package" link.



In case IndraWorks is updated, all existing libraries remain available on the computer. It is not necessary to install an "older" IndraWorks package again.

7.1.3 Requirements for communication

When exclusively ready-made technology functions are used with the drive-integrated PLC (MLD), it is not necessary to establish a connection between IndraLogic and the drive. Communication with IndraWorks MLD is only required to configure the drive.

If you wish to use your own functions, the "IndraDrive Cs" range requires an Ethernet connection for programming the drive-integrated PLC (MLD).

IndraWorks MLD supports the establishment of a connection to connected devices. The selection is made via the menu bar **Project ► Scan for devices....**

After the connection to IndraWorks MLD has been established, you can directly go online in IndraLogic (**Online ► Login**). The communication settings are automatically preset; other settings are not required.

7.2 Programming in IndraWorks MLD

In IndraWorks MLD, the drive functionality is represented in a device tree. The dialogs required to parameterize the drive and MLD can be accessed via the **MLD** branch.

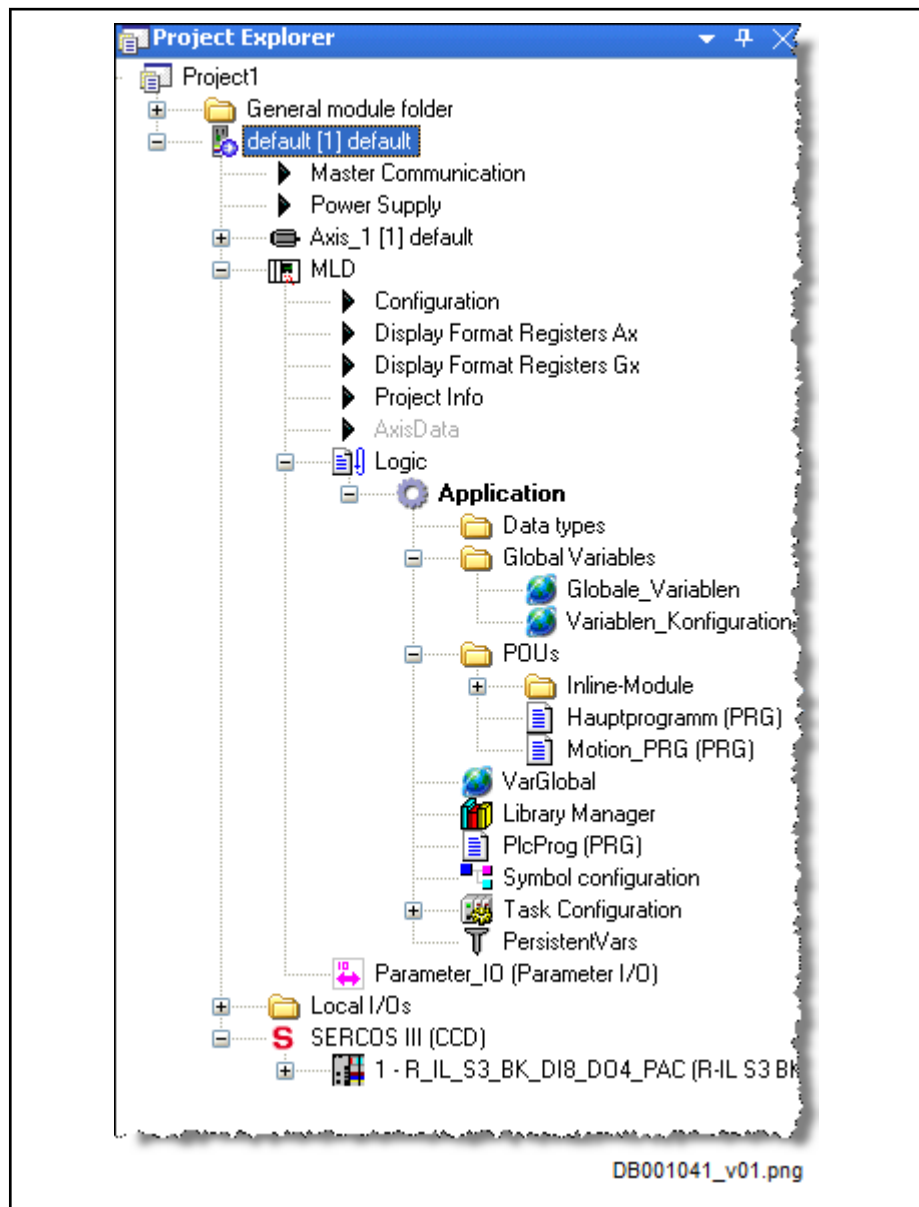


Fig. 7-1: Project Explorer with MLD enabled



The MLD-specific branches in the project tree are only visible when the functional package has been enabled!

Double-clicking the logic branch opens a dialog in which you can select how the program is to be created in IndraWorks MLD:

- **Free programming**
 - Global variable list, symbol configuration and data types, default program (PlcProg)
 - Global variable list, symbol configuration and data types
- **Using existing projects**
 - Creating from IndraLogic 1G project
 - Creating from export file or CoDeSys 3.x project

- **AxisInterface sample project**

The axis interface bundles and enhances PLCopen motion function blocks and provides an easy-to-use interface for drive functionality. Less code and more efficient commands speed up the program development of applications (see "[Using the "AxisInterface" for programming Rexroth IndraMotion MLD](#)").

- **GAT compact sample project**

GAT compact (Generic Application Template) is a ready-to-run programming frame in which the application code is entered at the predefined points. Professional and well-structured application programs with prepared elements using the dialog-based code generation of the GAT wizard are created. GATcompact is available in IndraWorks 14V14 with MPx20 and above (see "[Automated program generation with GAT compact](#)").

Apart from that, it is possible to use technology functions. This is a complete (compiled) PLC project that can consist of multiple function blocks (conforming to IEC) (see "[Using technology functions](#)").

7.2.1 Using technology functions

Overview

A complete (compiled) PLC project that can consist of multiple function blocks (conforming to IEC) is called technology function.

A project is created by combining IEC or firmware function blocks and mostly provides a complex and comprehensive functionality.



For some technology functions there are specific dialogs in the commissioning software IndraWorks. The dialogs support the commissioning and operation of the technology functions. In addition, there is an application description for each technology function.

Technology functions can be used in various ways:

The technology function can be loaded as a parameter file via commissioning software (e.g., IndraWorks)

- Technology function supplied as a package containing the PLC project and possibly additional parameters (parameter file without source code).
- It is impossible to access the PLC source code via IndraLogic.
- There is no debug function and no programming option.
- The handling is comparable to an invariably programmed firmware function.

Technology function as a library (individual function blocks)

- The library does not contain any source code (access is password-protected).
- The individual function blocks have been combined in groups (libraries) (e.g., variable error reactions [return motion, etc.]).
- The full debug function is available and programming is possible (similar to "DISC" macros of the EcoDrive range).

Parameters involved

The following parameters are used in conjunction with technology functions:

- P-0-1350, PLC control word
- P-0-1351, PLC status word
- P-0-1352, PLC user program administration data
- P-0-1353, PLC user program area 0
- P-0-1354, PLC user program area 1
- P-0-1355, PLC user program area 2
- P-0-1356, PLC user program area 3
- P-0-1357, PLC user program area 4
- P-0-1358, PLC user program area 5
- P-0-1360, PLC program identifier
- P-0-1361, PLC program name
- P-0-1367, PLC configuration
- (possibly other register parameters of the PLC)

Notes on commissioning and programming**Loading and activating technology functions**

Technology functions can be loaded in two ways:

- Loading as parameter file via the commissioning software (e.g. IndraWorks. Double-clicking "Technology functions" in the MLD folder opens the menu for managing the technology functions, cf. "P-0-1353, PLC user program area 0" to "P-0-1358, PLC user program area 5"). In this case, it is only necessary to load and then parameterize the desired technology function (parameterization has to comply with the pre-programmed functions).
- Loading the project via IndraLogic

Technology functions

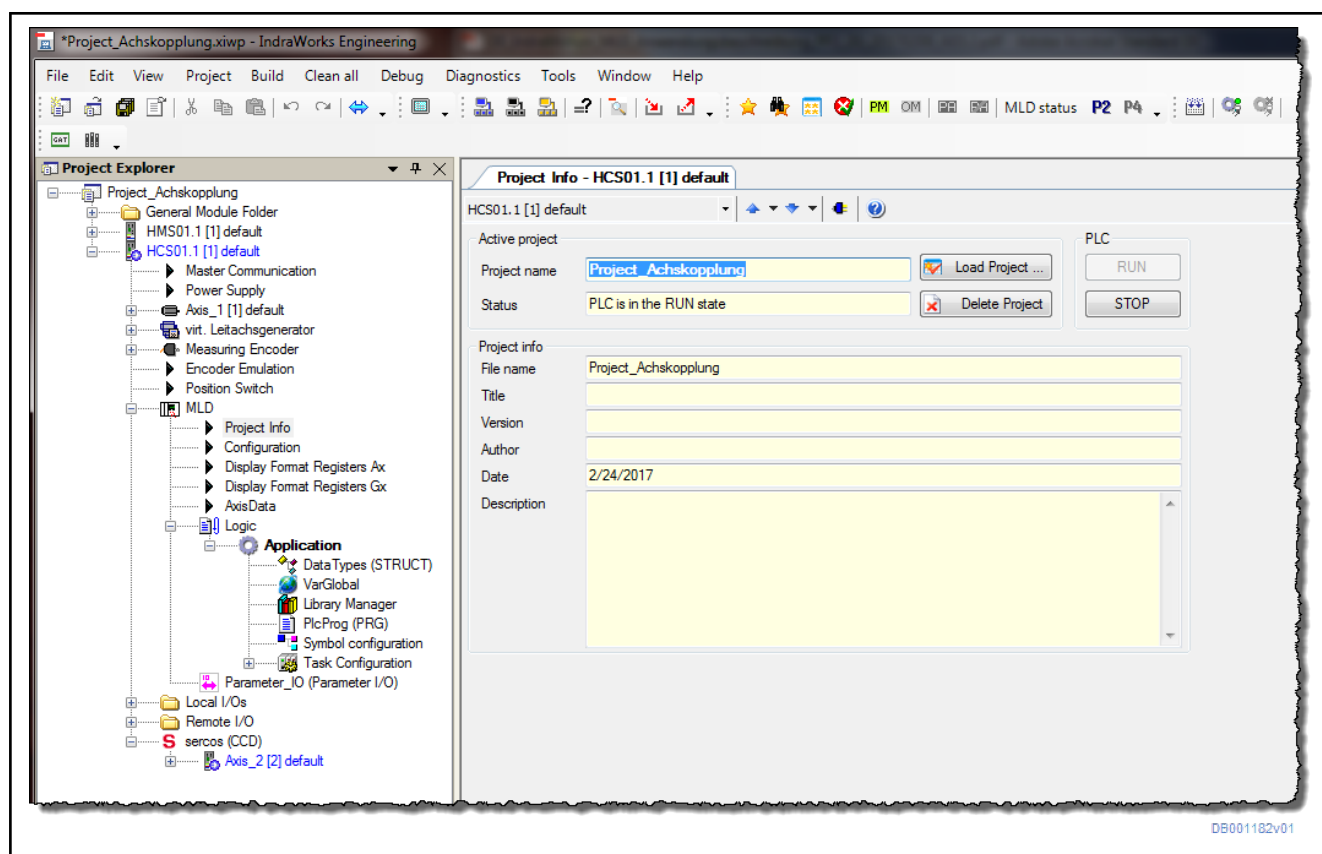


Fig. 7-2: Technology functions

The technology functions are supplied as a prepared package. It mainly consists of a file with the "*.spj" extension. This file contains a complete PLC project in compiled form; source code and PLC variables are not visible. With the above dialog, a technology function can be loaded to the drive.



Regardless of how the technology function is loaded to the drive (via IndraWorks or IndraLogic), it is stored as a project in the parameters (P-0-1352, P-0-1353, P-0-1354, P-0-1355, P-0-1356,... and, if necessary, P-0-1367) so that it is contained in a parameter backup.



Technology functions are mainly programmed in such a way that they are automatically started when the drive is booted up (cf. "P-0-1367, PLC configuration"). If this is not the case, they can be activated via "P-0-1350, PLC control word" after the project was loaded.

Parameterization and commissioning

The technology function can be parameterized in different ways:

- Via commissioning software (e.g., IndraWorks MLD) using dialogs specifically made for the technology function (e.g., "Productivity Agent")
- Via the visualization option in IndraLogic (see also "[Visualization, engineering and connection via OCI](#)")
- Via a screen of the operator and visualization device specifically programmed for this purpose that allows the PLC parameters to be accessed

- Via any higher-level control unit



For the settings and parameterizations specific to the technology function, please refer to the corresponding documentation.

Diagnostic and status information

The following parameters are used for running diagnostics on PLC projects (technology function):

- P-0-1360, PLC program identifier
- P-0-1361, PLC program name
- P-0-1351, PLC status word

7.2.2 Free programming of Rexroth IndraMotion MLD

Connection to the drive

General information

Communication between the PLC programming user interface IndraLogic and the drive is made available via the IndraWorks MLD commissioning software.

The connection is set in IndraWorks MLD. With "Scan for devices", for example, the connection can be automatically set.

IndraLogic is started from IndraWorks MLD by double-clicking the "Logic" branch in Project Explorer. IndraLogic automatically applies the communication setting from IndraWorks MLD. The IndraLogic dialog for setting the communication parameters is not required.



IndraLogic should only be started from IndraWorks MLD. Operating IndraLogic without the embedding in IndraWorks MLD is not supported.

Ethernet interface (TCP/IP)

At the drive, the Ethernet Engineering port (HCS01) is used. An Ethernet interface with TCP/IP protocol is required for communication. The settings of the TCP/IP communication can be made in IndraWorks via the context menu of the drive (parameters involved: P-0-1531, P-0-1532 and P-0-1533).

MLD settings in IndraWorks

Requirements

The drive should have been fully commissioned before programming MLD.

Settings

The settings for the "IndraMotion MLD" function can be configured under the "MLD" branch. The screenshot below shows the options provided in the IndraWorks Project Explorer:

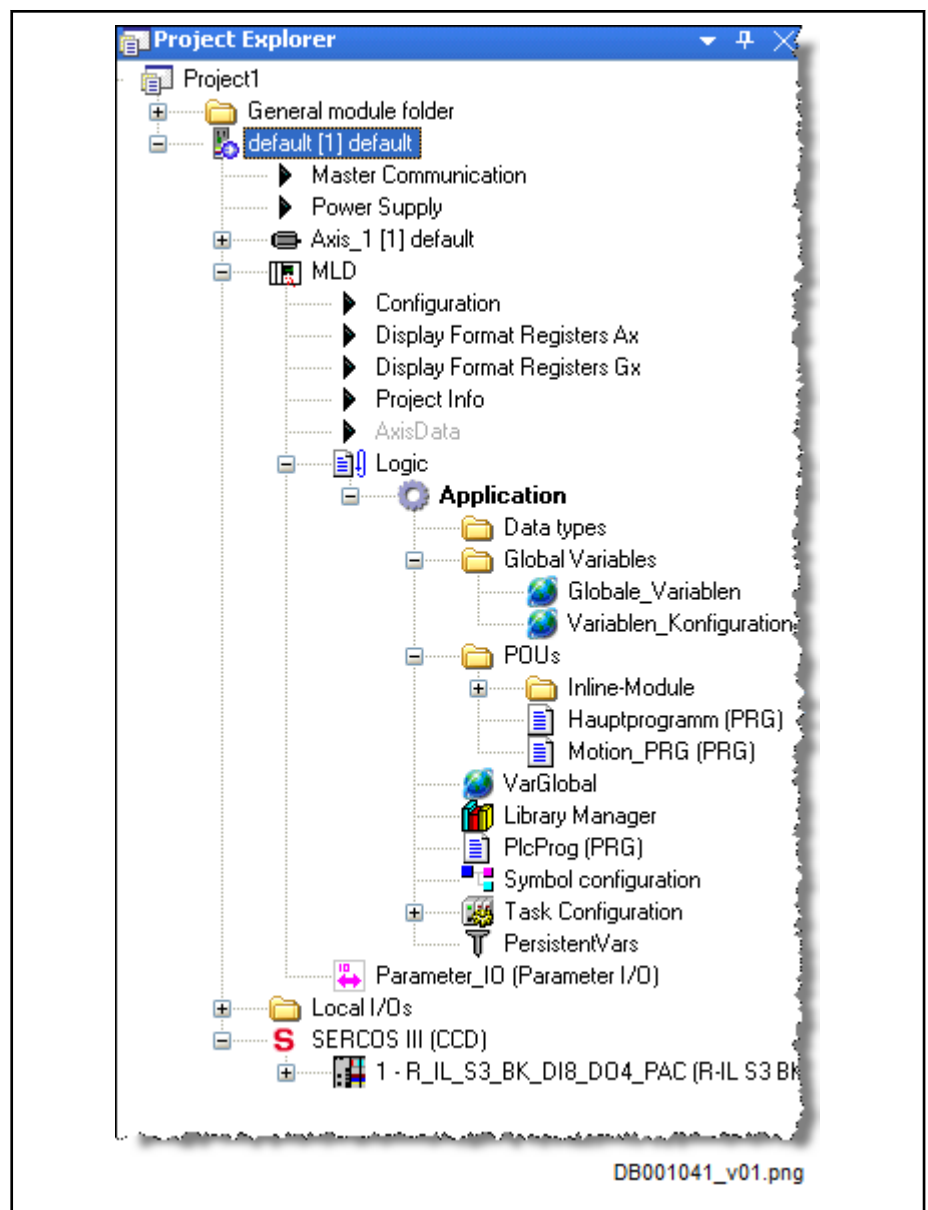


Fig. 7-3: Excerpt from the IndraWorks Project Explorer



The "Configuration" dialog can only be called if a connection to the drive has been established or offline simulation was started in IndraLogic.

"MLD" context menu

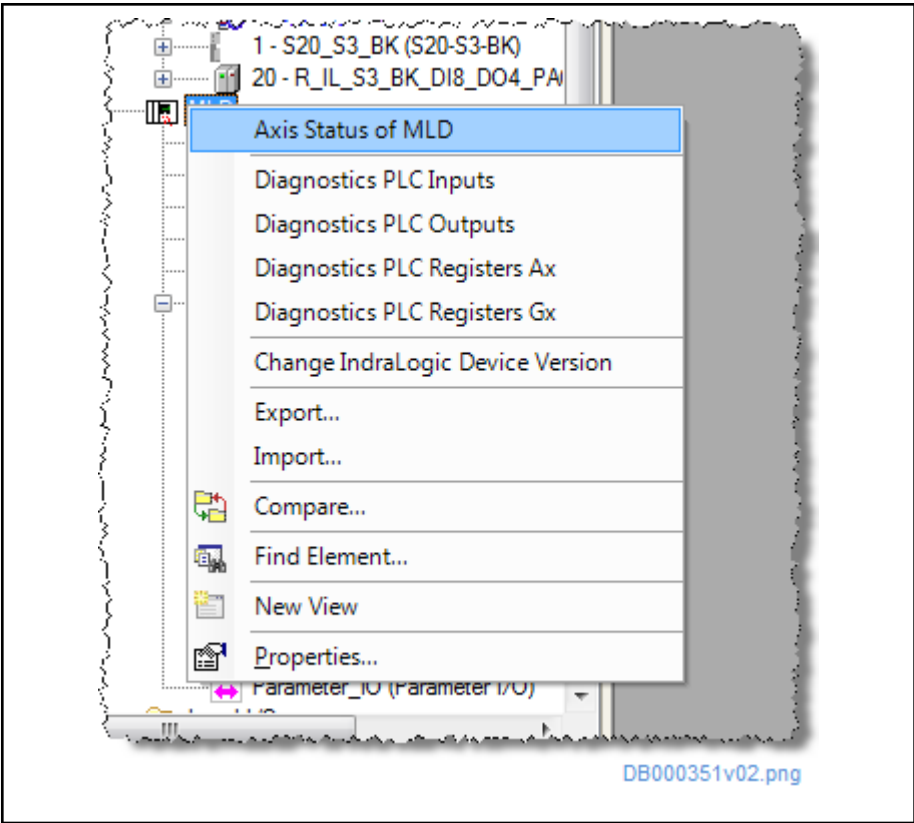


Fig. 7-4: "MLD" context menu

MLD axis status

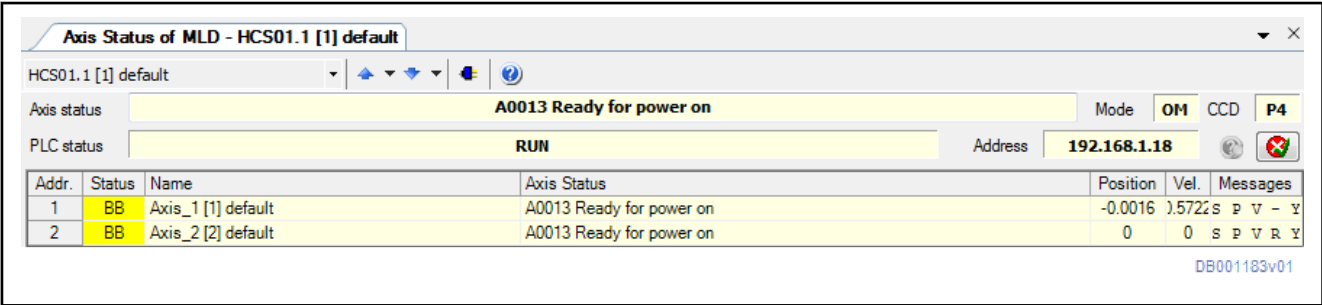


Fig. 7-5: Axis status of an MLD-M CCD group

Diagnostics for PLC inputs

In this dialog, the state of the process image of the inputs can be controlled. Inputs assigned to the PLC are displayed in this dialog.

i **Optimizing the inputs in IndraLogic:** The inputs which are not used in the PLC program are not displayed in IndraLogic. This dialog displays the entire process image of MLD, regardless of its use in the program.

Diagnostics PLC outputs

In this dialog, the state of the process image of the outputs can be controlled. Outputs assigned to the PLC are displayed in this dialog.

Diagnostics PLC registers Ax

This dialog displays the contents of the PLC registers A0 to A31, as well as PLC registers AT0 and AT1. The contents of PLC registers A0 to A31 are dis-

played in the format configured under "Display format registers Ax". The contents of the PLC registers can be edited from the dialog.

The PLC registers AT0 and AT1 are text registers and can contain a maximum of 255 characters each.

The PLC register AL0 is a list register with 8,192 elements (4 bytes each). The format set in "Display format registers Ax" applies to all 8,192 elements.



The Ax PLC registers are not buffered, i.e., their contents are lost in case voltage fails.

Diagnostics PLC registers Gx

This dialog displays the contents of the PLC registers G0 to G31, as well as PLC registers GL0 to GL2. The contents of the PLC registers G0 to G31 are displayed in the format configured under "Display format registers Gx". The contents of the PLC registers can be edited from the dialog.

The PLC registers GL0 to GL2 are list registers with 1,024 elements (4 bytes each). The format set in "Display format registers Gx" applies to all 1,024 elements.



All Gx PLC registers are buffered, i.e., their contents are retained in case voltage fails.

Change IndraLogic device version

The "Change IndraLogic device version" function can be used to change the device version (IndraWorks package) or add a new device version. Device versions (IndraWorks packages) can only be changed once any critical errors have been corrected. In general, changed device versions are included with IndraWorks releases.

Export / import

PLC program sections are exported and imported here. The default export file extension is "*.iwx". Only corresponding files can be imported.

Compare

The "Compare" function allows two project files to be compared to one another. Both the source file and the target file have to be specified.

Find element

The "Find element" function allows searches for words/expressions in a PLC project.

Configuration

Double-clicking "Configuration" calls a dialog that allows the start behavior of MLD to be set and some settings for motion control to be configured.

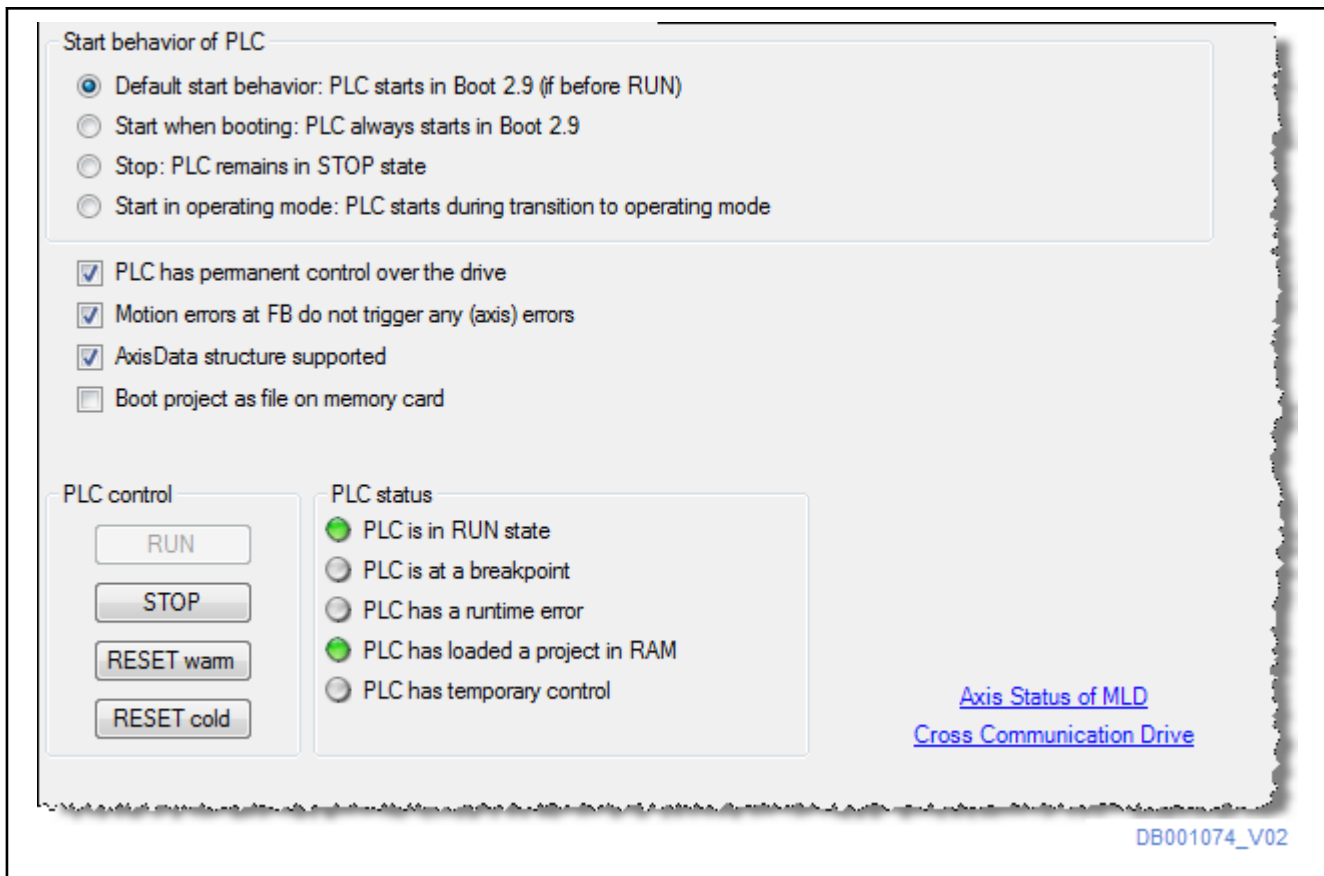


Fig. 7-6: Configuration dialog

- With "Start behavior of PLC", define the behavior of the drive-integrated PLC during the start-up sequence:
 - "Default start behavior"

The PLC maintains its state. This means that it starts when the drive is booted up, if it had been in the RUN state before the last switch-off. Otherwise, it remains in the STOP state.
 - "Start when booting up"

The PLC always starts when the drive is booted up, regardless of its previous state (STOP/RUN).
 - "Stop"

The PLC remains in STOP, regardless of its previous state.
 - "Start in operating mode"

The PLC only starts after the drive is in operating mode.
- With "PLC has permanent control over the drive", the drive can be configured with MLD as a stand-alone motion controller for control tasks (check box selected), or as an intelligent servo axis for extending the drive functionality (check box **not** selected).



If "PLC has permanent control over the drive" is selected, the storage mode is also automatically set such that parameter changes will be lost after the control voltage is switched off (S-0-0269="1").

- With "Motion errors at FB do not trigger any (axis) errors", it is possible to have the drive error F2150 generated in the case of errors which are detected by the corresponding PLC function blocks when commanding the axes (check box selected). If the drive is not supposed to react automatically, do not select the check box.
- "AxisData structure supported" check box: For motion tasks there is the optional global structure "AxisData" which simplifies access to cyclic data of the axes. Support of this data structure over all axes permanently consumes computing time and can be switched on or off, if required.
- The "Cross Communication Drive" link opens the dialog of the settings for multi-axis applications. The "multi-axis application" functionality is available with firmware version MPC18 VRS.
- The section "PLC status" can be used to determine whether or not a PLC project has been loaded and which state it is in.
- The PLC program can be started ("Run") and stopped ("Stop") via "PLC control".



Starting and stopping the PLC in this dialog is only useful in exceptional instances. Normally, this will be done in the IndraLogic programming system.

Display format registers Ax

Double-clicking "Display format registers Ax" opens a dialog of the same name. This dialog can be used to set the display format and decimal places of the global PLC registers A0 to A31 (P-0-1270 to P-0-1301) and of the list parameter "P-0-1368, PLC Global Register AL0".

When selecting formats SIGNED_DEC and UNSIGNED_DEC, the number of decimal places can be selected from 0 to 7.

Display format registers Gx

Double-clicking "Display format registers Gx" opens a dialog of the same name. This dialog can be used to set the display format and decimal places of the global PLC registers G0 to G31 (P-0-1370 to P-0-1385, P-0-1316 to P-0-1331) and of the list parameters global PLC registers GL0 to GL2 (P-0-1389, P-0-1311, P-0-1312).

When selecting formats SIGNED_DEC and UNSIGNED_DEC, the number of decimal places can be selected from 0 to 7.

Project info

Double-clicking "Project info" opens a dialog of the same name. This dialog can be used to display information (status, project name/file name, date created, etc.) about the project. The PLC program can be started and stopped with this function.

AxisData



The "AxisData" menu item may be grayed out and may have to be enabled in "Configuration".

Double-clicking "AxisData" opens a dialog of the same name. This dialog can be used to parameterize global cyclic data for motion programs.

"AxisData" replaces the local RtcR/RtcW real-time channel in MLD-2G. It is possible to have 5 freely definable command values and 6 actual values.

Application

The "Application" branch contains various functions for operating (online/offline) and editing (data import/add/export/import/compare) the PLC program. Right-clicking the Application branch opens the following context menu. The context menu of the "Logic" branch is similar to that of the "MLD" branch. The context menu for the "Application" branch is described below.

The context menu differs depending on whether IndraLogic is "offline" or "online":

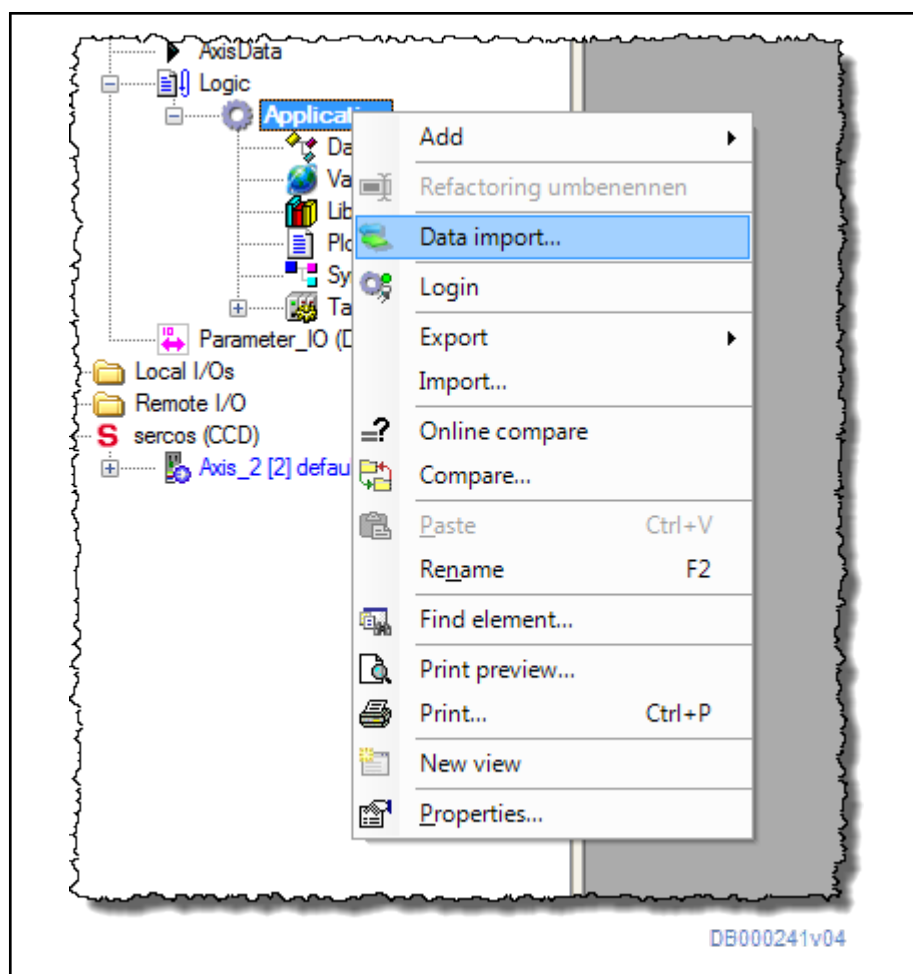


Fig. 7-7: Application branch context menu (IndraLogic offline)

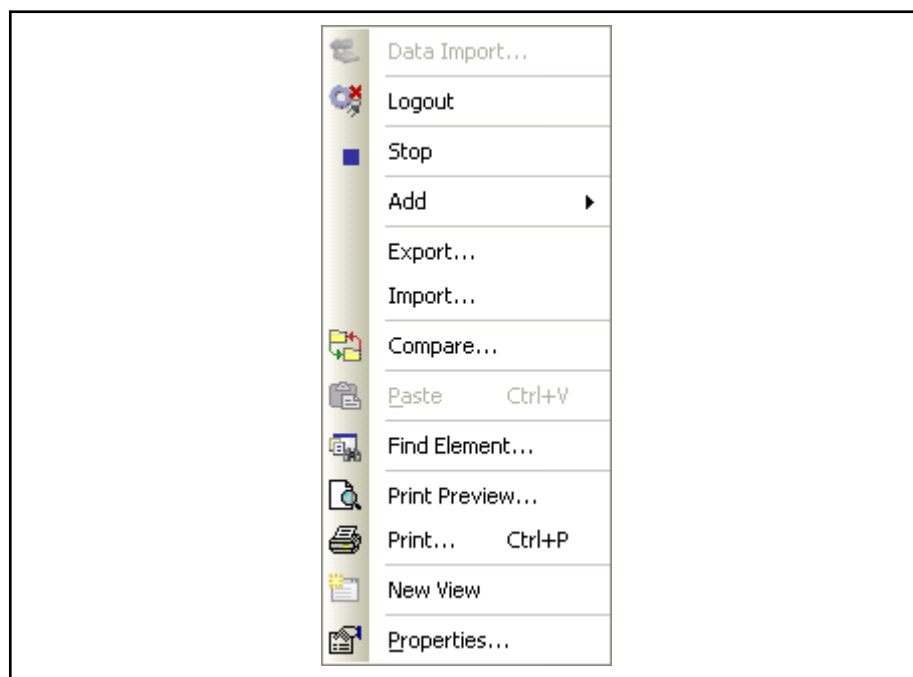


Fig. 7-8: Application branch context menu (IndraLogic online)

- **Data import** adds an IndraLogic project to this project. Projects created in MLD-1G can also be added to MLD-2G.
- The 2nd and 3rd menu items in this context menu change dynamically. **Login/Logout** or **Start/ Stop** appear depending on the state of the opened IndraLogic project.
 - With **Login**, a connection between IndraLogic and the drive is established (or the simulation program starts) and the system switches to online mode. Selecting the "Login" command automatically finishes compiling or loads a program to the drive, if these have not yet been done.
 PLC projects can only be edited and loaded to the drive when logged in.
Logout terminates the connection to the drive.
 - **Start** switches the PLC to "RUN mode", while **Stop** switches the PLC to "STOP mode".
- **Add** adds additional function blocks, POU's (PLC subprograms), tasks, data servers, data types, etc. to the PLC project.
- **Export / import** exports or imports PLC programs, function blocks, functions, etc. The file extension for these export files is "*.iwx".
- **Compare** is used to compare two PLC projects to each other.
- **Rename** is used to change the names of the elements under the "Application" branch.
- **Find element** allows searches for words/expressions in a PLC project.
- **Print preview** shows a preview of the printout of the selected file.
- **Print** allows all of the components in the IndraLogic project to be selected. Once the components have been selected, **Next >>** goes to the next dialog, where the printout can be set up. From here, **Finish** is used to automatically start the print job on the default printer.



Other print settings, such as selecting another printer, can be configured by going to **File ► Print settings....**

- **New view** displays the selected program section in its own window.
- **Properties...** is where project settings can be viewed and some of them edited.

Local I/Os The digital and analog inputs/outputs can be divided between drive functions and MLD using the dialogs in the "Local I/Os" branch. Assigning them to the MLD also sets the configuration in the process image (see "[MLD communication interfaces and data channels](#)").

Sercos The context menu can be used to configure MLD as a multi-axis control (MLD-M). The MLD-M multi-axis application is available with firmware version MPC-18VRS:

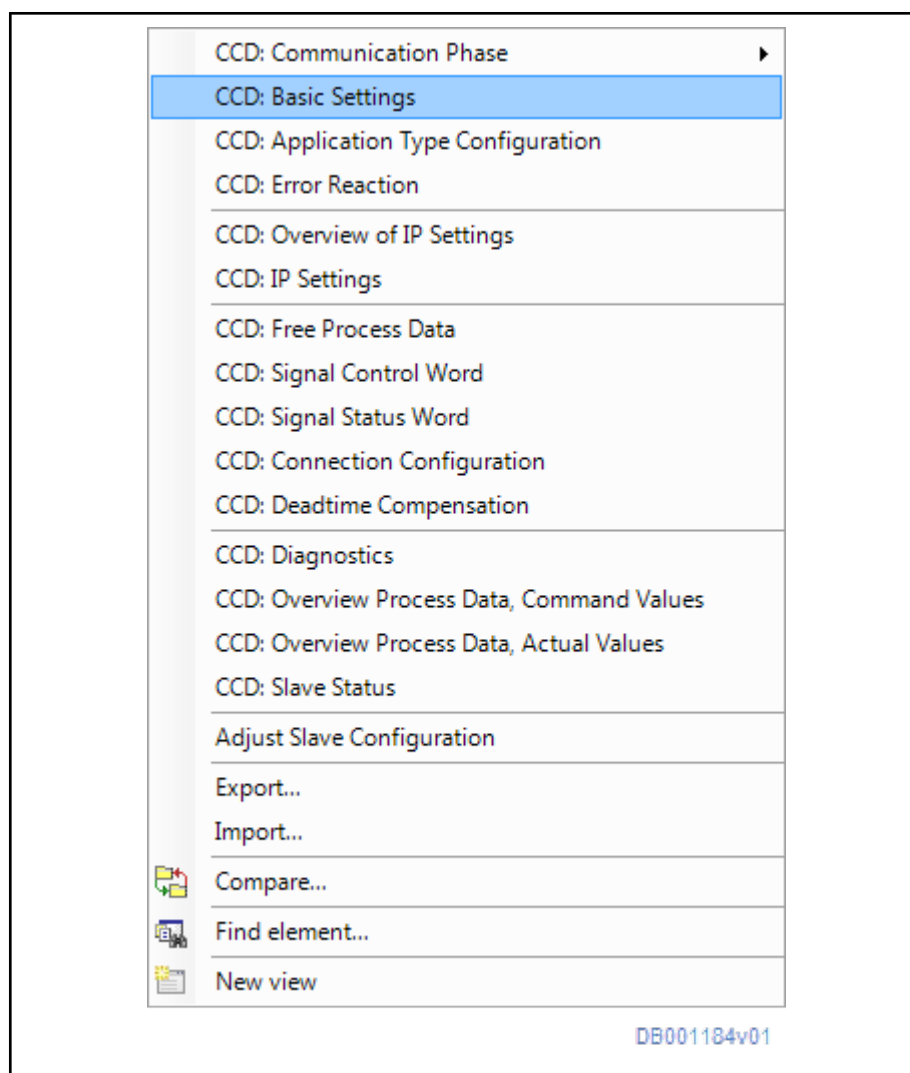


Fig. 7-9: Context menu of the Sercos branch

- "CCD: Basic settings" calls the dialog of the same name:

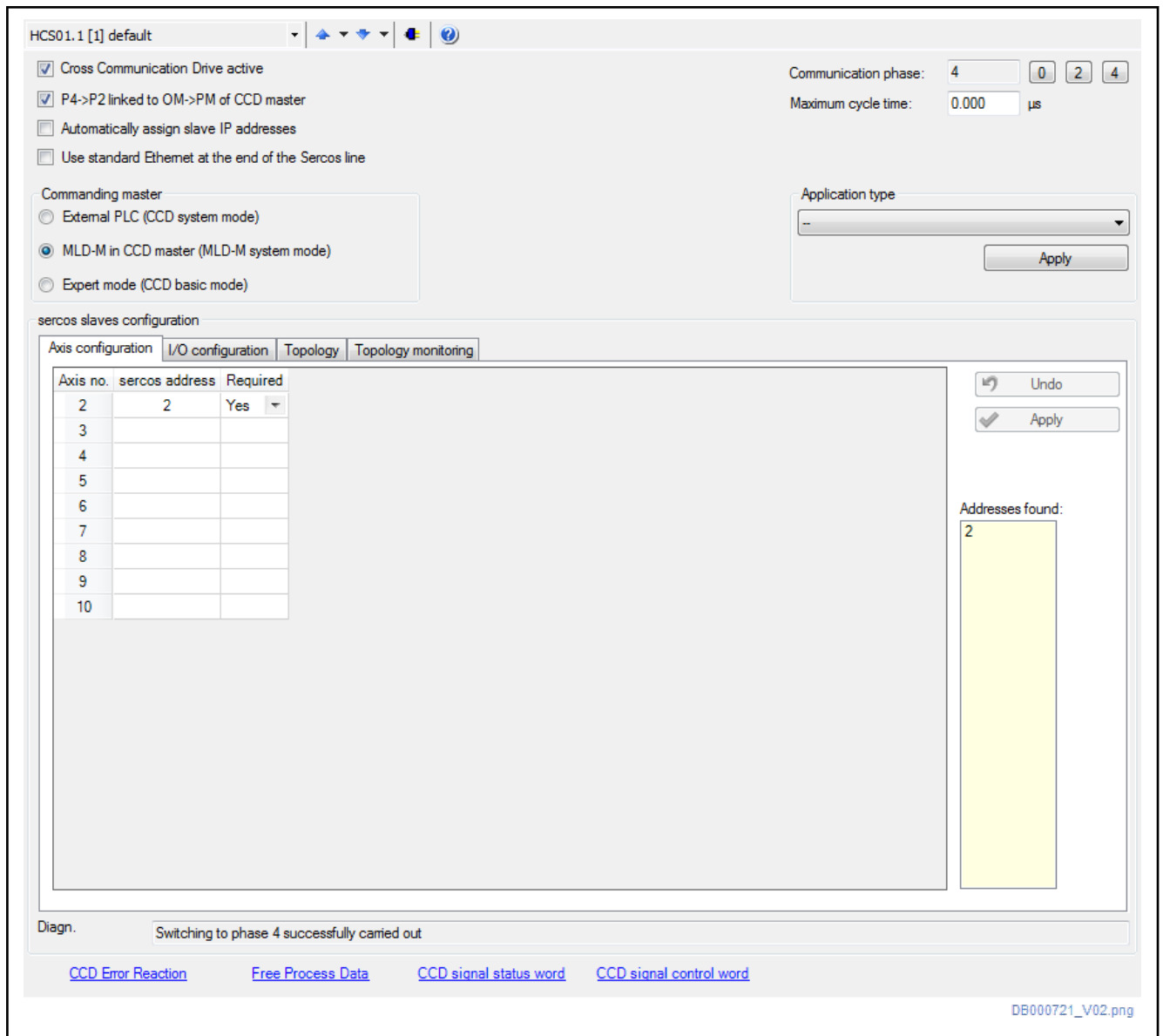


Fig. 7-10: "CCD: Basic settings" dialog

- "Cross Communication Drive active": Cross communication can be activated and deactivated here.
- "Commanding master": Selecting "MLD-M in CCD master (MLD-M system mode)" runs MLD as a multi-axis control.
- "Communication phase": Shows the current Sercos phase.
- "Maximum cycle time": The Sercos cycle time can be configured here.
- "Sercos slaves configuration": The slave axes controlled by MLD-M can be configured here.
 - The "Axis configuration" tab lists the maximum number of configurable slave addresses in a table; the drive addresses found can be configured. Only the configured drive addresses are inserted into the CCD ring. [The device addresses of the I/Os (Inline I/O, PLC drives, etc.) can be configured in the project tree.] The addresses of the configured drives are displayed (P-0-1801.0.10). In the

"Required" column, the configured drive addresses can be temporarily excluded from the CCD ring by selecting "No".

"Addresses found": Here, all addresses of the CCD slaves found via Sercos are displayed (P-0-1803.0.3).

- On the "IO configuration" tab, all device addresses are displayed that are created and used as IO device.
- On the "Topology" tab, the topology of the configured device addresses is displayed.
- On the "Topology monitoring" tab, the topology monitoring can be activated and the CCD ring can be restored if a ring break has been detected.



All axes listed here can be accessed via MLD. The address to be set in the PLC is based on the order of configured slaves. Addressing takes place as follows:

- "Axis1" always addresses the local axis (which is not displayed in this case)
- "Axis2" addresses the first configured slave.
- "Axis3" addresses the second configured slave.
- ...
- "Axis10" addresses the ninth configured slave.

Control panel

The control panel is used to check whether or not the optional "ML", "MA" or "TF" expansion package (as of MPx-20) has been enabled in the drive and whether or not the PLC functionality can be used.

After control voltage has been switched on, the drive is booting up. During the booting process, messages will appear on the display. Before the "Boot 2.9" message appears, the "Esc" and "Enter" keys have to be simultaneously pressed and held. If the PLC has been enabled ("ML" or "MA" firmware package licensed), the "PLC ?" message appears on the display.

The arrow keys on the control panel can be used to switch between "Run PLC" and "Stop PLC". If an operable PLC project has been stored in the drive, it can be started with "Run PLC". The option shown on the display can be selected by pressing "Enter".

Important information on specific IndraLogic 2G functionalities

IndraLogic 2G is the PLC programming interface for different PLC systems that has been integrated in IndraWorks MLD.

Observe the following:

- System events are not supported.
- The parameter manager is not supported. Network variables are supported in version MPx-18VRS and above.
- The "Update unused I/Os" functionality is not supported. Instead, the I/O ranges are displayed in IndraWorks MLD.
- The "Retain forcing" functionality is not supported.
- The task runtimes are not displayed in IndraLogic 2G. Instead, the runtimes can be determined in the program with the "MX_IECTaskGetLoad" function from the "MX_PLCCopen.library".

- The "Persistent variables" functionality is only available if a memory card is used. Due to the system, this function can be selected for all projects in the PLC variable declaration.

Safe programming

Using pointers

Pointer accesses are monitored at runtime. This prevents programs from accessing memory outside of the PLC data areas. An incorrect access attempt generates a PLC exception with F6010.

DANGER

Lethal injury or property damage from vertical axes moving inadvertently or dropping suddenly!

⇒ When using pointers, incorrect access can have unforeseeable consequences, so exercise extreme caution.

Stack check

Stack requirements are checked during compilation. This prevents a program with excessive stack requirements from being compiled.

DANGER

Lethal injury or property damage from vertical axes moving inadvertently or dropping suddenly!

⇒ Using functions with large data volumes can have unforeseeable consequences. The stack requirement for function blocks is minimal, as their data are stored in the PLC data. Functions, however, have no instance and store all data in the stack. Avoid using big data structures or fields, or create function blocks.

Array access (exceeding the range)

DANGER

Lethal injury or property damage from vertical axes moving inadvertently or dropping suddenly!

⇒ If there is no "CheckBounds" function in the project, indexed access to arrays is not checked (see also documentation "PLC Program Development with Rexroth IndraLogic 1.0"). If the index is outside of the allowed range, foreign data are read or overwritten. This presents the risk of incorrect access having unforeseeable consequences.



For new projects created with version MPx-18, the corresponding check function is contained in the automatically loaded "CheckRtv" library.

Subrange types (exceeding the range)

⚠ DANGER

Lethal injury or property damage from vertical axes moving inadvertently or dropping suddenly!

⇒ If there is no "CheckRangeSigned" or "CheckRangeUnsigned" function in the project, access to subrange types is not checked (see also documentation "PLC Program Development with Rexroth IndraLogic 1.0"). If the value is outside of the allowed range, this can have undesired effects in the PLC program. This presents the risk of incorrect access having unforeseeable consequences.



For new projects created with version MPx-18, the corresponding check function is contained in the automatically loaded "CheckRtv" library.

Division by zero

⚠ DANGER

Lethal injury or property damage from vertical axes moving inadvertently or dropping suddenly!

⇒ If the project does not contain any functions used to check for division by zero (see also documentation "IndraLogic 2G Programming Guide"), this can have undesired effects in the PLC program. When the divisor is 0, incorrect access can have unforeseeable consequences.



For new projects created with version MPx18, the corresponding check function is contained in the automatically loaded "CheckRtv" library.

Access to variables via programming system or interface is only byte-consistent

When a 2- or 4-byte variable is read, the read process can be interrupted by a change in the variable in a PLC task. When several contained bytes are simultaneously changed, the wrong value may be temporarily transmitted (e.g., one byte old, one byte new).

7.2.3 Using the "AxisInterface" for programming Rexroth IndraMotion MLD

The "AxisInterface" function is provided in the "MX_TechInterface.library".

The axis interface bundles and enhances PLCopen motion function blocks and provides an easy-to-use interface for drive functionality. Less code and more efficient commands speed up the programming of applications.

The axis interface contains control signals and parameters for the various operation modes of the master axis and slave axis, as well as options for setting selected process values.



The "AxisInterface" function can be activated by double-clicking the Logic branch.



The "MX_TechInterface" library is described in the IndraWorks online help and in the printed documentation "Rexroth IndraMotion MLC 14VRS, Technology libraries".

Other features that should be noted when using the axis interface in the IndraMotion MLD system are described in the IndraMotion MLD library description.

7.2.4 Automated program generation with "GAT compact"

"GAT compact" (Generic Application Template) is a sample project, based on IEC 61131-3. It is made available by the IndraWorks engineering framework "IndraMotion MLD" as of version 14V14 and a wizard can be used for loading into the drive and start up as of firmware version MPx-20. "GAT compact" contains the following partial functions:

- SFC with steps for initialization, manual and automatic mode and error status
- Integrated error management
- Integrated axis handling
- Commissioning visualizations
- Prepared HMIs

"GAT compact" sample project

Introduction

The description below will help you to create a Motion Logic application with one axis based on the "GAT compact" example project in only a few minutes.

The sample project is ready-to-run; it may be freely extended and modified. GAT compact uses the AxisInterface and contains a machine mode SFC as well as troubleshooting. Additionally, it contains prepared commissioning visualizations and HMIs.

With the sample project, the local axis can be moved in automatic mode in velocity control with an adjustable velocity. In manual mode, for example, the axis can be moved by means of visualization.

Requirements

"GAT compact" is available in the IndraWorks Engineering Framework "IndraMotion MLD" as of version 14V14.

A drive controller is required which contains at least firmware version MPx-20. Moreover, the "IndraMotion MLD" function packages has to be enabled.

In order for the "GAT compact" sample project to work, the PLC has to be configured as follows:

- PLC has permanent control over drive (P-0-1367, bit 4)
- Support of the AxisData structure (P-0-1367, bit 6)

In IndraWorks, the dialog for configuring the PLC can be called by double-clicking "Configuration" in order to make the necessary settings:

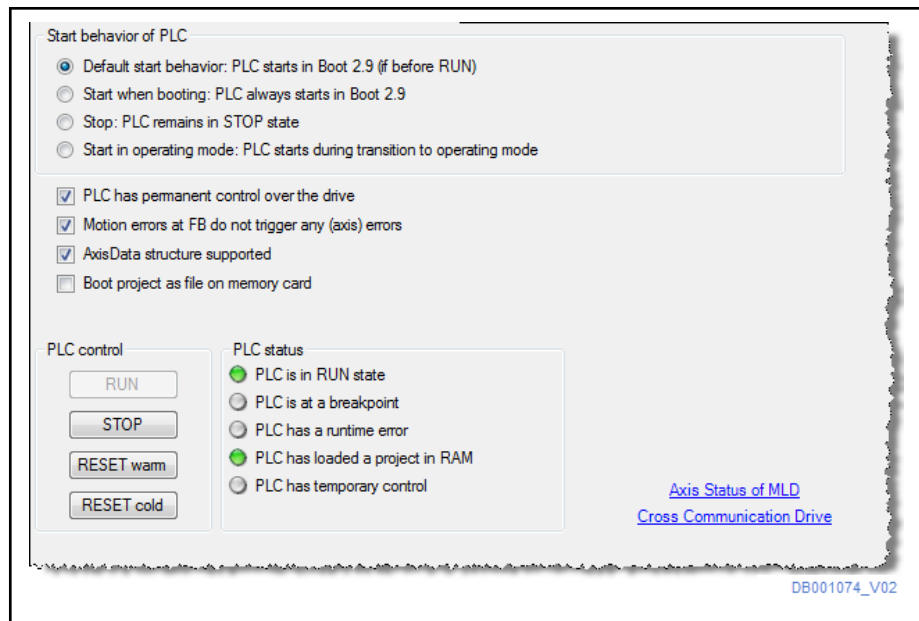


Fig. 7-11: Configuration dialog

First steps with the "GAT compact" sample project

Inserting "GAT compact"

1. Insert a GAT-compatible device (e.g. drive controller "IndraDrive") into an IndraWorks project using an MLD project.

The "Create logic" dialog appears.

2. In the "Create logic" dialog, select "GAT compact example project" and confirm your selection by means of **OK**.



If no logic has been created yet, you can call the "Create logic" dialog by double-clicking the Logic branch in the Project Explorer.

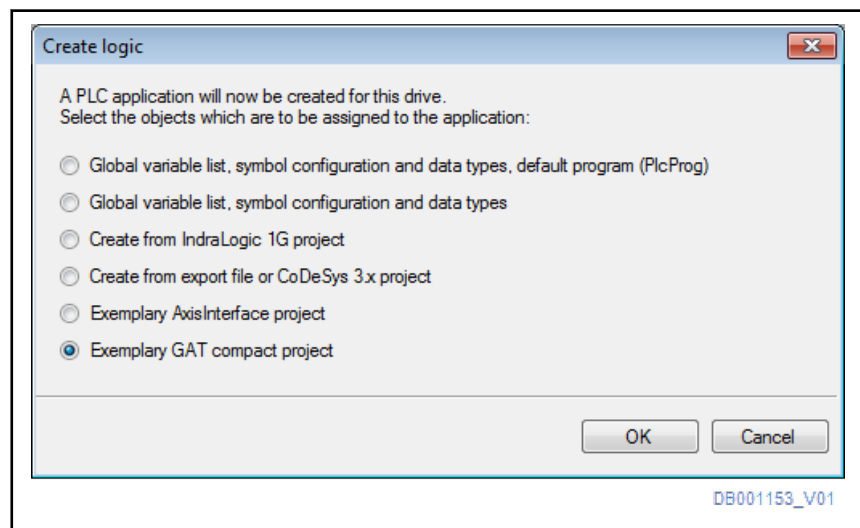


Fig. 7-12: Logic creation

The "GAT compact sample project" is automatically inserted into the PLC application

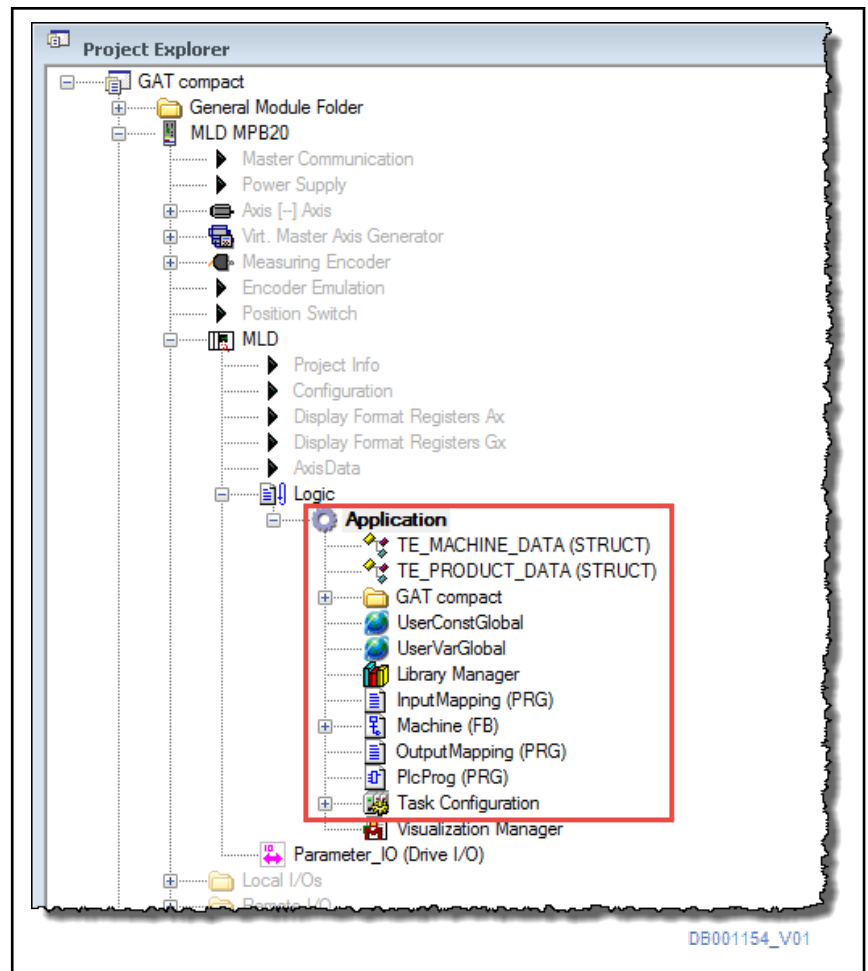


Fig. 7-13: "GAT compact" sample project

Project loading and starting

Load and start the "GAT compact" sample project by carrying out the following steps:

1. Login
2. Load the PLC application to the control
3. Start the PLC application

Operation via visualization

The "System_Overview_All_Axis" commissioning visualization available in the "GAT compact" sample project allows for easy operation of the "GAT compact" sample project.

1. Open the visualization by double-clicking "System_Overview_All_Axis" in the "GAT compact" window.

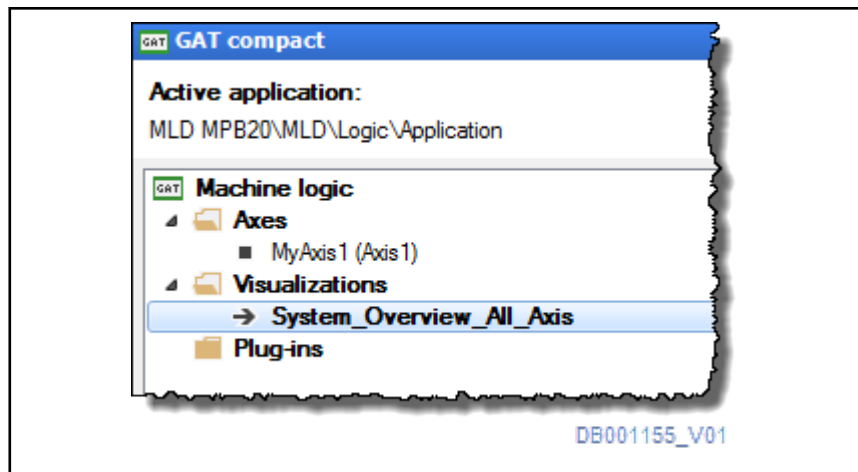


Fig. 7-14: Open visualization



The "GAT compact" window can be opened via **View ► Other windows ► GAT compact**.

The "System_Overview_All_Axis" visualization is opened.

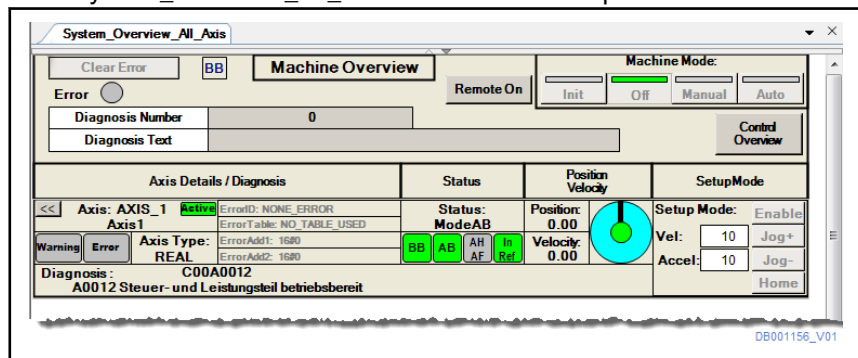


Fig. 7-15: "System_Overview_All_Axis" visualization

- In the visualization, select **Remote On** in order for the visualization to take over control over the "GAT compact" sample project

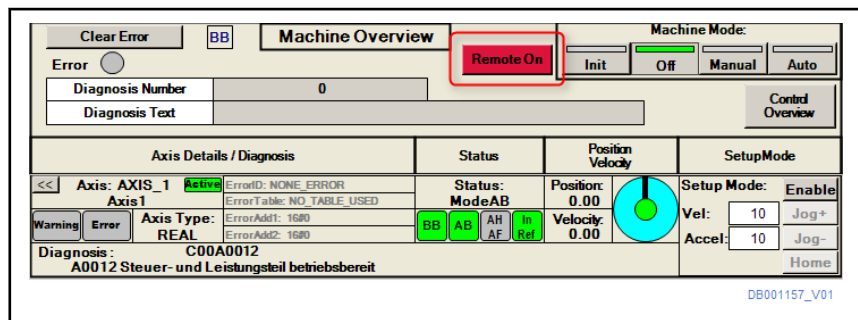


Fig. 7-16: Taking over control over the "GAT compact" sample project by means of the visualization

- Select **Manual** and then **Enable**.
Using **Jog+** and **Jog-**, you can move the axis.

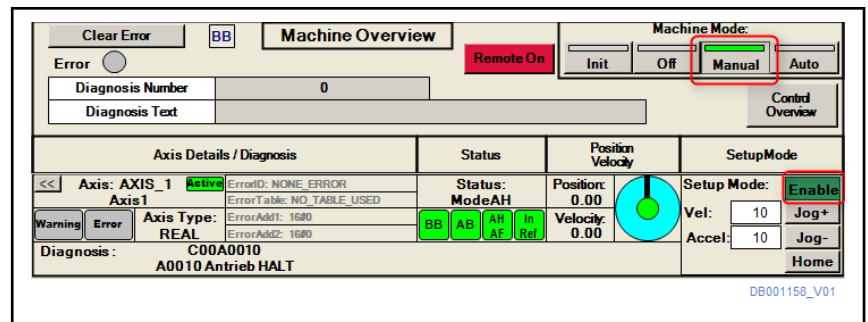


Fig. 7-17: Moving the axis in manual mode

4. Activate the automatic mode by means of **Auto**.

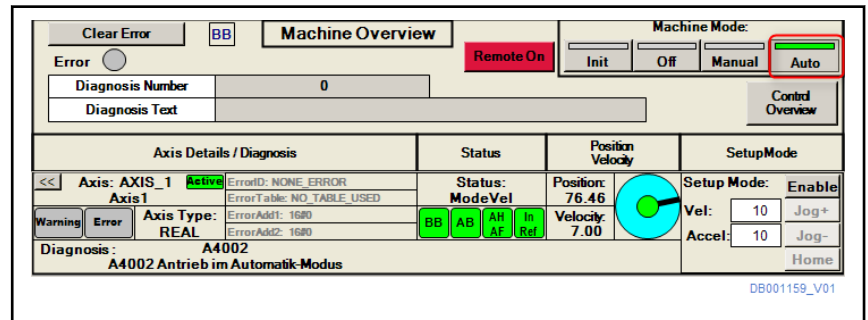


Fig. 7-18: Moving the axis in automatic mode

The axis moves continuously with 7 rpm as in the "Auto" operating mode, a motion at continuous velocity is programmed.

Control of the operating modes by means of the "Machine" functional block

The "Machine" functional block is the main element of the "GAT compact" sample project. The function block contains an SFC and the implementation of operating modes.

Double-clicking the "Machine" function block opens the operating mode SFC.

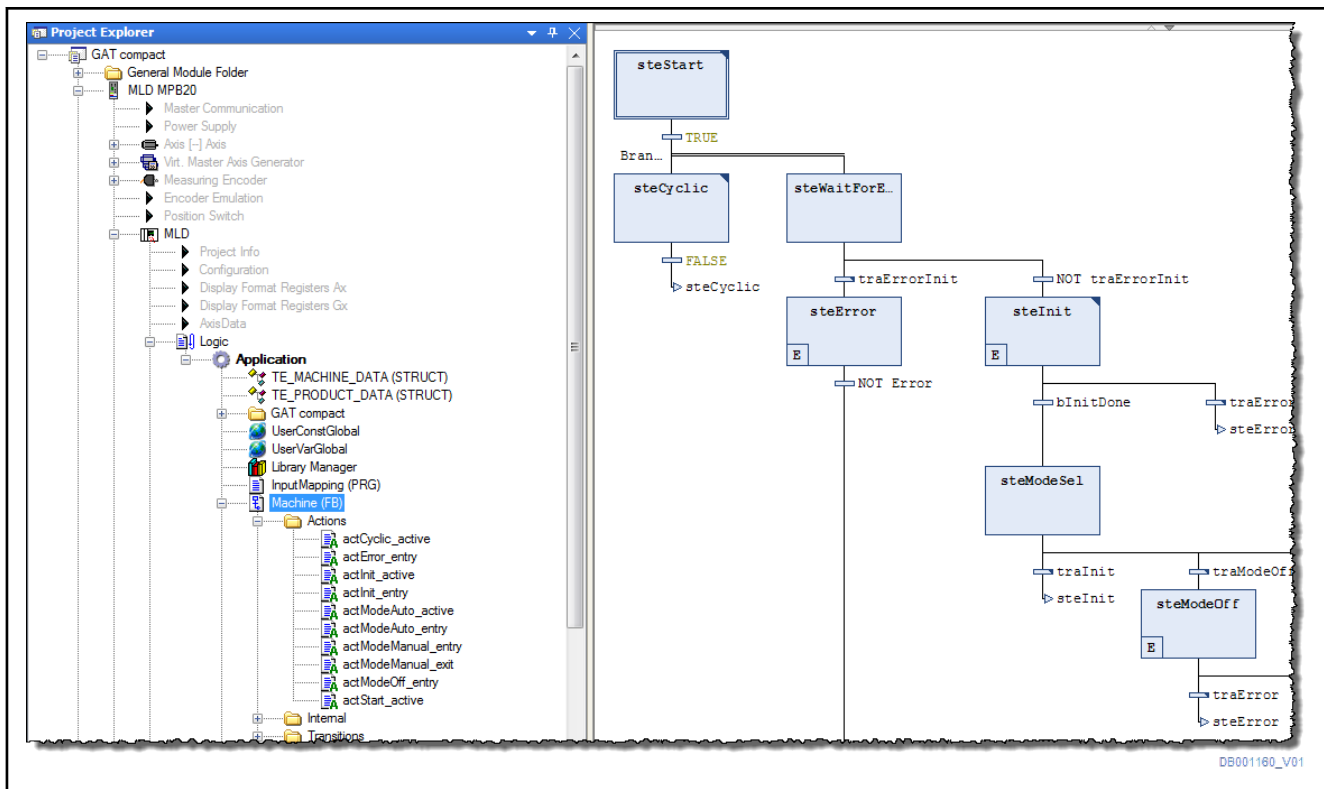


Fig. 7-19: "Machine" function block

The "actModeAuto_entry" input action of the "Machine" function block is processed once after selection of the "AUTO" operating mode. This action starts the axis motion. The axis is set to velocity mode (ModeVel).

The "actModeAuto_active" action is processed cyclically on an ongoing basis if the "Auto" operating mode is active. Here, the command velocity for the axis and the displayed velocity are cyclically copied. Consequently, changes in the velocity will immediately take effect in the "Auto" operating mode.

The "actModeManual_entry" input action stops all axes by means of the FOR loop via all axes and specification of "ModeAb" operating mode.

The "actCyclic_active" action is permanently and cyclically called independently from the machine operating mode. The action contains the call of the AxisInterface and error management.

The "actError_entry" input action is processed once after an error occurs. It contains the error reaction. In the example, all axes of the axis container are decelerated to standstill. Additional deceleration examples are included commented out.

7.3 PLC project structure

7.3.1 General information

The PLC project consists of multiple files and is part of an IndraWorks project:

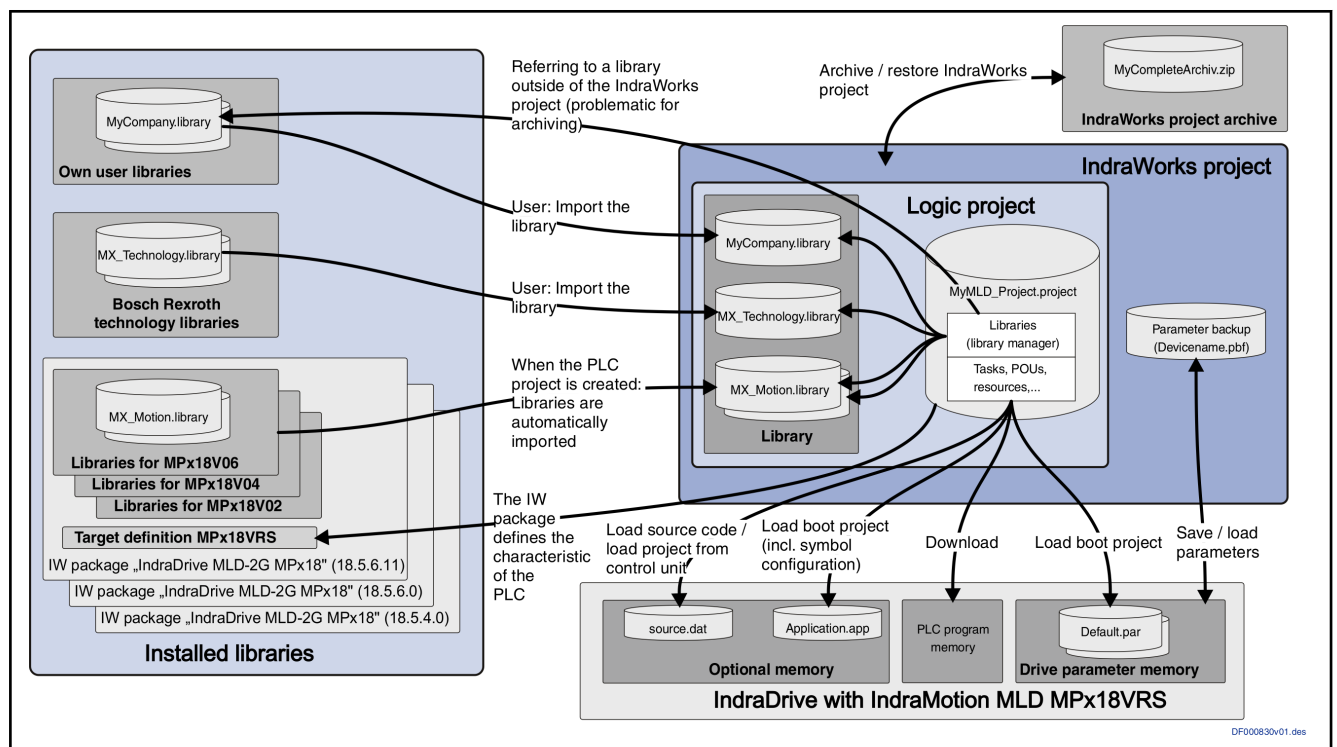


Fig. 7-20: Essential PLC project files in the context of the IndraWorks project

The left side of the above figure shows the IW packages and libraries installed on the PC. It also shows optional libraries which were created by the installation of technology packages or by the user's installation.

The bottom of the figure shows the drive with the relevant data memories. When the external memory card is available, the PLC source code can be loaded or symbol information for HMIs can be loaded. In this case, the parameter memory is also contained on the memory card (not visible in the figure).

The top of the figure shows an IndraWorks project archive. When the project is archived, it is transferred to a single packed file. The IndraWorks project itself consists of structure information, an optional parameter backup and an optional PLC project. The PLC project is contained in a subdirectory and contains all files created by IndraLogic. The most important of these files is the one with the *.project extension which contains the PLC source code and project settings. The PLC directory contains a subdirectory called "LIB". When a new project is created, the system copies all libraries of the appropriate release of the current target to this subdirectory.

The actions which the user can carry out are marked with arrows and will be explained in detail in the following chapters.

7.3.2 Saving projects in the drive

When a PLC project is created and loaded, different files are generated in the target (drive).

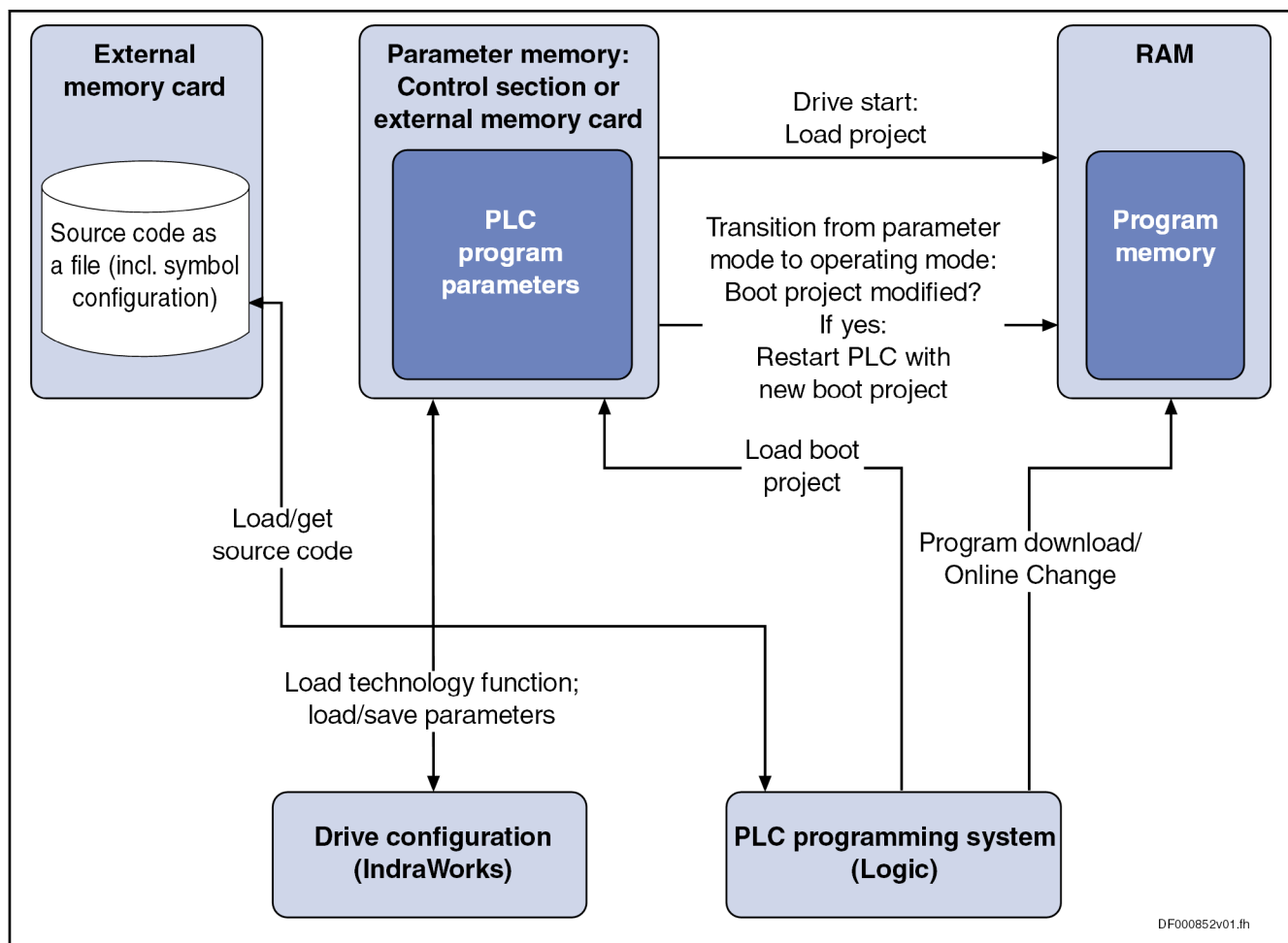


Fig. 7-21: Saving projects in the drive

The "program download" only loads the PLC program to the RAM of the drive. When the boot project is transmitted to the drive with the **Load boot project** menu item, IndraLogic first transmits several files that, by default, are stored in the drive in coded and compressed form as a set of several parameters.

- **Operation without external memory card**

If an external memory card is not available, the source code is stored in parameters.

- **External memory card as active memory**

If an external memory card is available, the PLC project as source code and the symbol configuration can be additionally stored in the drive. In this case, the boot project and the symbol configuration are contained in the parameter file on the external memory card.

7.3.3 Boot project

The boot project can be loaded and saved with IndraWorks MLD in the form of parameters (P-0-1352 .. P-0-1358). This means a backup of the parameters contains the boot project, i.e., the binary code, but not the source code.

The boot project of the PLC has the following properties:

- It consists of several system files. They contain the binary project without symbolic information. With **Create boot project** (online only), the system files are automatically loaded to the drive.

- In the drive, the system files:
 - ... are distributed to list parameters P-0-1352 to P-0-1358 and stored.
 - ... are stored on the external memory card as a file, which requires that a memory card be used and the corresponding "MLD configuration" be made in IndraWorks or P-0-1367 be parameterized.
- The boot project first does not have any influence on the program currently running in the RAM. Only after the drive has been switched on again is the boot project loaded to the RAM and possibly started.
- As an alternative, changed boot projects are loaded when the drive switches to operating mode (P3 to P4).

7.3.4 File system in the drive

The table below shows all files available in the drive. Many of the files are not directly visible to the user or are automatically managed. All "system file" type files are directly managed by the drive firmware. The PLC system files are managed by IndraLogic.

This means that there are files which are used internally and only exist in the memory. Furthermore, there are files available in the form of parameters, independent of the external memory card. All other files are contained on the memory card.

In the "Documentation" directory, the user can store their own files to keep records of their installation/machine.

Medium	Directory	File	Type	Content	Access
Parameter P-0-1352	-	Application.sts	PLC system file	PLC status	Logic context menu: Start/Stop
	-	Application.crc	PLC system file (P-0-1367.0.8=0)	Boot application checksum	Logic context menu: Create boot project
Parameters P-0-1353 to P-0-1358 (IndraDrive only)	-	Application.app	PLC system file (P-0-1367.0.8=0)	Boot application	
Internal flash (HydraulicDrive only)	-	Application.app	PLC system file (P-0-1367.0.8=0)	Boot application	
External memory card	Plc	Application.crc	PLC system file (P-0-1367.0.8=1)	Boot application checksum	Logic context menu: Load source code
		Application.app	PLC system file (P-0-1367.0.8=1)	Boot application	
		Archive.prj	PLC system file	Source code archive	
	Documentation	*.*	User files	Installation documentation	External
	Tools	*.*	User files	Tools	External
	User	*.*	User files	User files	PLC program with "Sys-File", FTP or IndraWorks Explorer
	Backup	*.*	PLC system file	Retain data and parameter backup	Logic: "Backup" command

Tab. 7-1: File system in the drive

7.3.5 External memory card as a storage medium

When MLD is used, the external memory card is particularly important. The memory card is permanently plugged and works as an active parameter memory. (The drive-integrated PLC can also be used without the memory card.)

Storing the source code of a PLC application

The "Load source code" function stores the complete PLC program on the drive/external memory card, i.e., the entire source code is stored in addition to the binary code. This allows the PLC project to be restored later by using the external memory card.



It is useful to also archive the PLC project on the PC. For this purpose, there is a menu-driven archiving option in the programming system. We recommend the entire IndraWorks project to be archived with IndraWorks MLD.

Restoring a PLC project from the loaded source code

A source code stored on the external memory card can be loaded from the drive via the **File ► Open...** menu item. When doing this, observe that when a project is loaded from the control unit, this project is not restored with the same file structure. There is an upload directory in which all additional files, such as libraries, are stored. To embed the libraries, the library path is set in such a way that it points to the upload directory. If a project restored in such a way is to be stored on the PC again, the files from the upload directory have to be copied to the library directory of the project. For further information see IndraLogic 2G Help.

Symbol configuration for HMI (e.g. VCP)

When an external memory card is available, the symbol configuration is loaded to the external memory card (µSD-Card) with the program code. The symbol configuration can be used to symbolically access PLC variables via the communication interface. This is necessary for displaying PLC project variables on the comfort control panel or an external operator terminal.

Via the symbol configuration it is possible to set whether and which symbols are to be stored.

Generally, only the required symbols should be selected to reduce the loading times (download) and to achieve high processing velocity of symbol access by an HMI or comfort control panel.

After transmission, the symbols are loaded in the drive. This requires dynamic memory which, however, is limited. Select less symbols if the memory is not sufficient for the symbols.

The memory actually required depends on the number of symbols, their names and the data types.



The available memory depends on where the boot project is stored.

Storing user data on the external memory card using the PLC

User files can be accessed from the PLC program. The "SysFile"/"SysFileAsync" and "SysDir"/"SysDirAsync" libraries are provided for this purpose.

Storing user data on the external memory card using an FTP server

The external memory card can be accessed via an FTP server.

7.3.6 IndraWorks package

The different firmware versions and firmware releases are supported by the corresponding IndraWorks packages. Each firmware version includes its own IndraWorks package. The IndraWorks packages are installed during the installation of IndraWorks MLD or can be subsequently installed via the "Change IndraLogic device version" dialog (Project Explorer: **MLD ► Change IndraLogic device version**) by clicking the "Install IW package" link.

7.3.7 Libraries

Functionalities and resources (variables, data types, etc.) are provided via libraries. There are different categories of libraries:

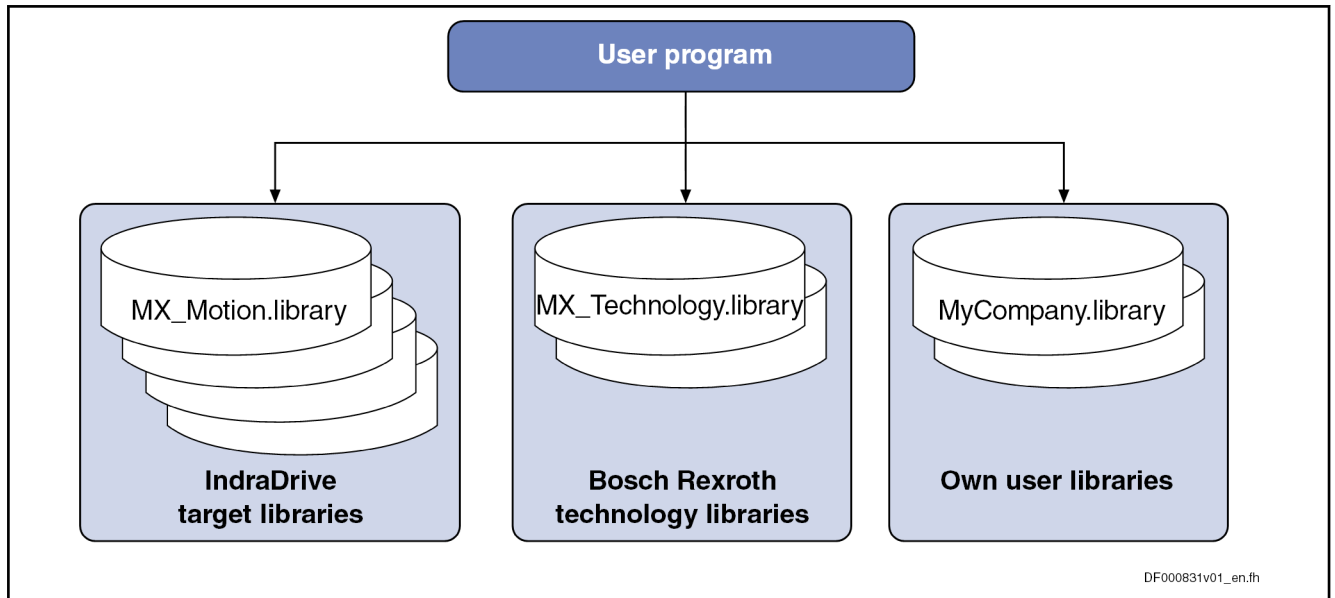


Fig. 7-22: Library categories

IndraWorks includes a collection of libraries.

In addition, libraries with technology function blocks available. These libraries use the installed libraries and make an extended functionality available.

Optionally, the user can include other libraries, for example company-specific ones.



Libraries alien to the target (libraries of other targets or libraries from alien sources) cannot be used in MLD.

7.4 Programming and commissioning steps

7.4.1 Creating a new project

Since a PLC project always is a component embedded in the IndraWorks project, a new project has to be created in IndraWorks MLD in the first step. This new project is empty at the beginning. There are two possibilities to include a drive in the project:

- **To include a drive in the project in offline mode**, left-click to select a drive (IndraDrive) from the device library. Keep mouse button pressed and drag the drive to the top branch in the Project Explorer. When the selected drive is dropped, the configuration wizard for the drive starts.
- **If the drive is already available**, it can be directly connected via a serial connection or an Ethernet connection. The "Scan for devices" command is used to automatically identify the drive which can then be included in the project. It is not necessary to set the basic configuration.

If the PLC functionality has not yet been activated in the functional packages, this has to be done now (see Functional description of firmware "Enabling functional packages").

Double-clicking the logic branch creates an empty PLC project. Afterwards, the PLC configuration can be started by double-clicking again or via the context menu.

7.4.2 Importing a PLC Project

Objective To use existing PLC projects which were created with an earlier IndraLogic version without MLD, proceed as follows:

- Required steps to follow**
1. Create an IndraWorks project.
 2. Initialize the control unit in the Project Explorer by double-clicking the logic branch.
 3. Use the context menu to select **Application ► Data import...**

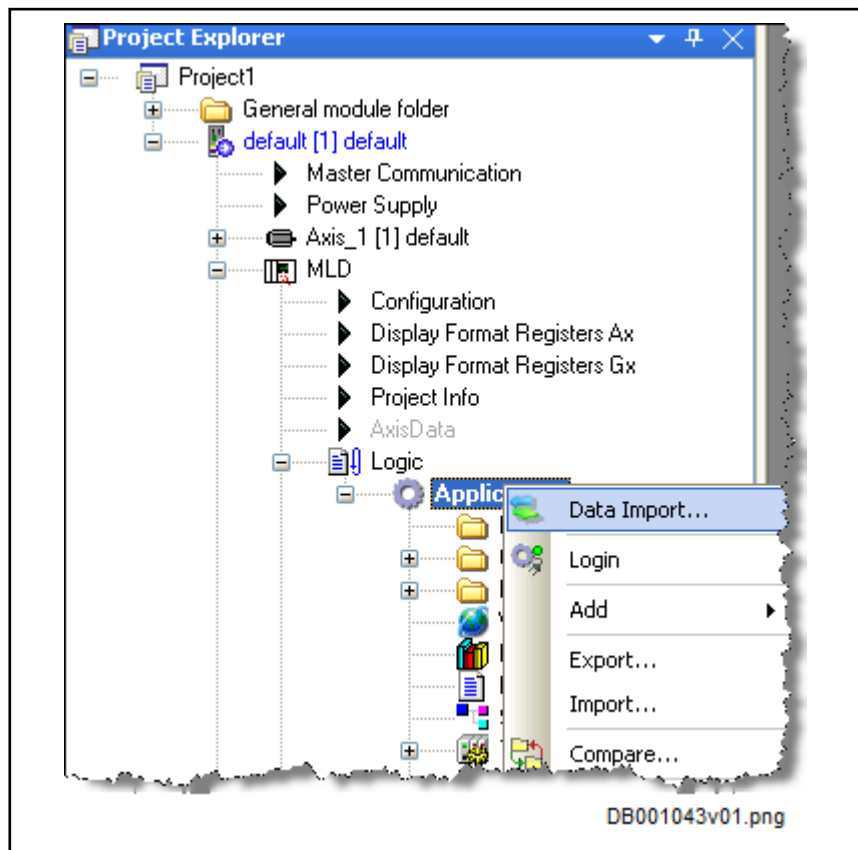


Fig. 7-23: Project tree with Logic branch

4. In the "Select project" dialog box, select the desired project file.
5. Clicking the "Open" button starts the import process.



Observe the following points when importing PLC projects:

- The trace configuration and sampling trace contained in the source project are not imported. Workaround: Store the sampling trace in a *.trc file and the trace configuration in a *.tcf file. Both files can be loaded after IndraLogic was imported.
- Network variables cannot be imported (error message).
Workaround: Write the global network variables in the source project to an *.iwx file using the **Export project...** function in IndraLogic. Import this *.iwx file to the target project using the **Import project...** function in IndraLogic.
- The target settings (Resources tab) and the settings under **Project ► Options** are not imported, but set to default values. These settings have to be checked and, if necessary, adjusted.
- When your own function blocks written in CFC are imported, the graphic connections and the function block designations are lost.

7.4.3 Exporting an IndraLogic project (*.project file) from IndraWorks MLD

Objective To use an IndraLogic project outside of the IndraWorks environment, proceed as follows:

- Required steps to follow**
1. Right-click the Logic branch in the Project Explorer.
The context menu opens.
 2. Select **Save as....** This menu item is only available in the context menu, when the IndraLogic project has already been stored in the IndraWorks project.
 3. Assign a file name to the IndraLogic project.



The device version is not exported together with the IndraLogic project! Make sure that the right device version is available when the exported IndraLogic project is to be used.

7.4.4 Loading the IndraLogic project to the drive

While the program is created, the IndraLogic project can be loaded to the drive in various ways:

- "Download"
- "Online change"
- "Load boot project"

Download With "Download", only the currently running project is changed. The boot project loaded at the next start is not affected. A boot project has to be loaded if a project is to run continuously in the drive, i.e., at the next start.

Online change "Online change" allows the project to be modified to a certain extent during commissioning without stopping the control unit. This does not affect any loaded boot project.

Note that "Online change", according to the program size, can temporarily interfere with the processing of the program. Time-critical tasks might possibly cause the watchdog to be triggered.

Load boot project

Even programs using the real-time channel can be changed in online mode. The system automatically modifies the configuration of cyclic data exchange.



You have to be "Online" to load the boot project (logged into the drive). Otherwise, the boot files will only be generated on the PC but not transmitted to the drive.



When the control section with an external memory card is used, the boot project can alternatively be stored on the memory card as a file (specific "MLD configuration" in IndraWorks or corresponding parameterization of P-0-1367).

The program compiled in binary form is stored in parameters in compressed form. For this purpose, a specific compression method is used which achieves compression to a remaining size of typically 35% to 50%.

The boot project is currently stored in the following parameters (the parameters are administrated by the system and must not be directly changed by the user):

- P-0-1352, PLC user program administration data
- P-0-1353, PLC user program area 0
- P-0-1354, PLC user program area 1
- P-0-1355, PLC user program area 2
- P-0-1356, PLC user program area 3



For the size of the boot project, there are approx. 650 kB available in parameters and 8 MB on the external memory card which allows storing big programs and filing the PLC sources and user files in parameters.

If a boot project is too big, it cannot be stored and an error message is generated while it is loading.

However, the loading process is not immediately aborted; after a while, another error message is displayed signaling that the boot project could not be generated.

Automatically loading the boot project

When a new project is created, the boot project is set to automatically generate after every download. This prevents the boot project from failing to generate although changes to the project are loaded.



If the transmission times are too long with large projects and a serial connection, the automatic loading of the boot project may be switched off in the target settings (IndraLogic). In this case, make sure to load the boot project before logging out!

7.4.5 Archiving / restoring an IndraWorks project

In IndraWorks, you can archive **entire projects** either on the local file system or on an FTP server (device or computer) connected via a network. These archives can be restored on the file system of the local computer.

The general IndraWorks documentation, chapter "Archiving and restoring projects" describes how to archive and restore an IndraWorks project.

As opposed to "archiving", IndraWorks also provides the option to **only export parts of projects** (see "[Archiving an MLD project](#)").

The following table shows more differences between "Archive" and "Export":

Property	Archive	Export
Entire project	✓	✓
Part of a project	–	✓
Compressed	✓	–
Password-protected	✓	–
Filing on the file system	✓	✓
Filing on an FTP server	✓	–

Tab. 7-2: Difference between "Archive" and "Export"

7.4.6 Loading the source code

After login, IndraLogic allows all PLC project information to be packed in a file with **Online ► Load source code**. If an external memory card has been plugged in the drive, this file (packed source code) can be written to the memory card.



It is recommended not to use this function.

7.4.7 Opening a project in the control unit

In IndraLogic, the dialog opened via **File ► Open** contains the "PLC..." button. When the connection to a drive controller has been established and you click this button, a project is opened in the control unit after a target has been selected and the communication parameters have been set.



It is recommended not to use this function.

7.4.8 Archiving an MLD project

Objective

To commission an MLD project in an identical installation, you can archive the project. The archived project contains all data and settings so that you can log in (establish an online connection) to a running installation without having to re-compile the MLD project.



It is recommended to generate a package with all required components (e.g. in the form of a CD or DVD) to make any PC, which meets the system requirements, a programming system for IndraMotion MLD. The package should contain the following components:

- The same or higher version of IndraWorks MLD as used when the archive was created.
- The archive (.zip) of the MLD project.

Preparing the MLD Project for archiving

To make sure that you can log in to the running MLD project without repeated compilation after the archived project has been restored, you have to carry out the following steps in IndraWorks before archiving the project:

1. Call **Project ► PLC compiler version...** and make sure that the latest PLC compiler version has been selected.

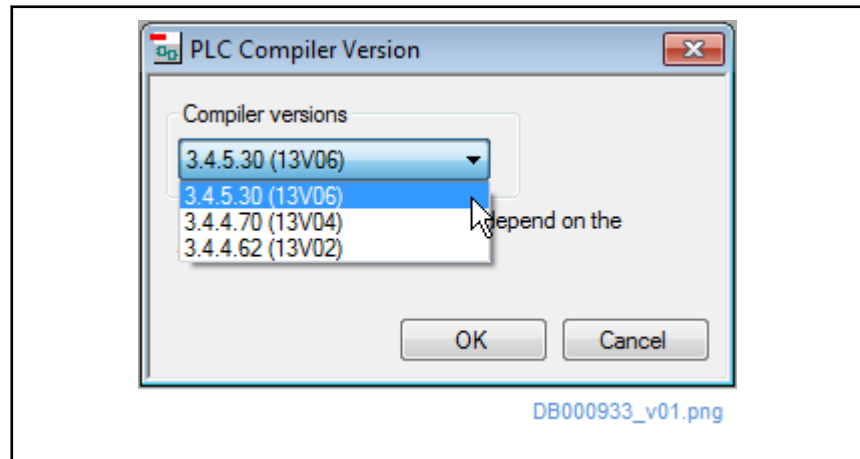


Fig. 7-24: Check the PLC compiler version

2. Execute **Build ► Clean all**.
3. Execute **Build ► Generate code**.
4. Execute **Debug ► Login**.
5. Execute **Debug ► Start**.
6. When the boot application is not implicitly generated in the properties of the application during the download, this must be done via **Debug ► Create boot application**.
7. Execute **Debug ► Logout**.

Archiving the MLD project

After the MLD project has been prepared for archiving, it can be archived.

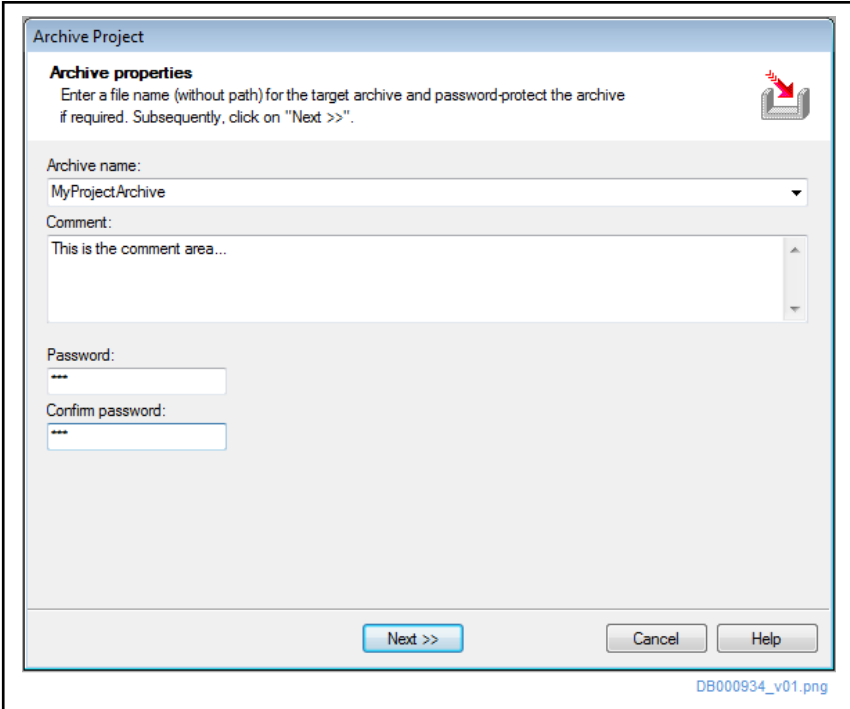


The project can be archived in online and offline mode.

In offline mode, the MLD project is archived with the offline data. Before archiving starts, the "Archiving of drive parameters" dialog provides the possibility of updating the offline data.

The following steps must be carried out for archiving the IndraWorks MLD project:

1. Start IndraWorks MLD.
2. Select **Project ► Archive....**
3. Define archive name and comment for the archive.



Archive Project

Archive properties
Enter a file name (without path) for the target archive and password-protect the archive if required. Subsequently, click on "Next >>".

Archive name:
MyProjectArchive

Comment:
This is the comment area...

Password:

Confirm password:

Next >> Cancel Help

DB000934_v01.png

Fig. 7-25: Assigning the archive properties



The archive can optionally be protected with a password.

4. Click **Next>>**.
5. In the following dialog, make the archive target settings [store the archive on a file system and/or on an FTP server (device or computer) connected via a network].

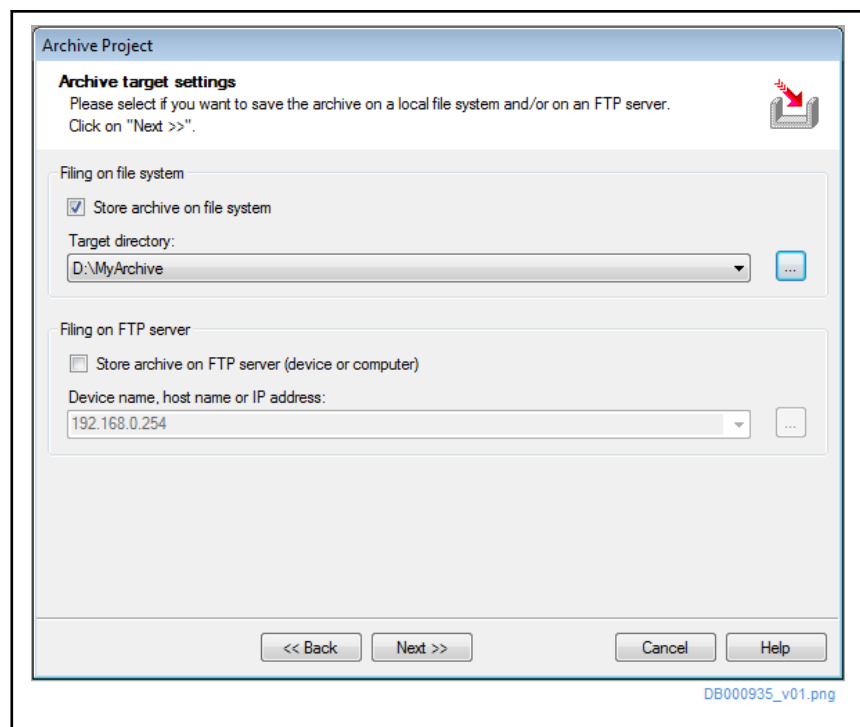


Fig. 7-26: Making the archive target settings

- **Filing on file system**

If you would like to store the archive on the file system, tick the **Store archive on file system** check box; the **Target directory** edit field can then be written. Define the directory of the file system in which the archive is to be stored.

- **Filing on FTP server**

If you would like to store the archive on an FTP server, tick the **Store archive on FTP server (device or computer)** check box; the **Device name, host name or IP address** edit field then can be written. Define the target device on which the archive is to be stored. There are four options to do this:

1. IP address
OR
 2. Host name of the target device
OR
 3. Selection via the selection list. It contains all FTP-compatible devices of the active project and the last five target devices used for archiving (device name, IP address or host name).
OR
 4. With the ... button, the target device can be selected from a list of all FTP-compatible devices of the active project.
6. Click **Next>>**.
⇒ A connection to the device that has been set is automatically established. Disruptions in the connection to the target device are displayed by means of an error message.
7. In the following dialog, the archive components must be selected:

For each archivable element or device shown in the left navigation zone, select the scope of the archiving options to be carried out in the right zone.



In the standard case, all elements should be selected!

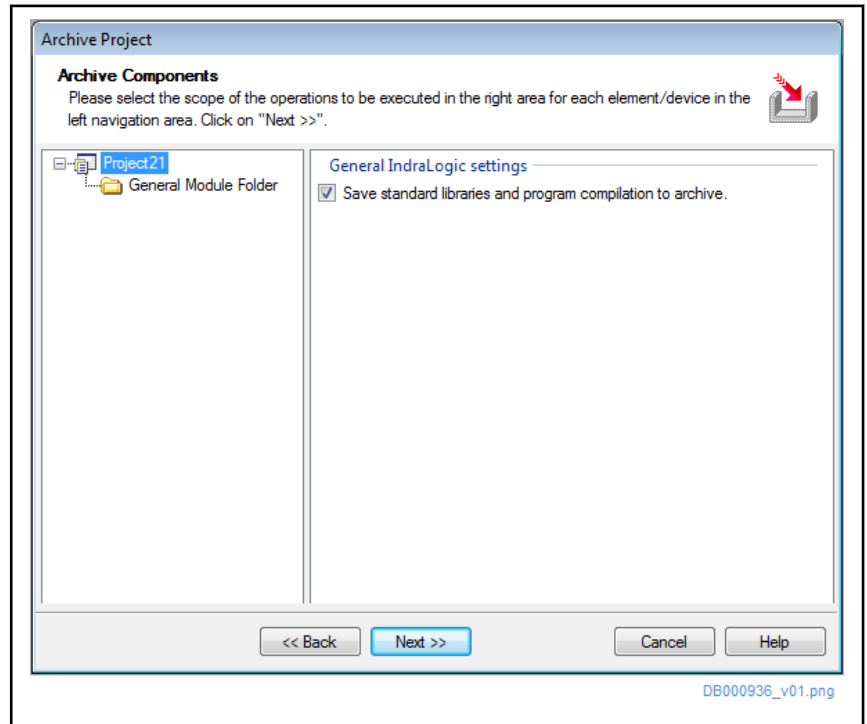


Fig. 7-27: Selecting the archive components

8. Click **Next>>**.
9. The inputs can be checked in the following dialog.

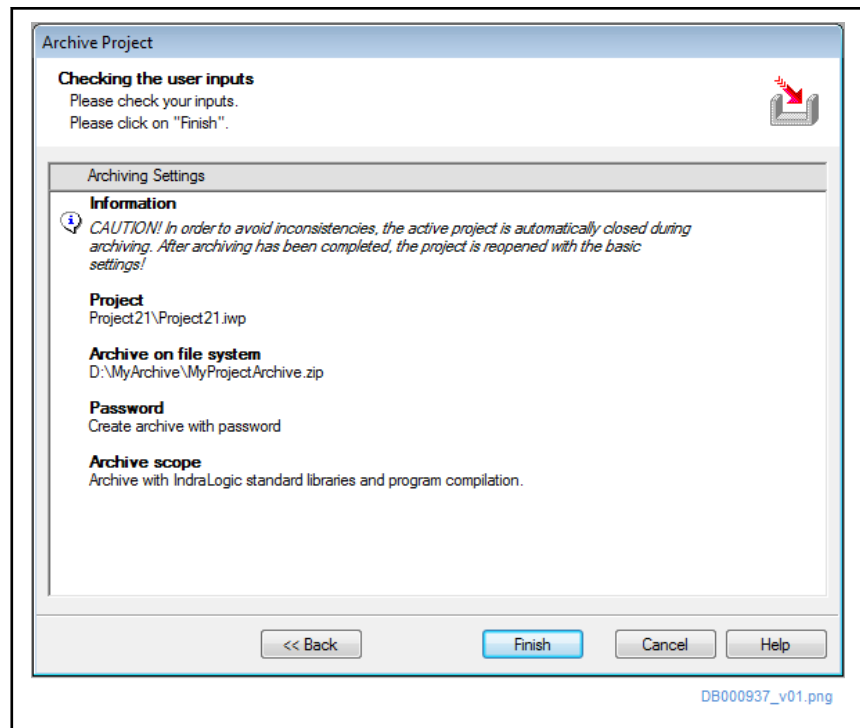


Fig. 7-28: Checking the user inputs for the archiving options

10. To create the archive, click the **Finish** button.

Restoring an archived IndraWorks MLD project

The following steps describe what to do to restore an archived MLD project.

Make sure that the same or a higher version of IndraWorks MLD as used when the archive was created is installed on the PC with which the archived MLD project is restored.

When the archived MLD project is to be used in a CCD group, the CCD group has to be commissioned before the MLD project is restored:

1. Set the following parameters according to configuration of the installation:
 - CCD slave: Drive address (P-0-4089.0.3) and, if necessary, selection of functional packages (P-0-2003).
 - CCD master: If necessary, selection of functional packages (P-0-2003) and setting of TCP/IP communication (P-0-1531 ... P-0-1533).



Changes to enabled functional packages and the TCP/IP communication settings only take effect on the next restart of the drive.

2. In IndraWorks MLD, set desired type of communication (serial or TCP/IP) and go online with the drive (CCD master) (select "Switch Online" in the context menu of the drive branch).

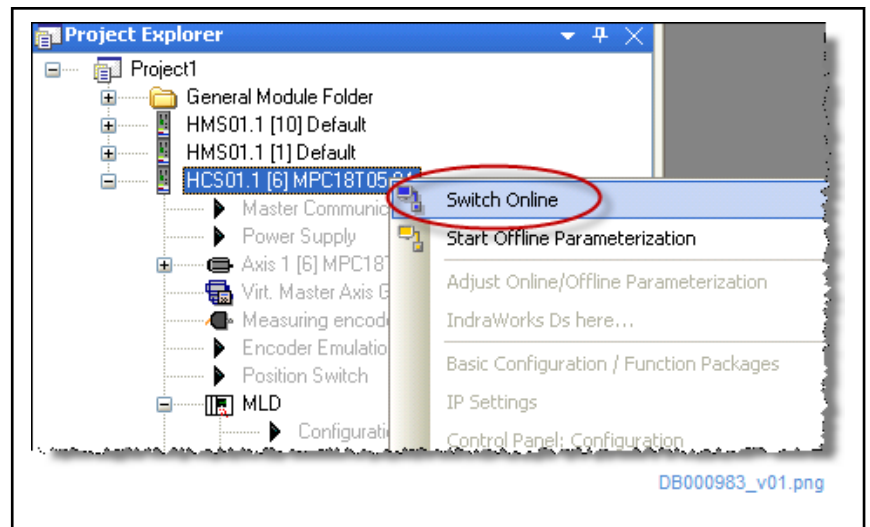


Fig. 7-29: "Switch Online" for the drive

3. In the function tree, call the context menu of **Sercos**, select **CCD: Basic Settings** and configure the CCD group.

The archived IndraWorks MLD project can be restored as follows:

1. Select **Project ► Restore....**

⇒ The "Restore project from archive", "Select restoring type" dialog opens.

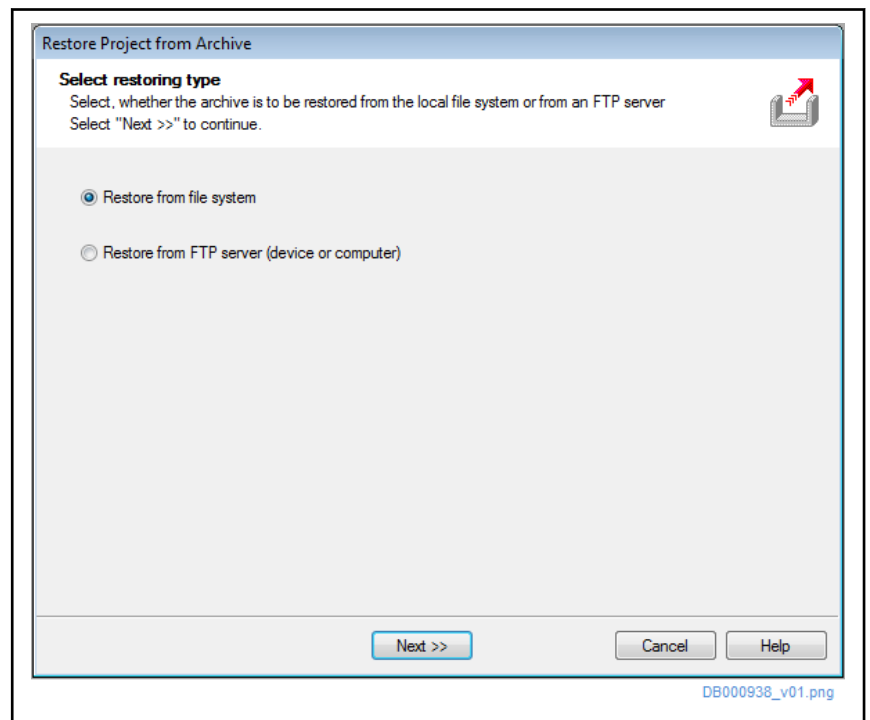


Fig. 7-30: Selecting the restoring type

2. Select "Restore from file system" or "Restore from FTP server (device or computer)".
3. Click **Next>>**.
4. a.) When "Restore from file system" was selected, the archive has to be selected in the following dialog.

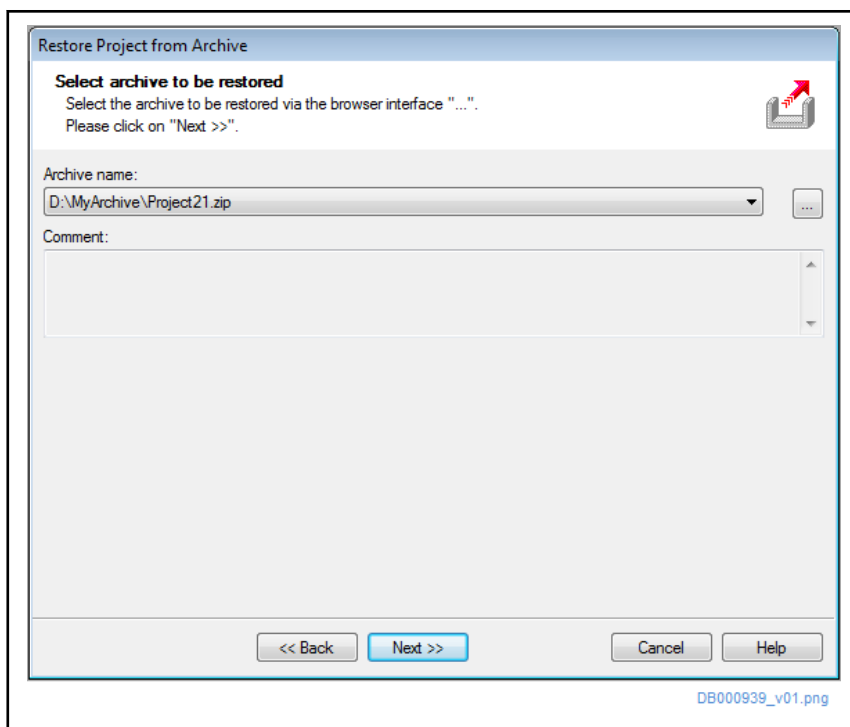


Fig. 7-31: Restoring an archive from the file system

b.) When "Restore from FTP server (device or computer)" was selected, a connection to the respective device has to be established.

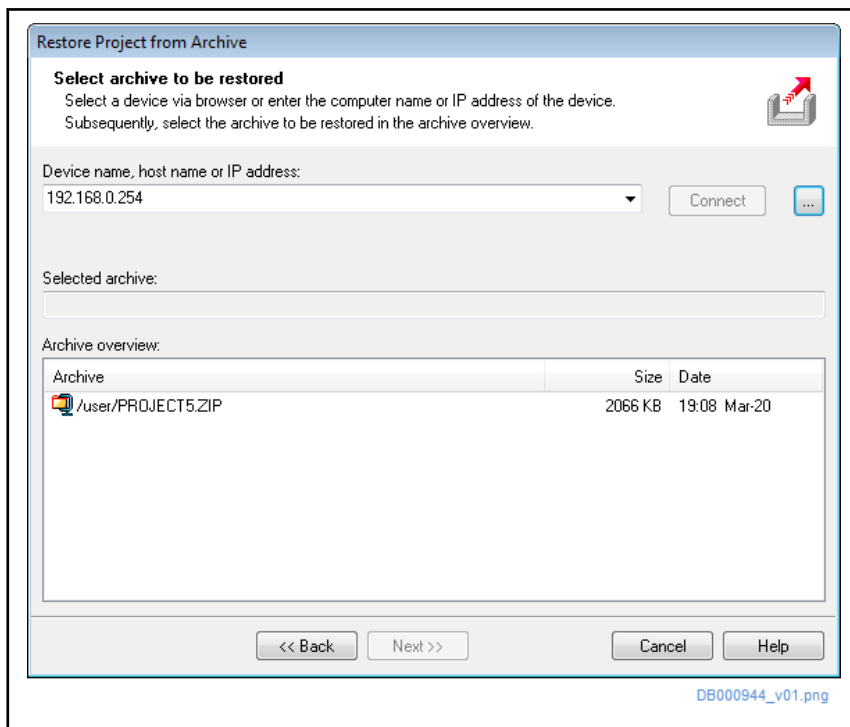


Fig. 7-32: Restoring an archive from an FTP server

There are four options to do this:

4.1 IP address

OR

4.2 Host name of the target device

OR

- 4.3 Selection via the selection list. It contains all FTP-compatible devices of the active project and the last five target devices used for restoring (device name, IP address or host name).

OR

- 4.4 With the ... button.

When the device is selected via the selection list or via the device browser, a connection to the selected device is automatically established. When the IP address or the host name is input, the connection to the target device is established via **Connect**.

5. After the connection has been established, all archives available on the device are displayed in the "Archive overview" list.
Select the archive to be restored.
6. Click **Next>>**.
7. Select the target directory in which the project is to be restored.

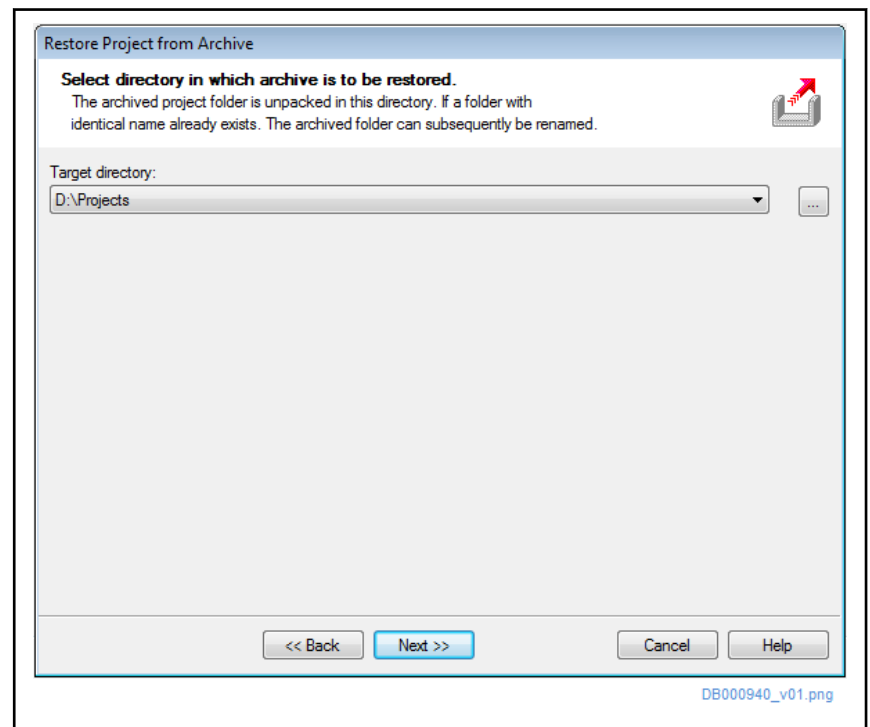


Fig. 7-33: Selecting the target directory

8. Click **Next>>**.
9. The inputs can be checked in the following dialog.

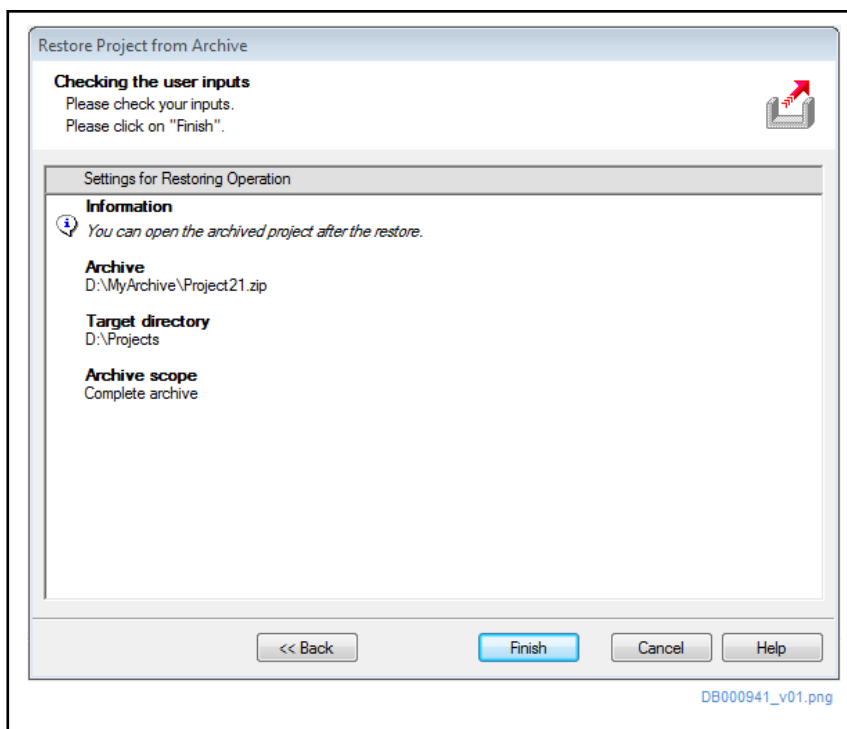


Fig. 7-34: Checking the inputs

10. To restore the archive, click the **Finish** button.
11. If the archive has been protected with a password, the password has to be entered now.

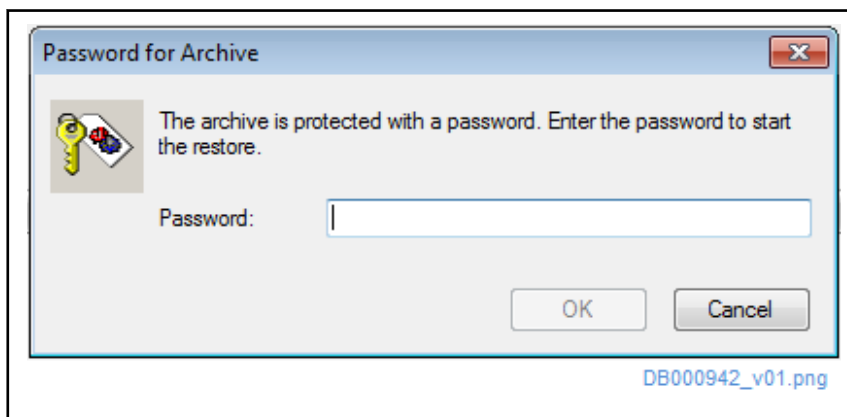


Fig. 7-35: Entering the password for the archive

12. The "Summary" dialog provides the option of selecting **Open project when closing the wizard**. If this option is selected, the restored project is opened after you have clicked **Close**.

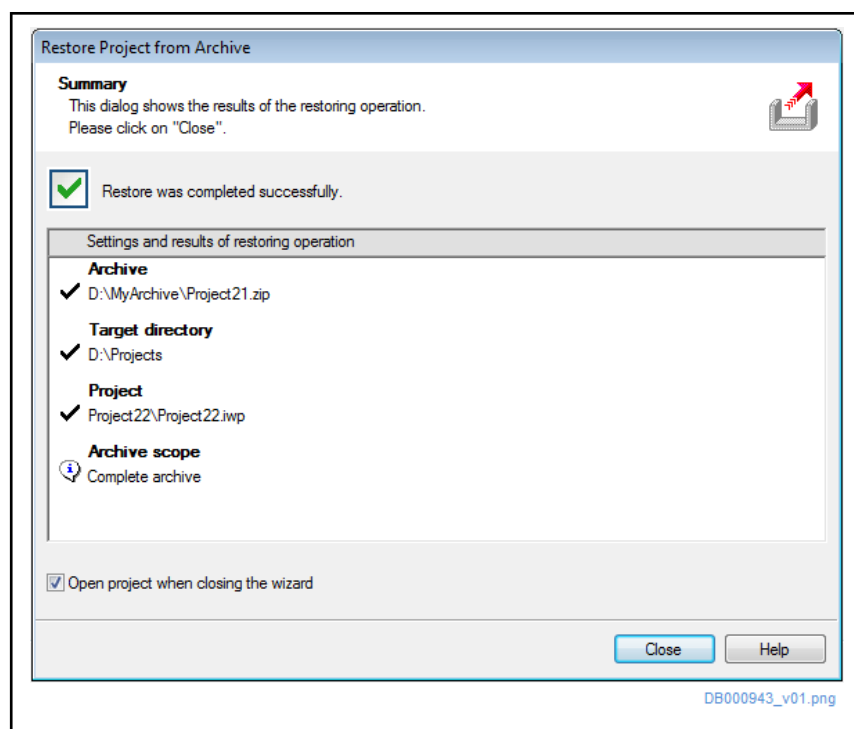


Fig. 7-36: Summary of the restored MLD project



Any changes to enabled functional packages only take effect on the next restart of the drive.

13. If the firmware version in the restored MLD project is the same as the firmware version of the connected drive, you can go online with the drive.

7.4.9 Importing an exported MLD project

The following steps describe what to do to import an exported MLD project.

Make sure that the same or a higher version of IndraWorks MLD as used when the export was created is installed on the PC with which the exported MLD project is imported.

When the exported MLD project is to be used in a CCD group, the CCD group has to be commissioned before the MLD project is restored:

1. Set the following parameters according to configuration of the installation:
 - CCD slave: Drive address (P-0-4089.0.3) and, if necessary, selection of functional packages (P-0-2003).
 - CCD master: If necessary, selection of functional packages (P-0-2003) and setting of TCP/IP communication (P-0-1531 ... P-0-1533).



Changes to enabled functional packages and the TCP/IP communication settings only take effect on the next restart of the drive.

2. In IndraWorks MLD, set desired type of communication (serial or TCP/IP) and go online with the drive (CCD master) (select "Switch Online" in the context menu of the drive branch).

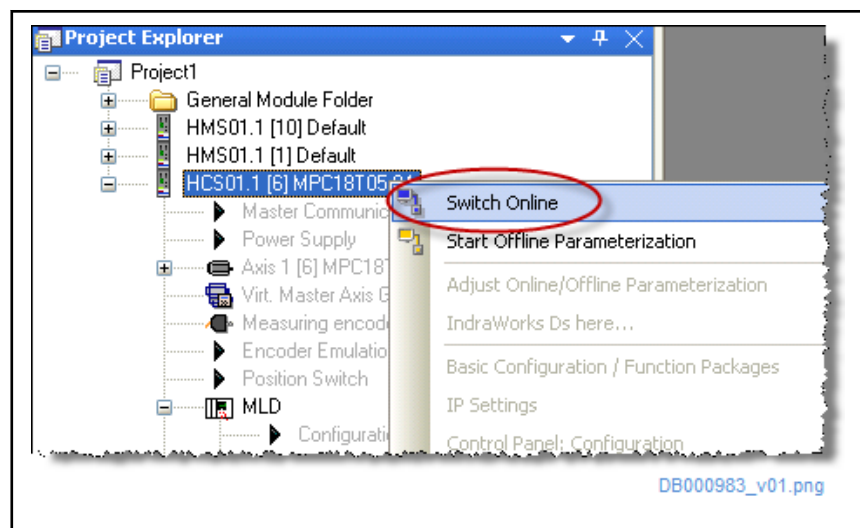


Fig. 7-37: "Switch Online" for the drive

3. In the function tree, call the context menu of **Sercos**, select **CCD: Basic Settings** and configure the CCD group.

The exported IndraWorks MLD project can be imported as follows:

1. Select **Project ► Import....**
⇒ The standard "Open file" Windows dialog opens.
2. Select the MLD project and click **Open**.
⇒ The import wizard opens.
3. Select in the project which parts of the project are to be imported.

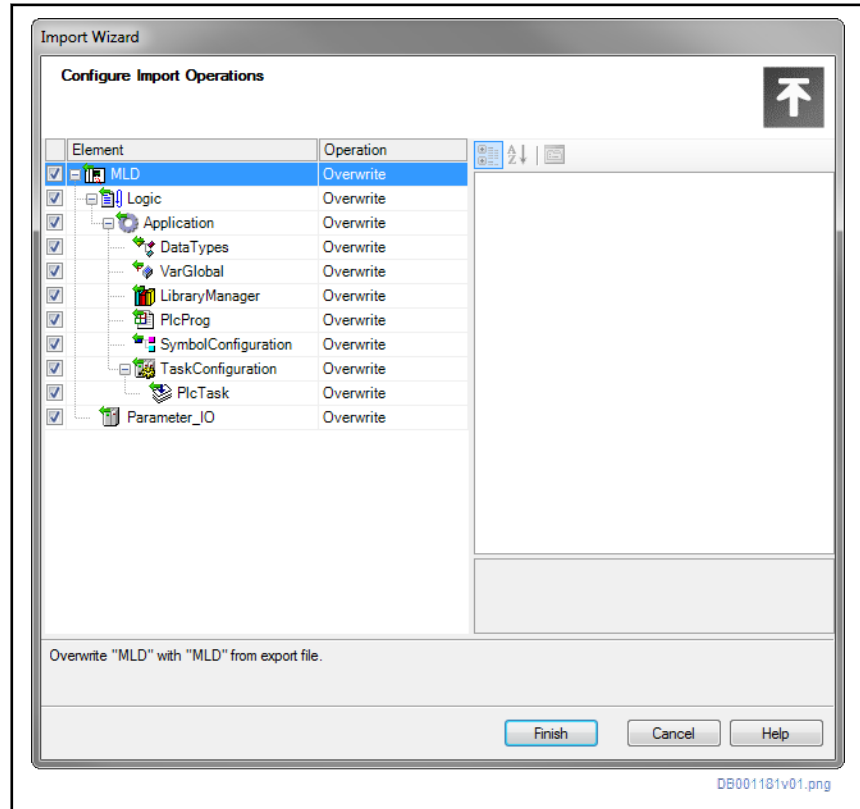


Fig. 7-38: Select the parts of the project that are to be imported

4. Start the import process by clicking **Finish**.

⇒ After the import process, a summary is displayed.

7.4.10 Updating the IndraLogic device version

Due to bug fixes, additional libraries, etc., it may be necessary to subsequently install an IndraLogic device version (IndraWorks packages). IndraWorks packages normally are provided by Bosch Rexroth support or supplied with a new IndraWorks MLD version.

The dialog for subsequently installing IndraWorks packages can be opened in the Project Explorer via the context menu **MLD ► Change IndraLogic device version**.

In the "Change IndraLogic device version" dialog, IndraWorks packages can be subsequently installed by clicking the "Install IW package" link.



In case IndraWorks is updated, all existing libraries remain available on the computer. It is not necessary to install an "older" IndraWorks package again.

7.4.11 Changing the libraries valid for a PLC project

Updating libraries of an existing PLC project

When a PLC project is created, the libraries current at this time are automatically copied to the project directory.

If the libraries in a PLC project are to be updated, it is necessary to install an IndraWorks package (IW package). The IndraWorks package contains information on the device version and on all the required libraries.

Installing the IW package

1. In the Project Explorer, open the context menu at the **MLD** branch and select **Change IndraLogic device version**.
The "Change IndraLogic device version" dialog opens.
2. Click the "Install IW package" link:

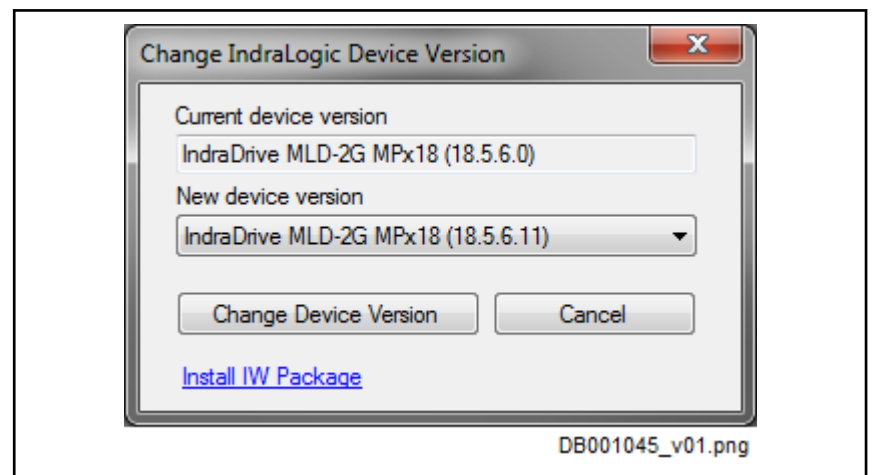


Fig. 7-39: Installing the IW package

The "Open file" Windows dialog opens.

3. Select the package to be installed. (Only files with the ".iworkpackage" extension can be selected!)



The installation can take some time.

After the installation a new device version is available in the drop down list for selection.

4. Select the new device version via the drop down list.
5. Click the **Change device version** button.
6. After the device version has been changed, the project should be cleaned:
From the main menu, go to **Build ► Clean all**.
7. The cleaned project has to be recompiled:
From the main menu, go to **Build ► Rebuild all**.



The cleaned and recompiled project will be loaded to the control unit with the next "Login".

Adapting PLC project to a different IndraLogic device version

If a PLC project is to be adapted to a different firmware release or a different firmware is to be selected, the device version has to be changed. The procedure is described under "[Compatibility between versions](#)", in the "Changing the device version" instructions.

7.4.12 Importing an external library file

To import an external library file, proceed as follows:

1. Start IndraWorks MLD.
2. In the Project Explorer, open the context menu (by right-clicking) of the Logic branch.
3. Select **Import libraries....**
The "Import an IndraLogic library file" dialog opens.
4. Select the library to be imported and confirm with "Open".



To import multiple libraries in the same directory, it is recommended to navigate with, e.g., Explorer. In Explorer, copy the path from "Address", enter it in "File name" in the import dialog and confirm with Enter.

7.5 Drive as PLC device

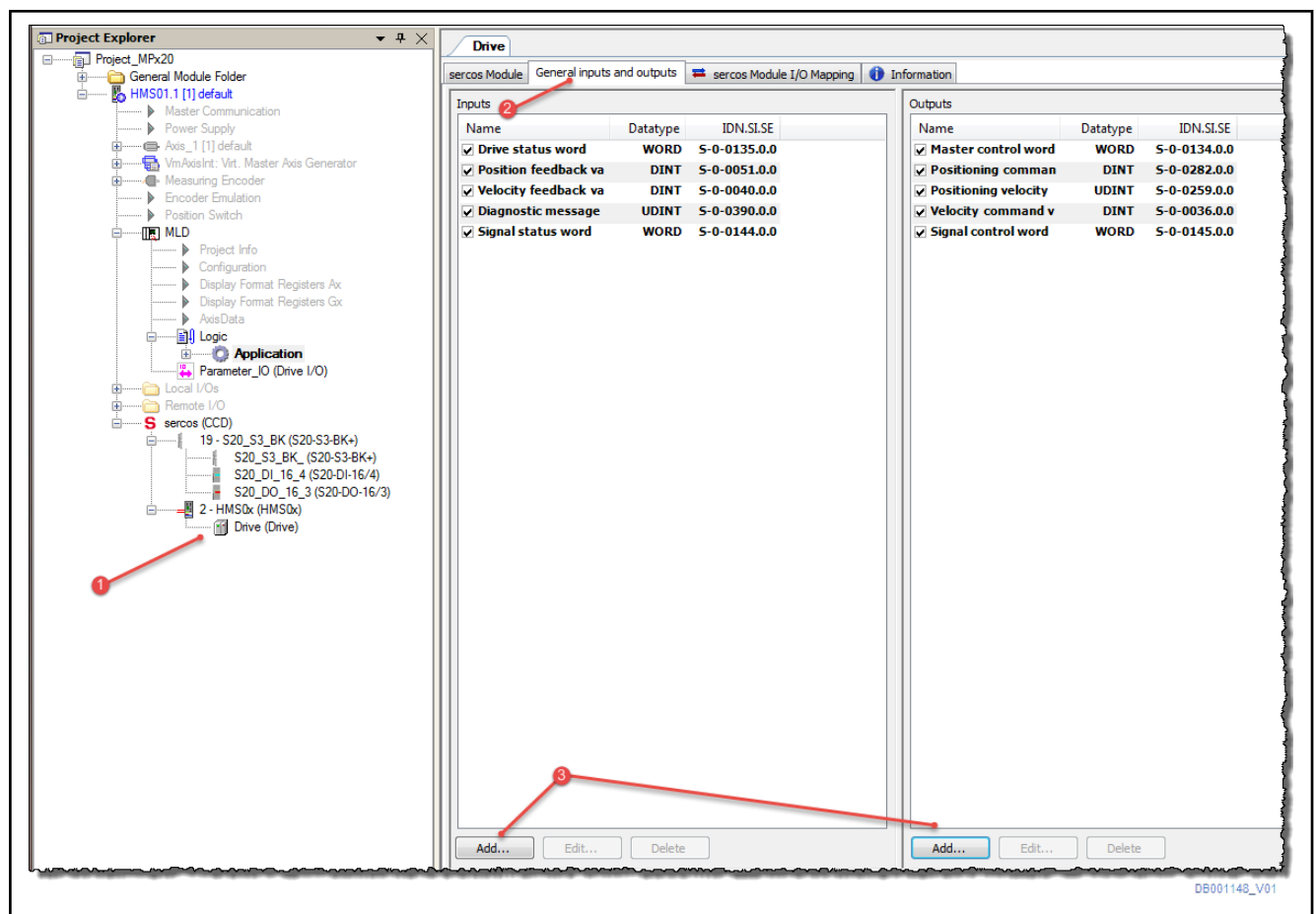
7.5.1 General information

In contrast to an axis, a drive as PLC device has no motion functionality, that means that the functional blocks of the "PLCopen.library" library cannot be used. The drive is PLC-commanded and freely programmable.

In the IndraWorks project, the drive can be created by dragging the device from the library window ("Peripherals / Sercos" folder) to the Sercos branch. IndraDrive devices are located in the library of the "Sercos" peripherals in MPx18VRS and above. Example: IndraDrive Cs single-axis device as HCS0x.

Drives of any manufacturers can be subsequently installed in the device database.

To configure the cyclic communication and diagnostics of the PLC devices, IndraWorks dialogs are available.



- 1 Double-click the Sercos device "HCS0x" below the Sercos branch
- 2 Select the "General inputs and outputs" tab
- 3 Select or delete parameters using "Add"

Fig. 7-40: Configuring the cyclic communication

For IndraDrive, configure the communication and parameterize the drives via IndraWorks Ds.

7.5.2 Sercos III drive as PLC device

Overview

By way of example, the implementation of the following motion sequence is given below:

- Positioning to an absolute position
- Relative positioning (target position = current position + position specification)
- Additive positioning (target position = last target position + position specification)
- Velocity control
- Drive Halt

Any number of motion sequences and drive-related functions can be implemented. In this example, only a small part of the IndraDrive functionality is used.

The drive commissioning steps can be followed using this section. At the end, it should be possible to move the drive with the described motion sequence using the program listing (see [chapter "PLC programming" on page 227](#)). The required steps are described at the end of the respective section.



The usage as real axis (slave axis in the MLD-M system) has many advantages compared to the usage as PLC device with free programming described here. For usage as PLC device, advanced knowledge of the Sercos parameters used and additional programming effort are required.

Creating the PLC drive

An IndraWorks project with an MLD master was created. The "MLD" functional package has been activated. The related device description file of the drive that is to be embedded as PLC device is installed.

1. Drag the drive from the library window (folder "Peripherals", folder "Sercos") to the Sercos branch and configure the Sercos address.
2. Login to the PLC program (compile PLC program and load it to the drive).
3. The PLC drive is automatically recognized, the CCD group is switched into Sercos phase "P2".
4. The master drive and the configured slave drive can now be switched into P4 (operation mode).

Parameterization

For IndraDrive, parameterize the drives via IndraWorks. For other drives, use the manufacturer-specific tool.

As an alternative to "IndraWorks Ds here...", communication can take place directly between PC and drive. The PC can be connected to a free Sercos port or communicate via a "Sercos III NetSwitch" (ordering no.: R911328254 HAWA HGS NS-S3-1NRT). The Windows command "route" can also route the IP communication via the control.



Instead of using IndraWorks Ds, an IndraDrive can also be dragged from the library (Drive and Control) to the main branch (parallel to the control) in the IndraWorks project of the control. A previously created parameter file using IndraWorks Ds is useful for the configuration.

With Sercos, the parameters to be cyclically transmitted are configured, in contrast to other field buses, below the Sercos branch in IndraWorks.

To implement the motion sequences given in [chapter "Overview" on page 221](#), at least the following parameterization has to be made using IndraWorks Ds:

- Primary operation mode "velocity control"
- Secondary operation mode 1 "torque control"

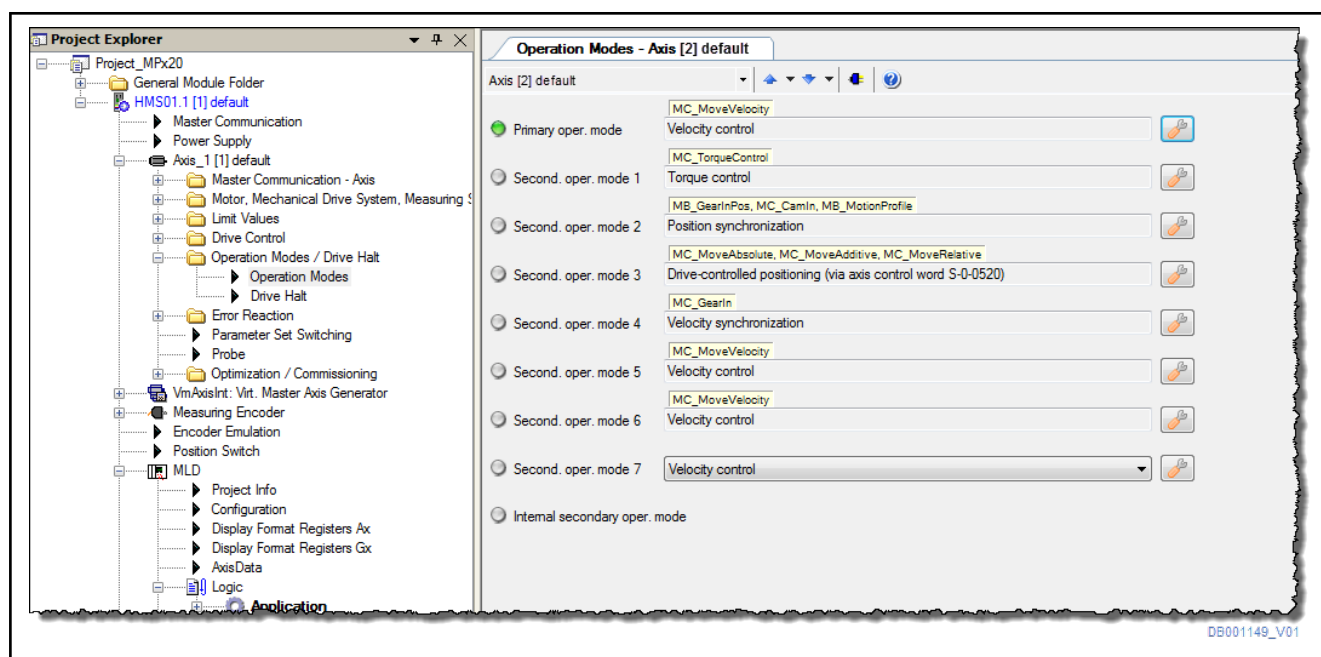


Fig. 7-41: Parameterizing operation modes using IndraWorks Ds

- Signal control word
 - Bit 0: S-0-0346, bit 0 "Applying positioning command value via toggling"
 - Bit 1: S-0-0346, bit 3 "Type of the positioning command value, 0: absolute, 1: relative (depending on bit 4)"
 - Bit 2: S-0-0346, bit 4 "Dedicated point for positioning command values, 0: last effective target position (S-0-0430), 1: active actual position value (S-0-0386)"
- Signal status word:
 - Bit 0: S-0-0331, bit 0 "Status "n_feedback = 0""
 - Bit 1: S-0-0342, bit 0 "Status "Target position attained""
 - Bit 2: S-0-0330, bit 0 "Status "n_feedback = n_command""
 - Bit 3: S-0-0403, bit 0 "Status of the reference encoder actual position value (encoder 1 or 2), 0: relative, 1: homed"

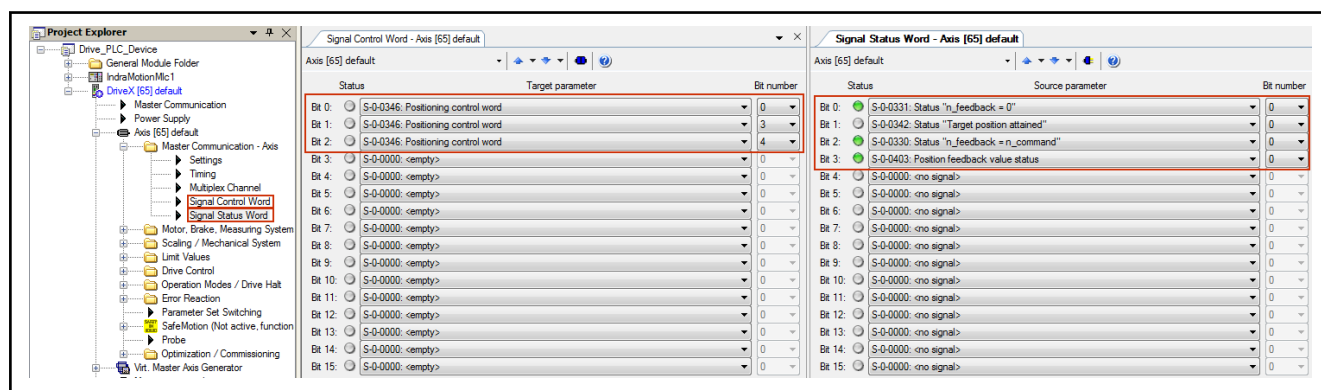


Fig. 7-42: Parameterizing signal control and signal status words using IndraWorks

Also set other parameters of the drive, e.g., motion limit values.



To read or write parameters of a PLC device from the PLC program, use function blocks from the RIL_SERCOSIII.library, folder "AcyclicCommunication".

Parameterizing the drive with direct connection

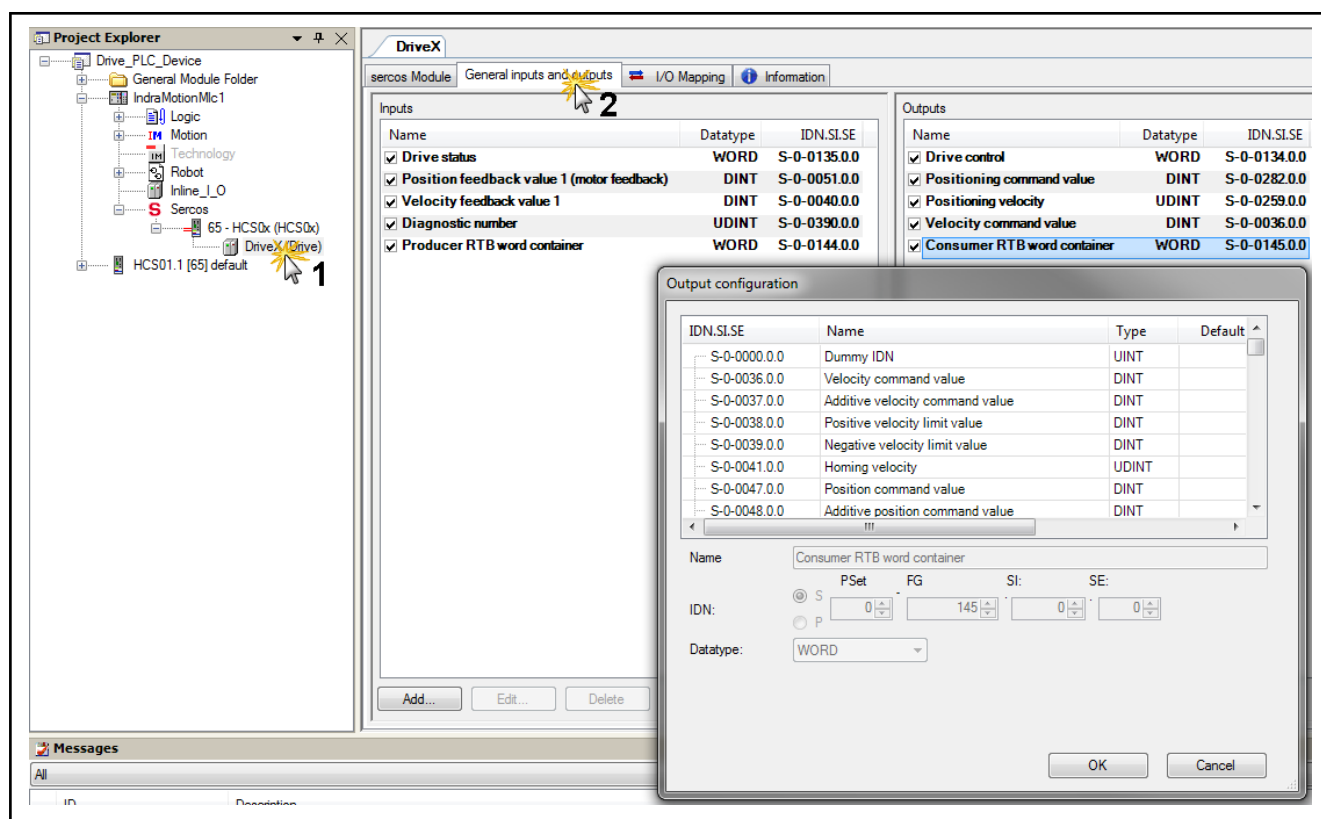
When using "IndraWorks Ds here..." (right-click the Sercos device), the steps 1 and 2 are omitted.

Only for direct connection: The PC is connected to an open Sercos port or via a "Sercos III NetSwitch". The IP address of the PC is configured with an address matching the control and drive IP address.

1. Enter the bridge address noted down in "[Creating the PLC drive](#)", point 4, into the IndraWorks project as IP address of the control. (Switching online should be possible again.)
2. Start IndraWorks Ds. "Select connection"→"IP address search". Enter and search IP address of the drive (noted down in "[Creating the PLC drive](#)", point 5). Connect to drive.
3. Parameterize the operation modes in IndraWorks Ds as described in "[Parameterizing operation modes using IndraWorks Ds](#)".
4. Parameterize the signal control word and signal status word in IndraWorks Ds as described in "[Parameterizing signal control and signal status words using IndraWorks](#)".
5. Further parameterization of the drive is to be executed in IndraWorks Ds; motion limit values, Drive Halt acceleration bipolar, controller parameters, status messages, etc.

Configuring the cyclic data

Data to be cyclically transmitted can be configured in IndraWorks below the Sercos branch in the "General inputs and outputs" dialog.



1 Double-click the Drive branch below the Sercos device

2 Select General inputs and outputs

Fig. 7-43: Exemplary configuration of cyclic parameters in the drive telegram (AT) (inputs) and master data telegram (MDT) (outputs)

In this example, the parameters required for the operation modes "positioning" and "velocity control" are configured.



If the device description file contains the required information (e.g. which parameters can be configured cyclically, name and attribute of the parameters), the configurator provides a drop-down list and suggests the required settings. If this information is not available, no drop-down list is provided and the user has to make the settings.

The entered name is subsequently displayed in the I/O mapping in the "Channel" column (see figure "Assigning PLC variables to the configured parameters".)

Depending on the parameter attribute, the correct data type (with suitable length; with or without sign) has to be selected in the PLC:

Data length in attribute	Format in attribute	Data type in PLC
2 bytes	BIN or HEX	WORD
4 bytes	BIN or HEX	DWORD
2 bytes	DEC_MV	INT
4 bytes	DEC_MV	DINT

Notes on commissioning and application

Data length in attribute	Format in attribute	Data type in PLC
2 bytes	DEC_OV	UINT
4 bytes	DEC_OV	UDINT

Tab. 7-3: Assignment parameter attribute ↔ data type in the PLC

The parameters can be assigned to the PLC variables in the "I/O mapping" tab. New variables can be generated or input and output parameters can be mapped to existing PLC variables.

New variables are generated in the following example:

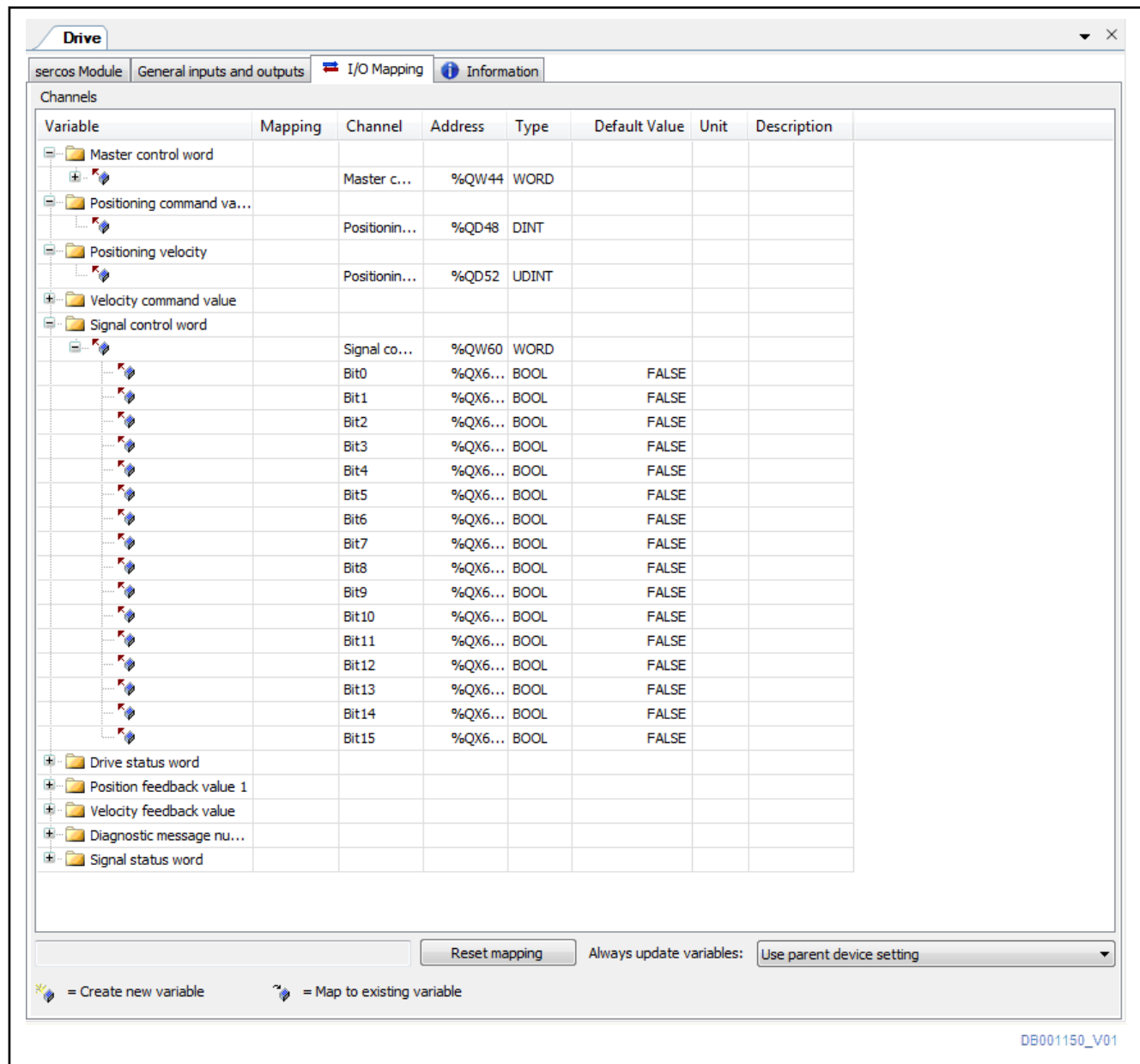


Fig. 7-44: Assigning PLC variables to the configured parameters

Configuring the cyclic data

1. Double-click the "Drive" branch of the drive in the IndraWorks project. Select the "General inputs and outputs" tab.
2. The "S-0-0135, Drive status word" parameter is to be configured.

Check the attribute in the help regarding S-0-0135:

- Length: 2 bytes
 - Format: BIN
 - For the inputs, select "Add"
3. The "S-0-0051, Actual position value encoder 1" parameter is to be configured.

Check the attribute in the help regarding S-0-0051:

- Length: 4 bytes
 - Format: DEC_MV
 - For the inputs, select "Add"
4. Enter the remaining parameters until all parameters have been configured as shown in figure "[Exemplary configuration of cyclic parameters in the drive telegram \(AT\) \(inputs\) and master data telegram \(MDT\) \(outputs\)](#)".
5. Go to the "I/O mapping" tab and enter the PLC variables in the "Variable" column as shown in figure "[Assigning PLC variables to the configured parameters](#)".
6. Load the PLC program; it should now be possible to switch to "P4" via IndraWorks.

PLC programming

The configured PLC variables of the cyclically configured parameters can now be freely connected in the PLC.

By default, the programming does not have to be performed in a specific task. Only in case of specific requirements, as for Gantry axes, the specifications have to be made in the motion-synchronous or Sercos-synchronous task.

Program example

This sequence was implemented in the following example:

- Start in MODE_OFF = AB. Set the "OpMode" variable to 20 (ABS_POS_MODE) to start the sequence
- ABS_POS_MODE = AF, absolute positioning with 4 rpm to 42 degrees
- REL_POS_MODE = AF, relative positioning with 2 rpm by 142 degrees in clockwise direction
- ADD_POS_MODE = AF, relative positioning with 3 rpm by 42 degrees in anti-clockwise direction
- VEL_MODE = AF, velocity control with 42 rpm for 5 seconds
- MODE_AH = AH, stopping with parameterized "Drive Halt acceleration bipolar" S-0-0372
- End in MODE_OFF = AB again. Switching off drive enable when standstill is reached

By writing one of the values REL_POS_MODE, ADD_POS_MODE or VEL_MODE to the "OpMode" variable, the sequence can also be started at these positions.

PLC programming

1. Open the "PlcProg" in the IndraWorks project.
2. Copy the "[Declaration](#)" program listing into the declaration part.
3. Copy the "[Implementation](#)" program listing into the implementation part. (Do not copy page break or delete it subsequently.)

Notes on commissioning and application

4. Compile, load and start PLC program.
5. Switch power on at the drive "bb"→"Ab". Check reference (boDriveX_Homed in the PLC project).
6. Set breakpoints at the transitions to the next operation mode (comment "// set breakpoint here to check") to check the correct processing.
7. Set the "OpMode" variable to 20. This starts the sequence and the first motion to the breakpoint can be executed.
8. Restart the PLC program up to the next breakpoint and check the motion sequence. Repeat until all steps are processed.
9. Set the "OpMode" variable to 30, 40 or 50. This starts the sequence of the respective operation mode.
10. For the final program implementation, sections can be copied and used from this program.

Declaration

```

PROGRAM PlcProg
VAR
  OpMode:INT:=MODE_DRV_OFF; // selection operation mode and state machine
  fbTON: TON;
  rActPosition: REAL; // actual position as real
  rActVelocity: REAL; // actual velocity as real
  boError: BOOL; // error flag from status word
  boWarning: BOOL; // warning flag from status word
END_VAR

VAR CONSTANT
  // operation modes and state machine
  MODE_DRV_OFF :INT:=10;
  ABS_POS_MODE :INT:=20; ABS_POS_MODE_1 :INT:=21; ABS_POS_MODE_2 :INT:=22;
  REL_POS_MODE :INT:=30; REL_POS_MODE_1 :INT:=31; REL_POS_MODE_2 :INT:=32;
  ADD_POS_MODE :INT:=40; ADD_POS_MODE_1 :INT:=41; ADD_POS_MODE_2 :INT:=42;
  VEL_MODE :INT:=50; VEL_MODE_1 :INT:=51;
  MODE_DRV_AH :INT:=60;
  // control word S-0-0134
  CTRL_WORD_OFF:WORD :=16#0000;
  CTRL_WORD_AH:WORD :=16#C000;
  CTRL_AF_PRIMARY_OP_MODE:WORD :=16#E000;
  CTRL_AF_SECONDARY_OP_MODE:WORD:=16#E100;
  // status word S-0-0135
  MASK_STATUS_AF_OP_MODE:WORD :=16#C700;
  STATUS_AF_PRIMARY_OP_MODE:WORD :=16#C000;
  STATUS_AF_SECONDARY_OP_MODE:WORD:=16#C100;
  // Weighting
  WEIGHTING_FACTOR_POS: DINT:=10000; // S-0-0078 = -4
  WEIGHTING_FACTOR_VEL: DINT:=10000; // S-0-0046 = -4
END_VAR

```

Implementation

```

CASE OpMode OF
  MODE_DRV_OFF:
    // Ab, switched OFF
    wDriveX_ControlWord := CTRL_WORD_OFF; // all OFF...

  ABS_POS_MODE:
    // start absolute positioning to 42°
    boDriveX_PosRelNotAbs:=FALSE; // S-0-0346.3 0=absolute Pos, 1=relative Pos
    //boDriveX_PosRelNotAdd; // S-0-0346.4 0=additive (last target), 1=rel. (actual pos.)
    udiDriveX_PositionVelocity:=DINT TO UDINT(4*WEIGHTING_FACTOR_VEL); // 4 rpm * 10000
    diDriveX_PositionCmdValue:=42*WEIGHTING_FACTOR_POS; // 42 degree * 10000
    wDriveX_ControlWord := CTRL_AF_PRIMARY_OP_MODE; // AF and primary operation mode
    boDriveX_PosToggle := NOT boDriveX_PosToggle; // start positioning
    OpMode:=ABS_POS_MODE_1;

  ABS_POS_MODE_1:
    // check acknowledge power AND homed
    IF (wDriveX_StatusWord AND MASK_STATUS_AF_OP_MODE) = STATUS_AF_PRIMARY_OP_MODE
    AND boDriveX_Homed = TRUE
    THEN
      IF boDriveX_InPosition = FALSE THEN // in position can be set from last command
        OpMode:=ABS_POS_MODE_2; // next
      END IF
    END IF

```

```

        END_IF
    END_IF

ABS_POS_MODE_2:
    // check in position
    IF boDriveX_InPosition THEN
        // here we should be at 42°
        OpMode := REL_POS_MODE; // set breakpoint here to check
    END_IF

REL_POS_MODE:
    // start relative positioning 142° clockwise
    boDriveX_PosRelNotAbs:=TRUE; // S-0-0346.3 0=absolute Pos, 1=relative Pos
    boDriveX_PosRelNotAdd:=TRUE; // S-0-0346.4 0=additive (last target), 1=rel. (actual pos.)
    udiDriveX_PositionVelocity:=DINT TO UDINT(2*WEIGHTING_FACTOR_POS); // 2 rpm
    diDriveX_PositionCmdValue:=142*WEIGHTING_FACTOR_VEL; // 142 degree * 10000
    wDriveX_ControlWord := CTRL_AF_PRIMARY_OP_MODE; // AF and primary operation mode
    boDriveX_PosToggle := NOT boDriveX_PosToggle; // start positioning
    OpMode:=REL_POS_MODE_1;

REL_POS_MODE_1:
    // check acknowledge power
    IF (wDriveX_StatusWord AND MASK_STATUS_AF_OP_MODE) = STATUS_AF_PRIMARY_OP_MODE THEN
        IF boDriveX_InPosition = FALSE THEN // in position can be set from last command
            OpMode:=REL_POS_MODE_2; // next
        END_IF
    END_IF

REL_POS_MODE_2:
    // check in position
    IF boDriveX_InPosition THEN
        // here we should be at 42°+142°=184°
        OpMode := ADD_POS_MODE; // set breakpoint here to check
    END_IF

ADD_POS_MODE:
    // start additive positioning 42° counterclockwise
    boDriveX_PosRelNotAbs:=TRUE; // S-0-0346.3 0=absolute Pos, 1=relative Pos
    boDriveX_PosRelNotAdd:=FALSE; // S-0-0346.4 0=additive (last target), 1=rel. (actual pos.)
    udiDriveX_PositionVelocity:=DINT TO UDINT(3*WEIGHTING_FACTOR_VEL); // 3 rpm
    diDriveX_PositionCmdValue:=-42*WEIGHTING_FACTOR_POS; // -42 degree * 10000
    wDriveX_ControlWord := CTRL_AF_PRIMARY_OP_MODE; // AF and primary operation mode
    boDriveX_PosToggle := NOT boDriveX_PosToggle; // start positioning
    OpMode:=ADD_POS_MODE_1;

ADD_POS_MODE_1:
    // check acknowledge power
    IF (wDriveX_StatusWord AND MASK_STATUS_AF_OP_MODE) = STATUS_AF_PRIMARY_OP_MODE THEN
        IF boDriveX_InPosition = FALSE THEN // in position can be set from last command
            OpMode:=ADD_POS_MODE_2; // next
        END_IF
    END_IF

ADD_POS_MODE_2:
    // check in position
    IF boDriveX_InPosition THEN
        // here we should be at 42°+142°-42°=142°
        OpMode := VEL_MODE; // set breakpoint here to check
    END_IF

VEL_MODE:
    // start velocity control with 42 rpm
    diDriveX_VelocityCmdValue:=42*WEIGHTING_FACTOR_VEL; // 42 rpm * 10000
    wDriveX_ControlWord := CTRL_AF_SECONDARY_OP_MODE; // AF and secondary operation mode 1
    OpMode := VEL_MODE_1;

VEL_MODE_1:
    // check acknowledge power
    IF (wDriveX_StatusWord AND MASK_STATUS_AF_OP_MODE) = STATUS_AF_SECONDARY_OP_MODE THEN
        fbTON(IN := TRUE, PT:= T#5S);
        // 5 seconds in velocity control
        IF boDriveX_InVelocity AND fbTON.Q=TRUE THEN
            // here we should rotate with 42rpm
            OpMode := MODE_DRV_AH; // set breakpoint here to check
            fbTON(IN := FALSE);
        END_IF
    END_IF

MODE_DRV_AH:
    // stop the drive until StandStill is reached
    wDriveX_ControlWord := CTRL_WORD_AH; // AH

```

Notes on commissioning and application

```

IF boDriveX_StandStill THEN
    OpMode := MODE_DRV_OFF; // set breakpoint here to check
END_IF

END_CASE

//actual values
rActPosition := DINT_TO_REAL(diDriveX_ActualPosition)/DINT_TO_REAL(WEIGHTING_FACTOR_POS);
rActVelocity := DINT_TO_REAL(diDriveX_ActualVelocity)/DINT_TO_REAL(WEIGHTING_FACTOR_VEL);

boError:= wDriveX_StatusWord.13;
boWarning:=wDriveX_StatusWord.12;
// dwDriveX_DiagnosisNumber; // diagnosis number
// Clear error: write S-0-0099 to b0011 with IL_SIIISvcWrite()

```

7.6 Supply unit as PLC device

7.6.1 General information

Using supply units of type HMV05¹⁾ (Product range IndraDrive ML) and KMV03 (product range IndraDrive: Mi) are provided with supply units with separate master communication. These supply units can be operated with firmware PSB-2x.

Supply units of this type can be referred to as "intelligent" supply units. They have a bus address, different parameters to parameterize functions, a device control word and a device status word.



Refer to the relevant project planning manual for information on "intelligent" supply units.

- KMV03: Project Planning Manual with material number R911335703
- HMV05: Project Planning Manual with material number R911344279

Supply units with separate master communication are supported by IndraLogic XLC/IndraMotion MLC/IndraMotion MLD as PLC devices.

7.6.2 Parameterization

The supply unit can be created in the IndraWorks project by dragging and dropping it from the library window (folder "Periphery", folder for the relevant field bus) to the field bus node. For Sercos, find the relevant supply unit in the library of the periphery devices "Sercos".

The supply unit can be parameterized in IndraWorks Ds. (A parameterization via single parameters is possible via IndraLogic XLC/IndraMotion MLC.)



IndraLogic XLC / IndraMotion MLC: Right-click on the Sercos device and on "IndraWorks Ds here..." to directly call IndraWorks Ds.

As an option to "IndraWorks Ds here...", the communication between PC and supply unit can take place directly. The PC can be connected to a free port of the Sercos device or can communicate via a "Sercos III NetSwitch" (order no.: R911328254, HAWA HGS NS-S3-1NRT). Optionally, the IP communication can be routed via the control using the Windows command "route".

¹⁾ HMV05 is the designation of a supply unit consisting of HNA05 mains connection module, HMU05 universal inverter and CSB02.5 control section with firmware for supply units (FWA-INDRV*-PSB-...). HMV05 is not an official product designation and is used for documentation purposes only.

The parameters to be transmitted cyclically can be parameterized in IndraWorks below the Sercos node.

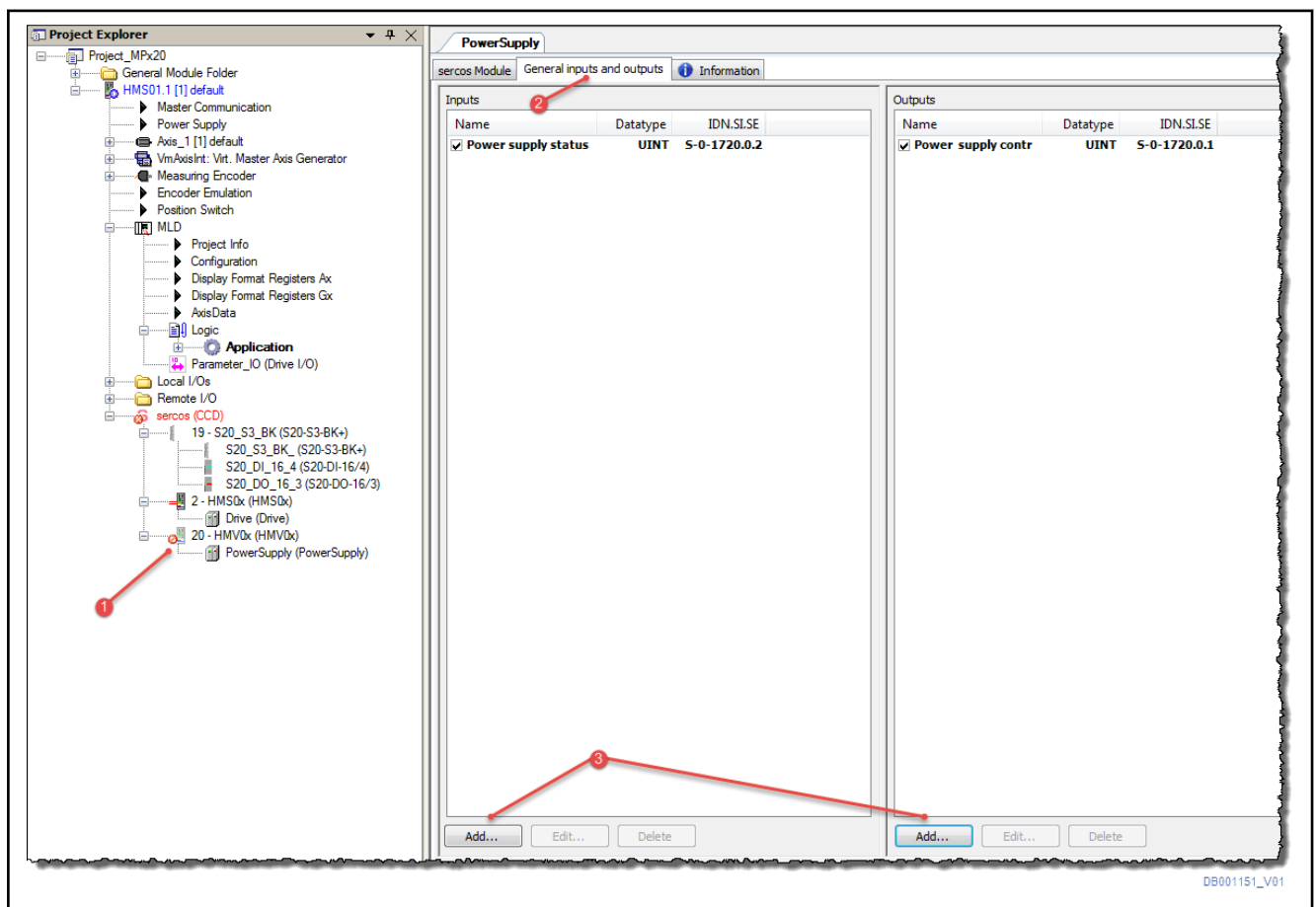
Establishing the connection and parameterizing the supply unit

When using "IndraWorks Ds here..." in IndraLogic XLC/IndraMotion MLC (right-click on the Sercos device), step 1 and 2 do not apply.

1. Connect the supply unit to a free Ethernet network connection or via a USB port using an Ethernet adapter. Set IP address of the Ethernet network connection or of the Ethernet adapter to the supply device area 172.31.x.x.
2. Start IndraWorks Ds. **Connection selection ► IP address search**. Connect to device.
3. If required, parameterize the supply unit again in IndraWorks Ds.

7.6.3 Configuring the cyclic data

Data to be cyclically transmitted can be configured in IndraWorks below the Sercos node in the "General inputs and outputs" dialog.



- 1 Double-click on the PowerSupply node below the Sercos device
- 2 Select the "General inputs and outputs" tab
- 3 Select or delete parameters using "Add"

Fig. 7-45: Example configuration of the cyclic parameters in the drive telegram AT (inputs) and master data telegram MDT (outputs)

The example configuration is a minimum configuration. "S-0-1720.0.1, power supply control word" as PLC output is required to command the supply unit

and "S-0-1720.0.2, power supply status word" as PLC input to read out the status.



If the device description file contains the required information (e.g. which parameters can be configured cyclically, parameter name and attribute), the configurator provides a drop-down menu and proposes the required settings. If information is not available and not drop-down list is available, select the settings manually.

Configuring the cyclic data

1. Double-click on the supply unit node in the IndraWorks project. Select the "General inputs and outputs" tab.
2. "S-0-1720.0.2, power supply status word" has to be configured.

In the S-0-1720.0.2 help, check the attribute:

- Length: 2 bytes
- Format: BIN
- Select **"Add"** at the inputs

3. Parameter "S-0-1720.0.1, power supply control word" has to be configured.

In the S-0-1720.0.1 help, check the attribute:

- Length: 2 bytes
- Format: BIN
- Select **"Add"** at the outputs

4. Assignment Variable ↔ Parameter

The PLC variable parameter can be assigned in the "I/O image" tab. New variables can be generated or input or output parameters can be mapped to existing PLC variables.

Change to "I/O image" tab. Enter the PLC variables in the "Variable" column as explained in [Example configuration of the cyclic parameters in the drive telegram AT \(inputs\) and master data telegram MDT \(outputs\)](#).

The entered name is displayed in the "Channel" column in [PLC variable assignment to configured parameters](#).

Depending on the parameter attribute, the correct data type (with matching length; with or without sign) has to be selected in the PLC.

Data length in attribute	Format in attribute	Data type in the PLC
2 bytes	BIN or HEX	WORD
4 bytes	BIN or HEX	DWORD
2 bytes	DEC_MV	INT
4 bytes	DEC_MV	DINT
2 bytes	DEC_OV	UINT
4 bytes	DEC_OV	UDINT

Tab. 7-4: Assignment parameter attribute ↔ data type in PLC

The variables are regenerated in this example:

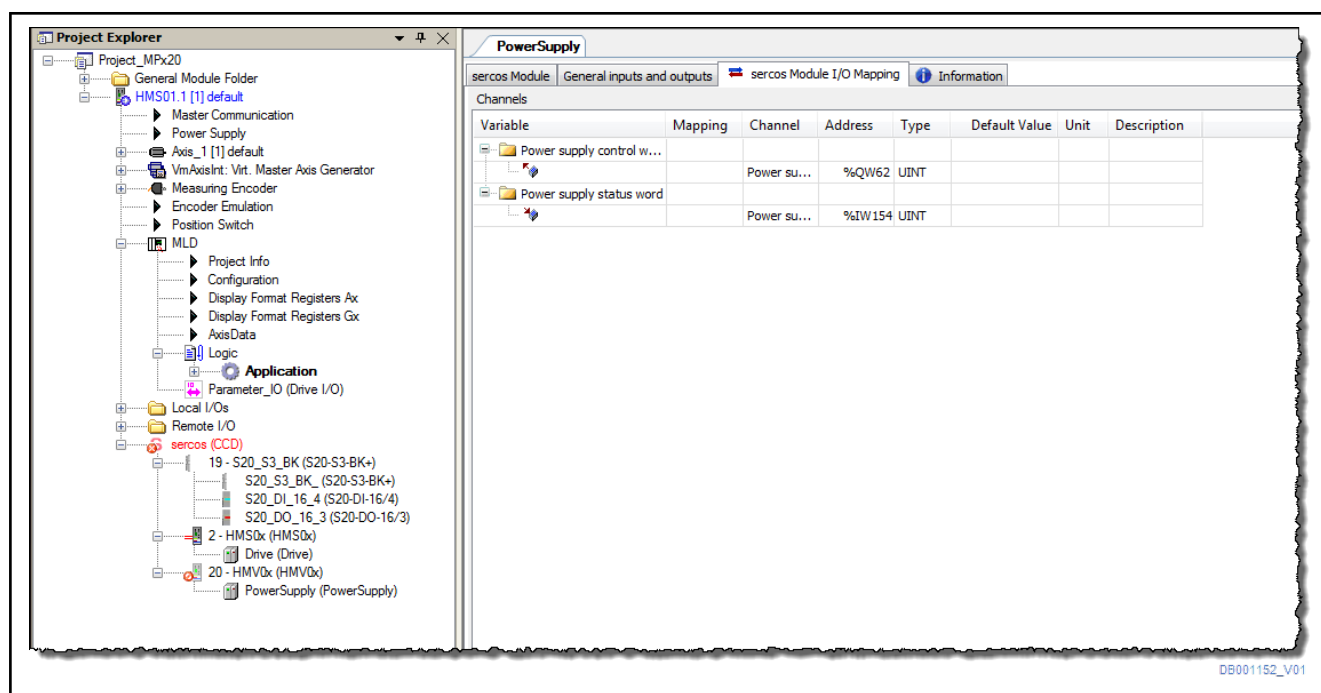


Fig. 7-46: PLC variable assignment to configured parameters

7.6.4 PLC programming

The configured PLC variables of the cyclically configured parameters for control and status word can be used in the PLC program.



As an option to a PLC program, supply units can be controlled via functions, provided in an IndraWorks library:

- **Library:** RMB_SercosIII_Util
- **Group:** PowerSupply
- **Target Systems:**
IndraMotion MTX 14VRS
IndraMotion MLC 14VRS, Function package PLC
IndraMotion MLD with MPx20

For the bit assignment of the supply units parameter (S-0-1720.0.1, Power supply control word / S-0-1720.0.2, Power supply status word), refer to the parameter description.

The supply units can be commanded in any cyclic task (e.g. "PlcTask" and "PlcProg").

Program example

Power on and power off is executed in the following example:

- **1. State "Switch on main contactor"**
Control word: Set bit 12 (16#1000)
Status word: Query bits 4, 5, 7 (16#00B0), advance state
- **2. State "Switch on main contactor (load DC)"**
Control word: Set bits 12, 14 (16#5000)
Status word: Query bits 4, 5, 6, 7 (16#40F0), advance state
- **3. State "Enable command operating mode"**

Notes on commissioning and application

Control word: Set bits 12, 14, 15; Bit 8, Bit 9 are not set for primary operation mode (16#D000)

Status word: Query bits 4, 5, 6, 7, 14, 15 (16#C0F0)

Connected axes can be commanded

- **4. state "Power off"**

Control word: delete all bits (16#0000)

PLC programming

1. Open the "PlcProg" program in the IndraWorks project.
2. Copy the program listing "Declaration" to the declaration part.
3. Copy the program listing "Implementation" to the implementation part. (Do not copy page break or delete it again after Copy&Paste).
4. Compiling, loading and starting of PLC program.
5. Setting the power requirement "bSetPower".
6. Verify if Power On has been executed (status word: Bits 4, 5, 6, 7, 14, 15).

⇒ From this program, parts can be copied and used for customized solving of a task.

Declaration

```

PROGRAM PlcProg
VAR
    iStatePowerSupply: INT;                (* state for power supply *)
    bSetPower: BOOL;                      (* set power *)
END_VAR

VAR CONSTANT
    STATE_ENABLE_MAINS_CONTACTOR : INT := 1;    (* state enable mains contactor *)
    STATE_ENABLE_POWER_SUPPLY    : INT := 2;    (* state enable power supply, load dc voltage *)
    STATE_ACTIVATE_POWER_SUPPLY  : INT := 3;    (* state activate power supply *)
    STATE_POWER_OFF              : INT := 4;    (* state power off *)
END_VAR

(* S-0-1720.0.1 Power supply control *)
(* Bit 15: power supply activation *)
(* Bit 14: power supply enable *)
(* Bit 13: reserved *)
(* Bit 12: mains contactor *)
(* Bit 11: reserved *)
(* Bit 10: DC discharging *)
(* Bit 09: selection of OpMode *)
(* Bit 08: selection of OpMode *)
(* Bit 07: reserved *)
(* Bit 06: reserved *)
(* Bit 05: reserved *)
(* Bit 04: reserved *)
(* Bit 03: reserved *)
(* Bit 02: reserved *)
(* Bit 01: reserved *)

(* S-0-01720.0.2 Power supply status *)
(* Bit 15: voltage boost ON/OFF *)
(* Bit 14: DC bus okay *)
(* Bit 13: error *)
(* Bit 12: warning *)
(* Bit 11: brake resistor *)
(* Bit 10: DC discharging *)
(* Bit 09: selection of OpMode *)
(* Bit 08: selection of OpMode *)
(* Bit 07: feedback signal of mains contactor *)
(* Bit 06: mains barrage *)
(* Bit 05: hardware enable *)
(* Bit 04: mains status *)
(* Bit 03: reserved *)
(* Bit 02: reserved *)
(* Bit 01: reserved *)

```

Implementation

```

(* set power request and check status *)
IF bSetPower AND wStatusWord = 16#00A0 OR wStatusWord = 16#0020 THEN
    iStatePowerSupply := STATE_ENABLE_MAINS_CONTACTOR;
ELSIF bSetPower AND wStatusWord = 16#00B0 THEN
    iStatePowerSupply := STATE_ENABLE_POWER_SUPPLY;
ELSIF bSetPower AND wStatusWord = 16#40F0 THEN
    iStatePowerSupply := STATE_ACTIVATE_POWER_SUPPLY;
ELSIF bSetPower AND wStatusWord = 16#C0F0 THEN
    iStatePowerSupply := STATE_POWER_OFF;
END_IF

(* state machine for power supply *)
CASE iStatePowerSupply OF

    STATE_ENABLE_MAINS_CONTACTOR: (* enable mains contactor *)

```

```

wControlWord := 16#1000;    (* set bit 12 *)

IF wStatusWord = 16#00B0 THEN    (* check bit 4, 5, 7 *)
    iStatePowerSupply := STATE_ENABLE_POWER_SUPPLY;
ELSE
    iStatePowerSupply := STATE_ENABLE_MAINS_CONTACTOR;
END_IF

STATE_ENABLE_POWER_SUPPLY: (* enable power supply, load DC voltage *)

wControlWord := 16#5000;    (* set bit 12, 14 *)

IF wStatusWord = 16#40F0 THEN    (* check bit 4, 5, 6, 7 *)
    iStatePowerSupply := STATE_ACTIVATE_POWER_SUPPLY;
ELSE
    iStatePowerSupply := STATE_ENABLE_POWER_SUPPLY;
END_IF

STATE_ACTIVATE_POWER_SUPPLY: (* enable power supply, operation mode constant DC voltage *)

wControlWord := 16#D000;    (* set bit 12, 14, 15 *)

IF wStatusWord = 16#C0F0 THEN    (* check bit 4, 5, 6, 7, 14, 15 *)
    iStatePowerSupply := STATE_POWER_OFF;
ELSE
    iStatePowerSupply := STATE_ACTIVATE_POWER_SUPPLY;
END_IF

STATE_POWER_OFF: (* power off *)

IF NOT bSetPower THEN
    wControlWord := 16#0000;    (* reset bits *)
END_IF

IF NOT wStatusWord = 16#0040 THEN    (* check bit 6 *)
    iStatePowerSupply := STATE_POWER_OFF;
END_IF

END_CASE

```

7.7 Comparing the "IndraDrive Cs" range with the "IndraDrive C"/"IndraDrive M" ranges

7.7.1 Introduction

The drive-internal PLC (IndraMotion MLD) of the "IndraDrive Cs" range features some functional and hardware-related changes compared to the "IndraDrive C"/"IndraDrive M" ranges. This affects, among other things, the application of PLC programs. The following sections describe the changes involved, as well as any aspects that have to be observed when PLC programs are ported.

7.7.2 Physical interfaces

- **Inputs and outputs**

"IndraDrive" provides an analog **input**. There is no analog **output**.

"IndraDrive" features digital inputs and configurable digital inputs/outputs.

The IO modules in the "Rexroth Inline" product range allow additional analog and digital inputs/outputs to be connected.

- **Engineering interface**

"IndraDrive Cs" features an Ethernet interface. This interface is used for programming and downloads.

- **Memory card**

The standard control panel does not feature any option of using a memory card.

The Advanced control panel features a slot for a microSD memory card.

7.7.3 Byte order

"Big Endian" / "Little Endian"

The firmware MPx08 and below used the "Little Endian" byte order. The "Big Endian" byte order is used in MPx17 and above.

I/O channel (PII, POI)

Introduction

The I/O channel is the contact of IndraMotion MLD to external devices, since it allows digital and analog inputs/outputs to be evaluated and addressed. To use the I/O channel in the PLC user program, variables have to be assigned to specific memory cells (process image). ["PII" (process input image) starts with parameter P-0-1390 et seq., "POI" (process output image) starts with parameter P-0-1410 et seq.]

Variables are assigned to specific memory cells using the percent character "%" and a character string which defines whether an input or an output is to be addressed and which data type is used.

%QX7.5 and %Q7.5: Output bit 7.5
 %IW215: Input word 215
 %QB7: Output byte 7
 %MD48: Double word at memory location 48 in the flag

Tab. 7-5: Examples

The program detail below shows an example of the declaration of variables with different data types which are assigned to the process image.

Program:

```
PROGRAM PLC_PRG
VAR
Dig_IN_DWORD AT %ID0: DWORD;
Dig_IN_WORD AT %IW2: WORD;
Dig_IN_BYTE AT %IB5: BYTE;

Dig_OUT_DWORD AT %QD2: DWORD;
Dig_OUT_WORD AT %QW0: WORD;
Dig_OUT_BYTE AT %QB7: BYTE;
END_VAR
```

Parameter/addressing relation tables

The following tables show the relation between the parameters for the process input/output image and the corresponding addressing types. The tables contain only three parameters to illustrate the principle. The distribution shown applies to all parameters of the inputs (P-0-1390 to P-0-1409) and outputs (P-0-1410 to P-0-1429).

P-0-1390		P-0-1391		P-0-1392		...
IB0 Bit15...bit8	IB1 Bit7...bit0	IB2 Bit15...bit8	IB3 Bit7...bit0	IB4 Bit15...bit8	IB5 Bit7...bit0	...
IW0		IW1		IW2		...
ID0				...		

Tab. 7-6: Process input image (Big Endian: MPx17 and above)

P-0-1410		P-0-1411		P-0-1412		...
QB0 Bit15...bit8	QB1 Bit7...bit0	QB2 Bit15...bit8	QB3 Bit7...bit0	QB4 Bit15...bit8	QB5 Bit7...bit0	...
QW0		QW1		QW2		...
QD0				...		

Tab. 7-7: Process output image (Big Endian: MPx17 and above)

Network communication

If it is intended to establish network communication with other devices using the MLD (e.g., via TCP/IP), the conversion functions for the byte order used should always be employed so that the user programs can be run with all firmware versions.

Usually, all communication libraries provide functions which execute this conversion. The "SysLibSockets" library includes, e.g., the following conversion functions:

- SysSockHtons (host to network): Control to network, 16 bits
- SysSockNtohs (network to host): Network to control, 16 bits
- SysSockHtonl (host to network): Control to network, 32 bits
- SysSockNtohl (network to host): Network to control, 32 bits

The following data types have to be converted for transmission via a network:

Data type	Control to network	Network to control
WORD, INT, UINT	SysSockHtons()	SysSockNtohs()
DWORD, DINT, UDINT, TIME	SysSockHtonl()	SysSockNtohl()

The BYTE, CHAR and STRING data types do not have to be converted. The byte order for the REAL and LREAL data types is not specified. The correct setting can only be determined by trying.

Accessing files with MLD

Files can only be accessed in conjunction with an ADVANCED control panel. If files are accessed using the "MX_SysLibFile" library, the byte order also has to be observed. In the figure below, variable "ardiBuffer" (32-bit values) is used to write ASCII characters to a text file ("Test.txt"). The characters are stored differently on systems with different byte orders. Especially when user programs are ported between systems with differing byte order, a check has to be run to verify whether the data are still consistent after the porting. If this is not the case, either the order has to be changed manually or the values have to be converted to 8-bit values.

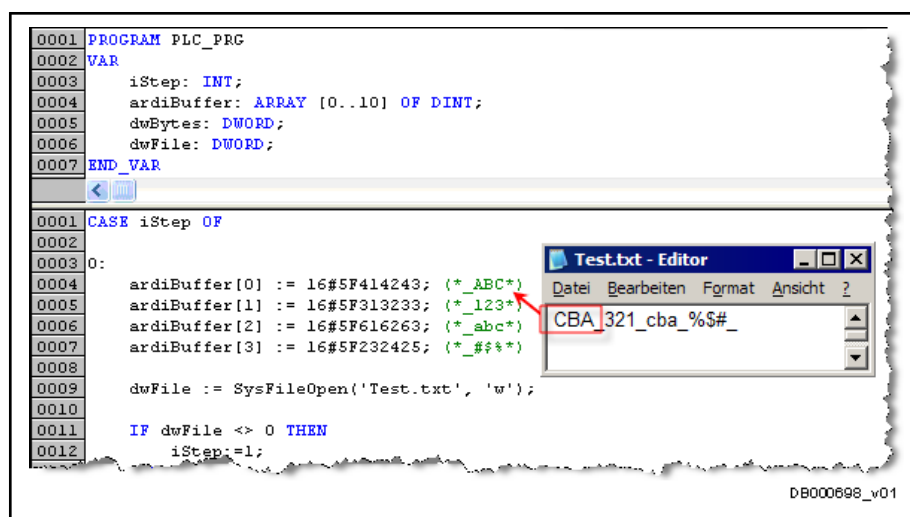


Fig. 7-47: Checking the byte order

There are two options for adapting the PLC program:

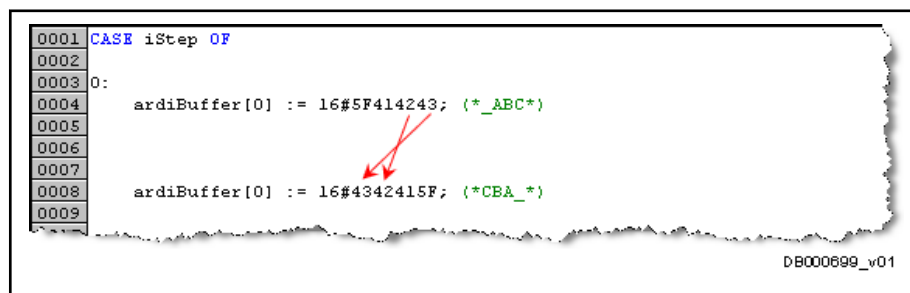


Fig. 7-48: Changing the byte order

- or -

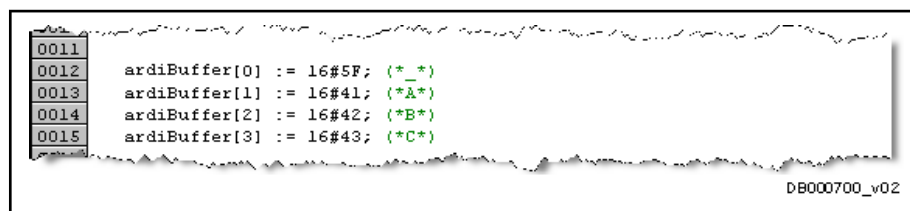


Fig. 7-49: Writing byte by byte

Address access with different data types

Access via flags

The "Merker_LByte" and "Merker_HByte" flags are used to write the bytes no. 200 and 201. The "Merker_WORD" flag with the "%MW100" (100 * 2byte=200) address is to read a 16-bit value. "Merker_WORD" in this example has the same address as "Merker_LByte".

For systems with different byte orders, "Merker_WORD" is used to read different values.

0001	Merker_WORD (%MW100) = 256	
0002	Merker_LByte (%MB200) = 1	
0003	Merker_HByte (%MB201) = 0	
0004		
0001	Merker_LByte := 1;	Merker_LByte = 1
0002	Merker_HByte := 0;	Merker_HByte = 0
0003	Merker_WORD;	Merker_WORD = 256
0004		
0005		

DB000702_v01

Fig. 7-50: Value with Big Endian (MPx17 and above): "Merker_WORD = 256"

To compensate the byte order in the PLC program, the addressing of the byte flags simply has to be adjusted in the variable declaration.

0001	Merker_WORD (%MW100) = 1	
0002	Merker_LByte (%MB201) = 1	
0003	Merker_HByte (%MB200) = 0	
0004		

DB000703v01

Fig. 7-51: Adjusting the addressing of the byte flags in the variable declaration

Access via pointers

In a test program, values are assigned to the "arWORD" field of the "WORD" type. A pointer initialized with the start address of the field is to be used to read a value of the "DWORD" type. This value is checked in an IF query. Depending on the result, a "bDummy" variable is set. In the first figure, the IF query is successful and the "bDummy" variable is set to TRUE.

0001	arWord[0]:=3;	arWord[0] = 3
0002	arWord[1]:=1002;	arWord[1] = 1002
0003		
0004	ptDWORD:=ADR(arWord);	ptDWORD = <0cd189a0>
0005	ptDWORD^;	ptDWORD^ = 65667075
0006		
0007	IF (ptDWORD^ = 65667075) THEN	ptDWORD^ = 65667075
0008	bDummy := TRUE;	bDummy = TRUE
0009	ELSE	
0010	bDummy := FALSE;	bDummy = TRUE
0011	END_IF	
0012		

DB000704_v01

Fig. 7-52: Value with Little Endian (MPx08 and below)

When the exemplary program is ported to the "IndraDrive Cs" range, a different value is obtained via the pointer due to the "Big Endian" conversion. This makes the IF query inconsistent. In this example, the IF query would have to be adjusted.

0001	arWord[0]:=3;	arWord[0] = 3
0002	arWord[1]:=1002;	arWord[1] = 1002
0003		
0004	ptDWord:=ADR(arWord);	ptDWord = <0c418838>
0005	ptDWord^;	ptDWord^ = 197610
0006		
0007	IF (ptDWord^ = 65667075) THEN	ptDWord^ = 197610
0008	bDummy := TRUE;	bDummy = FALSE
0009	ELSE	
0010	bDummy := FALSE;	bDummy = FALSE
0011	END_IF	
0012		

DB000705_v01

Fig. 7-53: Value with Big Endian (MPx17 and above)



If function blocks are used to write list parameters, e.g., "MB_WriteListParameter", the field with the values to be written should always have the same data length as the list parameter. Otherwise, the problems described above will occur.

7.7.4 MLD performance comparison

A performance comparison measurement should be made with regard to MLD.

The "MX_IECTaskGetLoad" function block of the "MX_Base" library should be used for the performance comparison measurement. The "MX_IECTaskGetLoad" function block is used to activate the extended run-time measurement. It displays information on the task load.

The task load is the percentage ratio of the calculating time (t_{used}) used for the PLC program to the maximum available calculating time in a task cycle (t_{cycle}):

$$\text{Task load} = (t_{\text{used}}/t_{\text{cycle}}) [\%]$$

It is recommended to determine the task load of the user programs. If the load is rather high (more than 85%), it might be necessary to adjust the task cycle time or to use an "IndraDrive" controller with Advanced performance.

8 Programming information

8.1 Industrial standards for programming

8.1.1 General information

To allow PLC programs to be transferred easily and quickly from one target to the other, industrial standards for programming were established.



All function blocks made available by Bosch Rexroth have been programmed according to the standards.

8.1.2 IEC 61131

The definition of the industry standard for programming in automation in the form of **IEC 61131-3** has established the basic principles for a combined logic and motion control. These are, in essence:

- Defining the **SFC (Sequential Function Chart)** as a powerful programming tool for data flow-oriented programs, such as controllers
- Defining the Pascal-esque **ST (Structured Text)** high-level language as a powerful instrument even for programming complex mathematical functions
- Defining basic data types from **BOOL** to **DINT** (32-bit integer) or **REAL** (32-bit float), as well as arrays and structures
- Defining function blocks that can be instantiated
- Defining **standard function blocks** (counters, time functions, edge detection, etc.), as well as **standard functions** (mathematical functions, string processing, comparisons, etc.)

Complying with IEC 61131 ensures future compatibility of systems from different suppliers; programs conforming to the standard can run on all automation devices.

The programming languages defined in the standard improve exchange of control programs from different manufacturers. According to the standard, control programs can be written in the following languages:

- Instruction List (IL)
- Ladder Diagram (LD)
- Function Block Diagram (FBD)
- Structured Text (ST) and
- Sequential Function Chart (SFC)

Among other things, the standard contains definitions and functional features for the PLC and the links to other relevant ISO/IEC standards. Its status is "International Standard" (IS).

8.1.3 PLCopen

The motion control market displays a wide variety of incompatible systems and solutions. In businesses where different systems are used, this incompatibility induces considerable costs for the end-users, learning is confusing, engineering becomes difficult and the process of market growth slows down.

Standardization would certainly reduce these negative factors. Standardization means not only the programming languages itself (as it is done within the worldwide IEC 61131-3 standard), but also standardizing the interface towards different motion control solutions. As a result, the programming of

these motion control solutions is less hardware-dependent. The reusability of the application software is increased and the time and effort involved in training and project planning are reduced.

PLCopen has the following objectives:

- **Performance:** Users write their programs very close to the hardware with dedicated functions, in order to get the highest performance possible as dictated by their environment. This limits the users in their options with respect to the target hardware and the reusability of the control software and increases the training costs.
- **Functionality:** The second user option allows for a very broad range of software functionality that can be offered. This can be very helpful to the user, but will rarely lead to high performance. Moreover, training costs are very high.
- **Standardization:** The third corner, standardization, is primarily focused on reusability across different systems from different suppliers, including integrated, distributed and networked systems, as well as reduction in training investments. Due to the general character of this definition, the performance on different architectures can be less optimal than hard coding. Due to this, standardization should not be expected to offer maximum performance but can get very close to the maximum functionality, meaning that the bottom of the triangle is very short.

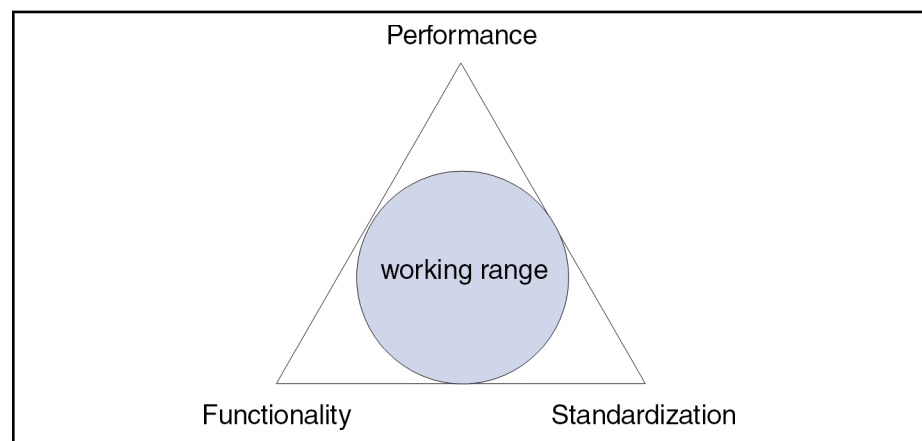


Figure name DF000146

Fig. 8-1: Objectives of PLCopen

The trend towards transferring PLC programs from one target to the other as easily and quickly as possible is intensified by the official release of the specification of the PLCopen Motion Task Force. This specification defines functions and function blocks for single-axis positioning, as well as electronic gear functions. In addition, a basic procedure for implementing motion-triggering function blocks was defined.

8.2 Libraries for Rexroth IndraMotion MLD

8.2.1 General properties of the libraries



This section provides an overview of the libraries for Rexroth IndraMotion MLD. For details, please refer to the library description ("Rexroth IndraMotion, MLD Libraries as of MPx-18").

The function blocks contained in the libraries for Rexroth IndraMotion MLD are based on the PLCopen specification and all of them, in principle, act in the same way.

Other function blocks that are required but not specified in PLCopen have the same syntax and semantics and some basic properties:

- The function blocks normally are called cyclically and provide information at their outputs, such as "Done", "Error", "CommandAborted".
- The processes are normally started with a rising edge at the "Execute" input. .
- Once completed or aborted, the corresponding output ("Done", "Error", "CommandAborted") is set. This output stays TRUE until "Execute" is reset. If "Execute" is already FALSE, each output remains in its status for one cycle (call) and then reverts to FALSE.
- The drive behaves according to the state machine defined in PLCopen. The user, however, cannot read the particular status.
- All motion function blocks are also working in a state machine that displays the status of the function block. This makes it possible, e.g., to detect whether or not a travel command has already been "picked up" by the drive.

Other features of the libraries or user interface:

- Instance behavior described in standard
- Parallelism of motion command values
- Command values: Position, velocity, acceleration
- AXIS_REF axis addressing (predefined variable in library)
- Error handling ["Error", "ErrorID" and "ErrorIdent" outputs]
- Unassigned inputs
- Cascading of function blocks

8.2.2 Libraries

Supported libraries

NOTICE

Property damage caused when non-supported libraries are included.

Only the libraries listed below are allowed to be included.

The following functions, data types and structures are combined in libraries available to the user.

When a project is created with an IndraDrive target, some libraries are automatically loaded. The user can include more libraries in the project.

Library	Description
IecSfc	Provides IEC steps conforming to standard in "Sequential Function Chart" (SFC)
IoDrvEtherNetIP	An EtherNet/IP scanner is provided. Using this scanner, other EtherNet/IP devices can be integrated as slaves (e.g., barcode reader, RFID, flow rate meter, laser distance meter, I/O, HMI,...).
MX_Base (IndraDrive) / MY_Base (HydraulicDrive)	• Data types or structures • Cyclic parameters as direct variables (system-wide variables) • Axis structures (for a multi-axis system [MLD system mode])
MX_CanL2 (IndraDrive only)	Contains data types, functions and function blocks for CAN communication on level 2
MX_CommonTypes	Contains data types and structures most of which are only used internally
MX_CheckRtv	For program-internal use only! Functions for checking or signaling runtime errors
MX_Debug	For internal test purposes only (laboratory)!

Programming information

Library	Description
MX_PLCOpen	• Functions for diagnostics • Function blocks/functions for drive control • Function blocks/functions for parameters • Functions for scaling • General functions
MX_TechInterface (Firmware version 20 and above)	The axis interface bundles and enhances PLCopen motion function blocks and provides an easy-to-use interface for drive functionality. Less code and more efficient commands speed up the programming of applications. The axis interface contains control signals and parameters for the various operation modes of the master axis and slave axis, as well as options for setting selected process values.
MX_TechBase (Firmware version 20 and above)	Contains basic functionalities such as control functions, functions to query system information and trigonometric functions.
MX_TechMotionBase (Firmware version 20 and above)	Contains functionalities such as probe function, position monitoring, position switch and functions to manage the cyclic data channel.
MX_TechMotion (Firmware version 20 and above)	Contains a function block to control the tension with regard to a command value. For this purpose, the measured value of a force transducer is evaluated.
MX_TechWinder (Firmware version 20 and above)	Contains function blocks providing functions for winding and unwinding fabric webs (winders).
MX_TechCrosscutCrossseal (Firmware version 20 and above)	Contains function blocks for cross sealer and cross cutter applications.
RIL_Fieldbus	Provides common data types (field bus types) for the Rexroth field bus libraries
RIL_HMI_Utility	Contains function blocks supporting the HMI devices
RIL_LoopControl	Provides basic elements and controllers of control technology
RIL_ModbusTCP	Provides function blocks which enable communication between Ethernet devices supporting the Modbus protocol
RIL_SercosIII (IndraDrive only)	Provides functions which enable communication between the PLC programming environment and the Sercos III nodes
RIL_SocketComm	Functions and function blocks for using TCP/UDP communication
RIL_Utility	Functions and function blocks for converting and influencing different data types
RMB_SercosIII_Util (Firmware version 20 and above)	Function blocks for axis control via Sercos bus
RMB_TechCam (IndraDrive only; firmware version 20 and above)	Function blocks for creating cam tables
CmplecTask	System library containing, amongst others, interface functions used to switch off a watchdog (and switch it back on afterwards) (see MLD Application Manual: "Task monitoring (watchdog)")
SysDir / SysDir23 ^{*1}	Functions for synchronously accessing a file directory system on the target
SysFile / SysLibFile23 ^{*1}	Functions for synchronously accessing files
SysFileAsync / SysFileAsync23 ^{*1}	Functions for asynchronously accessing files
SyslecTasks23	Supports the "SysIECTaskGetConfig" function that is used to read the task configuration
SysMem / SysMem23 ^{*1}	Functions for memory management
NetVarUdp	For program-internal use only! Functions for processing network variables
Standard	Functions and function blocks that are required by IEC 61131-3 as standard blocks for an IEC programming system
SysCallback23 ^{*1}	Functions for activating defined callback functions for runtime events

Library	Description
SysSem / SysSem23 ^{*1}	Functions for creating and using semaphores for task synchronization
SysSockets / SysSockets23 ^{*1}	Functions supporting the access to sockets for communication via TCP/IP and UDP
SysStr23 ^{*1}	Functions for handling strings
SysTime	Provides functions for reading and setting the real-time clock of the local system, as well as enabling various conversions of the time data. NOTE: To begin with, please verify whether the "RIL_Uutilities" library can fulfill your requirements.
SysTimeRTC	Provides functions to read or write the system time of the drive in the IEC 61588 format ("S-0-1305.0.1, System time").
Util	Function blocks that can be used for BCD conversion, bit/byte functions, mathematical auxiliary functions, as controllers, signal generators, function manipulators and for analog value processing
OSCAT	Free "OSCAT" ("Open Source Community for Automation Technology") PLC library. This open-source library contains many useful functions in the areas of "automation technology" and "building automation".

***1** Libraries with the extension "23" (e.g., "SysSockets23") are used to port projects; for new projects, use the libraries without the extension "23"

Hierarchic structure of the libraries

The figure below uses the "IndraDrive MP18" target to illustrate how the libraries build on one another. In principle, the user does not have to include any library, but can use the automatically included libraries.



The "MX_PLCOpen" library automatically includes its subordinate libraries ("MX_Base", "MX_CommonTypes").

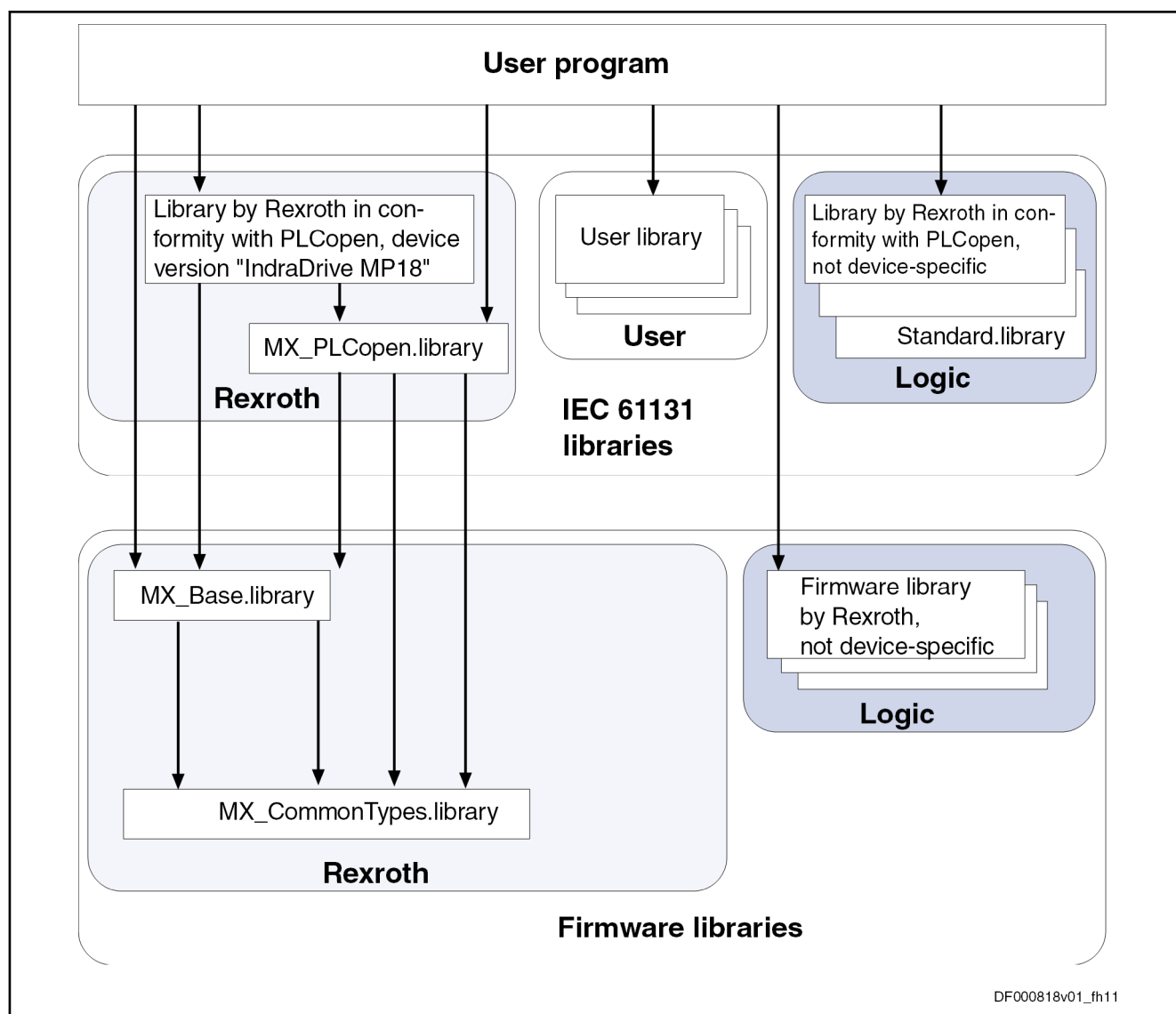


Fig. 8-2: Overview - hierarchic structure of the libraries

8.3 Retain data backup and restore

Retain data can be backed up and restored. This may be done manually or automatically, e.g. after system events or cyclically time-controlled.

The following options are only two of the possible applications:

- Backup in case of unintended loss of the retain data ([Manual retain data backup and restore](#))
- [Saving persistent variables upon Reset origin and initializing with saved persistent data upon PLC restart](#)

8.3.1 Manual retain data backup and restore

Retain data can be backed up and restored manually by saving the values of the persistent variables (retain data) in a recipe and reloading them afterwards, as shown in the following example.

The example assumes that a persistent variable list with the following content was created:

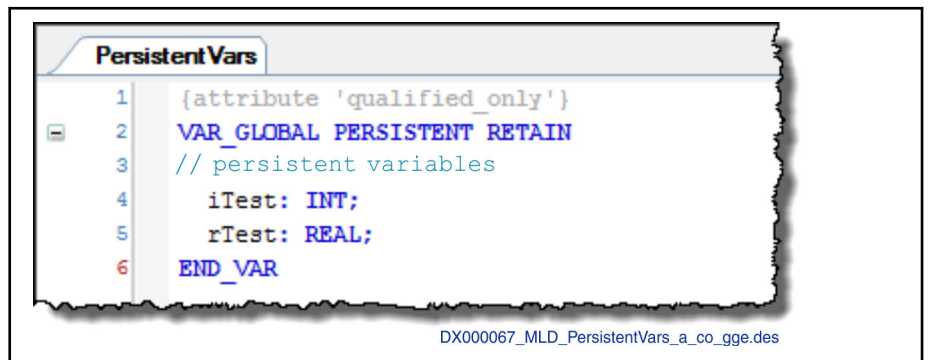


Fig. 8-3: Persistent variable list

1. Start IndraWorks and log on to the PLC.
2. Double-click the persistent variable list.
⇒ The editor for the persistent variable list opens.
3. Open the context menu of the editor for the persistent variable list and select the **Save current values in recipe** menu.

Programming information

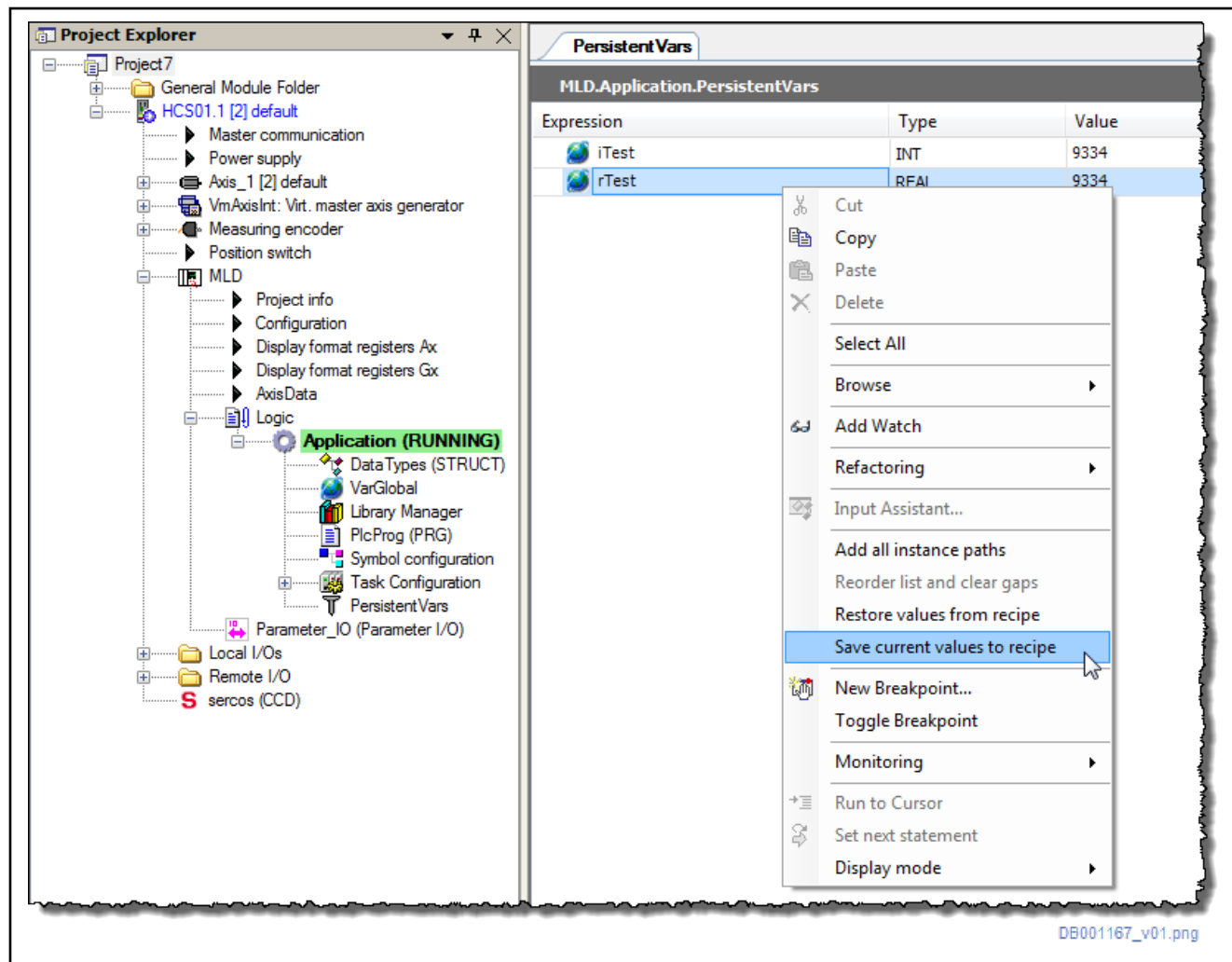


Fig. 8-4: Context menu of the editor for the persistent variable list

⇒ The recipe manager and a recipe definition are created for the persistent variable list in the Project Explorer below the application branch (here "PersistentVariables" recipe definition):

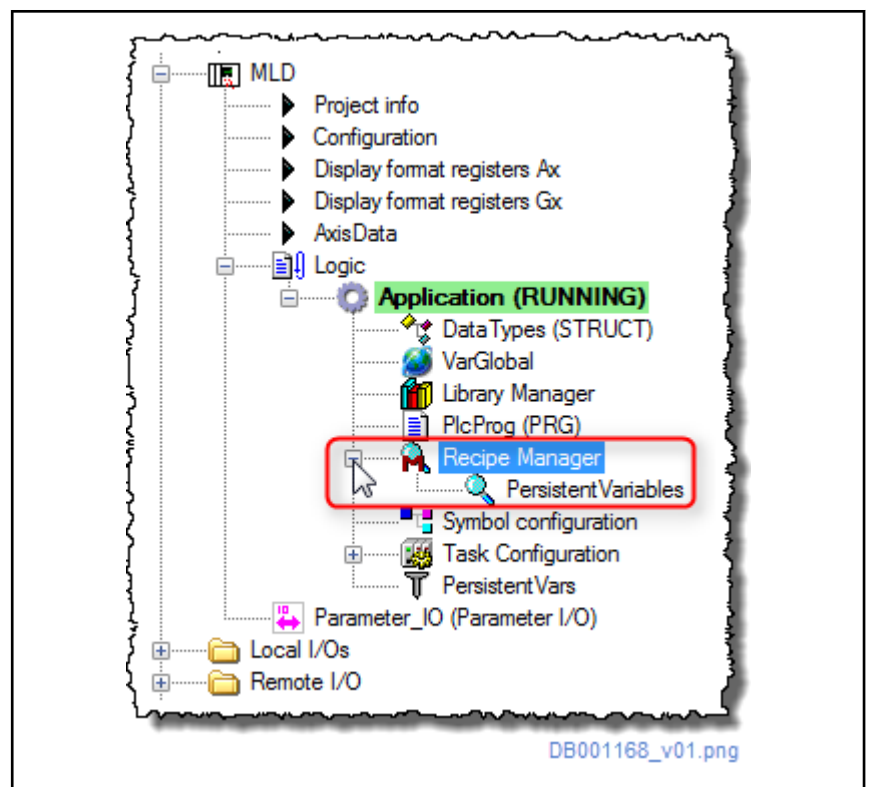


Fig. 8-5: Recipe manager

4. Open "PersistentVariables" (double-click).

Persistent Variables							
Variable	Type	Name	Comment	Minimal Value	Maximal Value	Current Value	PersVars
PersistentVars.iTest	INT					32278	13216
PersistentVars.rTest	REAL					32278	13216

DB001169_v01.png

Fig. 8-6: "PersistentVariables" dialog box

5. In addition to the current values of the persistent variables, the "PersVars" recipe is displayed. Point the mouse at one of the cells in the "PersVars" column and open the context menu.
 ⇒ The available commands for the "PersVars" recipe are displayed:

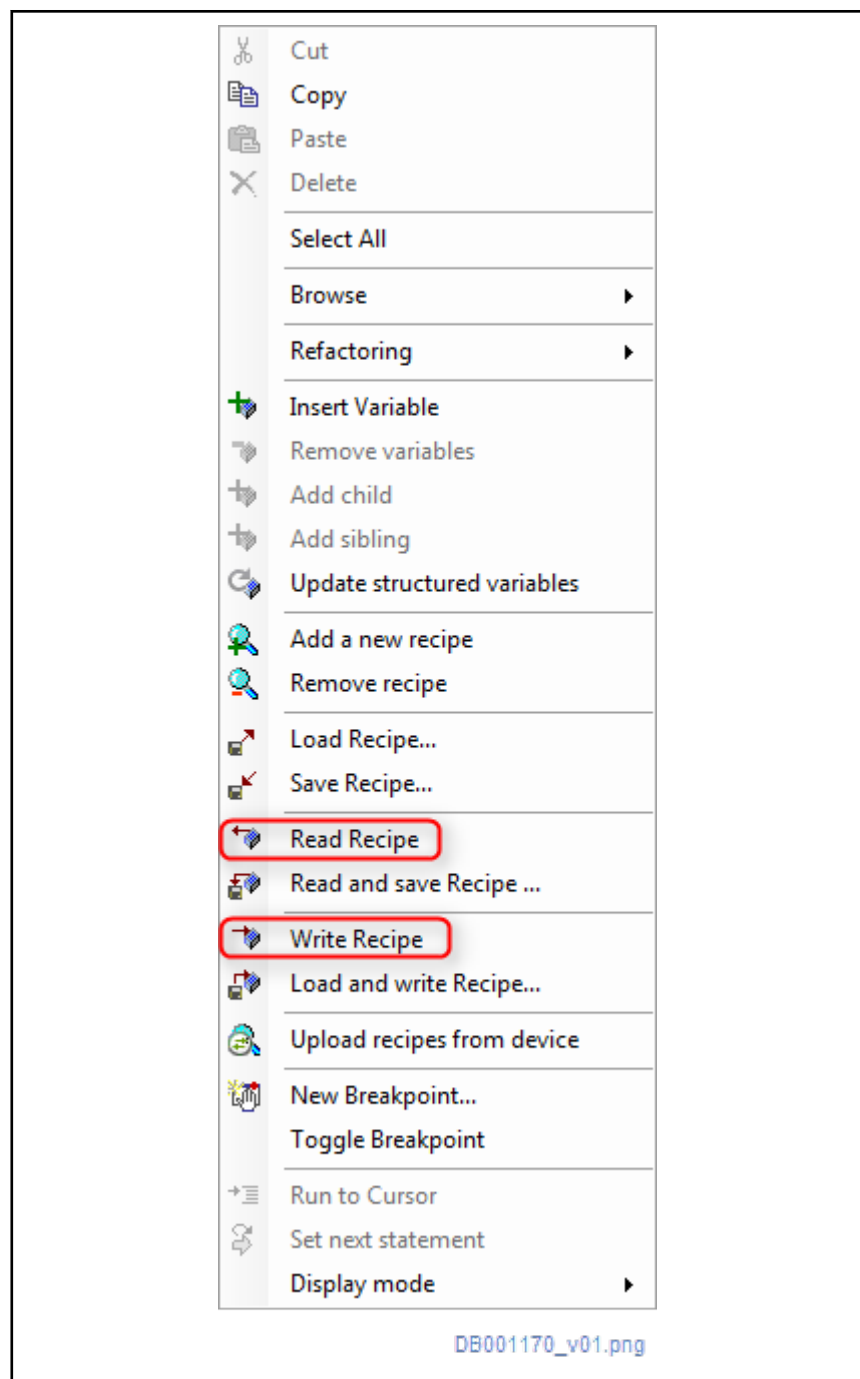


Fig. 8-7: Context menu of a recipe

- "Read recipe" applies the current values of the persistent variables in the PLC to the corresponding cells of the "PersVars" column.
- "Write recipe" overwrites the current values of the persistent variables in the PLC with the values of the corresponding cells of the "PersVars" column. So the values are written by the recipe to the PLC.
- With "Save recipe...", the recipe is saved as a file (*.txtrecipe) on the PC and reloaded at a later point in time with "Load recipe...".

8.3.2 Saving persistent variables upon "Reset origin" and initializing with saved persistent data upon PLC restart

The following example shows how persistent variables are automatically saved upon "Reset origin¹⁾" and how they can be initialized with the saved values after the PLC program has been reloaded and the PLC started.

The example assumes that a persistent variable list with the following content was created:

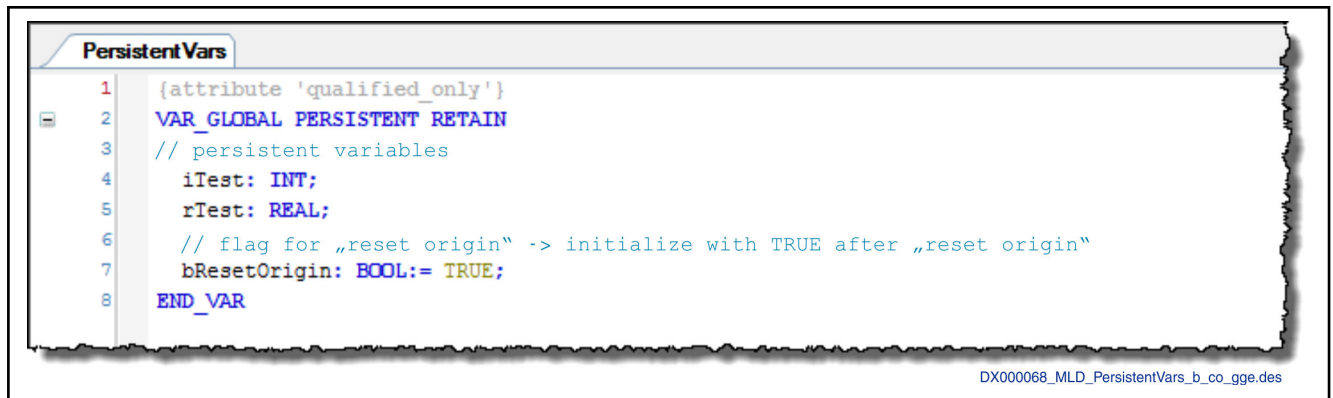


Fig. 8-8: Persistent variable list

1. Start IndraWorks and log on to the PLC.
2. Double-click the persistent variable list.
⇒ The editor for the persistent variable list opens.
3. Open the context menu of the editor for the persistent variable list and select the **Save current values in recipe** menu.

¹⁾ The "Reset origin" command resets all variables of the active application to their initialization values and deletes the application program from the control

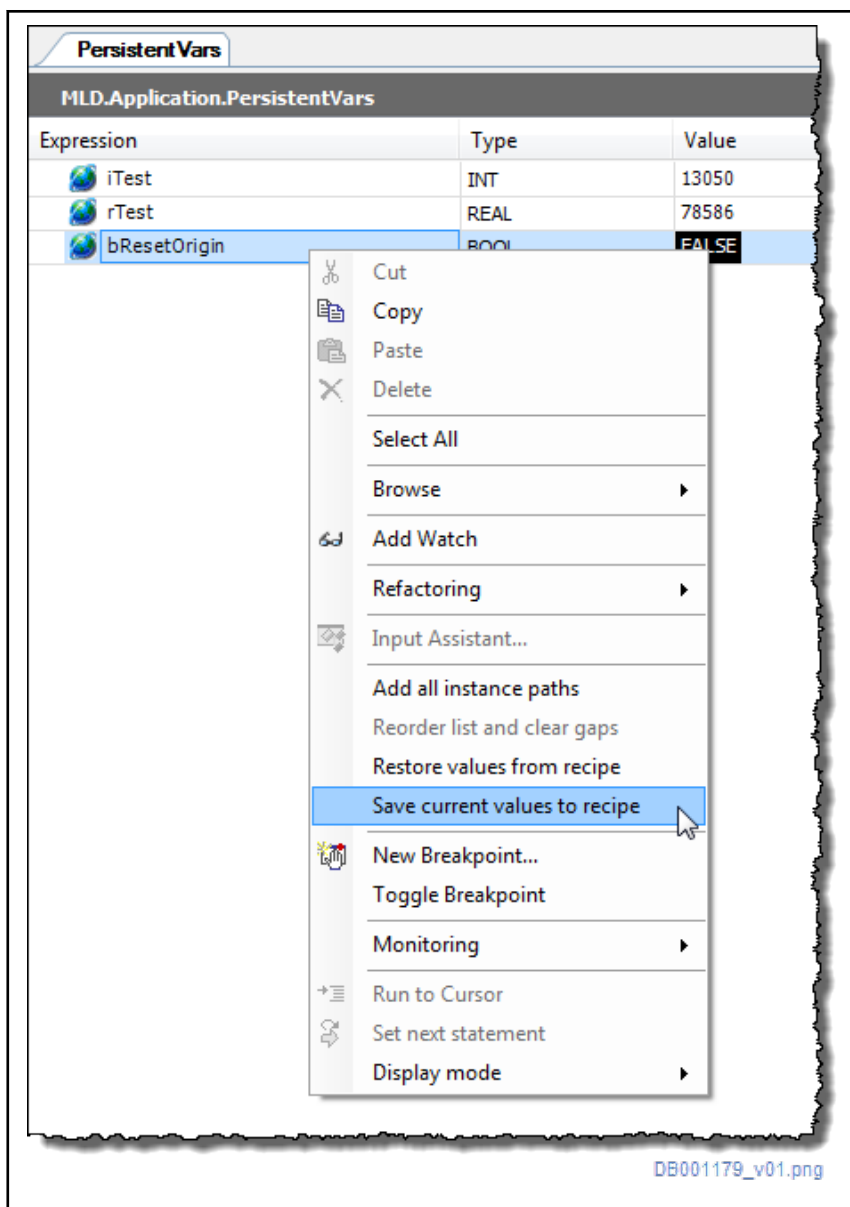


Fig. 8-9: Context menu of the editor for the persistent variable list

⇒ The recipe manager and a recipe definition are created for the persistent variable list in the Project Explorer below the application branch (here "PersistentVariables" recipe definition):

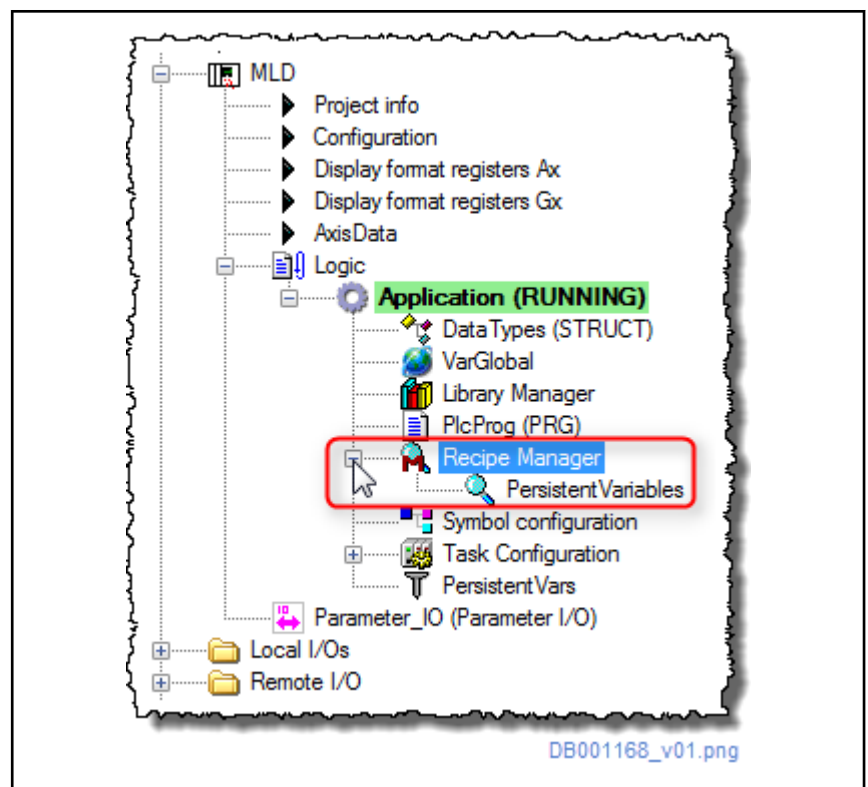


Fig. 8-10: Recipe manager

4. Open the recipe manager (double-click "Recipe manager" in the Project Explorer) and go to the "General" tab.
5. Activate the check boxes **Recipe management in the PLC** and **Save recipe changes in recipe files automatically**.

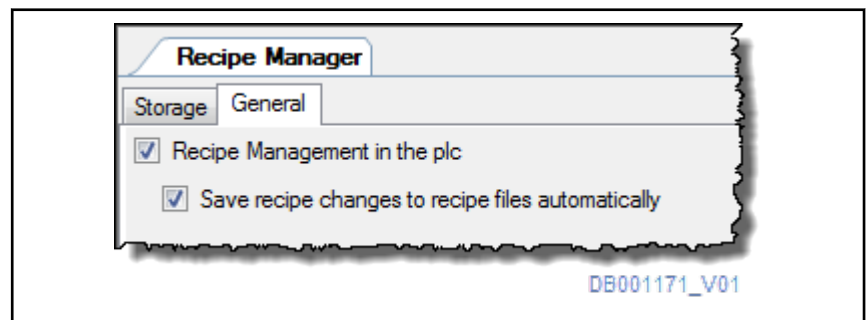


Fig. 8-11: Configuring the recipe manager

6. Open the task configuration and go to the "System events" tab.
7. Use the **Add event handler...** button to add the two "PrepareReset" and "PrepareStart" events. For each event, specify a function that is to be called upon occurrence of the event. In the example, the functions were named "MyReset" and "MyStart".

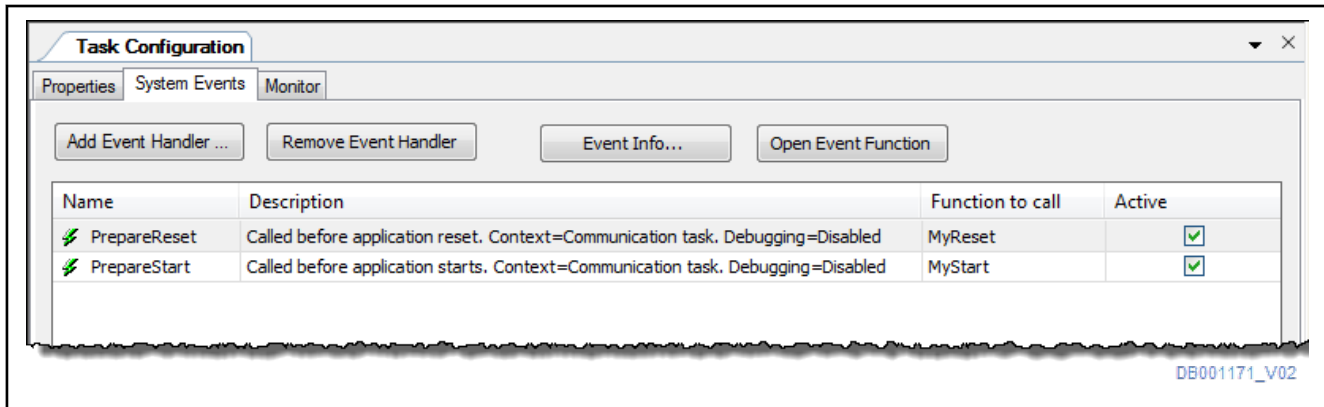


Fig. 8-12: Add event handler

8. The "MyReset" function is automatically created by IndraWorks when the "PrepareReset" EventHandler is added and the function to be executed (MyReset) is specified.

The "MyReset" function is used to program what has to be done in the case of PLC reset.

In the following example, the values of the persistent variables are read to the "PersVars" recipe and saved as a file ("MyPersVars.txtrecipe") in the case of "Reset origin":



Fig. 8-13: Possible implementation of the requirement in the "MyReset" function

9. The "MyStart" function is automatically created by IndraWorks when the "PrepareStart" EventHandler is added and the function to be executed ("MyStart") is specified.

The "MyStart" function is used to program what has to be done in the case of PLC start.

In the following example, the values are loaded from the "MyPersVars.txtrecipe" recipe file to the "PersVars" recipe and applied to the persistent PLC variables. After the one-time read-in of the persistent variables, the .PersistenVars.bResetOrigin variable is set to FALSE in order to prevent the persistent variables from being filled with

the values from the recipe upon every PLC start, but only after "Reset origin":

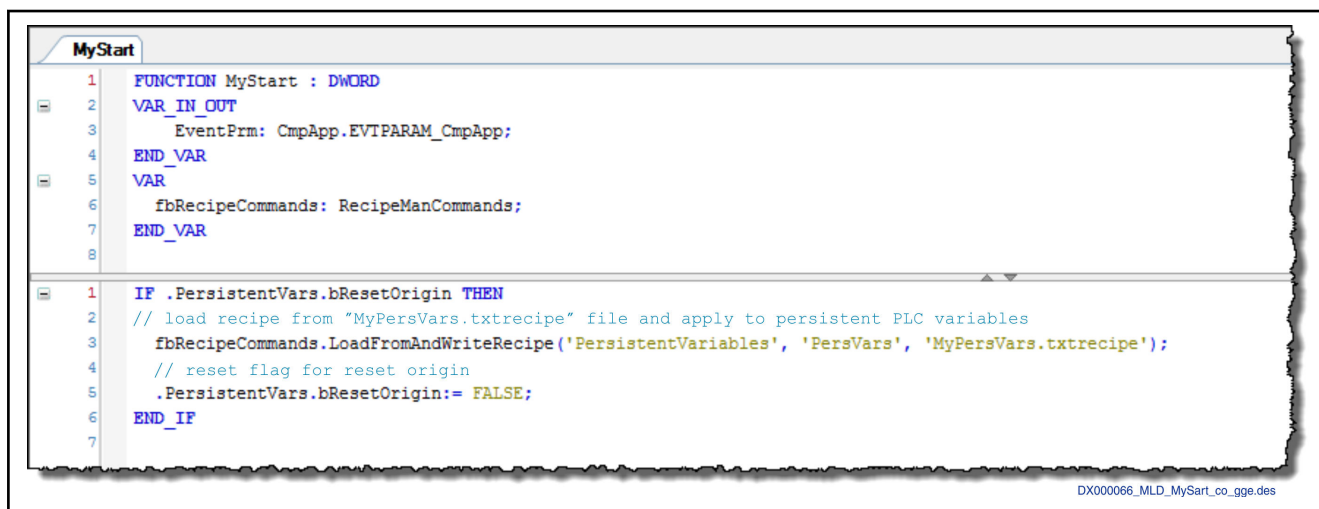


Fig. 8-14: Possible implementation of the requirement in the "MyStart" function

8.4 Accessing files on the optional memory card

The file system can be accessed synchronously and asynchronously.

Synchronous access to the file system

In the case of synchronous access to the file system, a function is called which only returns after the output has been completed. This is disadvantageous for a typical PLC task, since this task "waits" for the function during this time. It is therefore recommended that these tasks be executed in a low priority task without watchdog or with a very long watchdog time, depending on the requirements.

Asynchronous access to the file system

In the case of asynchronous access to the file system, cyclically-oriented function blocks are used. As is the case with the motion function blocks, the tasks are activated with an edge at the input. The status of the task is provided in further calls. The function blocks have to be cyclically called until they signal "bDone". Depending on the function, the return information is contained in another output.

The runtimes of the function blocks are relatively short so that they can be used in a cyclic task.

The actual task is processed in a separate, low-priority system task. This task is not constricted, even if the capacity of the PLC is fully used. Its velocity, however, depends on the load of the drive.

The library with function blocks for asynchronous access to the file system is suited for use in a typical cyclic PLC task with watchdog.

Libraries for accessing the file system

The system libraries "SysFile", "SysFileAsync" and "SysDir" are available to access the file system.

Folder structure for accessing the file system

The files are stored in a specific "User" directory on the memory card. All function blocks and functions are called with the file name and optionally a preceding subdirectory (user sub-folder).

The PLC only allows the "User" folder to be accessed. For reasons of safety, it is impossible to access any other folder and file with the PLC.

When the directories are listed in the Windows Explorer, only the content of the "User" folder and any sub-folders it contains are displayed. The "User" directory itself is not displayed, as if "User" were a separate partition. Accessing the root branch from the user's point of view is possible without specifying the directory, with single "\" and with double "\\" (Windows-compatible).

File access The "Open" functions do not open the files exclusively so that other access to the same file is possible.

As long as the PLC program is running, the files can be kept open. However, they are forced to close when the PLC program is reset with "Reset" or the memory card is removed. In the latter case, the error F2006 is generated.

A maximum of 8 files can be opened at the same time. If more files are opened, an error is generated.

"SysDirOpen" may be started a maximum of 4 times without reading all entries with "SysDirRead". If it is called more often, an error is generated.

When the **PLC is reset**, open files and directory search handles are closed and all resources are released.

DANGER

Data might be lost in case voltage fails while data are written to the memory card. In the worst case, the memory card cannot be read any more thereafter.

To prevent data loss, a backup of the complete memory card should be made at regular intervals.

When a memory card cannot be read any more in the drive, it might be possible to repair it with a PC and a card reader using the Windows "Error-checking" function. In case of an error, "Lost Clusters" are combined into files.

8.5 Start behavior and boot project

8.5.1 General information

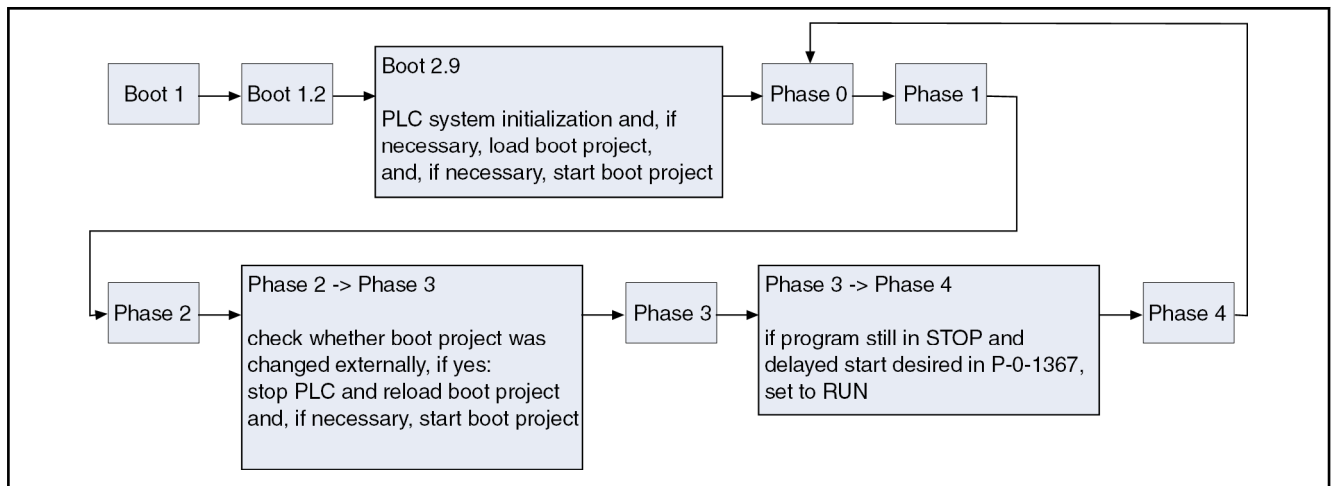
As the drive-integrated control unit runs in parallel as a task, it is initialized together with the drive. As regards the different communication phases, the drive-integrated control is also subject to the conventions determined for the drive firmware.

See also Functional description of firmware "Device-internal state machine".

8.5.2 Brief description

The drive-integrated PLC (Rexroth IndraMotion MLD) is activated in the booting process (Boot 2.9). It first initializes itself and checks whether a correct boot project is available. If yes, it is loaded. Depending on the configuration, the drive-integrated PLC is set to RUN, remains in STOP or is only started in phase 4 (operating mode) (parameterized via "P-0-1367, PLC configuration").

The figure below illustrates the booting process and the communication phase sequence of Rexroth IndraMotion MLD:

**Figure name** DF000133*Fig. 8-15: Rexroth IndraMotion MLD run-up***Pertinent parameters**

The following parameters are used in this context:

- P-0-1367, PLC configuration
- S-0-0127, C0100 Communication phase 3 transition check
- S-0-0128, C0200 Communication phase 4 transition check

Pertinent diagnostic messages

The following diagnostic messages are generated during the run-up phase:

- C0100 Communication phase 2 to 3 transition check
- C0200 Communication phase 3 to 4 transition check

If an error occurs during the transition checks, this is signaled with a diagnostic message of pattern C01xx or C02xx.

Booting up

During boot-up

- the PLC operating system is initialized (unless prevented before Boot 2.9)
- a boot project is loaded if necessary (if available)
- the boot project is started if necessary (unless prevented via "P-0-1367, PLC configuration")

C0100 Communication phase 2 to 3 transition check

In the transition command from communication phase 2 to communication phase 3 (also see "C0100 Communication phase 2 to 3 transition check"), the following steps are run:

- A check is run to find out whether the boot project was changed. If the boot project was changed, the PLC is stopped and the boot project is reloaded.
- If necessary, the reloaded program is started (unless prevented via "P-0-1367, PLC configuration").

C0200 Communication phase 3 to 4 transition check

In the transition command from communication phase 3 to communication phase 4 (compare "C0200 Communication phase 3 to 4 transition check"), the following steps are run:

- If an available boot project has not been started, it is started now.
- The system control, i.e., the access to the device control of the drive, is started.

8.5.3 Notes on application and programming

The paragraphs below contain some information on the configuration (parameterization) and handling of MLD during programming and use.

Configuring the start behavior	The start behavior (boot-up) is configured by parameterizing "P-0-1367, PLC configuration" (bits 0 and 1).
Configuring device control	Bit 4 of "P-0-1367, PLC configuration" is used to configure the control over the drive, i.e., access to the internal device control.
Boot project	During the change from phase 2 to phase 3, a check is run to find out whether the last change of the boot project took place via Parameter-Write (i.e., not by IndraLogic). If yes, the PLC is de-initialized and reinitialized. It thereby starts the boot project which has just been loaded.
When does the PLC run?	The PLC starts running during boot-up after the system tasks have been enabled, that is also in phase 2. When you switch between phases 2 and 4, the PLC keeps running. It is only stopped for a short time between phases 2 and 4 if the PLC boot project was changed and restarted with the new project.
Emergency mechanism to stop the PLC	Since a boot project is automatically loaded during drive run-up, a mechanism has been implemented with which the start of PLC programs can be prevented: • In the boot-up phase, before Boot 2.9 is displayed at the control panel, the "ESC" and "ENTER" keys have to be simultaneously pressed and held at the control panel. • The automatic start of a PLC boot project is prevented and the display reads "PLC ?". • By pressing the arrow keys (arrow down or arrow up) the display switches between "Run PLC" and "Stop PLC". If "Stop PLC" is confirmed with "ENTER", no project is loaded. If "Run PLC" is confirmed with "ENTER", a stored project is started or loaded.

8.6 Guidelines for programming with IndraLogic

8.6.1 Overview

This section is intended to impart knowledge about the basics of structuring and programming PLC projects with IndraLogic for the "Rexroth IndraMotion MLD" target.

The description includes suggestions and recommendations for PLC programming in the form of basic guidelines. They are intended to allow consistent use of software structures.

Basic knowledge of the IndraLogic PLC programming system is required (see documentation "IndraLogic 2G Programming Guide").



Observe the guidelines fully for optimal results. If this is not possible, attempt to follow them as closely as possible.

8.6.2 Structuring PLC projects

Task management

We basically recommend creating a task configuration so that the automatic single-task mechanism does not take effect (a max. of 4 tasks is possible).



If a task configuration is created, the (single-task) main program **PLC_PRG** is not used.

Based on a preemptive time-controlled and event-controlled, as well as constantly "freewheeling" task, the task configuration always provides a solution optimally adjusted to the task.

In general, the number of tasks used should be as low as possible but as high as appropriate for the task. This makes PLC projects clearly structured and easy to test.

Multi-program technology

A main program has to be assigned to each task. If required, this program can contain further blocks (programs and function blocks) (multi-program technology). This provides a high degree of flexibility as regards the calling and controlling of further function blocks. In addition, the order within the respective main program is therefore unequivocal.



It is possible to assign multiple programs to any task. The processing of the programs corresponds to the order assigned. However, further control of these programs is then only possible within the programs themselves, for which reason this variant is not recommended.

By naming the function blocks, the order of display in the "POUs" register (tree) can be influenced. It is advantageous to place the main programs at the beginning, e.g., "_prMain_<machine number>".

Within the function block view, the arrangement of the function blocks can be structured using further sub-folders. It is therefore possible to make a functional arrangement in groups, thus giving the entire project a clearer structure.



Details in the programs/function blocks can be seen via the "POUs" register (tree) or in the program editor via <Alt>+<Enter>. The calling point alone determines the location in chronological processing. Where applicable, the function blocks should be named so the order can also be seen in the tree.

8.6.3 Programming languages

Overview

A total of "5+1" programming languages are available in IndraLogic:

- Instruction List (IL)
- Structured Text (ST)
- Ladder Diagram (LD)
- Function Block Diagram (FBD)
- Continuous Function Chart (CFC)
- + Sequential Function Chart (SFC)

The basic programming language that should be used depends on the application.

Recommended basic programming languages

Logic processing	Ladder Diagram (LD) should be used as the basic programming language. This applies specifically to the basic logic that is commissioned by the machine installer and maintained and, if necessary, programmed by the system owner (especially external I/O handling).
Motion Control	The programming language used for "Motion Control" functions depends on the regulations of the system owner.

If interfaces and functional principles of function blocks have been exactly documented, it is less important which programming language was chosen. A separate programming language can be selected for each new block.



This also applies to each new action.



Actions can be used within function blocks, too, and not only within sequences of Sequential Function Chart (SFC). These actions then use the same declaration as the function block itself.

Notes on SFC programming

IEC steps should be used as often as possible (**Tools ► Use IEC-steps**). At an IEC step it is possible to program a maximum of 9 actions and one input and output action each.



The order in which actions are processed per step is alphanumeric and not based on the graphical order in the SFC sequence!

The following applies:

- Details in the actions can only be seen via the function block view.
- Transitions are used as step-enabling conditions and, as is the case with actions, can also be programmed in any language.

In a PLC cycle, the active steps/actions are always executed first and then the transition is calculated. When the transition has been fulfilled, the sequence is advanced in the next PLC cycle, but the actions of the previous step(s) are post-processed before the new step/action(s). It is therefore worth considering the use of a freewheeling task, because in this case processing is continued almost immediately after changing to another step.

Programming with the simplified steps is possible for simple sequences. Step and action are fused to form one logic implementation per step. Details in the logic can be seen directly via the step.

8.6.4 Global data

Accessing global data is possible in all function blocks.

Global data are normally used for cross communication between function blocks and can therefore relieve the call interface of function blocks. In addition, it is not necessary to adjust the call interface repeatedly in case of modifications. This is to advantage in that the appearance is maintained and the user documentation of the PLC project does not have to be changed.

The global data are declared in the "Resources" register. There, other global data lists and sub-folders can be created in order to achieve a better structure.

Declarations of global data can in particular be used as a central point of the declaration of I/O data in order to have a total list of I/O data in the PLC project (if desired).

8.6.5 Program header

The program header provides quick and clear information about the assigned program code.

The completely filled header contains the following pieces of information:

- Function/intended use of the PLC program in a few words
- Current version

- Name of the programmer
- Date of last change
- Company for which the programmer works
- IndraLogic targets on which the PLC program can run
- Detailed description of the function/intended use of the PLC program, as well as information on specific preparations or handling

The header should be written in English.



Using the header is highly recommended.

The template below can be used to achieve a uniform header. To do this, copy the text to the clipboard. Then paste the content of the clipboard into the declaration section in IndraLogic before all other code.



If **Declaration in table form** has been selected in the IndraLogic editor, you have to paste the content of the clipboard under the "Info" tab.

Header template

Program:

```
(*#####*)
(*#####*)
(* General Header *)
(*-----*)
(* Short description      : *)
(* Version                : *)
(* Name                   : *)
(* Date                   : *)
(* Company                : *)
(* Target                 : *)
(* Functional description : *)
(* Handling specials      : *)
(*-----*)
(* Additional Header *)
(*-----*)
(* *)
(* *)
(* *)
(* *)
(* *)
(*-----*)
(*#####*)
(*#####*)
```

8.6.6 History

The history documents changes in the code.

The completely filled history contains the following pieces of information:

- Version
- Name of the programmer
- Date of change
- Description of change

The header should be written in English.



Using the history is highly recommended.

The template below can be used to achieve a uniform history. To do this, copy the text to the clipboard. Then paste the content of the clipboard into IndraLogic at the end of the declaration.



If pasted at the end of the declaration, the history is no longer visible when the user changes to the tabular view and subsequently makes changes in the declaration.



If **Declaration in table form** has been selected in the IndraLogic editor, you have to paste the content of the clipboard under the "Info" tab page, directly after the header.

History template

Program:

```
(******)
(*Modification - History*)
(*-----*)
(* Version      : *)
(* Name         : *)
(* Date         : *)
(* Comment      : *)
(**)
(* Version      : *)
(* Name         : *)
(* Date         : *)
(* Comment      : *)
(**)
(* Version      : *)
(* Name         : *)
(* Date         : *)
(* Comment      : *)
(*-----*)
(******)
```

8.6.7 Type identifiers

General information

Standardized naming of types increases the legibility of the program code. This makes it much easier for third persons to become acquainted with the code and reduces the time required for troubleshooting.

In principle, we distinguish two types:

- Types which can contain program code. This includes programs, function blocks and functions.
- Types which do not contain program code. This includes structures, arrays and enumerators, as well as all IEC data types, such as String, Integer or Real.

Type identifiers for programs, function blocks and functions

According to PLCopen, the type identifiers for programs, functions and function blocks were named in English in case-sensitive form.



There is a conflict between PLCopen and the IEC standard. Functions and function blocks according to IEC standard are written in all caps (e.g., R_TRIG, TON, etc.). In contrast, the names of functions and function blocks of PLCopen are case-sensitive (e.g., "MC_MoveAbsolute").



As far as possible, the programming guidelines are based on the PLCopen standard.

Programs

Type identifiers for programs have no underscores, prefixes or suffixes.
Example: "MainProgram"

Functions and function blocks Abbreviations for system-specific and system-independent functions and function blocks were defined for Bosch Rexroth. They have to be added to the type identifier as a prefix with underscore.

Prefix	System-dependent	Assignment	Description	Example
MC	No	PLCopen	100% PLCopen function blocks	MC_MoveAbsolute
MB	No	Motion-oriented as per PLCopen	Function blocks with motion-oriented functionalities that are not PLCopen-certified, but based on PLCopen	MB_WriteParameter
MH	Yes	MLC	Hydraulic functions, only available for MLC	MH_HydrControl
IL	No	-	All system-independent functions and function blocks that are neither based on PLCopen nor motion-oriented	IL_SercosAttribute
ML	Yes	MLC	MLC-specific functions and function blocks	ML_ReadParameter-Bool
MS	Yes	Synax	Synax-specific functions and function blocks	MS_ReadSingleParameter
MSV	Yes	Synax + VisualMotion	Functions and function blocks for the Synax and VisualMotion systems	MSV_ReadMaxValue
MT	Yes	MTX	MTX-specific functions and function blocks	MT_NcBlk
MV	Yes	VisualMotion	VisualMotion-specific functions and function blocks	MV_Hysteresis
MX	Yes	MLD (IndraDrive and HydraulicDrive)	Functions and function blocks, specifically for the drive-integrated PLC	MX_SetDeviceMode

Tab. 8-1: Overview of prefixes for functions and function blocks



Function blocks and functions marked with "IL", "MC" or "MB" are not necessarily included on or supported by every system (e.g., MTX, SYNAX, etc.). In fact, each system that includes a function block with "IL", "MC" or "MB" in its name has to implement this function block in such a way that its system-wide external behavior is the same. This means that the appearance and behavior of, e.g., the function block "IL_SercosAttribute" has to be externally the same in every system that includes this function block.

Type identifiers for structures, arrays, enumerators and IEC data types

According to PLCopen, the type identifiers for structures, arrays and enumerators are generated in uppercase letters only. To improve legibility, underscores were partly used.



No type identifier is used, if arrays have been directly defined in the declaration.

If structures, arrays or enumerators are linked to a specific system, the names normally contain prefixes:

System	Prefix	Example
MLC	ML	ML_ReadParameterBool
Synax	MS	MS_ReadSingleParameter
Synax + VisualMotion	MSV	MSV_ReadMaxValue
MTX	MT	MT_NcBlk
VisualMotion	MV	MV_Hysteresis
MLD	MX	MX_SetDeviceMode

Tab. 8-2: Prefixes for system-dependent types

8.6.8 Instance identifiers

General information

It is possible to generate copies (instances) of data types. Instance identifiers are case-sensitive; they have been given English names.

Instance identifiers of complex data types

The prefixes listed below were used for instances of complex data types. The prefixes are added to the name with lower-case letters and without underscore.

Data type	Prefix	Example instance	Example type
Function blocks	Fb	fbJogMode	MT_Jogging
Structures	St	stComData	MX_COM_DATA
Arrays	Ar	arStateInfo	MV_STATE_INFO
Enumerators	En	enDiagData	ML_DIAG_DATA

Tab. 8-3: Instance identifiers of complex data types

Instance identifiers of simple data types

Prefixes for instances of simple data types

The prefixes listed below were used for instances of simple data types. The prefixes are added to the name with lower-case letters and without underscore.

Data type	Prefix	Example	Memory assignment	Data type designation	Data type description
BOOL	b	bVar	1 bit	Bool	Bit-oriented Boolean format
BYTE	by	byVar	8 bits	Byte	Bit-oriented byte format
WORD	w	wVar	16 bits	Word	Bit-oriented format of single word length
DWORD	dw	dwVar	32 bits	Double Word	Bit-oriented format of double word length
LWORD	lw	lwVar	64 bits	Long Word	Bit-oriented format of quadruple word length
SINT	si	siVar	8 bits	Short Integer	Signed integral format of abbreviated length
INT	i	iVar	16 bits	Integer	Signed integral format of single length

Data type	Prefix	Example	Memory assignment	Data type designation	Data type description
DINT	di	diVar	32 bits	Double Integer	Signed integral format of double length
LINT	lii	liVar	64 bits	Long Integer	Signed integral format of quadruple length
USINT	usi	usiVar	8 bits	UnsignedShort Integer	Unsigned integral format of abbreviated length
UINT	ui	uiVar	16 bits	Unsigned Integer	Unsigned integral format of single length
UDINT	udi	udiVar	32 bits	Unsigned Double Integer	Unsigned integral format of double length
ULINT	uli	uliVar	64 bits	Unsigned Long Integer	Unsigned integral format of quadruple length
REAL	r	rVar	32 bits	Real	Real numeric format of single length
LREAL	lr	lrVar	64 bits	Long Real	Real numeric format of double length
STRING	str	strVar	8 bits per character	String	String of 1-255 characters (ANSI code possible)
WSTRING	wstr	wstrVar	16 bits per character	Wide String	String of 1-65535 characters (UNI code possible)
TIME	t	tVar	32 bits	Time	Time format
DATE	d	dVar	32 bits	Date	Date format
TIME_OF_DAY	tod	todVar	32 bits	Time Of Day	Time of day format
DATE_AND_TIME	dat	datVar	32 bits	Date And Time	Date and time format
POINTER TO ???	p???	ptrVar	32 bits	Pointer To ???	Pointer/address of a variable with specific data type
POINTER TO DWORD	pdw	pdwVar	32 bits	Pointer To Double Word	Example: Pointer to a double word variable

Note At present, IndraLogic does not support data types LWORD, LINT, ULINT and WSTRING.

??? Data type of variable to which the pointer is pointing

Tab. 8-4: Instances of simple data types

Suffixes for instances of simple data types

The instance names of simple data types can additionally be given suffixes to identify their origins. They have been added to the name as lower-case letters with underscore.

Suffix	Significance	Example
i	Hardware input	bAxisTravelLimit_i
q	Hardware output	bLockDoor_q
gb	Global variable	stControlState_gb
m	Local flag	iComData_m

Tab. 8-5: Examples of suffixes of simple data types

8.6.9 Definition of standard interfaces at function blocks

Introduction

Most of the function blocks feature an input for activation and an output that is signaling error-free processing. In addition, an output is often necessary which indicates the processing time. Additional outputs are defined for the indication of errors.

Standardized naming and identical behavior of these standard interfaces increase comprehension, reduce the time required to become acquainted with the code and so relieve the support department.

Function blocks can be used to encapsulate complex tasks such that they can be reused and to address them via defined interfaces. Processing can take place in either state-controlled or edge-controlled form.

State-controlled	If repeating its task again and again when having reached a defined state after switch-on, a function block is state-controlled.
Edge-controlled	If executing its task exactly once after switch-on, a function block is edge-controlled.

Standard inputs at function blocks

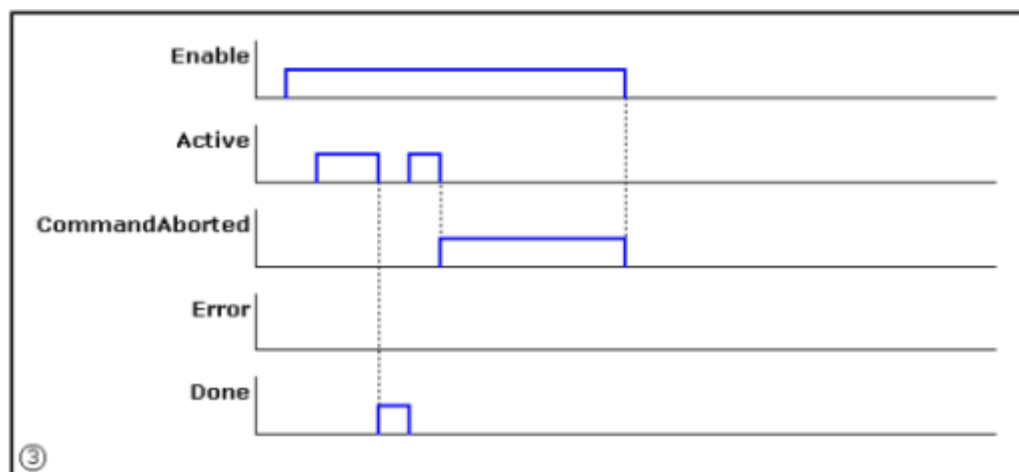
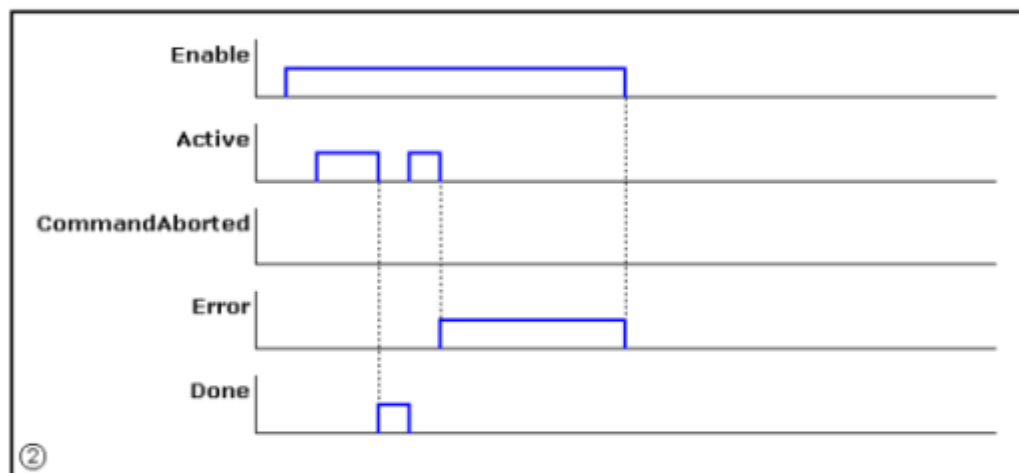
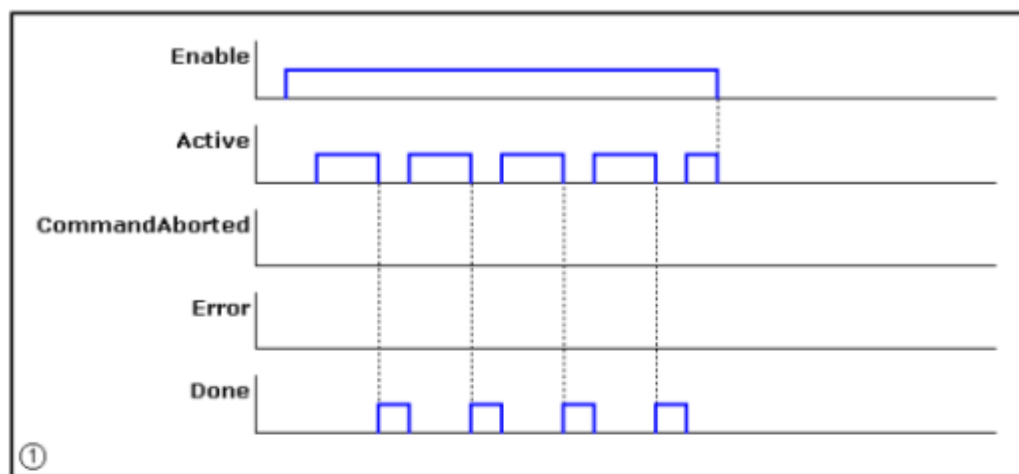
To make it possible to recognize from outside whether a function block is state-controlled or edge-controlled, two different variable names are used to activate the function block:

- The "Enable" activation input is used at function blocks which work in state-controlled form.
- The "Execute" activation input is used at function blocks which work in edge-controlled form.

Standard outputs at function blocks

- The "Done" output signals that a function block has successfully executed its task and possibly provided data are valid.
- Each function block has a specific target, i.e., a task (e.g., communication, closed-loop control, motion control etc.). As long as a function block is working on its actual task, this is to be signaled via the "Active" output. Any preprocessing is not signaled with this output.
- If the processing of a task could not be successfully completed, this is signaled by the "Error" output.

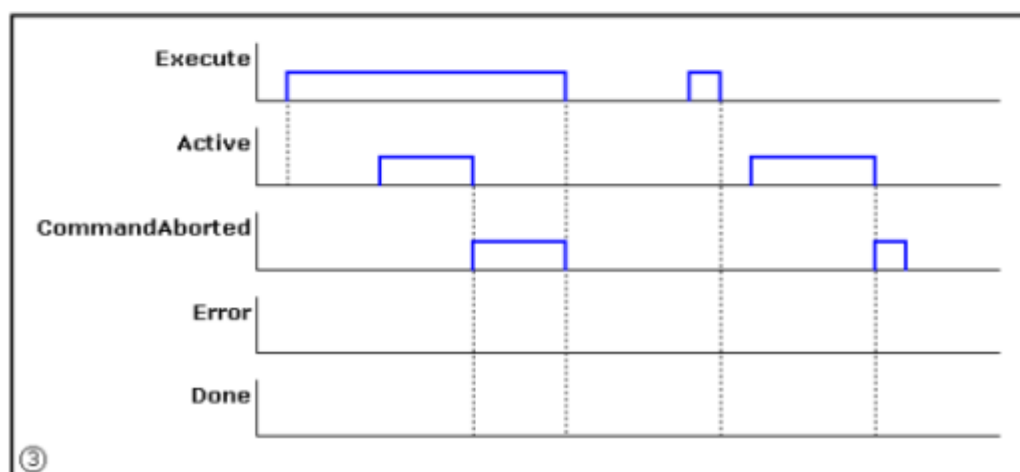
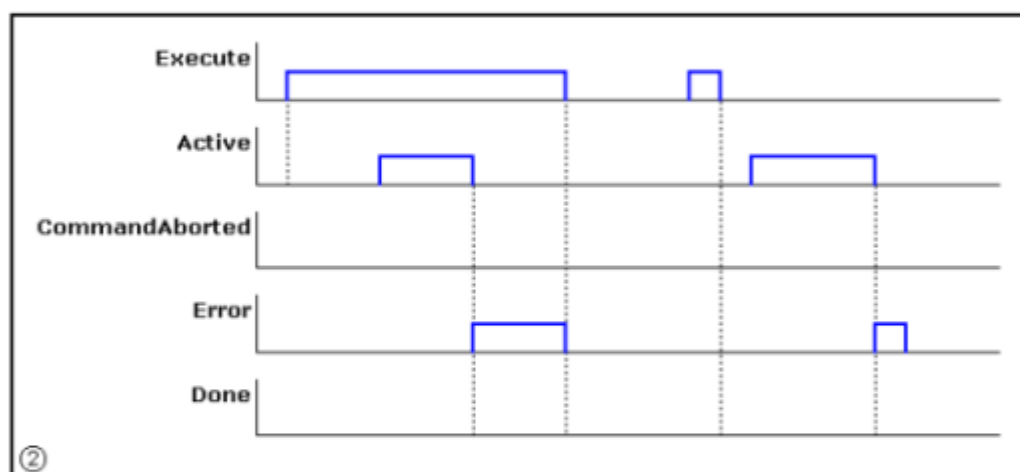
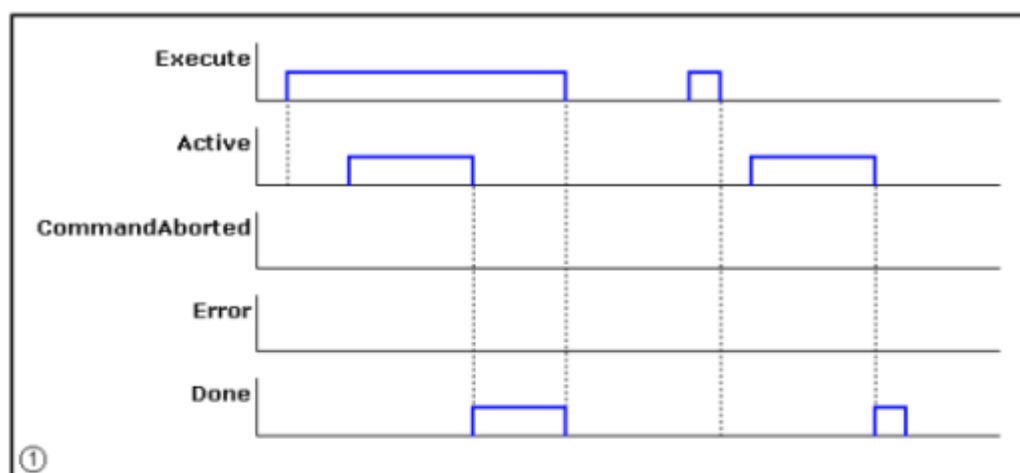
Signal-time diagram for state-controlled function blocks



- 1 State-controlled function block processing successfully completed
- 2 State-controlled function block processing aborted with error

- 3** State-controlled function block processing interrupted
Fig. 8-16: Signal-time behavior of state-controlled function blocks (with "Enable" input); for single axes, the "Active" output is not set!

Signal-time diagram for edge-controlled function blocks



- 1 Edge-controlled function block processing successfully completed
- 2 Edge-controlled function block processing aborted with error

- 3 Edge-controlled function block processing interrupted
 Fig. 8-17: *Signal-time behavior of edge-controlled function blocks (with "Execute" input); for single axes, the "Active" output is not set!*

8.6.10 Error handling

Error handling for Bosch Rexroth function blocks has been standardized; it consists of three diagnostic levels:

- Level 1: When it is TRUE, the output "Error" signals that an error has been reported. "Error" is BOOL type.
- Level 2: An enumerator ("ErrorID") provides basic information about the error (basic classification).
- Level 3: For detailed information, there is a structure ("ErrorIdent") in which the origin of the error and the exact error code can be read.

See also "[IndraMotion MLD error handling](#)"

8.6.11 Version assignment of libraries

Version function IndraMotion MLD provides the option of creating internal and external libraries. Internal libraries are programmed completely in an IEC 61131 programming language, while external libraries are implemented in the firmware.

The fact that the program code of a POE for an external library is linked during download via the POE identifier can be used for linking the POE and thereby the library to a specific IW package.

The version assignment, the IW package linking and the keeping of a history are made available for Bosch Rexroth libraries by a standardized version function in a consistent folder structure.

Message during download When a firmware library is included in a user project, all functions/function blocks contained in the firmware library are linked during the download. (POE linking and version protection might possibly not have been implemented on all targets.)

If one or more functions cannot be linked, the download is aborted with an error message.

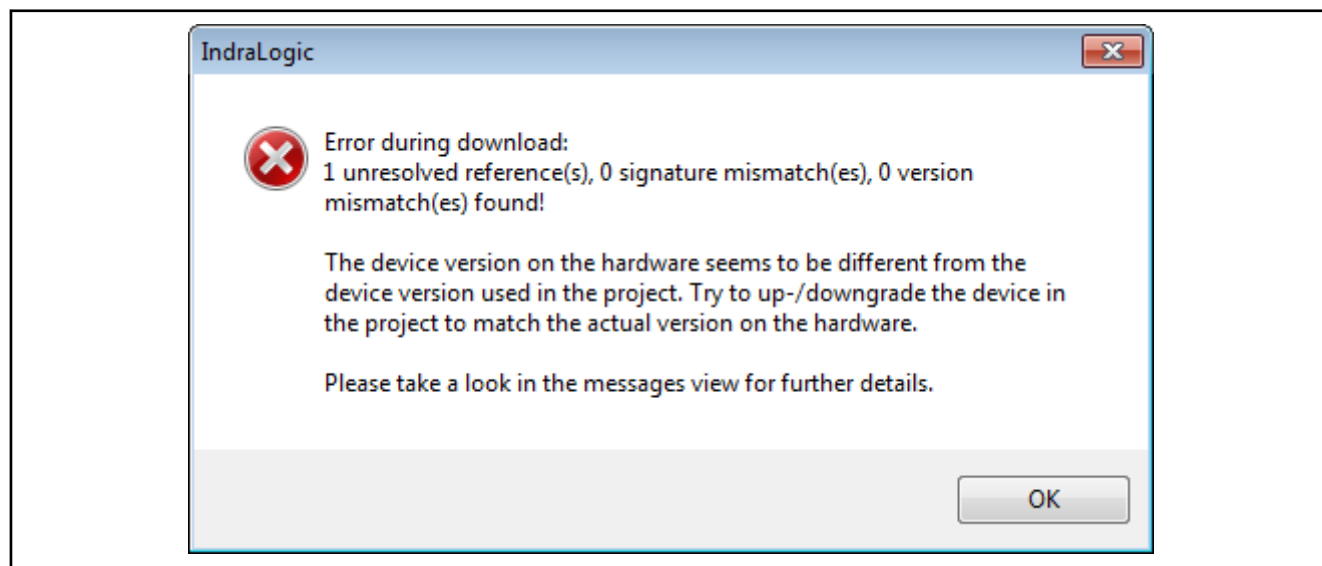


Fig. 8-18: *Message of a failed version check*

The error message contains a list indicating which functions/function blocks from which library could not be linked. A reference list (e.g., from the Firm-

ware Release Notes) helps finding out which libraries and functions/function blocks are made available in which firmware.

8.6.12 Code optimization

Requirements on the real-time behavior of the application

Drive control and drive-integrated PLC share the computing power of the drive, the drive control taking up a large part of the computing power of the drive. To use the available computing time in the best possible way, it is first necessary to check which demands are made on the real-time behavior of the application:

- Which functionalities have to be called in certain time intervals or in the case of certain events?
- What happens if a function is not completed at the desired point of time?
- Is it necessary to use a task in synchronism with master communication or a task in synchronism with CCD²⁾?

Task system

The first approach to the task configuration of the application results from the requirements on the real-time behavior of the application. Generally, several tasks should be used to separate functions with long runtimes from functions which must run in real time. Each task requires computing time to manage and control the task system. The more tasks have been configured, the more computing time must be spent for the management of the tasks.

Generally, only the necessary code should be processed in time-critical tasks. Too short intervals should not be used for the tasks.

Using tasks of different velocities only makes sense if relatively few actions have to be processed very fast and other actions run in a slower task (e.g., a task with $t\#1\text{ms}$ and a task with $t\#10\text{ms}$).

Besides, it is necessary to consider whether a watchdog already has to be triggered when a task is exceeded once, or whether the watchdog can be more tolerant.



As an alternative to a multi-task system, all fast actions can at first be carried out in a task and then only one slower action at a time (CASE instruction).

However, if the slower actions require more computing time, they have to be carried out in a separate task.

String processing

The processing of strings is relatively slow. If possible, it is to be avoided in the case of rapid task cycles.

IEC language selection

The programming language "Structured Text" (ST) is recommended for performance-optimized programming. "ST" also provides clearly structured programming.

Program code

- Bit processing is clearly slower than byte processing. This applies to addressed variables and bit access in words (e.g., $\%IX0.0$ inputs, $\%QX0.0$ outputs, flags or bit access).
- Division of integer types is clearly slower than REAL division.

I/O configuration

The I/O mapping to BOOL variables (8 bits long) is clearly faster than bit variables (e.g., $\%IX0.0$).

Using IEC constants

Using `VAR CONSTANT` is faster than using constant values in the form of normal variables. Variable addressing is not required, the value is directly used.

Parameter access

- Do not access parameters more often than necessary.

²⁾ Task in synchronism with CCD is only possible with MLD-M

	• Process string parameters in a separate task, because this requires a lot of computing time. • Access to parameters via direct variables is significantly faster than functions. • Access to parameters via functions is faster than function blocks. • Accessing local list parameters (Axis1) requires a relatively long computing time and should either be outsourced to a slower task or carried out element-by-element with "MX_fReadParamDINT()" / "WriteParamDINT()".
Functions / function blocks	• If the same function blocks or functions are used several times in a task cycle, they should be called in direct succession, if possible (cache effect!). • Only call function blocks as long as necessary - especially cyclic function blocks. • A constant value can already be assigned to a function block parameter at declaration. • Outputs of function blocks can be directly read, it is not necessary to assign them to a variable.
Check function	The check function detects and avoids incorrect access. The checks are run for pointer access, array access, range limits, division by 0... When the frequently processed code has been sufficiently checked, this automatic check (implicit check function) can be switched off. <i>Example:</i> Switching off the implicit check function In this case, runtime checks are not run within the POU: <pre>{attribute 'no check'}</pre> <pre>FUNCTION MyFunc : INT</pre>
Structured programming	• Using subfunctions or subfunction blocks only provides higher performance if these functions/function blocks are used several times. • By means of CASE instruction, different code branches can be systematically processed according to the status of the functionality or application. • Transmission parameters / return values to functions and function blocks should be as compact as possible. In the case of large data quantities, access the data in global form, or transmit them via VAR_IN_OUT or REFERENCE.

8.7 VCS – Version Control System

Using the "Version Control System" (VCS), IndraWorks allows for working with versioned projects stored on a team server. Several IndraWorks users can work at such versioned projects simultaneously.



Information on the installation and set-up of team servers to be used with IndraWorks is available in the IndraWorks Engineering Application Manual.

With IndraWorks 14V14 and above, the "Version Control System" can also be used for versioning and archiving MLD projects. The following functions can be carried out:

- Projects can be processed exclusively or non-exclusively
- Partial projects can be separated
- Local changes can be combined, coordinated with changes in the team project

In addition, the "Version Control System" provides the following options:

- Recording of all changes (who, when, why)
- Changes can be undone
- Project versions can be stored
- Differences between project versions are displayed
- Old project versions can be restored

A new team project can be created by calling the **Version management ► Create team project...** menu item from the context menu of the project branch.



All directories and files provided with a lock are assigned to the "Version Control System". With closed locks, the directories and files are write-protected. Directories and files with open locks can be written.



More detailed information on the "Version Control System" (VCS) is available in the IndraWorks Engineering Application Manual.

9 Diagnostic and service functions

9.1 Overview and introduction

This section describes the diagnostic and service functions that can be primarily used for troubleshooting. The options of free PLC programming and using self-contained technology functions will be explained.

Generally, the following classification is made:

- Standard drive diagnostic functions
- MLD diagnostic and service functions
- General service functions

9.2 Device editor

9.2.1 Device editor, general information

Among other things, the device editor provides tab pages for configuring a device.

The "**device editor**" can be opened in the Project Explorer via the context menu entry **Open** of a control unit [e.g., drive controller with integrated PLC (IndraMotion MLD)]. When IndraMotion MLD V14 is used, the following tab pages are contained:

- [Status](#)
- [Files](#)
- [Task deployment](#)
- [Log](#)
- [PLC settings](#)
- [PLC shell](#)
- [Information](#)



The "Log" and "PLC shell" tab pages can be switched on and off:
Tools ► Options ► IndraLogic 2G ► General Settings.

9.2.2 Status

The "Status" tab page shows status information of the respective device and of the internal bus system.

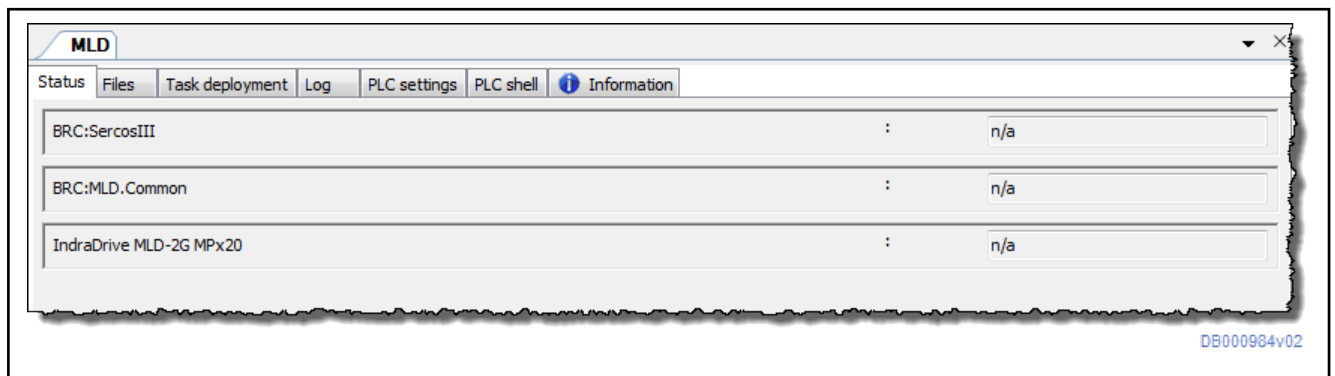


Fig. 9-1: Device editor: Status

9.2.3 Files

The "Files" tab page is used to exchange files between the PC (host) and the external memory card of the drive (IndraLogic runtime system).

It is only possible to access the "User" directory of the external memory card of the drive. There, directories / files may be deleted or directories created to store files belonging to the project, for example.



From an MLD program, the "User" directory can be accessed with the "SysFile"/"SysFileAsync" and "SysDir" libraries.

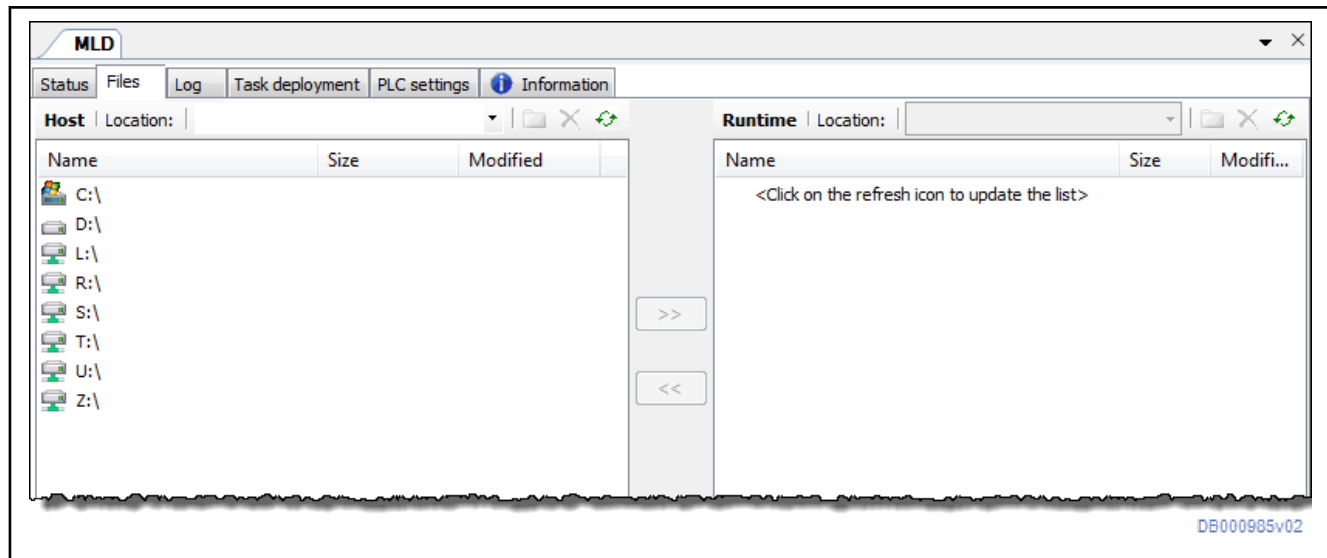


Fig. 9-2: Device editor: Files

9.2.4 Task deployment

After code has been generated for the application, the "Task deployment" tab page shows the inputs and outputs with their assignments to the defined tasks.

The tab page supports the troubleshooting as it shows where inputs or outputs are used in several tasks with different priorities. Multiple use may lead to undefined values due to overwriting.

9.2.5 Log

The "Log" tab page is used to display the logbook of the control unit, i.e. to display events that were recorded on the target.



The tab page can be switched on and off via **Tools ► Options ► IndraLogic 2G ► General Settings ► Enable PLC logger**.

The display shows:

- Events when starting and stopping the system (loaded components with versions)
- Application download and loading of the boot project
- Customized entries
- Log entries from I/O drivers
- Log entries of the DataServer

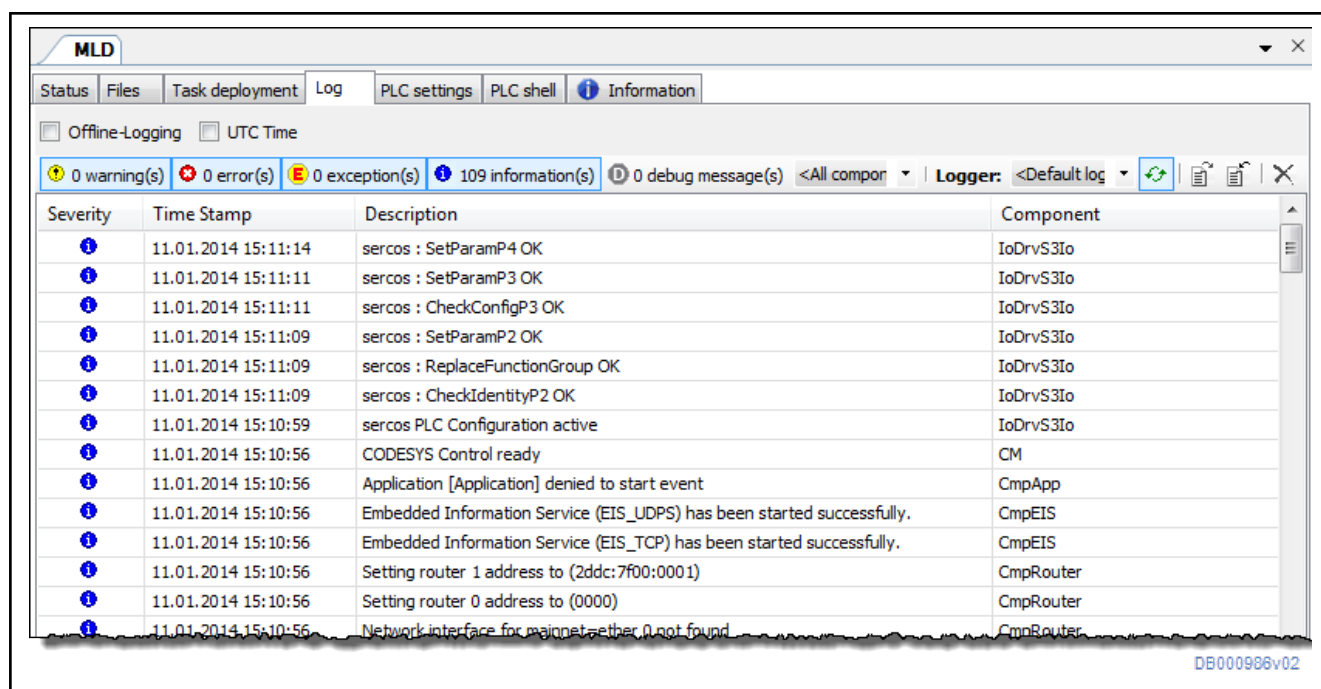


Fig. 9-3: Device editor: Log

A log entry is displayed with the following information:

- **Severity** (importance): There are four categories for the severity of the event: warning(s), error(s), exception(s), information. The display of each category can be shown or hidden via the corresponding button in the bar above the list. The respective number of log entries is displayed on the button for the respective category.
- **Time stamp**: Date and time, e.g. "30.06.2012 04:22"
- **Description**: Description of the event, e.g. "Import function failed of <CmpFileTransfer>"
- **Component**: Name of the respective component.
- **Com Name** (component name): An individual component can be selected in a drop-down list to display only related log entries. The default setting is "All components".
- **Logger**: The selection list contains the available recordings. The standard setting is the "<Default Logger>" preset by the target, at present identical to "PlcLog" for the IndraLogic runtime system.



- The list of logbook entries can be refreshed using the  button.
- The content of the list can be exported to an XML file. Click  to open the standard dialog to save a file. The file filter has been set to "xml-files (*.xml)". The list of logbook entries is stored in the selected directory with the specified file name. When the "Offline-Logging" option has been activated, the actions which do not refer to the connection to the control unit are recorded, too. Currently, this has only been implemented for the Safety version of the programming system.

9.2.6 PLC settings

The "PLC settings" tab page is used to specify how the PLC behaves in the stop state.

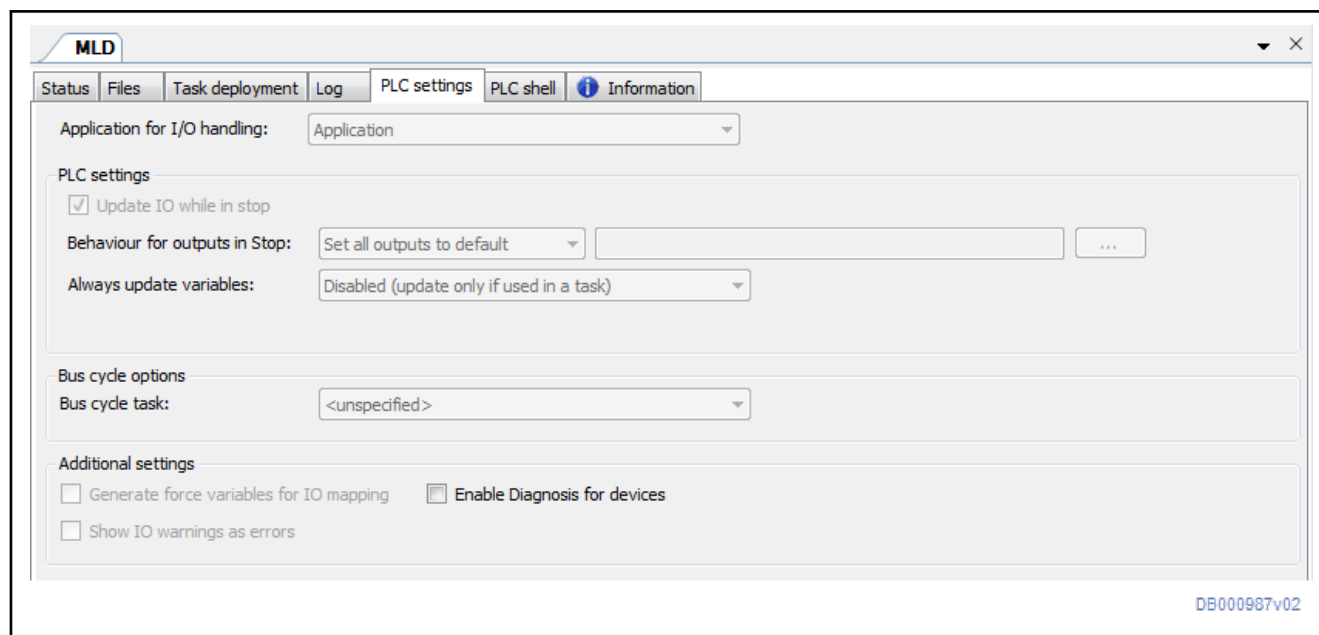


Fig. 9-4: Device editor, PLC settings

Application for I/O handling	If there are several applications available for the device, they are provided here in a selection list. The default application, which is automatically created with a default project, is always entered first.
PLC settings	Update I/Os while in stop: When this option has been activated (default), the values of the input and output channels are updated, even when the PLC goes to the stop status.
Behavior of the outputs in stop	The selection list provides the following options of how the values of the output channels are to be handled when the control unit goes to the stop status: • Keep current values: The current values are retained. • Set all outputs to default: The standard values resulting from the I/O image are assigned. • Execute program: The handling of the output values can be controlled using a program contained in the project. The name of the program can be specified here, and it will be executed when the control unit goes to the stop status. The "..." button opens the input assistant which can be used to facilitate the program selection.
Bus cycle options	Bus cycle task: The selection list provides all tasks defined in the task configuration of the active application (e.g. "MotionTask", "PlcTask" etc.).



Check the data of the tasks of your application and then enter the desired task.

Select a specific task which is to control the bus cycle or select the "<unspecified>" setting, when the task with the shortest cycle time, i.e. the fastest task, is to be used for this purpose.

9.2.7 PLC shell



The "PLC shell" function is used for diagnostic and debugging purposes. If you need to use this function, please contact our service department.

The tab page can be switched on and off via **Tools ▶ Options ▶ IndraLogic 2G ▶ General Settings ▶ Enable PLC logger**.

9.2.8 Information

The "Information" tab page shows general pieces of information on the device:

Name, vendor, groups, version, model number, description,... Normally, an image of the device is displayed.

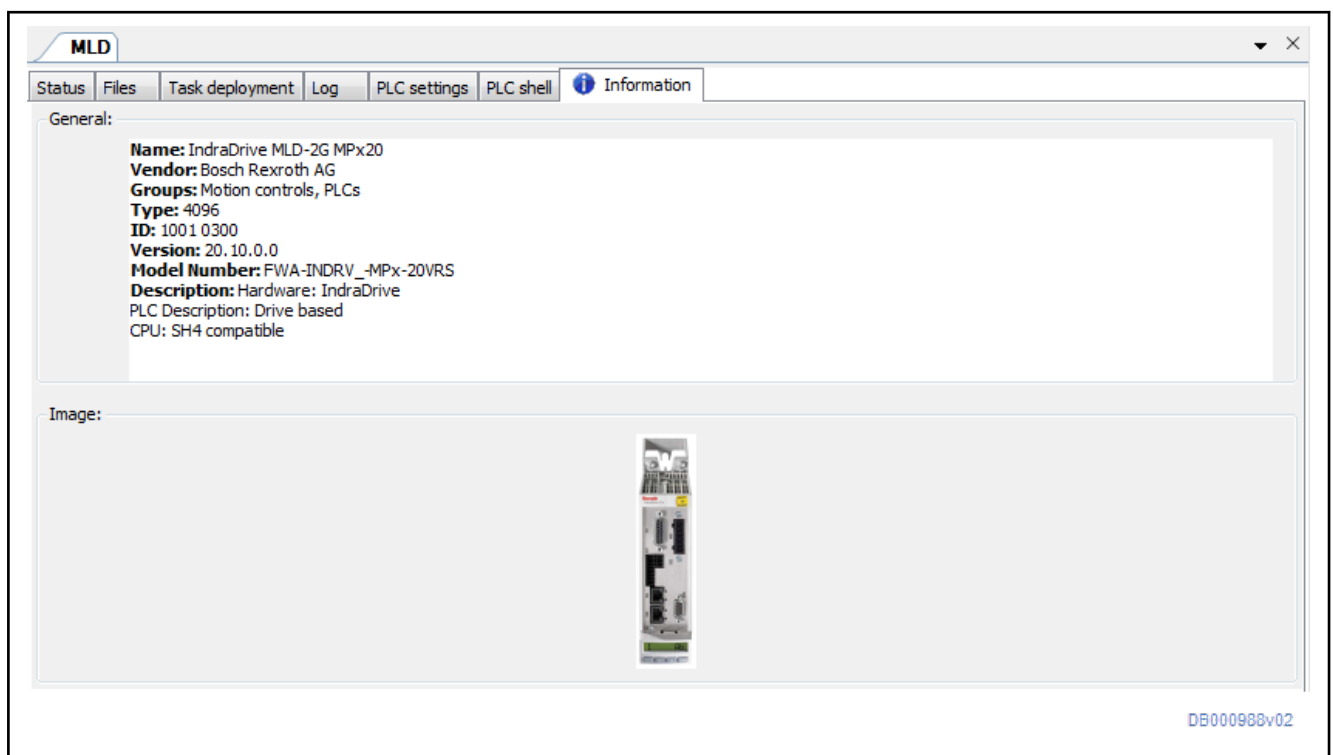


Fig. 9-5: Device editor, Information

9.3 Diagnostic functions

9.3.1 Standard drive diagnostic functions

Overview

For diagnosing the drive behavior and the integrated PLC, the following functions are available in the drive:

- **Oscilloscope function**

The oscilloscope function can be used to record drive-internal and external status variables (parameter contents).

It can be effectively used both for initial commissioning and trouble-shooting. Its functionality can be compared to that of a 4-channel oscilloscope.

The oscilloscope function supports different measuring modes in which you can carry out measurements. These modes are "Single Shot" and "Trending".

With "Single Shot", the recorded measured values are transmitted to the oscilloscope **subsequent to the measurement**.

The "Trending" measuring mode is the typical oscilloscope operation mode. In this mode, the measured values recorded by the device are **continuously** transmitted to the oscilloscope.

Related documentations on the oscilloscope function:

- Functional description of firmware: "Oscilloscope function"
- IndraWorks online help: "Oscilloscope function"
- or -

Printed edition of the Application Manual for "Rexroth IndraWorks" (document typecode "DOK-IWORKS-ENGINEE*V_{xx}-APRS" ["_{xx}" represents the respective IndraWorks version, e.g. "13"])

In addition, PLC variables of an MLD user program can be recorded with the oscilloscope function (see "Recording PLC variables with the oscilloscope function").

- **Analog output**

The "analog outputs" drive function allows analog signal values to be used for commissioning and optimizing drives with appropriate measuring devices (e.g., oscilloscope, multimeter), as well as for visualizing the contents of drive parameters (see Functional description of firmware "Analog outputs").

- **Patch function**

The patch function allows internal storage locations and PLC variables to be read (see Functional description of firmware "Patch function").

- **Error memory**

In the drive, all occurred errors, together with the corresponding count of the operating hours counter, are collected in an error memory on the control section (see Functional description of firmware "Error memory (power section and control section)").

9.3.2 MLD diagnostic functions

Overview

In conjunction with Rexroth IndraMotion MLD, there are other PLC-specific diagnostic functions available (for detailed descriptions of the functions, see the respective chapter in "PLC Program Development with Rexroth IndraLogic 1.0"):

- **Flow control**

When "flow control" has been activated, parts of the program executed during the last control cycle are highlighted.

- **Sampling trace**

Sampling trace allows the time flow of PLC variables to be exactly traced.

- **Watch and recipe manager**

The watch/recipe manager allows variables to be loaded and stored in summarized form.

- **Visualization**

PLC variables can be graphically represented or changed via the IndraLogic user interface.

- **Monitoring functions**

Both the runtime and the available resources are monitored in Rexroth IndraMotion MLD.

- **Simulation**

The PLC logic, for example, can be tested using simulation.

- **Debugging/troubleshooting** (breakpoints, watch, single step, writing, forcing)

Flow control

When "flow control" has been activated, parts of the program executed during the last control cycle are highlighted.



When flow control is used, take into account that it generates additional PLC code which strongly influences the processing velocity.

Sampling trace

Function Sampling trace allows the time flow of PLC variables to be exactly traced. (Direct variables are not recorded. If necessary, they should be copied to a flag that can be recorded.)

Use Observe the following aspects for use:

- If the selected sampling rate in the trace configuration is smaller than the PLC task interval, the scaling of the X-axis does not match the displayed data!
- What makes sense is a sampling rate of "0" (default=1 sampling operation per PLC cycle) or integral multiples of the task cycle time.



With sampling rate "0", there is 1 sampling operation per PLC cycle! Useful minimum sampling rate = PLC interval [ms].

Watch and recipe manager

The watch and recipe manager allows the values of specific variables to be displayed. It is also possible to preset variables to specific values and transmit them altogether to the control unit (see also documentation "PLC Program Development with Rexroth IndraLogic 1.0").

Visualization

During start-up, the integrated visualization is well suited for graphically representing PLC variables or change them via the IndraLogic user interface (e.g., when the inputs have only been wired in part). To illustrate project variables, IndraLogic makes available the option of visualization. This allows convenient operation and display dialogs to be generated for PLC projects (see also documentation "PLC Program Development with Rexroth IndraLogic 1.0").

Monitoring functions

In Rexroth IndraMotion MLD, both the runtime and the available resources are monitored in order to detect when they are inadvertently exceeded.

In the case of unauthorized access, a PLC exception is generated. These errors cause the drive error F6010 with axis deceleration (see "[IndraMotion MLD error handling](#)").

Simulation

In the simulation mode, the connection to the drive controller is not established. When logging in, the PLC is (partly) simulated on the PC. In this case, there are no real time and no firmware function blocks. The PLC logic, for example, can be tested using simulation.

Debugging/troubleshooting

For debugging and testing the created PLC programs, a debugger on source code basis has been integrated in IndraLogic. This allows breakpoints to be set and the tests to be carried out in individual steps. Variables can be displayed or overwritten.



What makes sense is a counter in each task. In this way it is possible to easily monitor whether the code is processed. Besides, it shows how often and how fast a task is running.

There are the following options available for debugging:

- **FlowControl** displays the parts of the program that are run through. "FlowControl" greatly slows down the PLC program!
- **Force lists** are used to permanently overwrite variables (e.g., simulation of a hardware switch). Force lists are cleared with "Logout", "Reset" and "Download".

Tracing PLC exceptions

General information

For PLC exceptions, such as division by zero or exceeding of the range, it is advantageous to be able to quickly find the corresponding position in the PLC source code.

Debugging after commissioning

After commissioning, the error "F6010 PLC runtime error" is generated when a PLC exception occurs. In this case, the programming interface jumps to the faulty program code and highlights it with a color. The parameter P-0-1365 contains information on the cause of error and on the affected POU (Program Organization Unit) (this information is also displayed in IndraWorks in the "Messages" window).

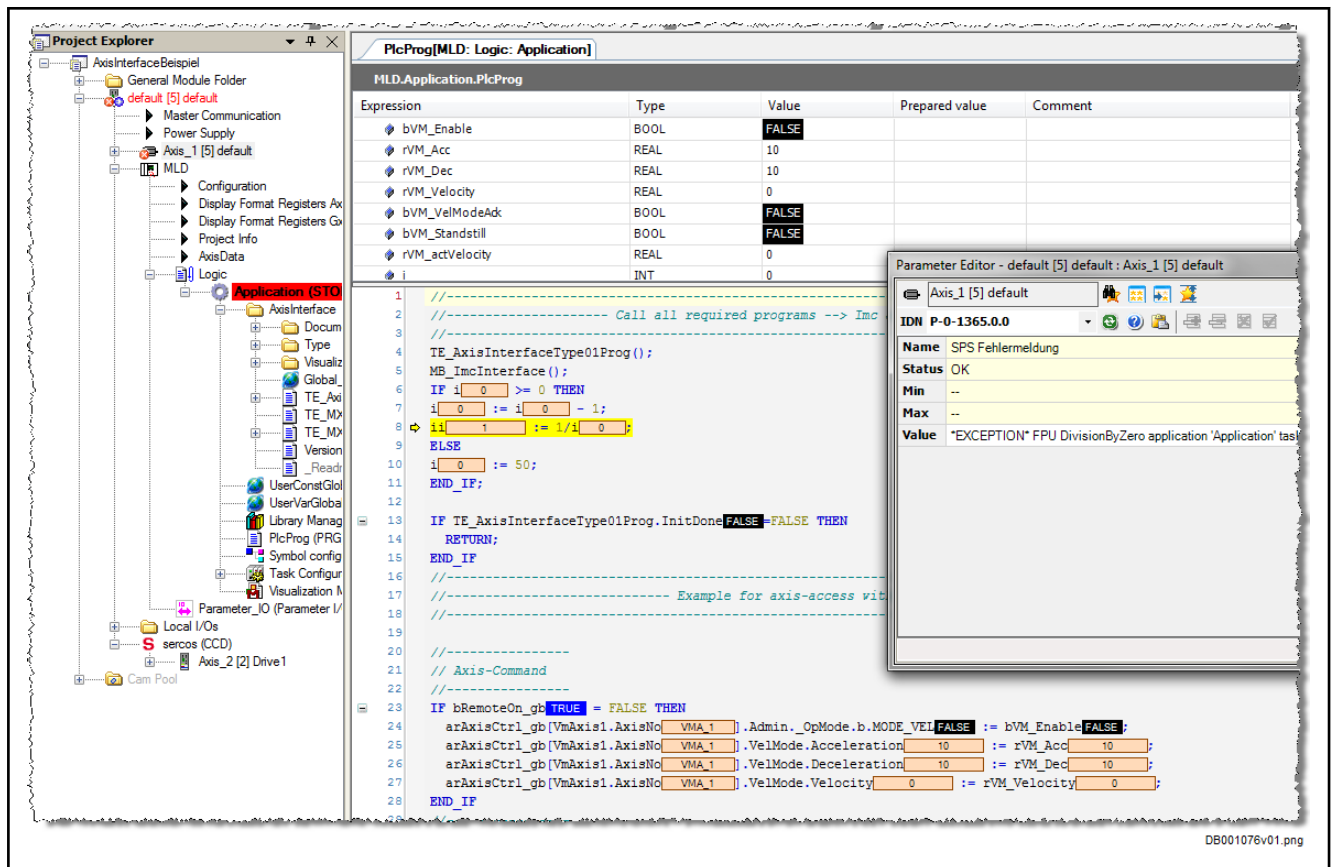


Fig. 9-6: Displaying a PLC exception after commissioning

How to proceed after an exception without IndraLogic at the drive

The parameter P-0-1365 contains the address information (can be read by accessing any parameter).

Program:

```

getp P-0-01365
SPS Fehlermeldung 126 Chars
= 'Id:000007E7 Hex; Time:000027BF s; Par:00000000 Text:WORD zero division; Pou:215; Of:232; Task = MyTask1;

```

With the POU number it is possible to identify the POU in which the error has occurred. If this POU is called by multiple tasks, the task in question can be evaluated. The offset ("Ofs") can be used for finding the program line. In IndraLogic, the corresponding function block can be retrieved with the second number in brackets at the POU's.

To find the code line, open the compile file <project name>.bpl and search the breakpoint list of the concerned function block. To do this, first search the function block with the help of its name or its number (in brackets). The list of the function block contains the line numbers with address offset and POU number. Convert the offset (232 in this case) to hexadecimal syntax (00E8 in this case) and search the line with the same or next higher offset (line 10 in this case). With this line number you can retrieve the position in IndraLogic.

ZeroDiff.bpl (compile file):

Program:

```

MYFB1 (0) (201): 0010-0134
Breakpoints:
0001 0010[215]
0002 001c[215]
0003 0028[215]

```

```

0004 0034 [215]
0005 0040 [215]
      0018 [217]
0006 0064 [215]
      0018 [217]
0007 0092 [215]
      0018 [217]
0008 00BE [215]
0009 00CA [215]
0010 0124 [215]
0012 0134 [215]
...

```

-> man erkennt die betroffenen Zeilennummern 9 bzw. 10

9.4 Service functions

9.4.1 Service functions of the PLC (IndraMotion MLD)

Service functions in IndraWorks (IndraLogic)

IndraWorks or IndraLogic provides the following service functions:

- **"Download" / "Online change" / "Load boot project"**
see ["Loading the IndraLogic project to the drive"](#)



Before leaving the installation, it is necessary to make sure that the boot project is updated, so that no outdated project is loaded after a restart.

- **"Clean all" / "Rebuild all"**
"Clean all" clears information of the last download and the last compilation process.
In contrast to (incremental) "Build", the complete project is recompiled with "Rebuild all". When this is done, the download information, however, is not deleted as it is the case with the "Clean all" command. Please note that you can exclude objects from compilation.
- **Reset**
 - "Reset" [corresponds to "Reset (warm)"]: Resets the program; retain variables retain their values
 - "Reset (cold)": Resets the program; retain variables are initialized
 - "Reset (origin)": Program and boot project are deleted

Stopping IndraMotion MLD (emergency mechanism)

Motivation Since a boot project is automatically loaded and, depending on the configuration, started during the run-up of the drive, a mechanism has been implemented which can prevent the start of PLC programs.

Instructions for use/sequence Proceed as follows to prevent PLC programs from starting while the drive is running up:

- In the boot-up phase, before Boot 2.9 is displayed at the control panel, the "ESC" and "ENTER" keys have to be simultaneously pressed and held at the control panel.
- The automatic start of a PLC boot project is prevented and the display reads "PLC ?".
- By pressing the arrow keys (arrow down or arrow up) the display switches between "Run PLC" and "Stop PLC".

If "Stop PLC" is confirmed with "ENTER", no project is loaded. If "Run PLC" is confirmed with "ENTER", a stored project is started or loaded.

9.4.2 Firmware replacement

Overview and introduction

There is a difference between a firmware release update and a firmware version upgrade when replacing the firmware

Note the following when replacing the firmware:

- For firmware release updates, the device version is updated, if necessary; it contains the libraries of old releases.
- Not every new firmware contains new or modified libraries. If there are no new or modified libraries, no new IndraWorks package is provided.
- Any new firmware version or a different characteristic includes a new IndraWorks package (examples: MPB17→MPC17, MPB17→MPB18)

Firmware release update

General information

With a firmware release update, a firmware of the same version is brought to a new release status (e.g., MPB17V08 to MPB17V10).

The general procedure for carrying out a firmware release update is explained in the Functional description of the firmware in the "Firmware replacement" chapter.



The paragraphs below only describe the MLD-specific behavior and procedure.

Use and content of Firmware Release Notes

The "Firmware Release Notes" documentation provides information on firmware versions and their compatibility.

The resolved defects relevant to Rexroth IndraMotion MLD are contained in the "Drive Firmware / Drive PLC" subchapter of the Firmware Release Notes for the respective firmware version (e.g., MP*18VRS).

Instructions for use

Note the following points:

- All parameters are retained with a firmware release update. This means the drive controller is ready for operation after replacing the firmware without any additional action required.
- A firmware release update contains resolved defects or compatible changes/functional enhancements.
- In general, the PLC programs that were compiled for a firmware are also operable with a higher firmware release. The project itself is stored in the parameters and is therefore retained.
- Projects can also be created for an old release by using the "old" suitable libraries.
- To use new functions (compatible functional enhancements), an IndraWorks package is made available that basically contains the new libraries.
- IndraWorks packages can be installed and managed with IndraWorks (see ["Device Update with IndraWorks" on page 287](#)).

Compatibility within a single version

With the general sales of a firmware version, the IndraWorks package is made available with IndraWorks MLD with its libraries for this version.

With IndraWorks MLD, the PLC projects are automatically created in such a way that they are compatible with the firmware which is used.

Rexroth IndraMotion MLD also provides version management, i.e., a distinction is also made with regard to compatibility:

- **Binary upward compatibility:** As soon as the firmware is in general sales, the PLC programs created with an old release still can be used in unchanged form. They are binary upward compatible.
- **Downward compatibility:** Within a version, the compiled projects are normally binary downward compatible as well. It is thereby possible to load a PLC project to an old release of the firmware.



All incompatibilities are checked and detected.

IndraWorks package

The IndraWorks package contains information on the device version and on all the required libraries.

- The target name contains information on device, characteristic and version (example: "MldStdDevice_V18.5.4.0.iwpackage").
- Not every new firmware does also contain library changes so that a new IndraWorks package is not always available.



If an existing project is to be adapted to a different firmware, this can be achieved with IndraWorks.

Firmware conflict

If a project does not match the firmware release, the corresponding message is output. This can happen in the following cases:

- When **old firmware** is to be edited with a **new IndraWorks package**, an incompatibility of the libraries is detected and causes a message during download or prevents the automatic loading of the program. The latter causes the error F6010.
- An internal check detects the conflict, an error message is output.

Corrective measure

To solve the problem, the following steps are required:

- Check the drive firmware (S-0-0030).
- Install the IndraWorks package (see ["Device Update with IndraWorks" on page 287](#)).
- Afterwards, the project can be loaded to the drive.

9.4.3 Firmware version upgrade

General information

Firmware version upgrade

When firmware in a drive controller is to be replaced by a new firmware version, this is called a firmware version upgrade (e.g., FWA-INDRV*-MPB-17V12-D5 is to be replaced by FWA-INDRV*-MPB-18V06-D5).

The general procedure for carrying out a firmware version upgrade is explained in the "Firmware Release Notes" documentation and in the Functional description of the firmware in the "Firmware replacement" chapter.



The paragraphs below only describe the MLD-specific behavior and procedure.

Use and content of Firmware Version Notes

The "Firmware Version Notes" documentation provides information on firmware versions and their compatibility, as well as differences compared to the previous version.

The changes relevant to Rexroth IndraMotion MLD are to be found in the subchapters "Notes on application"->"Incompatible new functions and functional changes"->"Optional device functions", or "Notes on application"->"Compatible new functions and functional changes"->"Optional device functions" in the Firmware Version Notes for each firmware version.

Compatibility between versions



A firmware version upgrade contains a new IndraWorks package and new or possibly incompatible function blocks. For this reason, an existing project has to be recompiled before the download. In some cases, it is necessary to adjust the PLC project!

Instructions for use

Note the following when replacing the firmware:

- Any new firmware version or different characteristic always includes a new package (MLD-2G) or target (MLD-1G) (example: MPH05→MPH07).
- Starting with the general sales, the packages of a firmware version are downward compatible so that, for example, a package for MPx18V12 projects is valid for MPx18V10 projects, too. This does not necessarily apply to the prototype status of a firmware!
- It might be necessary to install multiple packages if projects for different firmware versions are to be created.

Device Update with IndraWorks

For MLD-2G there are different releases of the firmware and of the devices with their libraries.

The latest version of the device can normally be operated with a firmware; it is not necessary to make any settings.

When, during the download of the PLC program, IndraWorks displays an error message pointing to a device that is out of date, the device version must be changed; if necessary, a device version (IW package) must be subsequently installed.

Changing the Device Version

1. In the Project Explorer, call the context menu at the **MLD** branch and select **Change IndraLogic Device Version**.
The "Change IndraLogic Device Version" dialog is called.
2. Select the new device version via the drop down list of "New device version".

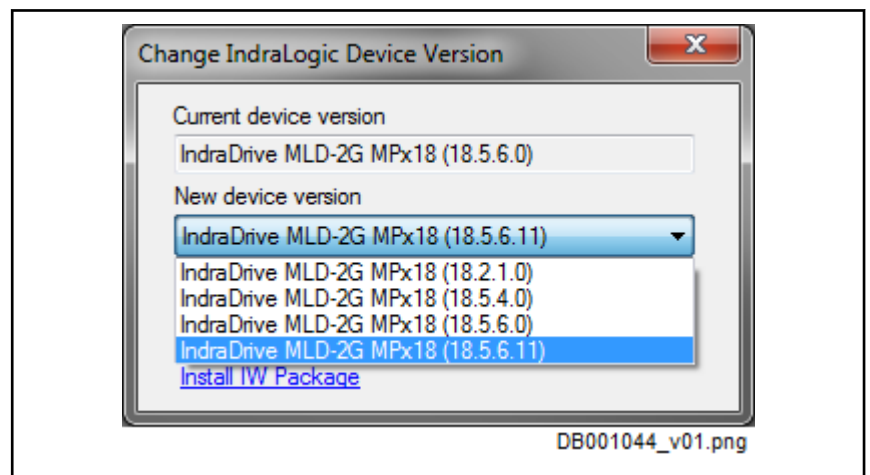


Fig. 9-7: Selecting the Device Version

3. Click the **Change Device Version** button.
4. After the device version has been changed, the project should be cleaned:

In the main menu, execute **Build ► Clean all**.

5. The cleaned project must be rebuilt:

In the main menu, execute **Build ► Rebuild all**.



The cleaned and rebuilt project will be loaded to the control unit with the next "Login".

An IndraWorks installation normally contains all the device versions required for the configuration of IndraMotion MLD; i.e. the device version is contained in the "New device version" drop down list of the "Change IndraLogic Device Version" dialog.

If a device is not available, it can be subsequently installed via the installation of an IW package. The current IW package is made available via the installation of a new IndraWorks version. In special cases (bug fixing in a version), the IW package can be procured from the service department.

Subsequently Installing a Device Version

A new device version can be added by installing a so-called IW package.

1. In the Project Explorer, call the context menu at the **MLD** branch and select **Change IndraLogic Device Version**.

The "Change IndraLogic Device Version" dialog is called.

2. Click the "Install IW Package" link:

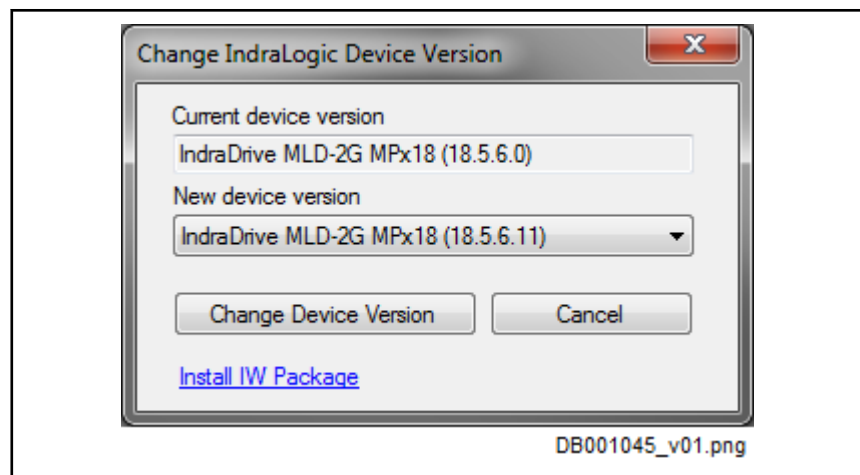


Fig. 9-8: Subsequently Installing a Device Version

The "Open File" Windows dialog is opened.

3. Select the package to be installed. (It is only allowed to select files with the ".iwpkg" extension!)



The installation can take some time.

After the installation, the new device version is available in the drop down list for selection.

4. Select the new device version via the drop down list.
5. Click the **Change Device Version** button.

6. After the device version has been changed, the project should be cleaned:

In the main menu, execute **Build ► Clean all**.

7. The cleaned project must be rebuilt:

In the main menu, execute **Build ► Rebuild all**.



The cleaned and rebuilt project will be loaded to the control unit with the next "Login".

10 Converting projects

10.1 Introduction

This chapter describes how to convert IndraLogic 1.x projects (MLD-1G) to IndraLogic 2G projects (MLD-2G). In particular, the changes in the fields of "Motion" and "PLC" will be described in detail.

Converting IndraLogic 1.x projects (MLD-1G) to IndraLogic 2G projects (MLD-2G) is only possible with IndraWorks 13VRS or IndraWorks 14VRS.



Check the project for conversion errors after the conversion.

The system has to be commissioned again after a successful conversion.



The projects developed with firmware version MPx08 and below and projects for firmware version MPx17 / MPx18 and MPx20 have a different memory organization (byte order). The memory organization is:

Up to MPx08: little endian

MPx17 / MPx18 / MPx20: Big Endian

The change in the memory organization may cause the following problems:

- When using BYTE addressing (%QB or %IB): Re-parameterize I/O configuration (interchange %QB0 and %QB1!)
- When using WORD addressing (%QW or %IW): No change is required! Preferred addressing!
- Using DWORD addressing (%QD or %ID): If possible, replace by WORD addressing / re-parameterize I/O configuration, if applicable, and/or re-wire module.

10.2 Requirements in MLD-1G

The following requirements in MLD-1G have to be checked/preset in order to convert an existing project from MLD-1G to MLD-2G:

Monitoring runtimes of the project

IndraMotion MLD-2G requires more calculating time than the implementation of IndraMotion MLD-1G. With small projects, the performance of IndraMotion MLD-2G is lesser than the performance of IndraMotion MLD-1G. With large projects, the lesser performance is compensated by the more compact program code.

NOTICE

After MLD-1G has been replaced by MLD-2G, the unused calculating time might possibly not suffice to process the MLD program.

Before replacing MLD-1G by MLD-2G, check the runtimes of the project. With IndraMotion MLD-1G, the load should be smaller than 70%.

After MLD-1G has been replaced by MLD-2G, check again if the unused calculating time suffices to process the MLD program. With IndraMotion MLD-2G, the load should be smaller than 90%.

The easiest way to check the calculating time is by using P-0-1364. The parameter is available in both versions and shows the unused calculating time.

The "MX_IECTaskGetLoad" function block provides a more precise analysis of the available time. In the "MX_IECTASKLOAD" structure, it contains run-time information of the task:

- "rLoad": Load of the task in the last cycle in percent in relation to the cycle time
- "rLoadMax": Maximum load of the task in percent in relation to the cycle time
- "rFreeTime": Unused calculating time of the task in μ s since the last cycle until the next start of the task
- "RMinFreeTime": Minimum unused remaining calculating time of the task in μ s until its next start



MLD-1G: The "MX_IECTaskGetLoad" function block can be found in the "MX_Base" library under **Tools**.

MLD-2G: The "MX_IECTaskGetLoad" function block can be found in the "MX_PLCCopen" library under **POUs ▶ Tools**.

Checking data types

Check the use of the data types to avoid compiling errors ahead of converting the project. Check the proper use of the "INT"/"WORD" and "DINT"/"REAL" data types. The compiler of MLD-2G's runtime system checks the proper use of the data types very strictly.

Target Settings

If a project is to be used that was already developed in MLD-1G, make sure to deactivate "Trace" in "Target Settings". If "Trace" remains activated, the PLC program cannot be imported.

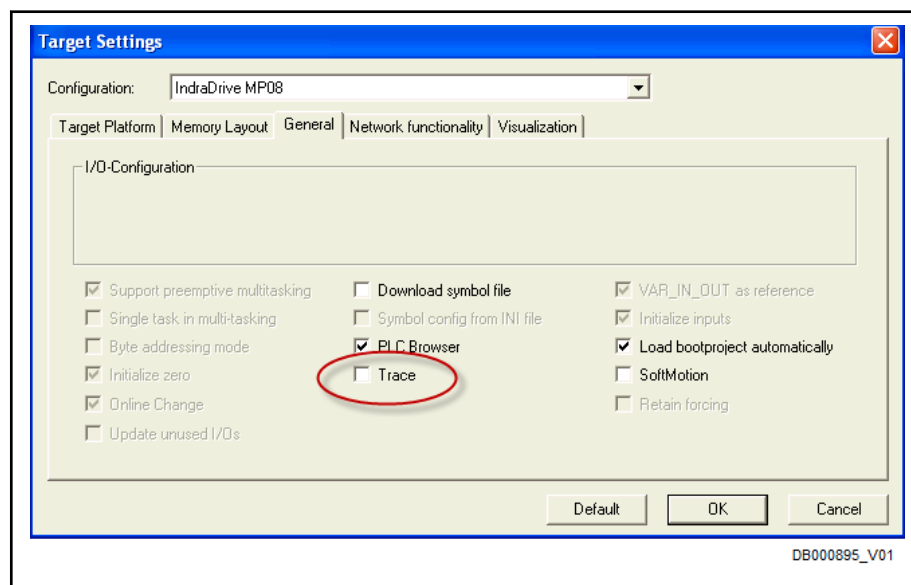


Fig. 10-1: Trace

10.3 Adding to MLD-2G

10.3.1 Adding a project

The user interface for programming and visualization of MLD-2G is integrated into IndraWorks in version IndraWorks MLD 13VRS and above.

Existing programs that were already developed in MLD-1G can be imported as follows:

1. In the IndraWorks Project Explorer, select the project into which you would like to import the MLD-1G project.
2. Expand the **MLD ► Logic** project tree and right-click on **Application**.
3. Select **Data import...** from the context menu.
⇒ The standard Windows dialog box for opening a file appears.
4. If necessary, set the file filter to "IndraLogic project file" to limit the variety of files to be opened to files with the ".pro" extension.
Select the project file you want to import.
⇒ The dialog box for selecting the objects to be imported opens.
5. Select the objects you want to import and confirm your selection with **OK**.

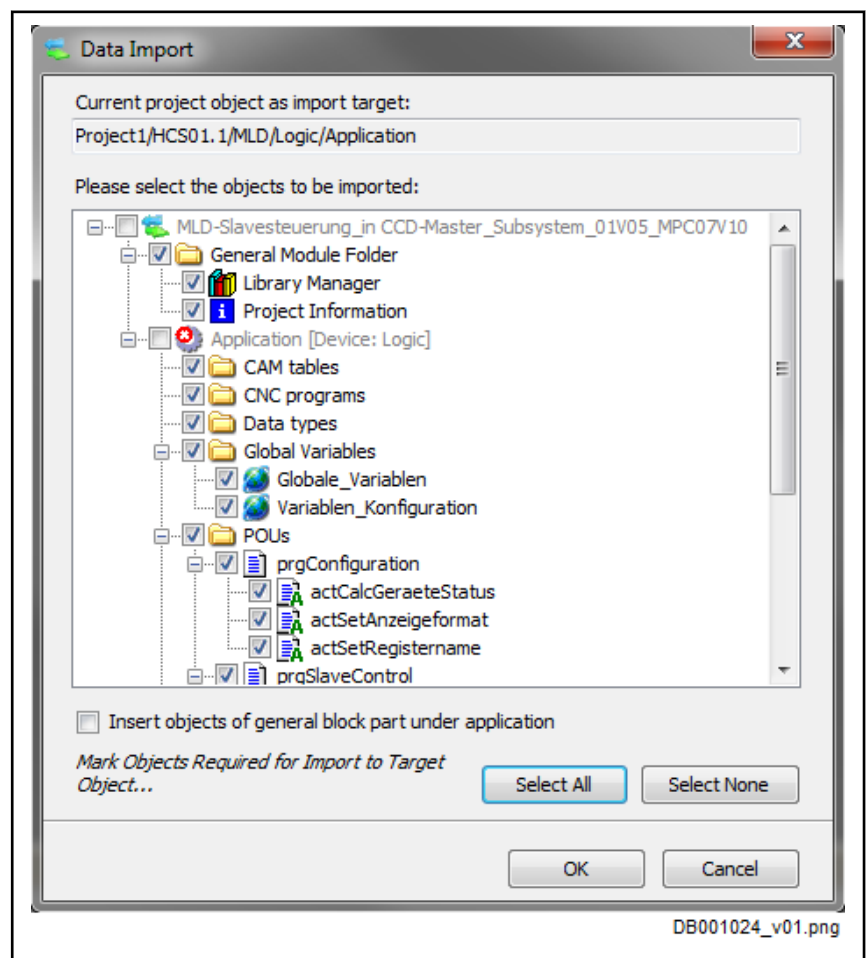


Fig. 10-2: Selecting objects for import

⇒ The MLD-1G project is converted.

10.3.2 Compiling errors after converting projects

Syntax of the Direct Variables

After porting the user program in MLD-2G, the direct variables can be changed by global "Find and Replace".

Example:

"Find and Replace"

MLD-1G: DV_P_0_1370 := DV_P_0_1371;

MLD-2G: DV_Axis[1].P_0_1370 := DV_Axis[1].P_0_1371;

Better:

DV_Axis[AXIS_1].P_0_1370 := DV_Axis[AXIS_1].P_0_1371;

Write Protection of Parameters P-0-1410 to P-0-1429

If the parameters P-0-1410 to P-0-1429 had been write-accessed in a project of MLD-1G, PLC outputs must instead be defined and written in MLD-2G.

Example:

MyOutput0 : WORD AT %QB0;

instead of DV_P_0_1410

MyOutput1 : WORD AT %QB2;

instead of DV_P_0_1411

Assigning Different Data Types

The corresponding access to the process image must be changed to writing PLC outputs.

When different data types are used, a warning is generated during the compilation run - the project is operable after the compilation.

Example:

Program:

```
diLagesollwert2 :DINT;
rLageistwert2 :REAL;
rLagedifferenz2 :REAL;
```

Warning is generated:

rLagedifferenz2 := diLagesollwert2 - rLageistwert2;

Correct:

rLagedifferenz2 :=

DINT_TO_REAL(diLagesollwert2) - rLageistwert2;

Remove the warning by converting data types, e.g.

- INT_TO_WORD
- DINT_TO_REAL etc.

Calling function blocks or their methods with the function block type

If function blocks or their methods are called with the function block type, an error is generated during the compilation run.

Instance the function block or call it without a prefix within the function block to correct this.

Example:

MX_PID_ControllerPL.Init_Regler(); (→generates error)

Init_Regler(); (→no error)

Functions with Dummy Arguments

With MLD-1G: Functions cannot be programmed without parameter; a parameter must be set, but is not used.

With MLD-2G: Functions can be programmed **without** parameter.

10.3.3 Adjustments in MLD-2G

Checking the task system

The task system has to be checked after the program is imported. IndraLogic MLD-2G supports different task systems:

Events:

- Cyclic task
- Event task
- Free-running task (behavior las in MLD-1G)
- External event task
 - CCD_SYNCHRONIZED_TASK
 - FKM_SYNCHRONIZED_TASK
 - PositionLoopEvent

The "PositionLoopEvent" allows for the processing of certain PLC programs that are configured in a task in the position controller cycle.

The standard installation of IndraWorks MLD comprises the MLD package without "PositionLoopEvent". The "PositionLoopEvent" is available with an additional IW package (e.g. MldPosLoopDevice_V20.12.0.0.iwpackage). The IW package for the "PositionLoopEvent" is available from the support and can be installed retroactively.

I/O variables cannot be accessed with the "PositionLoopEvent". I/O variables can be indirectly accessed in two ways:

- Reading the I/Os in the cyclic task and copying them to the flag. Accessing the flag in the "PositionLoopEvent".
- Directly accessing parameters in the "PositionLoopEvent" (when this has to be done quickly!)



It is impossible in MLD-2G to create a project with only event-controlled tasks. At least one cyclic or freewheeling task has to be created.

System events:

MLD-2G supports different system events:

- StartDone (call after start of the application)
- StopDone (call after stop of the application)
- PrepareReset (call before reset of the application)
- ResetDone (call after reset of the application)



- Only the system events described here may be used.
- When using system events, make sure as little code as possible is processed.

10.4 Converting projects MPx18 --> MPx20

When converting a PLC project of firmware version MPx18 to a project for firmware version MPx20 please note that the declaration of the "AXIS_REF" variables has changed. The "AXIS_REF" type variables are used for PLCOPEN function blocks. For this reason, reference is made to PLCOPEN for the error message. The error message is not clearly an indication of an incorrect declaration.

The declaration of the variables has to be adjusted. The previous "VAR_IN_OUT" declaration has to be changed to now include the declaration: "VAR_IN_OUT CONSTANT"

11 Service and support

Our worldwide service network provides an optimized and efficient support. Our experts offer you advice and assistance should you have any queries. You can contact us **24/7**.

Service Germany Our technology-oriented Competence Center in Lohr, Germany, is responsible for all your service-related queries for electric drive and controls.

Contact the **Service Hotline** and **Service Helpdesk** under:

Phone:	+49 9352 40 5060
Fax:	+49 9352 18 4941
E-mail:	service.svc@boschrexroth.de
Internet:	http://www.boschrexroth.com

Additional information on service, repair (e.g. delivery addresses) and training can be found on our internet sites.

Service worldwide Outside Germany, please contact your local service office first. For hotline numbers, refer to the sales office addresses on the internet.

Preparing information To be able to help you more quickly and efficiently, please have the following information ready:

- Detailed description of malfunction and circumstances
- Type plate specifications of the affected products, in particular type codes and serial numbers
- Your contact data (phone and fax number as well as your e-mail address)

Index

Symbols

.NET Libraries..... 164

A

Additional documentations..... 4
 Application (context menu)..... 185
 Appropriate use..... 29
 Applications..... 29
 Archive..... 207
 Archiving MLD projects..... 272
 Assignment of Logic MLD-M Axis Number to
 Sercos Address of CCD Slaves..... 68
 AT..... 42
 Axis..... 43
 Axis Addressing..... 55
 Axis Availability for Functions and Function
 Blocks..... 67
 Axis commanding..... 47, 52
 Axis control..... 43, 47
 AxisData cyclic axis data..... 105
 AxisData structure..... 107
 AxisInterface..... 42, 192

B

Basic principles..... 15
 Boot project..... 256
 Boot Project..... 17

C

C0241..... 77
 C0266..... 77
 CCD..... 42
 CCD data containers..... 143
 CCD master/CCD slave..... 42
 Configuration, device..... 275
 Configuring MLD with IndraWorks..... 48
 Configuring the inputs and outputs..... 131
 Connecting VE* via Ethernet..... 155
 Control panel..... 190
 Control unit
 Logger..... 276
 Converting projects..... 291
 Adjustments in MLD-2G..... 295
 Correcting compiling errors..... 293
 Requirements in MLD-1G..... 291
 Transferring a project from MLD-1G..... 292

D

Data channel
 I/O channel..... 102
 Parameter channel..... 119
 Device configuration..... 275
 Device control..... 43
 Device control with MLD-M..... 44
 Device editor..... 275

Logger..... 276
 Task deployment..... 276
 Device group..... 279
 Device model number..... 279
 Device name..... 279
 Device type..... 279
 Device version..... 279
 Direct variables..... 118
 Documentation
 Additional documentations..... 4
 Overview..... 4
 Record of revisions..... 1
 Reference documentations..... 4
 Documentation structure..... 4
 Drive as PLC device..... 220
 Drive errors..... 90
 Drive system..... 31

E

eal_sdk..... 162
 Application examples..... 162
 Documentation..... 162
 EAL4Android..... 164
 EAL4C..... 164
 EAL4DotNet..... 164
 EAL4Java..... 164
 EAL4LabVIEW..... 164
 Easy Automation Library..... 162
 Electric drive system..... 31
 Event logging..... 276
 Event tasks..... 74, 83
 External event task..... 74

F

Firmware requirements..... 23
 Free ticks..... 80
 Freewheeling task..... 74, 77
 Function..... 16
 Function Block..... 16
 Function block behavior..... 53
 Instantiation..... 54
 Timing..... 53

G

GAT compact..... 27
 Configuration..... 193
 Control of the operating modes..... 197
 First steps..... 194
 Partial functions..... 193
 GAT compact..... 42
 Global registers..... 122

H

Hardware requirements..... 22
 Helpdesk..... 297

Index

Hotline.....	297
I	
I/O channel.....	102
I/O configuration of MLD.....	131
IEC 61131.....	17, 241
IEC 61131.....	17
Importing a PLC Project.....	204
Inappropriate use.....	30
Consequences, disclaimer.....	29
IndraControl S20.....	15
IndraMotion for Handling.....	16
IndraMotion MLD.....	10
IndraMotion MLD data channels.....	99
IndraMotion MLD-M.....	12
IndraMotion MLD-S.....	11
IndraMotion, overview.....	9
IndraWorks package.....	219, 286
Information on the device.....	279
Instance identifiers	
of complex data types.....	264
of simple data types.....	264
Instructions for Use and Applications.....	22
Intelligent servo axis.....	25, 49
J	
Java.....	164
L	
LabVIEW.....	164
Libraries.....	18
General properties.....	242
Hierarchic structure.....	245
Supported libraries.....	243
Local axis.....	42
Lock symbol.....	273
Logger in runtime system.....	276
Logic (context menu): see Application.....	185
M	
Master communication.....	43
MDT.....	42
Means of representation	
Conventions of notation.....	1
Notations.....	1
Notes.....	2
MLD function blocks and parameters for axis	
commanding.....	70
MLD interfaces and data channels.....	97
MLD projects	
archiving.....	272
processing as team.....	272
versioning.....	272
MLD-M Axis Number.....	68
MLD-S, MLD-M, MLD.....	41
Motion control.....	42
Motion Control command interface	
Operation modes.....	129
Motion task in synchronism with CCD.....	75
Application.....	76
Motion task cycle time.....	75
Timing.....	76
Motion task in synchronism with master	
communication.....	74
Application.....	75
Motion task cycle time.....	75
Timing.....	75
N	
Network variables.....	77
Notes on application and programming.....	70
O	
OCI@Drives.....	158
Open Core Interface for drives.....	158
Application examples.....	164
Possible applications.....	160
Supported devices.....	160
Target platforms.....	160
P	
Parameter channel.....	119
Parameters and diagnostic messages for ax-	
is commanding.....	71
Parameters for general purpose.....	122
PELV.....	35
Performance data.....	21
Periodic (cyclic) tasks.....	74
Persistent variable backup.....	247
Persistent variable backup and restore.....	247
PII.....	42
PLC errors.....	90
PLC programming	
Global data.....	260
History.....	261
Program header.....	260
Task.....	16
Type identifiers.....	262
PLC Programming	
Boot project.....	17
Function.....	16
Function block.....	16
Program.....	16
Resources.....	16
PLC runtime.....	44
PLC settings.....	278
PLC shell.....	279
PLC warnings.....	89
PLCopen.....	17, 241
PLCopen function blocks.....	42
POI.....	42
Program.....	16
Programming languages.....	18
Notes on SFC programming.....	260

Overview.....	259
Recommended basic programming languages.....	259
Protective extra-low voltage.....	35

R

Read recipe.....	250
Real-time channel	
see AxisData cyclic axis data.....	100
Reference documentations.....	4
Remote axes.....	42
Resources.....	16
PLC configuration.....	16
Sampling trace.....	16
Task configuration.....	16
Variable configuration.....	16
Watch and recipe manager.....	16
Restore.....	207
Retain data backup.....	247
Retain data backup and restore.....	247
Rexroth IndraControl S20.....	15
Rexroth IndraMotion MLD	
Applications.....	24
Rexroth Inline.....	14, 15, 135
Rexroth Inline Block IO.....	14, 140
Rexroth Inline Modular IO.....	14
Run-up mode.....	42
Runtime system.....	44

S

Safety instructions for electric drives and controls.....	31
Save recipe as file on the PC.....	250
SDK	
Installation.....	162
Prerequisites.....	162
Sequential Motion Control (SMC).....	15, 27
Sercos Address of CCD Slaves.....	68
Sercos I/O.....	145
Configuration.....	145
Process images of the inputs/outputs.....	146
Sercos III drive as PLC device	
Configuring the cyclic data.....	224
Overview.....	221
Parameterization.....	222
PLC programming.....	227
Sercos peripherals	
(Frequency converter) EFC.....	15
Drive controller.....	15
RMB_SercosIII_Util.library.....	15
Service hotline.....	297
Setting the IP address.....	149
Signal-time diagram	
for edge-controlled function blocks.....	269
for state-controlled function blocks.....	267
SMC.....	15
Software requirements.....	24

Stand-alone Motion Logic Control.....	51
Stand-Alone Motion Logic Control.....	25
Stand-alone single-axis Motion Logic Control.....	50
Standard inputs at function blocks.....	266
Standard outputs at function blocks.....	266
Start behavior.....	256
State-of-the-art.....	29
Structuring PLC projects	
Multi-program technology.....	259
Task management.....	258
Supply unit as PLC device	
General information.....	230
Parameterization.....	230
PLC programming.....	233
Supply units as PLC device	
Configuring the cyclic data.....	231
Support.....	297
Supported task types.....	72
System behavior of MLD	
Notes on application and programming.....	257

T

Task.....	16
Task configuration.....	83
Task cycle times.....	83
Task deployment.....	276
Task monitoring.....	77
Task properties.....	74
Task system.....	72
Task System and priorities.....	73
Team server.....	272
Technical Data.....	19
Technology function.....	15
Diagnostic and status information.....	180
Documentation.....	180
Loading and activating.....	178
Overview.....	177
Parameterization and commissioning.....	179
Technology function blocks.....	42
Technology libraries.....	26
Technology packages.....	27
Terms.....	15
Timing.....	53
Turnkey solutions.....	27
Type identifiers	
for programs, function blocks and functions.....	262
for structures, arrays, enumerators and IEC data types.....	263

U

Updating the IndraLogic device version.....	219
Use	
Appropriate use.....	29
Inappropriate use.....	30
User program.....	27

V

VCS.....	272
Vendor name.....	279
Version assignment of libraries.....	270
Version Control System.....	272
Version function.....	270
Versioning MLD projects.....	272
Virtual master axis.....	42

W

Watchdog.....	77
Watchdog function.....	77
Write recipe.....	250

Notes

Bosch Rexroth AG

Electric Drives and Controls

P.O. Box 13 57

97803 Lohr, Germany

Bgm.-Dr.-Nebel-Str. 2

97816 Lohr, Germany

Phone +49 9352 18 0

Fax +49 9352 18 8400

www.boschrexroth.com/electrics



R911338914