rexroth
A Bosch Company

IndraMotion MTX 14VRS

OPC Communication

Project Planning Manual Edition 03
R911342636

Title IndraMotion
MTX 14VRS
OPC Communication

Type of Documentation Project Planning Manual

Document Typecode DOK-MTX***-OPC*COM*V14-PR03-EN-P

Internal File Reference RS-003e4faf1c4924aa0a6846a5018fbe0f-3-en-US-4

Change record Edition 03, 2019-05
See tab. 1-1 "Change Record" 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 Engineering Automation Systems Control Functions Motion and CNC, EbSc (TaDo/MePe)

Table of Contents

Page

1 About this documentation ... 1
1.1 Validity of the documentation ... 1
1.2 Purpose ... 1
1.3 Required and supplementing documentation MTX ... 2
1.3.1 Selection/compilation ... 2
1.3.2 Configuration ... 2
1.3.3 Commissioning ... 3
1.3.4 Operation ... 4
1.3.5 OEM Engineering ... 5
1.3.6 AddOns ... 5
1.4 Information representation ... 5
1.4.1 Safety instructions ... 5
1.4.2 Symbols used ... 6
1.4.3 Names and abbreviations ... 6
1.4.4 Customer feedback ... 6

2 Introduction ... 7
2.1 Exclusion of liability ... 7
2.2 OPC tag structure ... 7
2.3 OPC test client ... 9

3 Important instructions on use ... 11
3.1 Intended use ... 11
3.1.1 Introduction ... 11
3.1.2 Areas of use and application ... 11
3.2 Unintended use ... 12

4 Items ... 13
4.1 General information ... 13
4.1.1 DeviceTypeTest2 ... 13
4.2 Root ... 13
4.2.1 Overview ... 13
4.2.2 Root Items ... 13
4.3 Firmware ... 19
4.3.1 Overview ... 19
4.3.2 Firmware Items ... 20
4.3.3 Error codes ... 30
4.4 Sercos ... 31
4.4.1 Overview ... 31
4.4.2 Sercos Idents ... 31
4.4.3 Sercos Elements ... 31
4.4.4 Sercos Attributes and Data Types ... 32
4.4.5 Sercos Lists ... 33

Page

4.4.6 Requesting Several Elements ... 34
4.4.7 Sercos Attribute Cache ... 35
4.4.8 Sercos Error Codes ... 36
4.4.9 Sercos Items ... 38
4.5 Diagnostics and Logbook ... 45
4.5.1 Overview ... 45
4.5.2 Diagnostics ... 47
4.5.3 Logbook ... 57
4.6 IndraLogic ... 63
4.6.1 Overview ... 63
4.6.2 Firmware ... 64
4.6.3 Info ... 65
4.6.4 Intervention ... 67
4.6.5 Field bus ... 67
4.6.6 Variables ... 67
4.7 NC - Global ... 72
4.7.1 General information ... 72
4.7.2 Drives ... 92
4.7.3 CPL data ... 98
4.7.4 File-based Tables ... 101
4.7.5 Tables based on databases ... 105
4.7.6 Machine parameters ... 111
4.7.7 System data ... 115
4.7.8 System Functions ... 121
4.8 NC - channel-dependent ... 146
4.8.1 General Channel Information ... 146
4.8.2 Axis positions ... 153
4.8.3 Axis corrections ... 163
4.8.4 Adjustment Functions ... 173
4.8.5 Spindle Data ... 177
4.8.6 Process Data ... 183
4.8.7 Feed Values and Speed Values ... 196
4.9 NC Commands ... 202
4.9.1 Overview ... 202
4.9.2 Global commands ... 203
4.9.3 Channel-dependent commands ... 206
4.10 Error messages ... 218
4.10.1 Error handling ... 218
4.10.2 SAV Errors ... 220
4.10.3 MACODA Errors ... 222
4.10.4 TAD Errors ... 222
4.10.5 DBD Errors ... 223
4.10.6 CFG Errors ... 224
4.10.7 SD Errors ... 226
4.10.8 XML Errors ... 227
4.10.9 TAR Errors ... 232

Page

4.10.10 Unlock error 234

5 Service and support 237

Index 239

1 About this documentation

Versions of this documentation

Edition	Release Date	Note
01	2014-03	First edition
02	2017-12	Changes in the chapters "4.3 Firmware", "4.6 IndraLogic", "4.7 NC - global", "4.8 NC - channel-dependent", "4.9 NC commands" and "5.3 Bosch Rexroth OPC UA server"
03	2019-05	Changes in the chapter "4.7 NC global"

Tab. 1-1: Change Record

1.1 Validity of the documentation

Overview on target groups and product phases

The following graphic refers to the bordered activities, product phases and target groups of the present documentation.

Example: The target group "Programmer" can execute the "configuration" in the product phase "Engineering" using this documentation.

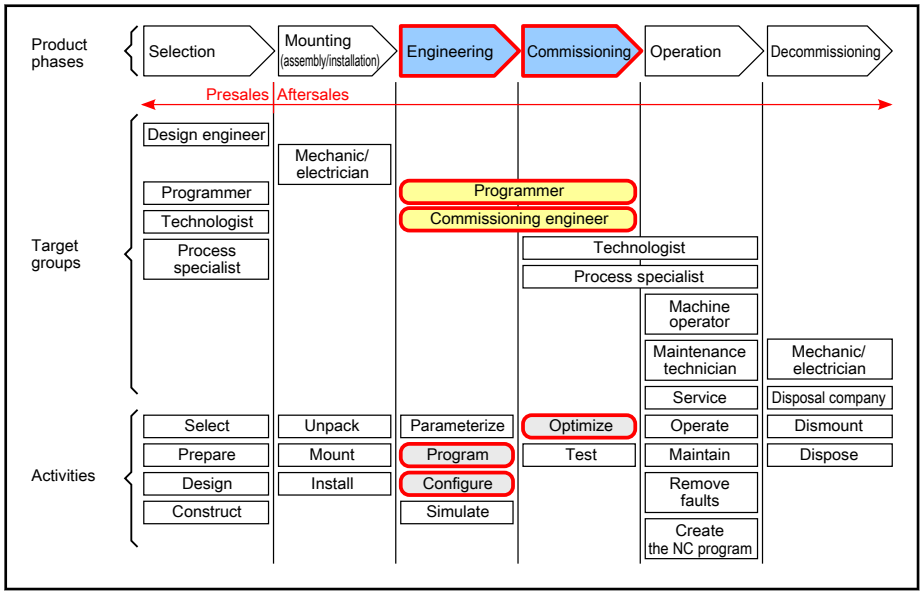

Fig. 1-1: Assigning this documentation to the target groups, product phases and target group activities

Target group

This documentation is intended for OPC client programmers and WinStudio project developers. OPC tags are described for the logical device MTX (LDX).

This documentation supports the user during the following phases:

- Engineering
- Commissioning

1.2 Purpose

This documentation is intended for OPC client programmers and WinStudio project developers to describe OPC tags for the logical device MTX (LDX).

The main component of the tag syntax is the so-called **Item** specified via the requested date using the name and individual parameters. The following documentation describes the syntax and the structure of the available items.

To determine a specific OPC tag, refer to chapter 2.2 "OPC tag structure" on page 7.

1.3 Required and supplementing documentation MTX

1.3.1 Selection/compilation

Documentation titles with type codes and part numbers

IndraMotion MTX 14VRS System Description DOK-MTX***-SYS*DES*V14-PRRS-EN-P, R911342640 This documentation describes the Rexroth IndraMotion MTX control. It includes the designs, technical data, interfaces as well as the configuration of the control components.
IndraMotion MTX 14VRS SafeLogic System Overview DOK-MTX***-SL**SYS*V14-PRRS-EN-P, R911341698 This documentation describes the use of the safety control SafeLogic in the IndraMotion MTX.

Tab. 1-2: MTX documentation overview - Selection/compilation

1.3.2 Configuration

Documentation titles with type codes and part numbers

IndraMotion MTX 14VRS Machine Parameters DOK-MTX***-MA*PAR**V14-RERS-EN-P, R911342630 This documentation describes design and modification of the Rexroth IndraMotion MTX parameters available. It also includes the functions of the NC configurator and its operation.
IndraMotion MTX 14VRS PLC Interface DOK-MTX***-PLC*INT*V14-PRRS-EN-P, R911342622 This documentation describes interface signals and program function blocks for the integrated PLC.
IndraMotion MTX 14VRS Functional Description - Basics DOK-MTX***-NC*F*BA*V14-RERS-EN-P, R911342626 This documentation describes the functions of the Rexroth IndraMotion MTX basic functions. The basic commissioning steps and the control functions are provided as description and handling instruction.
IndraMotion MTX 14VRS Functional Description - Extension DOK-MTX***-NC*F*EX*V14-RERS-EN-P, R911342628 This documentation describes the extended functions of the Rexroth IndraMotion MTX. The basic commissioning steps and the control functions are provided as description and handling instruction.
IndraMotion MTX 14VRS Functional Description - Special Functions DOK-MTX***-NC*F*SP*V14-RERS-EN-P, R911342624 This documentation describes the special functions of the Rexroth IndraMotion MTX. The basic commissioning steps and the control functions are provided as description and handling instruction.

IndraMotion MTX Free Form Surface Milling DOK-MTX***-FREEFORM***-APRS-EN-P, R911341435 This documentation describes the free-form surface machining process with the MTX control. CNC programs generated by a CAD/CAM system are the basis for the entire machining process. An overview of the following topics is provided: Description of the MTX or CNC functions for free-form surface machining, NC parameterization, basic conditions when generating CNC programs with CAM software.
Rexroth IndraMotion MTX Converting MTX Projects DOK-MTX***-PROCONV****-PRRS-EN-P, R911342484 This documentation provides support during the conversion of IndraMotion MTX 1.x projects to MTX 2G. The conversion process is described and potential challenges during the conversion are identified.

Tab. 1-3: MTX documentation overview - Configuration

1.3.3 Commissioning

Documentation titles with type codes and part numbers

Rexroth IndraWorks 13VRS Simulation DOK-IWORKS-SIMU****V13-APRS-EN-P, R911339256 This documentation describes the functions of the simulation components View3D, virtual control panel, virtual control and its operation in IndraWorks.
IndraMotion MTX 14VRS Commissioning DOK-MTX***-STARTUP*V14-CORS-EN-P, R911342620 This documentation describes the commissioning of the IndraMotion MTX control. Apart from a complete overview, commissioning and configuration of the axes and the user interface as well as the PLC data are described.
IndraWorks 14VRS Basic Libraries IndraLogic 2G DOK-IL*2G*-BASLIB**V14-LIRS-EN-P, R911343920 This documentation describes the system-comprehensive PLC libraries.
IndraWorks 14VRS Field Buses DOK-IWORKS-FB******V14-APRS-EN-P, R911341485 This documentation describes field bus and local periphery connections supported by the systems IndraLogic XLC, IndraMotion MLC and IndraMotion MTX. This documentation focuses on the configuration, parameterization, commissioning and diagnostics of the different periphery connections. The documentation is the basis for the online help.
IndraWorks 14VRS WinStudio 7.4 DOK-IWORKS-WINSTUD*V14-APRS-EN-P, R911341585 This "User Manual and Technical Reference Book" provides information for efficient working with the "Rexroth WinStudio"™ software. The document provides technical information and step-by-step instructions to create web-enabled HMI/SCADA programs.
IndraWorks 14VRS Software Installation DOK-IWORKS-SOFTINS*V14-CORS-EN-P, R911344286 This documentation describes the IndraWorks installation.
IndraWorks 14VRS Engineering DOK-IWORKS-ENGINEE*V14-APRS-EN-P, R911343566 This documentation describes the use of IndraWorks in which the Rexroth Engineering tools are integrated. It includes instructions on how to work with IndraWorks and how to operate the oscilloscope function.
Rexroth IndraWorks 12VRS FDT Container DOK-IWORKS-FDT*CON*V12-APRS-EN-P, R911334398 This documentation describes the IndraWorks FDT Container functionality. It includes the activation of the functionality in the project and working with DTMs.

IndraWorks 14VRS PLC Programming System IndraLogic 2G DOK-IWORKS-IL2GPRO*V14-APRS-EN-P, R911343571 This documentation describes the PLC programming tool IndraLogic 2G and its use. It includes the basic use, first steps, visualization, menu items and editors.
IndraWorks 14VRS HMI DOK-IWORKS-HMI*****V14-APRS-EN-P, R911343569 This documentation describes the functions, configuration and operation of the user interfaces IndraWorks HMI Engineering and IndraWorks HMI Operation.

Tab. 1-4: MTX documentation overview - Commissioning

1.3.4 Operation

Documentation titles with type codes and part numbers

Rexroth IndraMotion MTX 09VRS Shop Floor Programming Turning and Milling DOK-MTX***-SF*PROG*V09-AWxx-EN-P, R911324377 This documentation describes the operation and programming of the graphic NC programming of turning and milling as well as the workpiece simulation.
Rexroth IndraMotion MTX 12VRS Block Pre-Run DOK-MTX***-BLK*RUN*V12-APRS-EN-P, R911334379 This documentation explains to the machine manufacturer how to setup the "Block pre-run" function at the machine for the end user.
IndraMotion MTX 14VRS Programming Manual DOK-MTX***-NC**PRO*V14-RERS-EN-P, R911342634 This documentation describes the standard programming of the Rexroth IndraMotion MTX control. Apart from the basics of NC programming, the use of NC functions according to DIN 66025 as well as the NC functions with high-level language syntax and CPL functions are described.
IndraMotion MTX 14VRS Standard NC Operation DOK-MTX***-NC*OP***V14-APRS-EN-P, R911342632 This documentation describes the operation of the standard user interface of the NC control of the Rexroth IndraMotion MTX. It includes the operation of the interface, the NC program development as well as the tool management.
IndraMotion MTX 14VRS Standard NC Cycles DOK-MTX***-NC*CYC**V14-PRRS-EN-P, R911342638 This documentation describes the application of the standard cycles of the different technologies for Rexroth IndraMotion MTX control.
IndraMotion MTX 14VRS Multitouch DOK-MTX***-MULTI***V14-APRS-EN-P, R911384735 The Multitouch user interface of the NC control Rexroth IndraMotion MTX is described in this documentation.
IndraMotion MTX 14VRS NC Simulation DOK-MTX***-NC*SIM**V14-APRS-EN-P, R911379629 This documentation describes the NC Simulation for Rexroth IndraMotion MTX control.

xx Corresponding edition

Tab. 1-5: MTX documentation overview - Operation

1.3.5 OEM Engineering

Documentation titles with type codes and part numbers
Rexroth IndraMotion MTX 13VRS Automation Interface DOK-MTX***-AUT*INT*V13-APRS-EN-P, R911337274 This documentation describes the script-based access to IndraWorks project data via the Automation Interface. Different objects including code examples are described. The Automation Builder is also described in this manual.
Rexroth IndraMotion MTX 14VRS OPC Communication DOK-MTX***-OPC*COM*V14-PRRS-EN-P, R911342636 This documentation describes the syntax and the structure of the items for the communication with Bosch Rexroth devices.
Rexroth IndraMotion MTX 14VRS OPC UA Communication DOK-MTX***-OPC*UA**V14-APRS-EN-P, R911342513 This documentation describes the OPC UA communication of the IndraMotion MTX.

Tab. 1-6: MTX documentation overview - OEM Engineering

1.3.6 AddOns

Documentation titles with type codes and part numbers
Rexroth IndraMotion MTX 11VRS Action Recorder DOK-MTX***-ACR*****V11-APxx-EN-P, R911329943 This documentation describes the MTX action recorder. It includes the installation and commissioning as well as interface signals, application and operation.
Rexroth IndraMotion MTX 14VRS efficiency workbench MTX cta, MTX ega DOK-MTX***-EWB*****V14-APRS-EN-P, R911343916 This documentation describes the mode of operation and the area of application of the analysis tool IndraMotion MTX cta and IndraMotion MTX ega.
Rexroth IndraMotion MTX Remote Condition Monitoring DOK-MTX***-RCM*****V01-APRS-EN-P, R911334383 This documentation describes the operation of the Remote Condition Monitoring System.
Rexroth IndraMotion MTX visIREC User Documentation DOK-MTX***-VISIREC*V01-APRS-EN-P, R911344242 This documentation describes the analysis tool visIREC. The visIREC is used to optimize the free-form surface machining process. 2D or 4D display of path-related data. 2D or 4D display of coordinate-related data. Analyzing critical areas (path and orientation deviation). Comparing the programmed and interpolated NC blocks.

xx Corresponding edition

Tab. 1-7: MTX documentation overview - AddOns

1.4 Information representation

1.4.1 Safety instructions

If there are safety instructions in the documentation, they contain certain signal words (Danger, Warning, Caution, Notice) and sometimes a safety alert symbol (according to ANSI Z535.6-2006).

The signal word draws attention to the safety instruction and indicates the risk potential.

The safety alert symbol (warning triangle with exclamation mark) preceding the signal words Danger, Warning and Caution indicates hazards for persons.

⚠ 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 **may** occur.

⚠ CAUTION

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

NOTICE

In case of non-compliance with this safety instruction, material or property damage can occur.

1.4.2 Symbols used

Note Notes are represented as follows:

☞ This is a note for the user.

Tip Tips are represented as follows:

This is a tip for the user.

1.4.3 Names and abbreviations

Term	Explanation
NC	Numerical Control

Tab. 1-8: Names and abbreviations used

1.4.4 Customer feedback

Customer requests, comments or suggestions for improvement are of great importance to us. Please email your feedback on the documentations to Feedback.Documentation@boschrexroth.de. Directly insert comments in the electronic PDF document and send the PDF file to Bosch Rexroth.

2 Introduction

2.1 Exclusion of liability

⚠ WARNING **No liability for damages by customer OPC clients**

With the OPC, the access to all device data and the complete device functionality is not restricted.

Bosch Rexroth is not liable for damages and loss of data due to an unintended use of the OPC in "third-party" OPC client applications and WinStudio operating screens. Bosch Rexroth is also not liable if device-specific guidelines are not observed or ignored. This applies especially to the writing of device-sensitive and safety-relevant data.

2.2 OPC tag structure

An OPC tag is structured as follows. The parameters are separated by commas.

Syntax ![1],//[2]/[3],[4],[5] {[6],[7]}

1. Namespace
2. IP address or host name
3. Device name or device Guid
4. Item parameter list
5. Parameter
6. Line
7. Column

☞ The specification of namespace, host name, line and column are optional.

If no namespace is specified, data is provided in the VARIANT data type.

If no host name is specified, data is provided by the local OPC server. This corresponds to the value "localhost" or " 127.0.0.1".

The device can either be described via the name (e.g. "IndraMotion_MTX_P60") or the device Guid.

Type and number of the parameters differ for each item. For the specific parameters, refer to the respective item chapters.

If no {Line, column} is specified, the complete data response is provided as given in the documentation. This corresponds to the value "{-1,-1}".

Examples:

```
!BSTR,//machine01/
IndraMotion_MTX_P60,NC.SystemDataWatch_U,/
Sys_Simulation[1],1024 {0,0}
```

```
!BOOL,//machine02/
IndraMotion_MTX_P60,Plc.PVL,.OpMode.AUTO.Key {0,0}
```

```
!R8,//machine03/IndraMotion_MTX_P60,NC.Chan.AxisPosAcs,1
{0,7}
!R8,//machine04/
IndraMotion_MTX_P60,NC.Chan.ActRotAxisProgrammedSpeed,1
{3,7}
```

Namespace

The namespace describes the data type of the requested type.

The following namespaces can be selected:

COM data type	Namespace
VT_I1	!I1
VT_I2	!I2
VT_I4	!I4
VT_I8	!I8
VT_INT	!INT
VT_R4	!R4
VT_R8	!R8
VT_UI1	!UI1
VT_UI2	!UI2
VT_UI4	!UI4
VT_UI8	!UI8
VT_UINT	!UINT
VT_BOOL	!BOOL
VT_BSTR	!BSTR
VT_DATE	!DATE
VT_CY	!CY

Tab. 2-1: Namespace

☞ The OPC server might convert the original date (e.g. UI1) into the data type specified in the namespace (e.g. I4) taking the OPC convention into account.

One namespace specification should always be used. If the namespace is not specified, the requested date is provided in the VARIANT data type.

{Line, column}

An extension "{Line, column}" affects a partial result from a complex result that can extend of up to multiple lines and columns (matrix).

If the OPC tag is extended by "{Line, column}", a partial result can be specifically retrieved from the total result. Such tags are collected in the OPC server and the read access is optimized.

The following formats are permitted:

!R8,//machine01/ IndraMotion_MTX_P60,NC.Chan.AxisPosAcs,1	**Complete table**
!R8,//machine01/ IndraMotion_MTX_P60,NC.Chan.AxisPosAcs,1 {0,0}	Result from line 0 and column 0
!R8,//machine01/ IndraMotion_MTX_P60,NC.Chan.AxisPosAcs,1 {0,7}	Result from line 0 and column 7
!R8,//machine01/ IndraMotion_MTX_P60,NC.Chan.AxisPosAcs,1 {0, -1}	**Invalid.** Complete line is **not** supported
!R8,//machine01/ IndraMotion_MTX_P60,NC.Chan.AxisPosAcs,1 {-1, 7}	**Invalid.** Complete column is **not** supported

Tab. 2-2: {Line, column}

2.3 OPC test client

To test the OPC communication or the data response of the requested OPC tag, use the OPC client application "**OPC.TestClient.exe**". For more information, refer to the OEM SDK documentation.

☞ For the "OPC.TestClient.exe", go to the IndraWorks installation directory (e.g. "C:\Program Files\Rexroth\IndraWorks")

3 Important instructions on use

3.1 Intended use

3.1.1 Introduction

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

The products may only be used as intended. If they are not used as intended, situations occur that result in damage to property or injury to persons.

☞ Bosch Rexroth shall not assume any warranty, liability or payment of damages in case of damage resulting from a non-intended use of the products; the use shall solely bear all risks from unintended use of the products.

Before using Bosch Rexroth products, the following requirements have to be met to guarantee the intended use of the products:

- Anybody dealing with Bosch Rexroth products in any way is obliged to read and consent to the relevant safety instructions and the intended use.
- Hardware products may not be altered and have to remain in their original state; i.e. no structural changes are permitted. The decompilation of software products or the alteration of source codes is not permitted.
- Do not install or operate damaged or faulty products.
- It has to be ensured that the products have been installed as described in the relevant documentation.

☞ Ensure that the data present in the control or entered or read in by the user is correct before applying it to exclude unwanted axis motion. It can be the following invalid or old data:

- Part programs
- ZO tables
- Compensation tables
- Tool tables
- Permanent CPL variables
- Remanent PLC data
- Permanent system data

3.1.2 Areas of use and application

The Rexroth IndraMotion MTX control is used to

- program contour and machining technology (path feed, spindle speed, tool change) of a workpiece.
- guide a machining tool along a programmed path.

Feed drives, spindles and auxiliary axes of a machine tool are activated via sercos interface.

☞ This additionally requires I/O components for the integrated PLC which - together with the actual CNC - control the machining process as a whole and also monitors this process with regard to technical safety.

It may only be operated with the explicitly specified hardware component configurations and combinations and only with the software and firmware specified in the appropriate documentations and functional descriptions.

The Rexroth IndraMotion MTX provides the perfect CNC system solution for cutting and forming for the following technologies:

- Turning
- Milling
- Drilling
- Grinding
- Bending
- Nibbling
- Punching
- Contour cutting
- Handling

3.2 Unintended use

The use of the Rexroth IndraMotion MTX in application areas other than those specified or described in the documentation and technical data is considered as "unintended".

The Rexroth IndraMotion MTX must not be used if ...

- it is subjected to operating conditions not corresponding to the specified ambient conditions. Operation under water, under extreme temperature fluctuations or under extreme maximum temperatures is prohibited.
- the intended applications have not expressively been released by Bosch Rexroth. Therefore, please read the information given the general safety instructions!
- The Rexroth IndraMotion MTX may not be used in systems or machines connected to the internet via an unsecure network connection Otherwise, malfunctions or a control failure can result due to unauthorized access.

4 Items

4.1 General information

4.1.1 DeviceTypeTest2

Description Checks whether a communication connection is established to a certain device type.

Remark The item "DeviceTypeTest2" internally reads parameters of the device (e.g. firmware ID) used to identify a device type.

Availability Sercans, IndraDrive, MLC, MTX, IndraLogic

Item syntax DeviceTypeTest2

Read Checks whether a communication connection can be established to the device and whether the type of the configured device (logical device) corresponds to the type of the addressed device (field device).

Data format DeviceTypeTest2

row, col	Type	Description	Example
0,0	bool	TypeValidated flag Indicates whether the type of the configured device and the type of the addressed device match and whether a connection was established.	False

4.2 Root

4.2.1 Overview

Communication process functions are provided via the "Root" interface. Via this interface, the functions for scanning, diagnostics and configurations are provided.

Root is always provided as part of the communication.

4.2.2 Root Items

AutoItemAccess

Description Access mode of "AutoItems" of one device.

Some devices can address an online interface as well as an offline interface. A device-specific state is used to switch certain items between the online and offline interface.

Item syntax AutoItemAccess

Read Reading the "AutoItem" state of the device.

AutoItemAccess, Read

row, col	Type	Description	Example
0,0	long	State 1: SCPAccessOnlineWriteOffline (default) 2: SCPAccessOffline 3: SCPAccessOnline	2
1,0	string	State as text	"SCPAccessOffline"

Write Reading the "AutoItem" state of the device.

AutoItemAccess, Write

row, col	Type	Description	Example
0,0	long	State 1: SCPAccessOnlineWriteOffline (default) 2: SCPAccessOffline 3: SCPAccessOnline	2

CNF.AddComponent

Description Creates a new device or a driver.

☞ A device or driver is configured via specified configuration parameters. For details, refer to the documentation for the device and driver configuration.

Item syntax CNF.AddComponent,1

Write Creating device or driver.

Data format The parameter name (x,0) and the parameter value (x,1) are transferred as string for each parameter (x).

CNF.AddComponent,1

row, col	Type	Description	Example Adding a TCP/IP driver.
0,0	string	Parameter name "Name"	"Name"
0.1	string	Parameter value	"DR.IndraDrive"
1,0	string	Parameter name "ProgId"	"ProgId"
1,1	string	Parameter value	"IwSCPDriverEthernet.SCPDriverEthernet"
2,0	string	Parameter name "Guid"	"GUID"
2,1	string	Parameter value	"074cee90-0190-4cb4-9913-d7dc7d69ab34-On"
3.0	string	Parameter name "IpAddress"	"IpAddress"
3.1	string	Parameter value	"192.23.43.5"

row, col	Type	Description	Example Adding a TCP/IP driver.
4.0	string	Parameter name "Port"	"Port"
4.1	string	Parameter value	"5003"

Tab. 4-1: Adding a TCP/IP driver

CNF.RemoveComponent

Description Removes a device or a driver from the configuration.

Item syntax CNF.RemoveComponent

Write Device or driver is removed.

The items SDL, see chapter "SDL" on page 19, or GDL, see chapter "GDL" on page 18, do not display the device or driver anymore.

☞ Device or driver is removed from the list. Device or driver remains in the memory as long as it is used.

CNF.RemoveComponent

row, col	Type	Description	Example
0,0	string	GUID or name of device or driver.	"87979723-228C-4ff4-ACE2-97C5A3F8B8C8"

CNF.CMD

Description Saving and loading the communication configurations.

The default storage location of the configuration can be read and written.

This item is available at the "Root" interface.

☞ The file names are used by the respectively addressed communication process. Correspondingly, a remotely created process uses the path names on the remote computer.

Do no use relative paths.

Item syntax CNF.CMD

Read Reading the default storage location of the configuration file.

☞ The file name is read as user-specific entry from the "Windows" Registry.

CNF.CMD (Read)

row, col	Type	Description	Example
0,0	string	Constant "DefaultConfigFile"	DefaultConfigFile
1,0	string	Default storage location of the configuration file.	"c:\scpconfig.ini"

Write Writing the default storage location of the configuration file.

☞ The file name is written as user-specific entry into the "Windows" Registry.

CNF.CMD (Write)

row, col	Type	Description	Example
1,0	string	Default storage location of the configuration file.	"c:\scpconfig.ini"

Call

Loading and saving an SCP server configuration.

"Save" and "Load".

☞ Loading the configuration deletes all already activated devices and drives.

Saving overwrites an already existing file.

CNF.CMD (Call)

row, col	Type	Description	Example Saving the configuration in C:\scpconfig.ini
0,0	string	Command "Load" or "Save"	"Save"
1,0	string	Optional: Default storage location is determined from the Registry.	"c:\scpconfig.ini"

Example: Load

CNF.CMD (Call)

row, col	Type	Description	Example Activating the configuration from C:\scpconfig.ini
0,0	string	Command "Load"	"Load"
1,0	string	Optional: Default storage location is determined from the Registry.	"c:\scpconfig.ini"

Example: Save

The command with the argument data "Save" stores the current configuration of the SCP server including the devices and the driver configuration. An already existing file is overwritten.

CNF.CMD (Call)

row, col	Type	Description	Example
0,0	string	Command "Save"	"Save"
0.1	string	Optional: Default storage location is determined from the Registry.	"c:\scpconfig.ini"

Example (JavaScript)

```
var server;

function Connect()
{
```

```
  if (server == null)
  {
    server = new ActiveXObject("IwSCP.ScpServer");
    server.Connect("Root", "Loader");
  }
}

function LoadSCP()
{
  var item;
  var data;
  Connect();
  item = server.CreateItem("CNF.CMD",0);
  data = item.SyncRead(1);
  ConfigFile.value = data.GetValue(0,1);
  data.SetValue(0,0,"Load");
  item.SyncCall(data);
}

function SaveSCP()
{
  var item;
  var data;
  Connect();
  item = server.CreateItem("CNF.CMD",0);
  data = item.SyncRead(1);
  data.SetValue(0,0,"Save");
  item.SyncCall(data);
}
```

ComPortList

Description List of the available COM ports of the PC

Remark Via the serial interfaces, SCP can communicate with the devices. For this purpose, the serial driver is used. Via this item, the list of the available COM ports can be determined.

Item syntax ComPortList

Read Reading the current list of the COM ports and their status.

Data format Number of lines: Number of COM ports(x)

ComPortList

row, col	Type	Description	Example
x,0	string	ComPort name. The Windows name for the COM interface is provided.	COM2
x,1	long	ComPort number. The extracted number from the ComPort name.	2
x,2	bool	The COM interface is available. -->True The COM interface is assigned. --> False	false
x,3	string	Status text of the interface test. Is empty in case of an available interface.	Access denied

DTY

Description Type of device or driver.

Item syntax DTY

Read Reading the device type.

DTY

row, col	Type	Description	Example
0,0	string	Device type	MTX

GDL

Description Provides a list of all devices and drivers.

Item syntax GDL

Read Reading the global list of devices and drivers.

Data format Number of lines(x) corresponds to the number of devices and drivers.

GDL

row, col	Type	Description	Example
x,0	string	Device name	IndraMotion MTX
x,1	string	Device type	MTX
x,2	string	GUID	87979723-228C-4ff4-ACE2-97C5A3F8B8C83847

ListOfItems

Description List of items available at this device or driver.

Notes The list is provided based on the device type and the available components.

The list does not contain any description of the possible parameters. For the parameter description, refer to the documentation.

☞ The item is usually used for test purposes.

Item syntax ListOfItems or

LOI

Read One line (x) is provided for each supported item.

The item description is provided as short text in English.

ListOfItems

row, col	Type	Description	Example
x,0	string	Item name	"SPA2"
x,1	string	Short description of the item (English)	"Sercos parameter of the drive"

SCL

Description Devices or driver information.

Remark The functionality of a device consists of several components. Each of these components provides a set of items. An application can access this set.

Item syntax SCL

Read A list is output containing the names and the version information of the loaded components. The device or the driver type is in the first line (x=0).

Data format Four elements are provided for each line (x) and for each component.

SCL

row, col	Type	Description	Example
x,0	string	Component name	"Root"
x,1	long	MajorVersion	6
x,2	long	MinorVersion	11
x,3	long	Build number	151

SDL

Description List of subordinate devices or drivers.

Item syntax SDL

Read A list is output containing the names, type and GUID of the subordinate devices.

Data format One line (x) with three entries is provided for each subordinate device or driver.

SDL

row, col	Type	Description	Example
x,0	string	Device name. The name is output including the possible existing superordinate name.	"Mlc01/Drive_0"
x,1	string	Device type or driver type.	"SercosDrive"
x,2	string	Clear ID of the device or the driver (GUID)	"3528e452-ed32-4a4a-8ee5-6ee77e1dbc6e5"

A direct connection can be established via the name or via the GUID.

☞ Subdevices are defined using device name "/" subdevice name.

☞ In order to address a device on another computer via DCOM, the computer name or the IP address has to be placed in front of the device name.

E.g: "//<HostName>/Mlc01/Drive_0" or "//<IPAddr>/Mlc01/Drive_0".

4.3 Firmware

4.3.1 Overview

The firmware management items support a uniform access to the firmware of the individual devices.

Items for firmware detection

- **Firmware.DeviceFirmware**
  Reading firmware ID
- **Firmware.ComponentFirmware**
  Reading firmware ID of individual components
- **Firmware.ModuleFirmware**

Reading firmware modules of an individual component

- **Firmware.SubmoduleFirmware**
  Reading IDs of the submodules of a module

Items for download

- **Firmware.CheckDownloadRequirements**
  Checking the authorization for the firmware download
- **Firmware.Download**
  Downloading firmware or firmware parts into a device
- **Firmware.Reboot**
  Control reboot
- **Firmware.ResumeConnection**
  Detecting connection establishment after reboot
- **Firmware.Status**
  Reading download status
- **Firmware.StatusReset**
  Resetting download status information

Item	IndraDrive	IndraLogic	MLC	MTX
Firmware.DeviceFirmware Identification Layer 1	x	x	x	x
Firmware.ComponentFirmware Identification Layer 2	x			
Firmware.ModuleFirmware Identification Layer 3	x	x	x	x
Firmware.SubmoduleFirmware Identification Layer 4		x	x	

Tab. 4-2: Items for firmware detection

4.3.2 Firmware Items

Firmware.CheckDownloadRequirements

Description Checks whether a firmware download is allowed. If the download is not allowed, the reason is specified.

Item syntax **Firmware.CheckDownloadRequirements**

Call Synchronous/asynchronous

Access Read

Cyclic (opt.) Yes

Data format **Firmware.CheckDownloadRequirements, read**

row, col	Type	Description	Example
0,0	bool	Download allowed or not allowed	FALSE
0.1	long	Reason ID if the download is not allowed 1 - PLC program is running 2 - Sercos phase > 2	2

☞ **MLC**

Firmware.CheckDownloadRequirements uses the C-0-0023 Sercos parameter for the firmware download.

IndraDrive

Firmware.CheckDownloadRequirements uses the SIS services to exit and reboot.

Sercans

The download is always allowed.

Firmware.ComponentFirmware

Description Provides the ID and firmware ID of the individual components.

Item syntax **Firmware.ComponentFirmware**

Call Synchronous/asynchronous

Access Read

Cyclic (opt.) Yes

Data format Number of lines: Number of components

Firmware.ComponentFirmware

row, col	Type	Description	Example
x,0	long	Component ID	128
x,1	string	Component firmware ID	MPH-02T02.01-MS

Example IndraDrive with three components. The HPC component is provided with the MPH-02T02.01-MS firmware. The EMD components are provided with the EMD-02T01.01-MS firmware.

Firmware.ComponentFirmware

row, col	Type	Description
0,0	long	128
0.1	string	MPH-02T02.01-MS
1,0	long	1
1,1	string	EMD-02T01.01-MS
2,0	long	3
2,1	string	EMD-02T01.01-MS

☞ **IndraDrive, Sercans**

If the firmware download was aborted, the list of the firmware components could be incomplete.

IndraLogic, MTX, MLC

The Firmware.ComponentFirmware item is not necessary.

Firmware.DeviceFirmware

Description Provides the device firmware ID.

Item syntax **Firmware.DeviceFirmware**

Call Synchronous/asynchronous

Access Read

Cyclic (opt.) no

Data format **Firmware.DeviceFirmware, read**

row, col	Type	Description	Example
0,0	string	Firmware ID of the device	FWA-INDRV*-MPH-02T02.01-MS

☞ **IndraDrive**

The IndraDrive firmware update with MMC is only performed after an IndraDrive restart. The firmware ID can be read now.

MLC

Firmware.DeviceFirmware uses the C-0-0012 Sercos parameter to determine the device firmware ID.

Firmware.Download

Description Firmware or firmware parts are downloaded to a device (as continuous process).

Item syntax **Firmware.Download**

Call Asynchronous

Access Call

Cyclic (opt.) no

☞ The parameters are transferred via an SCPData object as key value pairs, whereas the parameters to be transferred depend on the respective device type.

The item returns if a communication is established again after a download or if an error occurred.

The following table provides an overview on the possible parameters

Parameter values

Key	Description	Type	Note
Firmware	Absolute path of the firmware file or the firmware directory	string	
CheckDownloadRequirements	Before downloading, check whether a download is allowed (e.g. PLC stopped, Sercos phase <=2)	bool	Optional Default = TRUE
Reboot	Reboot after download	bool	Optional Default = TRUE
Component	Component ID. Determines the device component to which a ComponentFirmware or a ModuleFirmware is loaded	string	Optional

Key	Description	Type	Note
ComponentFirmware	Component firmware ID. Specifies the component firmware loaded to the component <n> or specifies the component firmware to which the ModuleFirmware<n> is to be loaded	string	Optional
ModuleFirmware<n>	Module firmware ID. Specifies the ModuleFirmware to be loaded to the ComponentFirmware<n> of the Component<n>	string	Optional
UserLevel	1: Default user 2: Service 3: Engineer	long	Optional Default = 1
Date	Download date Format: "YYYY.MM.DD" Only for IndraDrive	string	Optional
Time	Download time Format: "HH:MM:SS" Only for IndraDrive	string	Optional
Comment	Download comment Only for IndraDrive	string	Optional

Example Firmware.Download

row, col	Type	Description	Example
0,0	string	Key: "Firmware"	Firmware
0.1	string	Value: Absolute path of the firmware file or the firmware directory	C:\Program Files\... \Firmware \FWAMPH02T02_ 02.IBF
1,0	string	Key: "CheckDownloadRequirements"	CheckDownload Requirements
1,1	bool	Value Optional, default = TRUE Determining whether download conditions are checked (e.g. PLC stopped, Sercos phase <= 2)	TRUE

Result Empty

☞ **IndraDrive**

- In user level "1", the download is only executed if firmware modules are available for all components.
- With plugged-in MMC card, the FWA download is only possible with compatible IBF files with compatibility list.
- The FirmwareComponent download requires Component<n> and ComponentFirmware<n>.
- The module download requires Component<n>, ComponentFirmware<n> and ModuleFirmware<n>.

MTX

The subsystem download requires the respective ModuleFirmware<n>.

MLC

In the "FTP_cmds.txt" file, description of a download, only the following commands can be used: update, bin, asc, put <file>, put <file1> <file2>, mkd, del, bye.

Error codes Empty

The item uses SCPErrNACK6 (106) to display an error.

Error codes

Detailed error number <long>	Description
0	No error
1	Firmware file or firmware directory is missing
2	Firmware file is defective (format error) or incomplete
3	The download is not allowed in the current device status
4	Device error (hardware problem or device firmware problem in the target device)
5	Error while verifying download
6	Error while writing

Tab. 4-3: Item Firmware.Download error codes

Firmware.ModuleFirmware

Description Provides the firmware modules of a component.

Remark Specification of a component ID is optional as not every device is provided with components. Logical devices without components do not support this firmware item.

Item syntax *Firmware.ModuleFirmware,(1)*

1. Component ID

Call Asynchronous

Access Read

Cyclic (opt.) no

Data format Number of lines: Number of modules

Firmware.ModuleFirmware

row, col	Type	Description	Example
x,0	string	Module ID	1
x,1	string	Module firmware ID	

Example Firmware.ModuleFirmware,128

row, col	Type	Description	
0,0	string	1. Module ID	1
0.1	string	1. Module firmware ID	MPH-02T02.01-MS - Application
1,0	string	2. Module ID	2
1,1	string	2. Module firmware ID	MPH-02T02.01-MS - Loader

☞ **IndraDrive, Sercans**

If the firmware download has been aborted, the list of the firmware modules could be incomplete.

Firmware.Reboot

Description Writing this item causes a control reboot. The item is not provided with any parameters.

Item syntax Firmware.Reboot

Call Synchronous/asynchronous

Access Write

Cyclic (opt.) no

☞ **MLC**

Firmware.Reboot uses the C-0-1020 Sercos parameter for the reboot.

Firmware.ResumeConnection

Description This item is used after a completed firmware download to detect when a connection to the device was established again. The item is not provided with any parameters.

Item syntax Firmware.ResumeConnection

Call Synchronous/asynchronous

Access Write

Waits either until a connection with the device has been established or until a timeout is reported.

Cyclic (opt.) Yes

☞ **MLC**

Firmware.ResumeConnection repeatedly reads the C-0-0012 Sercos parameter in order to detect a reestablished connection.

Firmware.Status

Description The download status is read.

Item syntax **Firmware.Status**

Call Synchronous/asynchronous

Access Read

Cyclic (opt.) Yes

Data format **Firmware.Status**

row, col	Type	Description	Example IndraDrive Status
0,0	long	Status ID 0 - Idle 1 - downloading 2 - Downloaded 3 - Canceling 4 - Canceled 5 - Error	1
1,0	string	Current device firmware configuration	FWA-INDRV*-MPH-02T02.01-MS
2,0	string	Current component ID	HPC01.1
3,0	string	Current component firmware ID	MPH-02T02.01-MS
4,0	string	Current module firmware ID	MPH-02T02.01-MS - Loader

row, col	Type	Description	Example IndraDrive Status
5,0	long	SIS download progress ID 0 - Idle (download not yet started or already completed) 1 - check download requirements 2 - get configured baudrate of device 3 - set communication parameters at loader default parameters 4 - read backup medium configuration state 5 - identify device configuration 6 - Get module information 7 - Get component mode 8 - Shutdown 9 - Set communication parameters at maximum transfer rate 10 - Delete firmware 11 - Write file info (MMC) 12 - Write file (MMC) 13 - Flash firmware 14 - Calculate checksum 15 - reboot 16 - Waiting until connection is restored 17 - Restore communication parameter 18 - Get device baud rate 19 - Get maximum download baud rate FTP download progress ID 0 - Idle (download not yet started or already completed) 1 - Extract firmware file 2 - check download requirements 3 - Get communication parameters 4 - downloading 5 - reboot 6 - resume connection	9
6,0	long [0..100]	Progress in progress ID (in percent) FTP: Progress refers to the complete download procedure SIS: Progress refers to the respective module	43
7,0	long	Number of modules	2

row, col	Type	Description	Example IndraDrive Status
8,0	long	Current module number	1
9,0	long	Error code	0
10,0	long	Detailed error code	0
11,0	string	Additional error information (e.g. file name)	

☞ **MLC, IndraLogic, MTX**

The current component ID and the current component firmware ID are currently empty.

Firmware.StatusReset

Description Resetting the download status information. The item is not provided with any parameters.

Item syntax **Firmware.StatusReset**

Call Synchronous/asynchronous

Access Write

Cyclic (opt.) no

Firmware.SubmoduleFirmware

Description Provides the submodule IDs in the module of a component.

Item syntax *Firmware.SubmoduleFirmware,(1),(2)*

1. Module ID
2. Component ID (optional)

Call Asynchronous

Access Read

Cyclic (opt.) no

Data format Number of lines: Number of submodules

Firmware.SubmoduleFirmware, read

row, col	Type	Description	Example
x,0	string	Submodule ID	1
x,1	string	Submodule firmware ID	

Example The firmware is provided with the 2 submodules 3S and Motion.

Firmware.SubmoduleFirmware,st

row, col	Type	Description	
0,0	string	1. Submodule - ID	3S
0.1	string	1. Submodule - firmware ID	IL-03V01-01-MS
1,0	string	2. Submodule - ID	Motion
1,1	string	2. Submodule - firmware ID	ML-03V01-01-MS

☞ **IndraDrive, MTX, Sercans**
The Firmware.SubmodulFirmware item is not necessary.

NC.LanguagesDownload

Description The items transfers, deletes and displays language texts. The transfer direction of the language files is from PC to NC only. Language files are only deleted in the /usrfep/text directory on the NC.

A read access to the item shows the language files on the installation directory of the PC and of the NC.

Remark The text files stored in the installation directory can either be of type ANSI, UNICODE or UTF8. Text files of the types ANSI and UNICODE are converted into UTF8 before the transfer. The UTF8 files are transferred without conversion.

When writing, the country code has to be specified with 3 digits.

Writing and deleting is very time-consuming in contrast to reading. Writing a complete language set requires 30 seconds. Deleting a language set requires approximately 15 seconds. Therefore, please consider that calling these items is executed asynchronously while writing or deleting.

To work faster with text files, an index file is automatically generated for each language file transferred to the NC. The index file is a binary file. Via the 0-based text number, there is a file pointer directly indicating the respective text in the respective language file. The index file is provided with the extended text file prefix "idx". The index file for the German system text is “idxsystexte.049”.

Item syntax **NC.LanguagesDownload,(1),(2)**

1. Path to system texts and OEM texts on the PC (optional)
2. Path to PFI texts on the PC (optional)

☞ If the item is created without path specification, it automatically determines the paths, in which the language files to be installed are located, from the Registry. The entry is: (HKEY_LOCAL_MACHINE\SOFTWARE\BOSCH\PC-PANEL \CurrentVersion\TextDir)

Restriction: Currently no PFI text processing!

Call Asynchronous

Access Read/Write

Cyclic (opt.) no

Data format **NC.LanguagesDownload, read**

row, col	Type	Description	Example: Reading all language files from the installation directory of the PC and of the NC
0,0	string	All language files in the installation directory	Systexts.049 28.01.2008 09:56:14 Systexts.044 24.01.2008 18:49:06 Oemtexts.049 24.01.2008 19:03:02 Oemtexts.044 24.01.2008 19:03:02
1,0	string	All language files within the NC	Systexts.049 28.01.2008 09:56:14 Oemtexts.049 24.01.2008 19:03:02

NC.LanguagesDownload, write

row, col	Type	Description	Example: Transferring language files from the installation directory of the PC to the installation directory of the NC
0,0	string	Specification of language file(s) - To be separated by a space in case of several files in the example: 044 = English language file 049 = German language file	049 044
1,0	long	Mode 0 = Write language file(s) 1 = Delete language file(s)	0

NC.LanguagesDownload, write

row, col	Type	Description	Example: Deleting language files from the installation directory of the NC
0,0	string	Specification of language file(s) - To be separated by a space in case of several files in the example: 044 = English language file 007 = Russian language file	044 007
1,0	long	Mode 0 = Write language file(s) 1 = Delete language file(s)	1

4.3.3 Error codes

Firmware management error uses SCPErrNACK6 (106) to display an error.

Detailed error number <long>	Description
0	No error
1	Firmware file or firmware directory is missing
2	Firmware file is defective (format error) or incomplete
3	The download is not allowed in the current device status
4	Device error (hardware problem or device firmware problem in the target device)
5	Error while verifying download

Tab. 4-4: Error codes in the firmware management

4.4 Sercos

4.4.1 Overview

Sercos parameters define a standardized access to the data of the Sercos devices. The schema of the Sercos parameters developed from the drive parameterization.

In Sercos, access to drive parameters is standardized.

There are more parameters described in the respective device documentation.

The Sercos ring is used as communication connection between controls and drives. On the Sercos ring, the participants are addressed via the "Sercos address". Controls usually implement the access to the drive parameters. The drive is addressed via the "Sercos address".

4.4.2 Sercos Idents

In order to identify a Sercos parameter, idents are used. They are coded as **strings**.

Ident format

X-Y-ZZZZ	16Bit ident number range	
X-Y-ZZZZ.SI[.SE]	32Bit ident number range	
X	Parameter type consisting of one letter. See also device description. Examples:	
	S	Standard parameter
	P	Product-specific parameter
Y	Parameter block. (0 ... 7)	
ZZZZ	Data block number (0000 ... 4095) Attention! The data block number is specified with 4 decimals.	
SI	Sercos instance Decimal without leading 0. (0..255)	
SE	Optional: Sercos element Decimal without leading 0. (0..255)	

Tab. 4-5: Sercos idents

4.4.3 Sercos Elements

Sercos parameters consist of several elements. Elements can be read explicitly.

Sercos elements

Element	Description	Display format	Example
Status	Parameter status	Hexadecimal 16 bits	"0x0000"
Name	Identifier of the Sercos parameter	String	"NC cycle time (TNcyc)"

Unit	Physical parameter unit	String	"µs"
Attribute	Type and scaling of the Sercos parameter	Hexadecimal 32 bits	"0x60110001"
Min	Minimum value	dep. on the attribute	"200.000"
Max	Maximum value	dep. on the attribute	"2000.000"
Value	Operating date of the Sercos parameter	dep. on the attribute	"1500.000"

Tab. 4-6: Sercos parameter elements

4.4.4 Sercos Attributes and Data Types

The type and the display format of the parameter are provided via the Sercos attribute. The attribute is transferred via a string, coded in 32-bit hex.

The device determines the attribute for a certain Sercos parameter.

☞ Binary data is converted to a display value in the communication interface.

Attribute coding

Description	Type	Bits
Evaluation factor	0..65535	0..15
Data length	0: 1 byte 1: 2 byte 2: 4 byte 3: 8 byte	16..17
List	0: No list 1: List	18
Parameter type	0: Operating date 1: Command	19
Display format	0: Binary 1: Decimal without sign 2: Decimal with sign 3: Hexadecimal without sign 4: String 5: Ident 6: Float 7: Sercos time	20..22

Decimal places	0..15	24..27
Write protection (CP2 ... CP4)		28..30

Tab. 4-7: Sercos parameter attribute coding

Evaluation factor A factor used to scale the device data. The evaluation factor is used for the data elements Min, Max and Value.

A calculation is carried out with the display formats binary, decimal and hexadecimal.

Data length Number of bytes. Used for Min, Max and Value. Also applies to the list elements. Strings are transferred with a data length of 1 byte.

List Flag to label lists.

Display format Interpretation labeling of the provided data. The data is interpreted according to the display format.

Decimal places Used in number formats for an additional evaluation.

Sercos display formats

Name	Data width in bytes	Coding	Example
Binary	2, 4, 8	0: FALSE 1: TRUE	"0000010100000100"
Decimal or sign	2, 4, 8	Unsigned decimal number	"12643" "123.234" for 3 decimals Evaluation factor and decimal points are taken into consideration.
Decimal w. sign	2, 4, 8	Signed decimal number	"432347" " -423823" "-24238.23" for 2 decimals Evaluation factor and decimal points are taken into consideration.
Hexadecimal	1, 2, 4, 8	Hexadecimal number	"0x1242" The evaluation factor is taken into consideration.
String	1	UTF8	"Hallo"
Ident	4	See chapter "Ident Formats"	"S-0-0030.4.6"
Float	4, 8	Floating point representation	"1.2354"
Sercos time	8	Acc. to IEC 61588	"2012-04-25T10:04:25Z 123456ns"

Tab. 4-8: Sercos parameter display formats

4.4.5 Sercos Lists

Operating data can also contain lists. This is indicated by the attribute.

Coding

List elements are separated by "\n".

Min., Max. and Values in lists

Specify the respective limit values for the elements in the list.

Transfer with element coding "D"

The value of the list is specified in one cell of the result.

Example:

Transfer with element coding "D"

SPA1, read

row, col	Type	Description	SPA1,3,S-0-0017,D
0,0	string	List	S-0-0001\n S-0-0002\n ... S-0-xxxx

Tab. 4-9: Sercos parameter list format "D"

Transfer with advanced element coding

In this transfer form, the list is transferred with the current length and the maximum length of the list.

Example:

List with the element coding 40

SPA1, read

row, col	Type	Description	SPA1,3,S-0-0017,40
0,0	string	Current list length	"400"
0.1	string	Maximum list length	"400"
0.2	string	List	"S-0-0001\n S-0-0002\n ... S-0-xxxx"

Tab. 4-10: Sercos parameter list format "40"

4.4.6 Requesting Several Elements

Via the Advanced format, several elements of a Sercos parameter can be read.

Element coding

Parameter element	Default format only one letter can be specified	Advanced format Values can be linked by means of "Or"
Data status	S	01H
Name	N	02H
Attribute	A	04H
Unit	U	08H
min. input value	L	10H

Max. input value	H	20H
Operating date	D	40H

Tab. 4-11: Sercos parameter element coding

Data format

The elements are returned in the above order

Example:

Requesting all Sercos elements

Reading result data from SPA1,3,S-0-0111,7f

row, col	Type	Description	SPA1,3,S-0-0111,7f
0,0	string	Data status	"0x0000"
1,0	string	Name	"Standstill current, motor"
2,0	string	Attribute	"0x43120001"
3,0	string	Unit	"A eff "
4,0	string	Min	"0.000"
5,0	string	Max	"500.000"
6,0	string	Value	"1.000"

Tab. 4-12: Sercos parameter order of the elements

Example:

Requesting all Sercos elements of a list parameter

Reading result data from SPA1,3,S-0-0017,7f

row, col	Type	Description	SPA1,3,S-0-0017,7f
0,0	string	Data status	"0x0000"
1,0	string	Name	"IDN list of all operating data"
2,0	string	Attribute	"0x70550001"
3,0	string	Unit	"--"
4,0	string	Min	"--"
5,0	string	Max	"--"
6,0	string	Current list length	"2116"
7,0	string	Maximum list length	"4986"
8,0	string	Value	"S-0-0000\nS-0-0001\n... ... P-0-4092\nP-0-4095"

Tab. 4-13: Sercos parameter, requesting all elements

4.4.7 Sercos Attribute Cache

Overview

In the communication layer, Sercos parameter elements are treated according to their structure. For this purpose, it is necessary to know the attribute value of the Sercos parameter. Usually, the attribute is always newly read again each time the Sercos parameter is accessed. This causes a

respective overhead. To avoid this, there is the attribute cache that buffers the attribute values of the Sercos parameters for future access.

It is possible to change attribute values in different Sercos parameters. These attribute values must not be cached. The attribute cache is activated via the configuration of certain Sercos parameters or parameter ranges.

Within a device, the cache is configured with regard to the axis. Also refer to SAC.Config, page 44

To configure the cache, auxiliary functions for the programming environment .NET are provided. See also SCPMnoSupport.dll, the classes Rexroth.IndraWorks.SCP.SPAAttributeCacheData and Rexroth.IndraWorks.SCP.SPAAttributeCache

4.4.8 Sercos Error Codes

Sercos errors are reported by the devices via the SCPErrNACK1 (101) IwSCP error.

sercos-specific errors are reported via a detailed error number.

Number (dec.)	Number (hexadec.)	Description
0	0x0000	No error
1	0x0001	Service channel is not opened
9	0x0009	Invalid closing of the service channel
144	0x0090	Control currently busy, request not possible. Try again later.
160	0x00A0	"Requirement invalid" e.g. access to the parameters S and P in initialization mode
176	0x00B0	"Selected element invalid" Only the "operating date" can be written
192	0x00C0	"Drive address invalid" The drive address is greater than permitted or drive is not active in the Sercos ring (deactivated or not available)
240	0x00F0	"Fatal error" At parameter transfer, a CLC internal error occurred (s. C-0-0041) that affects the data exchange
4097	0x1001	Ident number not present
4105	0x1009	Invalid access to element 1
5094	0x13E6	Conversion error of input/output format
5095	0x13E7	"File access error, check file system" Files are required to transfer a high amount of data. The memory of the RAM file system could be full.
8193	0x2001	Name not available
8194	0x2002	Name transfer too short

Number (dec.)	Number (hexadec.)	Description
8195	0x2003	Name transfer too long
8196	0x2004	Name cannot be changed
8197	0x2005	Name currently write-protected
12290	0x3002	Attribute transfer too short
12291	0x3003	Attribute transfer too long
12293	0x3004	Attribute cannot be changed
12294	0x3005	Attribute currently write-protected
16385	0x4001	Unit not available
16386	0x4002	Unit transfer too short
16387	0x4003	Unit transfer too long
16388	0x4004	Unit cannot be changed
16389	0x4005	Unit currently write-protected
20481	0x5001	Minimum input value is not available
20482	0x5002	Transfer of minimum input value too short
20483	0x5003	Transfer of minimum input value too long
20484	0x5004	Minimum input value cannot be changed
20485	0x5005	Minimum input value is currently write-protected
24577	0x6001	Maximum input value not available
24578	0x6002	Transfer of maximum input value too short
24579	0x6003	Transfer of maximum input value too long
24580	0x6004	Maximum input value cannot be changed
24581	0x6005	Maximum input value is currently write-protected
28674	0x7002	Operating date transfer too short
28675	0x7003	Operating date transfer too long
28676	0x7004	Operating date cannot be changed
28677	0x7005	Operating date currently write-protected
28678	0x7006	Operating date is smaller than the minimum input value
28679	0x7007	Operating date is larger than the maximum input value
28680	0x7008	Invalid operating date incorrect
28681	0x7009	Operating date is write-protected by a password
28682	0x700A	Operating date is currently write-protected as it was cyclically configured. (IDN configured in the MDT or AT, writing via service channel not allowed)

Number (dec.)	Number (hexadec.)	Description
28683	0x700B	Invalid indirect addressing (e.g. data container, list handling)
28684	0x700C	Operating date is currently write-protected due to other settings (e.g. parameters, operation mode, drive enable, drive ON)
28688	0x7010	Command is already active
28689	0x7011	Command cannot be interrupted
28690	0x7012	Command can currently not be executed (e.g. command cannot be activated in this phase)
28691	0x7013	Command cannot be executed (wrong parameters)
28940	0x710C	The entered value is no in the valid range of numbers
32769	0x8001	"Access currently not possible" Service channel assigned. No data transfer
32770	0x8002	"Failure in the service channel" The requested drive can currently not be accessed
32779	0x800B	Transfer has been aborted by the control as it has to communicate with the same drive (higher priority)
32780	0x800C	Access not allowed; service channel still active Latest transfer has not been completed yet and new request has been started

Tab. 4-14: Sercos parameter error codes

4.4.9 Sercos Items

SPA1

Description Reads and writes elements of the Sercos parameters.

Remark SPA1 provides access to control parameters if it is possible to access them in a Sercos parameter format. The control parameters are usually addressed via the drive address "0".

SPA1 also provides access to drive parameters if the drives are accessible via the controls. Drives are selected via the drive address.

Errors occurring when accessing Sercos parameters are displayed by the IwSCP error number "SCPErrNACK1".

Item syntax SPA1,(1),(2),(3)

1. Drive address

2. Ident

Ident format

X-Y-ZZZZ	16Bit ident number range
X-Y-ZZZZ.SI[.SE]	32Bit ident number range

X	Parameter type consisting of one letter. See also device description. Examples:	
	S	Standard parameter
	P	Product-specific parameter
Y	Parameter block. (0 ... 7)	
ZZZZ	Data block number (0000 ... 4095) Attention! The data block number is specified with 4 decimals.	
SI	Sercos instance Decimal without leading 0. (0..255)	
SE	Optional: Sercos element Decimal without leading 0. (0..255)	

Tab. 4-15: Sercos idents

3. Element coding

Element coding is used to address Sercos parameter data elements.

Element coding

Parameter element	Default format only one letter can be specified	Advanced format Values can be linked by means of "Or"
Data status	S	01H
Name	N	02H
Attribute	A	04H
Unit	U	08H
min. input value	L	10H
Max. input value	H	20H
Operating date	D	40H

Tab. 4-16: Sercos parameter element coding

Example:

SPA1,1,S-0-0003,L

Minimum input value of parameter S-0-0003 of drive 1

SPA1,1,S-0-0003,D:

Operating date of parameter S-0-0003 of drive 1

SPA1,1,S-0-0003,48:

Unit & operating date of parameter S-0-0003 of drive 1

Read The elements of the Sercos parameters are generally provided as strings.

Results are coded as follows:

Requested element	Result	Example
Data status	Hexadecimal word	"0x0000"
Name	String	"NC cycle time (TNcyc)"

Requested element	Result	Example
Attribute	Hexadecimal double word	"0x60110001"
Unit	String	"µs"
min. input value	Decimal word	"2000"
max. input value	Decimal word	"20000"
Value	decimal hexadecimal (max. 8 digits) binary (Max. 64 characters) Time (only for "S-" and "P"-parameters)	"1.5" "0x80" "1111000010101010" "2012-08-16T09:25:01.123456789Z"

Tab. 4-17: Return formats for Sercos elements

Write Only the operating date can be written if it is possible in the parameter.

Data format Number of lines:

n=Number of parameters requested

SPA1, read

row, col	Type	Description	SPA1,3,S-0-0111,08
0,0	string	Requested element	"A eff"

Tab. 4-18: SPA1, read

SPA1, write

row, col	Type	Description	SPA1,3,S-0-0111,40
0,0	string	Value to be assigned	"1.500"

Tab. 4-19: SPA1, write

Lists with simple format Requested with element coding "D"

SPA1, read

row, col	Type	Description	SPA1,3,S-0-0017,D
0,0	string	List	S-0-0001\n S-0-0002\n ... S-0-xxxx

Tab. 4-20: Lists with simple format - SPA1, read

Lists with length Requested with element coding "40".

SPA1, read

row, col	Type	Description	SPA1,3,S-0-0017,40
0,0	string	Actual length of the list in the drive	"400"
0,1	string	Max. length of the list in the drive	"400"
0,2	string	List	"S-0-0001\n S-0-0002\n ... S-0-xxxx"

Tab. 4-21: Lists with simple format - SPA1, read

SPA2

Description Reads and writes elements of the Sercos parameters of a drive.

Remark Errors occurring when accessing Sercos parameters are displayed by the IwSCP error number "SCPErrNACK1".

Item syntax SPA2,(1),(2)

1. Ident

Ident format

X-Y-ZZZZ	16Bit ident number range	
X-Y-ZZZZ.SI[.SE]	32Bit ident number range	
X	Parameter type consisting of one letter. See also device description. Examples:	
	S	Standard parameter
	P	Product-specific parameter
Y	Parameter block. (0 ... 7)	
ZZZZ	Data block number (0000 ... 4095) Attention! The data block number is specified with 4 decimals.	
SI	Sercos instance Decimal without leading 0. (0..255)	
SE	Optional: Sercos element Decimal without leading 0. (0..255)	

Tab. 4-22: Sercos idents

2. Element coding

Element coding is used to address Sercos parameter data elements.

Element coding

Parameter element	Default format only one letter can be specified	Advanced format Values can be linked by means of "Or"
Data status	S	01H

Name	N	02H
Attribute	A	04H
Unit	U	08H
min. input value	L	10H
Max. input value	H	20H
Operating date	D	40H

Tab. 4-23: Sercos parameter element coding

Example:

SPA2,S-0-0003,L:
Minimum input value of the parameter S-0-0003.
SPA2,S-0-0003,D:
Operating date of the parameter S-0-0003.
SPA1,S-0-0003,48:
Unit and operating date of the parameter S-0-0003.

Read The elements of the Sercos parameters are generally provided as strings.

Results are coded as follows:

Requested element	Result	Example
Data status	Hexadecimal word	"0x0000"
Name	String	"NC cycle time (TNcyc)"
Attribute	Hexadecimal double word	"0x60110001"
Unit	String	"µs"
min. input value	Decimal word	"2000"
Max. input value	Decimal word	"20000"
Value	decimal hexadecimal (max. 8 digits) binary (Max. 64 characters) Time (only for "S-" and "P"-parameters)	"1.5" "0x80" "1111000010101010" "2012-08-16T09:25:01.123456789Z"

Tab. 4-24: Return formats for Sercos elements

Write Only the value can be written.

Data format SPA2, read

row, col	Type	Description	SPA2,S-0-0111,08
0,0	string	Requested element	"A eff"

Tab. 4-25: SPA2, read

SPA2, write

row, col	Type	Description	SPA2,S-0-0111,40
0,0	string	Value to be assigned	"1.500"

Tab. 4-26: SPA2, write

Lists with simple format Requested with element coding "D".

SPA2, read

row, col	Type	Description	SPA2,S-0-0017,D
0,0	string	List	S-0-0001\n S-0-0002\n ... S-0-xxxx

Tab. 4-27: Lists with simple format - SPA2, read

Lists with length Requested with element coding "40".

SPA1, read

row, col	Type	Description	SPA2,S-0-0017,40
0,0	string	Actual length of the list in the drive	"400"
0,1	string	Max. length of the list in the drive	"400"
0,2	string	List	"S-0-0001\n S-0-0002\n ... S-0-xxxx"

Tab. 4-28: Lists with simple format - SPA1, read

DeviceAddress

Description Sercos address

Item syntax DeviceAddress

Read Determines the device address set on the device. The item is used especially if the device is accessed using point-to-point addressing, but the set device address turns out to be of interest anyhow.

Data format DeviceAddress

row, col	Type	Description	Example
0,0	long	Device address	5

DeviceAddressList

Description Drive addresses of a multi-axis device

Remark If a device disposed of several axes, the different axes are addressed via corresponding Sercos addresses.

Item syntax DeviceAddressList

Read Reading the drive addresses of the device.

Data format Per axis, one line (x) with the axis address is supplied.

DeviceAddressList

row, col	Type	Description	Example
x,0	long	Axis address	1

SAC.Config

Description Configuration of the attribute cache for Sercos parameter access. It is determined for each axis for which Sercos parameter the attribute can be buffered. Both, single idents and ident areas can be specified.

☞ The item is exclusively provided for the internal use. The configuration of wrong sercos idents may cause misinterpretations of Sercos parameter data.

Item syntax *SAC.Config[,(1)]*

1. Drive address

   The address of the drive the cache configuration of which is to be addressed.

   The parameter is necessary in Sercans and MLC. For IndraDrives and subdrives, the parameter is not to be specified.

☞ Control parameters are usually addressed via the drive address 0. --> SAC.Config,0

Ident Format Idents and area limits are to be specified in the format <Type>-<Set>-<Block>.

Type: One capital letter 'A' to 'Z'

Set: The parameter set 0 to 7

Block: The block number, 4-digit decimal number from 0000 to 4095.

Ident Areas They have 2 columns with min and max idents. The ident area is calculated from the respective min and max values of the specifications.

Example:

A-0-0100 to A-2-0200

All idents between A-0-0100 and A-0-200, between A-1-0100 and A-1-0200 and between A-2-0100 and A-2-0200

Availability SAC.Config Sercans, IndraDrive, EcoDrive, MLC, not MTX

Read Reading the idents and ident areas that are cached.

SAC.Config, Read

row, col	Type	Description	Example
0,0	string	Beginning of an ident area	C-0-0000
0,1	string	End of the ident area	C-3-0123
1,0	string	Single ident	C-0-0447

Write Writing ident areas and single idents.

SAC.Config, Write

row, col	Type	Description	Example
0,0	string	Beginning of an ident area	C-0-0000
0,1	string	End of the ident area	C-3-0123
1,0	string	Single ident	C-0-0447

When writing the configuration, the cached attribute values are deleted.

Call Deleting the cached attribute values.

SAC.Config, Call Argument

row, col	Type	Description	Example
0,0	string	Command: Has to contain Reset.	Reset

The cache configuration persists.

4.5 Diagnostics and Logbook

4.5.1 Overview

The diagnostics is one component in the MTX and in the IndraLogic devices allowing the access to messages and logbook. Messages are filed device-specifically. The corresponding items are available as part of the device functionality.

The logbook is available as individual part. The entries of different devices are collected here.

Diagnostic Items

- **System.MessageStore**

  Reading out the currently pending messages.
- **System.MessageStoreEx**

  Reading out the additional data of a message.
- **System.MessageStatus**

  Query whether messages are pending, changing or just came in.
- **System.ErrorStatus**

  Query whether messages of the "Error" classification are pending, changing or just came in.
- **System.InfoStatus**

  Query whether messages of the "Note" classification are pending, changing or just came in.
- **System.WarningStatus**

  Query whether messages of the "Warning" classification are pending, changing or just came in.
- **System.StartupStatus**

  Query whether messages of the startup prerequisite (startup status) classification are pending, changing or just came in.
- **System.SetupStatus**

Query whether messages of the 'Setup diagnostics" classification are pending, changing or just came in.

- **System.MessageReset**

  Resets the currently pending messages.

Logbook items

- **System.LogbookStore**

  Reading out the logbook data.

- **System.LogbookStoreEx**

  Reading out the additional data of a logbook entry.

- **System.LogbookClear**

  Deleting entries from the logbook.

- **System.LogbookDevices**

  Specifying the controls which are logged in the logbook.

- **System.LogbookStart**

  Starting the logbook logging.

Description

There are different types of diagnostic items:

- General commands

  With these commands, simultaneous requests of various types of diagnostic messages can be made, e.g. ProVi messages and MTX messages.

- Message-specific commands

  These messages have been defined specifically for a certain type of message, e.g. special commands for the ProVi criteria analysis

Even if the general commands provide data for all diagnostic types, different results are returned for each type.

Usually, there is always a defined part identical to all types, followed by the type-specific data. The type in question and the kind of additional data to be expected can be identified from a result in the defined part.

Parameter of the diagnostic items

The valid value ranges of the individual parameter types for the control diagnostic items are described in the following.

Filter: diagnostic type

There are different diagnostic types. They are defined in the IDL file "ScpDiagnosisComponent" by the enumerator "SCPDiagnosisType":

- "SCPDiagnosisType_MTX 0x01":

  MTX messages

- "SCPDiagnosisType_ProVi 0x02":

  ProVi messages

- "SCPDiagnosisType_IndraStep 0x04":

  SFC messages

If all diagnostic types are required, the value "SCPDiagnosisType_All" can be used. (In the item string, this value represents the string -1, e.g. System.MessagStore,-1,2,3,0).

To combine different types, the types in question can be connected via an OR operation.

Example:

MTX messages and ProVi messages

SCPDiagnosisType_MTX | SCPDiagnosisType_ProVi corresponds to 0x01 | 0x02 and results in 0x03

☞ Even if the value is defined as a hexadecimal value, a decimal value is expected as filter

Filter: rough classification

There are different types of diagnostic classes. They are defined in the IDL file "ScpDiagnosisComponent" by the enumerator "SCPDiagnosisClass":

SCPDiagnosisClass_Error	0x01	Error
SCPDiagnosisClass_Info	0x02	Note
SCPDiagnosisClass_Warning	0x04	Warning
SCPDiagnosisClass_Startup	0x08	Startup prerequisite
SCPDiagnosisClass_Setup	0x10	Setup diagnostics

Tab. 4-29:

If all diagnostic classes are required, the value "SCPDiagnosisClass_All" can be used. (In the item string, this value represents the string -1, e.g. System.MessagStore,2,-1,3,0).

To combine different classes, the types in question can be connected via an OR operation.

Example: Warning and setup diagnostics

SCPDiagnosisClass_Warning | SCPDiagnosisClass_Setup

corresponds to 0x04 | 0x10 and results in 0x14 as decimal value 20 in the string

Filter: error source

This filter defines the requested module number.

The requested module number is directly specified, e.g. System.MessageStore,2,1,3,0 (module number 3 is requested).

If data is to be requested for several modules, the module numbers are connected by a plus sign, e.g. System.MessageStore,2,1,2+3+5,0 (module numbers 2, 3, and 5 are requested).

If the data for all modules is to be requested or if no modules are defined, the value SCPErrorSource_All can be used. The latter is defined in the IDL file "ScpDiagnosisComponent" by the enumerator "SCPErrorSource". (In the item string, this value represents the string -1, e.g. System.MessagStore, 2,2,-1,0)

4.5.2 Diagnostics

System.MessageStore

Description Reading out all currently pending messages. When generating an item, the messages to be returned can be specified.

Remark This item is not suitable to record each message. If a message is only pending very shortly, it is not reported. If it is necessary to record each message, use the logbook.

Item syntax **System.MessageStore,(1),(2),(3),(4)**

1. Filter: diagnostic type. Several diagnostic types can be requested simultaneously.

SCPDiagnosisType_MTX	0x01	MTX messages
SCPDiagnosisType_ProVi	0x02	ProVi messages
SCPDiagnosisType_IndraStep	0x04	Step chain messages
SCPDiagnosisType_All	-1	all diagnosis types

2. Filter: rough classification. Several classifications can be requested simultaneously

SCPDiagnosisClass_Error	0x01	Error
SCPDiagnosisClass_Info	0x02	Note
SCPDiagnosisClass_Warning	0x04	Warning
SCPDiagnosisClass_Startup	0x08	Start requirement
SCPDiagnosisClass_Setup	0x10	Setup diagnosis

3. Filter: error source. Several error sources can be specified simultaneously.

The requested module number is specified

If data is to be requested for several modules, the module numbers are connected by a plus sign

If the data for all modules is to be requested or if no modules are defined, "–1" is specified.

4. Number of desired messages, 0 - All messages

Read One line is returned for each current message. If 3 messages have been requested for example, but only 1 message is pending, the result has only 1 line. The result has no line (0) if no message is pending.

The number of columns of a line depends on the diagnostic type of the message. Each line has a minimum of 9 columns defined as follows.

As every message can be unambiguously assigned to 1 diagnostic type, 1 rough classification and 1 error source, the returned results are unambiguous as well, i.e. there is no OR operation in the result anymore.

Example System.MessageStore

Data format For this item, different formats are provided depending on the message type. A new line (x) is provided for each message.

System.MessageStore, General part

row, col	Type	Description	Example
x,0	long	Diagnostic type of this message; also describes the data starting from column 9	
x,1	long	Rough classification of this message	
x,2	long	Error source of this message	
x,3	string	Error text with resolved placeholders	
x,4	double	Time stamp of the message with date and time of the "Variant Time" type. The time is given in Universal Time Coordinated (UTC). Filter: error source	

row, col	Type	Description	Example
x,5	long	Time index to sort messages with the same time stamp according to the time they occurred. If two messages have the same time index, the message with the smaller time index came in first. If both have the same index, they have been come in at the same time or the difference cannot be identified.	
x,6	bool	TRUE - Criteria analysis data available FALSE - No criteria analysis data available	
x,7	bool	TRUE - Further diagnostic information available (can be called via System.MessageStoreEx) FALSE - No further diagnostic information available	
x,8	string	Unique message ID. By this ID and the diagnostic type, messages can be distinguished even if several messages have the same message text. Is used as input parameter for further diagnostic commands.	

Additional columns of an MTX message

If "SCPDiagnosisType_MTX" is returned as diagnostic type, the columns are defined as follows:

System.MessageStore, addition MTX message

row, col	Type	Description	Example
x,9	string	Error, message number or note number	
x,10	long	Fine classification: message class (device-specific, ~ 20 with BRIC)	
x,11	string	Message class as text	
x,12	long	Channel or ID as channel-independent	
x,13	string	Channel name (configuration date)	

Additional columns of a ProVi message

If "SCPDiagnosisType_ProVi" is returned as diagnostic type, the columns are defined as follows:

System.MessageStore, addition ProVi message

row, col	Type	Description	Example
x,9	long	Message number	
x,10	long	Error category as number	
x,11	long	Message group as number	

Additional columns of an SFC message

If "SCPDiagnosisType_IndraStep" is returned as diagnostic type, the columns are defined as follows:

System.MessageStore, addition SFC message

row, col	Type	Description	Example
x,9	long	Message type 0 = Active step, 1 = Travel to initial position, 2 = Initial position reached, 3 = Monitor error, 4 = Time error, 5 = Start prerequisite message, 6 = Initial position Message	
x,10	string	SFC instance	

System.MessageStoreEx

Description Supplemented data to a message which were called up with System.MessageStore.

Remark The data type depends on the diagnostic type.

The result has always 1 line.

Item syntax *System.MessageStoreEx,(1),(2)*

1. Diagnostic type - Value returned in column 0 for System.MessageStore.
2. Unique message ID - Value returned in column 7 for System.MessageStore

Access Read

Data format The number of columns depends on the diagnostic type of the message. There is no uniformly defined part of the result. Depending on the diagnostic type, the following columns differ from each other.

Columns of an MTX message

If "SCPDiagnosisType_MTX" is returned as the diagnostic type, the columns are defined as follows.

System.MessageStoreEx, MTX message

row, col	Type	Description	Example
0,0	string	Cause text	
0,1	string	Remedy text	

Columns of a ProVi message

If "SCPDiagnosisType_ProVi" is returned as the diagnostic type, the columns are defined as follows.

System.MessageStoreEx, ProVi message

row, col	Type	Description	Example
0,0	string	Cause text from the message integrator	
0,1	string	Remedy text from the message integrator	
0,2	string	User notes from the message integrator	

row, col	Type	Description	Example
0,3	string	HTML link from the message integrator	
0,4	long	Type of POE in which the message is programmed. 1 = Program, 2 = Function block	
0,5	string	Name of POE in which the message is programmed.	
0,6	long	Type of detail in which the message is programmed. 1 = Implementation, 2 = Action 3 = Transition	
0,7	string	Name of detail, empty for implementation	
0,8	long	Network number or line in which the message is programmed	
0,9	string	Variable used as assignment for the ProVi message	
0,10	string	POE instance name	
0,11	string	Criteria analysis (for internal use only)	

Columns of an SFC message

If "SCPDiagnosisType_IndraStep" is returned as the diagnostic type, the columns are defined as follows.

System.MessageStoreEx, SFC message

row, col	Type	Description	Example
0,0	long	Type of POE in which the SFC is programmed. 1 = Program, 2 = Function block	
0,1	string	Name of POE in which the SFC is programmed.	
0,2	string	Step where the message comes from.	
0,3	long	Type of detail where the message comes from. 2 = Action 3 = Transition Only for message types "Monitor Error" and "Time Error". Always action for "Monitor Error".	
0,4	string	Action name from where the message comes. Only for message types "Monitor Error", "Time Error" and "Detail" = action.	

row, col	Type	Description	Example
0,5	long	Network number or line in which the message is programmed. Only for message type "Monitor Error".	
0,6	string	Criteria analysis Only for message type "Monitor Error", "Time Error" and "Startup prerequisite" message and "Initial Position" message. (for internal use only)	

System.MessageStatus

Description It is queried whether there are certain messages pending, whether these have changed or whether new messages just came in. When generating an item, the messages to be returned can be specified.

Remark This item is not suitable to record each message. If a message is only pending very shortly, it is not reported. If it is necessary to record each message, use the logbook.

Item syntax System.MessageStatus,(1),(2),(3)

1. Filter: diagnostic type. Several diagnostic types can be requested simultaneously

SCPDiagnosisType_MTX	0x01	MTX messages
SCPDiagnosisType_ProVi	0x02	ProVi messages
SCPDiagnosisType_IndraStep	0x04	Step chain messages
SCPDiagnosisType_All	-1	all diagnosis types

2. Filter: rough classification. Several diagnostic types can be requested simultaneously

SCPDiagnosisClass_Error	0x01	Error
SCPDiagnosisClass_Info	0x02	Note
SCPDiagnosisClass_Warning	0x04	Warning
SCPDiagnosisClass_Startup	0x08	Start requirement
SCPDiagnosisClass_Setup	0x10	Setup diagnosis

3. Filter: error source

The requested module number is specified

If data is to be requested for several modules, the module numbers are connected by a plus sign

If the data for all modules is to be requested or if no modules are defined, "–1" is specified.

Read

Example System.MessageStatus,?,?,?

Data format System.MessageStatus

row, col	Type	Description	Example
0,0	bool	TRUE - Messages are pending FALSE - No messages are pending	
0,1	double	Time stamp of the latest incoming message, with date and time of the "Variant Time" type. The time is given in Universal Time Coordinated (UTC).	
0,2	double	Time stamp of the message sent last, with date and time of the "Variant Time" type. The time is given in Universal Time Coordinated (UTC).	

Description

The results can be interpreted as follows:

- Are messages pending? For this purpose, column 0 is queried.
- Is the first message coming in or is the last message going out? Cyclical login of this item, when callback method is called, query column 0.
- Have messages been changed? Cyclical login of this item, when callback method is called, the messages have changed.
- Are there new messages? Keep the incoming time stamp. Cyclical login of this item, when the callback method is called and the incoming time stamp has changed, new messages came in.

System.ErrorStatus

Description It is queried whether there are errors pending, whether these have changed or whether new errors have been added. When generating an item, specify which errors to be returned.

Remark This item is not suitable to record each error. If an error message is only pending very shortly, it is not notified. If it is necessary to record each error, use the logbook.

Item syntax System.ErrorStatus,(1),(2)

1. Filter: diagnostic type. Several diagnostic types can be requested simultaneously.

SCPDiagnosisType_MTX	0x01	MTX messages
SCPDiagnosisType_ProVi	0x02	ProVi messages
SCPDiagnosisType_IndraStep	0x04	Step chain messages
SCPDiagnosisType_All	-1	all diagnosis types

2. Filter: error source. Several error sources can be specified simultaneously.

The requested module number is specified

If data is to be requested for several modules, the module numbers are connected by a plus sign

If the data for all modules is to be requested or if no modules are defined, "–1" is specified.

Read

Example System.ErrorStatus,?,?

Data format System.ErrorStatus

row, col	Type	Description	Example
0,0	bool	TRUE - Errors are pending FALSE - No errors are pending	
0,1	double	Time stamp of the latest incoming error, with date and time of the "Variant Time" type. The time is given in Universal Time Coordinated (UTC).	
0,2	double	Time stamp of the error sent last, with date and time of the "Variant Time" type. The time is given in Universal Time Coordinated (UTC).	

System.InfoStatus

Description It is queried whether there are notes pending, whether these have changed or whether new notes have been added. When generating an item, specify the notes to be returned.

Remark This item is not suitable to record each note. If a note is only pending very shortly, it is not notified. If it is necessary to record each note, use the logbook.

Item syntax System.InfoStatus

Read

Example System.InfoStatus

Data format System.InfoStatus

row, col	Type	Description	Example
0,0	bool	TRUE - Notes are pending FALSE - No notes are pending	
0,1	double	Time stamp of the latest incoming note, with date and time of the "Variant Time" type. The time is given in Universal Time Coordinated (UTC).	
0,2	double	Time stamp of the note sent last, with date and time of the "Variant Time" type. The time is given in Universal Time Coordinated (UTC).	

System.WarningStatus

Description It is queried whether there are warnings pending, whether these have changed or whether new warnings have been added. When generating an item, specify the warnings to be returned.

Remark This item is not suitable to record each warning. If a warning is only pending very shortly, it not notified. If it is necessary to record each warning, use the logbook.

Item syntax System.WarningStatus

Read

Example System.WarningStatus

Data format System.WarningStatus

row, col	Type	Description	Example
0,0	bool	TRUE - Warnings are pending FALSE - No warnings are pending	
0,1	double	Time stamp of the latest incoming warning, with date and time of the "Variant Time" type. The time is given in Universal Time Coordinated (UTC).	
0,2	double	Time stamp of the warning sent last, with date and time of the "Variant Time" type. The time is given in Universal Time Coordinated (UTC).	

System.StartupStatus

Description It is queried whether there are startup prerequisites pending, whether these have changed or whether new startup prerequisites have been added. When generating an item, specify the startup prerequisites to be returned.

Remark This item is not suitable to record each message. If a message is only pending very shortly, it is not notified. If it is necessary to record each message, use the logbook

Item syntax System.StartupStatus

Read

Example Concrete item with parameter, e.g. System.StartupStatus

Data format System.StartupStatus

row, col	Type	Description	Example
0,0	bool	TRUE - Start prerequisites are pending FALSE - No start prerequisites are pending	
0,1	double	Time stamp of the latest incoming startup prerequisite, with date and time of the "Variant Time" type. The time is given in Universal Time Coordinated (UTC).	
0,2	double	Time stamp of the startup prerequisite sent last, with date and time of the "Variant Time" type. The time is given in Universal Time Coordinated (UTC).	

System.SetupStatus

Description It is queried whether there are setup diagnostics pending, whether these have changed or whether new setup diagnostics have been added. When generating an item, specify the setup diagnostics to be returned.

Remark This item is not suitable to record each message. If a message is only pending very shortly, it is not notified. If it is necessary to record each message, use the logbook

Item syntax System.SetupStatus

Read

Example System.SetupStatus

Data format System.SetupStatus

row, col	Type	Description	Example
0,0	bool	TRUE - Setup diagnostics are pending FALSE - No setup diagnostics are pending	
0,1	double	Time stamp of the latest incoming setup diagnostics, with date and time of the "Variant Time" type. The time is given in Universal Time Coordinated (UTC).	
0,2	double	Time stamp of the setup diagnostics sent last, with date and time of the "Variant Time" type. The time is given in Universal Time Coordinated (UTC).	

System.MessageReset

Description Currently pending messages are reset. The item does not have any parameters. In 'Write', specify the messages to be reset.

Remark For future extensions. This item does not yet reset messages.

Item syntax System.MessageReset

Write

Example System.MessageReset

Data format System.MessageReset

row, col	Type	Description	Example
0,0	long string	Filter: diagnostic type. Several diagnostic types can be reset simultaneously.	
0,1	long string	Filter: rough classification. Several classifications can be reset simultaneously.	
0,2	long string	Filter: error source. Several error sources can be specified simultaneously.	

Filter: diagnostic type

SCPDiagnosisType_MTX	0x01	MTX messages
SCPDiagnosisType_ProVi	0x02	ProVi messages
SCPDiagnosisType_IndraStep	0x04	Step chain messages
SCPDiagnosisType_All	-1	all diagnosis types

Filter: rough classification

SCPDiagnosisClass_Error	0x01	Error
SCPDiagnosisClass_Info	0x02	Note
SCPDiagnosisClass_Warning	0x04	Warning
SCPDiagnosisClass_Startup	0x08	Start requirement
SCPDiagnosisClass_Setup	0x10	Setup diagnosis

Filter: error source

The requested module number is specified

If data is to be requested for several modules, the module numbers are connected by a plus sign

If the data for all modules is to be requested or if no modules are defined, "–1" is specified.

4.5.3 Logbook

System.LogbookStore

Description

Reading out the logbook messages.

Remark

When generating an item, the messages to be returned can be specified.

Each logbook entry is uniquely identified by the diagnostic type and a message ID.

If, for example, messages until time stamp X and from time stamp X are requested, both requests contain messages with the exact time stamp X. However, these double messages can be identified by the diagnostic type and the message ID.

This applies to outgoing messages as well. They are returned with a later time stamp. The diagnostic type and the message ID indicate to which incoming message they belong to.

Item syntax

System.LogbookStore,(1),(2),(3),(4),(5),(6)

1. filter: control

-1 messages of all configured controls of this logbook are returned.

If only the messages of a certain control are required, their GUID can be specified here.

2. filter: diagnostic type

Several diagnostic types can be requested simultaneously.

SCPDiagnosisType_MTX	0x01	MTX messages
SCPDiagnosisType_ProVi	0x02	ProVi messages
SCPDiagnosisType_IndraStep	0x04	Step chain messages
SCPDiagnosisType_All	-1	all diagnosis types

3. filter: rough classification

Several classifications can be requested simultaneously.

SCPDiagnosisClass_Error	0x01	Error
SCPDiagnosisClass_Info	0x02	Note
SCPDiagnosisClass_Warning	0x04	Warning
SCPDiagnosisClass_Startup	0x08	Start requirement
SCPDiagnosisClass_Setup	0x10	Setup diagnosis

4. filter: error source

Several error sources can be specified simultaneously.

The requested module number is specified

If data is to be requested for several modules, the module numbers are connected by a plus sign

If the data for all modules is to be requested or if no modules are defined, "–1" is specified.

5. Time stamp from which the messages are to be provided. Message with date and time of the "Variant Time" type. The time is given in Universal Time Coordinated (UTC).

All messages whose time stamp is more recent or identical are provided. If **0** is specified, all messages older than or identical to parameter 6 are returned. How old the measurement may be is not determined.

6. Time stamp up to which the messages are to be provided. Message with date and time of the "Variant Time" type. The time is given in Universal Time Coordinated (UTC).

All messages whose time stamp is older or identical are provided. If **0** is specified, all messages are returned that are newer than or identical to parameter 5. How recent the measurement may be is not determined.

Access Read

The number of columns of a line depends on the diagnostic type of the message.

Each message is uniquely assigned to 1 diagnostic type, 1 rough classification and 1 error source. Thus, the returned results are unambiguous. That mean that there is no OR operation in the result.

Each line has a minimum of 10 columns defined as follows.

Data format The number of lines corresponds to the number of logbook entries.

System.LogbookStore. General

row, col	Type	Description	Example
x,0	string	Device name	
x,1	string	Device GUID	
x,2	long	Diagnostic type of this message. Determines the data starting from column 10	
x,3	long	Rough classification of this message	
x,4	long	Error source of this message	
x,5	string	Error text with resolved placeholders	
x,6	double	Time stamp of the message with date and time of the "Variant Time" type. The time is given in Universal Time Coordinated (UTC)	
x,7	uint	1 - Message received 0 - Message output	
x,8	uint	1 – Further diagnostic information available (retrieve with System.LogbookStoreEx) 0 - No more diagnostic information available	
x,9	string	Unique message ID. By means of this ID and the type of diagnostic, messages can be unambiguously differentiated, for example to assign a departing message to the arriving message. Is used as input parameter for further diagnostic commands.	

Additional columns of an MTX message

If "SCPDiagnosisType_MTX" is returned as diagnostic type, the columns are defined as follows.

System.LogbookStore, addition MTX message

row, col	Type	Description	Example
x,10	string	Error, warning or note	
x,11	long	Fine classification Message class (device-specific, ~ 20 with BRIC)	
x,12	string	Message class as text	
x,13	long	Channel or ID as channel-independent	
x,14	string	Channel name (configuration date)	

Additional columns of a ProVi message

If "SCPDiagnosisType_ProVi" is returned as diagnostic type, the columns are defined as follows.

System.LogbookStore, addition ProVi message

row, col	Type	Description	Example
x,10	long	Message number	
x,11	long	Error category as number	
x,12	long	Message group as number	
x,13	bool	TRUE - Criteria analysis data available FALSE - No criteria analysis data available	

Columns of an SFC message

If "SCPDiagnosisType_IndraStep" is returned as diagnostic type, the columns are defined as follows.

System.LogbookStore, addition SFC message

row, col	Type	Description	Example
x,10	long	Message type 0 = Active step, 1 = Travel to initial position, 2 = Initial position reached, 3 = Monitor error, 4 = Time error, 5 = Start prerequisite message, 6 = Initial position Message	
x,11	string	SFC instance	
x,12	bool	TRUE - Criteria analysis data available FALSE - No criteria analysis data available	

System.LogbookStoreEx

Description Reading out more data regarding a logbook entry which has been queried with System.LogbookStore. The data type depends on the diagnostic type.

Item syntax System.LogbookStoreEx,(1),(2)

1. diagnostic type. Value which has been returned in column 0 for System.LogbookStore.

SCPDiagnosisType_MTX	0x01	MTX messages
SCPDiagnosisType_ProVi	0x02	ProVi messages
SCPDiagnosisType_IndraStep	0x04	Step chain messages
SCPDiagnosisType_All	-1	all diagnosis types

2. unique message ID. Value which has been returned in column 7 for System.LogbookStore

Read The number of columns depends on the diagnostic type of the message. In this case, there is no uniform part of the result.

Example

Data format

Columns of an MTX message

If "SCPDiagnosisType_MTX" is returned as the diagnostic type, the columns are defined as follows.

System.LogbookStoreEx, MTX message

row, col	Type	Description	Example
0,0	string	Cause text	
0,1	string	Remedy text	

Columns of a ProVi message

If "SCPDiagnosisType_ProVi" is returned as the diagnostic type, the columns are defined as follows.

System.LogbookStoreEx, ProVi message

row, col	Type	Description	Example
0,0	string	Cause text from the message integrator (currently still empty)	
0,1	string	Remedy text from the message integrator (currently still empty)	
0,2	string	User notes from the message integrator (currently still empty)	
0,3	string	HTML link from the message integrator (currently still empty)	
0,4	long	Type of POE in which the message is programmed. 1 = program, 2 = function block	
0,5	string	Name of POE in which the message is programmed.	
0,6	long	Type of detail in which the message is programmed. 1 = implementation, 2 = action, 3 = transition	
0,7	string	Name of detail, empty for implementation	
0,8	long	Network number or line in which the message is programmed	

row, col	Type	Description	Example
0,9	string	Variable used as assignment for the ProVi message	
0,10	string	POE instance name	

Columns of an SFC message

If "SCPDiagnosisType_IndraStep" is returned as the diagnostic type, the columns are defined as follows.

System.LogbookStoreEx, SFC message

row, col	Type	Description	Example
0,0	long	Type of POE in which the SFC is programmed. 1 = program, 2 = function block	
0,1	string	Name of POE in which the SFC is programmed.	
0,2	string	Step where the message comes from. Only for "Active Step" and "Time Error" message types	
0,3	long	Type of detail where the message comes from. 2 = action, 3 = transition. Only for "Monitor Error" and "Time Error" message types. Always "action" for "Monitor Error".	
0,4	string	Action name from where the message comes. Only for "Monitor Error", "Time Error" and "Detail" = "Action" message types.	
0,5	long	Network number or line in which the message is programmed. Only for message type "Monitor Error".	
0,6	string	String criteria analysis. Only for "Monitor Error", "Time Error", "Startup Status", "Reset", and "Diagnostic Type" message types.	

System.LogbookClear

Description Deleting logbook entries

The item is not provided with parameters. In "Write", specify which entries are to be deleted.

Remark When time ranges are specified, it is possible that the incoming entry of a message is deleted but its outgoing entry remains or vice versa.

Item syntax System.LogbookClear

Write Deleting the specified entries

Data format System.LogbookClear

row, col	Type	Description	Example
0,0	long string	Filter: diagnostic type. Several diagnostic types can be reset simultaneously.	-1
0,1	long string	Filter: rough classification. Several classifications can be reset simultaneously.	0x01
0,2	long string	Filter: error source. Several error sources can be specified simultaneously.	2+3+5
0,3	long string	Time stamp from which the entries are to be deleted. Message with date and time of the "Variant Time" type. The time is given in Universal Time Coordinated (UTC). Only the entries that are more recent or identical with regard to time are cleared. If 0 is specified, all entries whose time stamp is older than or identical to the parameter in column 4 are deleted. It is not determined how old the entry may be.	
0,4	double string	Time stamp up to which the entries are to be deleted. Message with date and time of the "Variant Time" type. The time is given in Universal Time Coordinated (UTC). All entries that are older or of identical date are deleted. If 0 is specified, all entries whose time stamp is more recent than or identical to the parameter in column 3 are deleted. It is not determined how old the entry has to be at least to be deleted.	

Filter: diagnostic type

SCPDiagnosisType_MTX	0x01	MTX messages
SCPDiagnosisType_ProVi	0x02	ProVi messages
SCPDiagnosisType_IndraStep	0x04	Step chain messages
SCPDiagnosisType_All	-1	all diagnosis types

Filter: rough classification

SCPDiagnosisClass_Error	0x01	Error
SCPDiagnosisClass_Info	0x02	Note
SCPDiagnosisClass_Warning	0x04	Warning
SCPDiagnosisClass_Startup	0x08	Start requirement
SCPDiagnosisClass_Setup	0x10	Setup diagnosis

Filter: error source

The requested module number is specified

If data is to be requested for several modules, the module numbers are connected by a plus sign

If the data for all modules is to be requested or if no modules are defined, "–1" is specified.

System.LogbookDevices

Description The logical devices that log their messages in the logbook are output.

Item syntax System.LogbookDevices

Write Device name and GUID

Data format System.LogbookDevices

row, col	Type	Description	Example
0,0	string	Name of the logic device	
0,1	string	GUID of the logic device	

System.LogbookStart

Description Starts logging the logbook data

The data is only written into the logbook if a general diagnostic item is logged in for the respective control.

This item can be used to log logbook data without another item already logged in.

Remark This item can be neither written nor read. It only has to be created.

Item syntax System.LogbookStart,(1)

Filter: control

-1: the messages of all configured controls in this logbook are logged.

If only the messages of a certain control are to be logged, the GUID can be specified here.

4.6 IndraLogic

4.6.1 Overview

Item overview

- Plc.PVT
  Reads the symbol information of a PLC project
- Plc.PVL
  Reads and writes variable values
- Plc.PVA
  Reads the addresses of symbolic PLC variables
- Plc.State2
  State information of the IndraLogic device component
- Plc.PVF
  Symbol information of a PLC variable
- Plc.ModStatus
  Status of the DP master
- Plc.ModDiagnose
  Diagnostics of a DP slave
- Plc.PVLK
  Access to PLC variables (structures, arrays)
- Plc.FirmwareVersion
  Reads the PLC firmware version
- Plc.FirmwareConnection
  Status of communication with the PLC
- Plc.FirmwareReboot

Activates the PLC reboot procedure

- **Plc.FirmwareStatus**
  PLC status
- **Plc.Plc.PVBS**
  Access to PLC variables (structures, arrays) as byte stream
- **Plc.StartStop**
  Switching PLC to RUN/STOP
- **Plc.FeldBus**
  Tunneling of the field bus data

4.6.2 Firmware

Plc.FirmwareConnection

Description Checks in time intervals of one second whether the communication with the PLC is present.

Remark Item is used during reboot.

Identical functionality as item **Firmware.ResumeConnection**

Item syntax **Plc.FirmwareConnection**

Write

Plc.FirmwareReboot

Description Starts the reboot in the PLC

Remark Communicating with the PLC during reboot!

Identical functionality as item **Firmware.Reboot**

Item syntax **Plc.FirmwareReboot**

Write

Plc.FirmwareStatus

Description Determines whether PLC status permits the start of the download

Remark Identical functionality as item **Firmware.CheckDownloadRequirements**

Item syntax **Plc.FirmwareStatus**

Read PLC running: "false" is provided

PLC in STOP: "true" is provided

Data format

Plc.FirmwareStatus, read

row, col	Type	Description	Example
0,0	string	Firmware status	false
0.1	long	???	1

Plc.FirmwareVersion

Description Determines the current PLC firmware version

Remark Identical functionality as item **Firmware.DeviceFirmware**

Item syntax **Plc.FirmwareVersion**

Read

Data format

Plc.FirmwareVersion, read

row, col	Type	Description	Example
0,0	string	Firmware version	FWC-ILsL40-RUN-03T00.00-NN

4.6.3 Info

Plc.ModDiagnose

Description Determines the diagnostics of a DP slave.

Remark The diagnostics applies to a slave of the DP master.

Item syntax **Plc.ModDiagnose,(1),(2),(3),(4),(5),(6),(7),(8)**

1. Master ID (currently 2000)
2. Master type ID (currently 1)
3. Master type index (currently 0)
4. Master bus address
5. Slave ID (currently 2001)
6. Slave ID (currently 1)
7. Slave index (currently 0)
8. Slave bus address

Read *Provides e.g. error information:*

- Error info flag 0 = Error info is a string
- Error info flag 1 = Error info is a binary array

Data format Number of columns: n = array length of the error info, for line 5

Plc.ModDiagnose, read

row, col	Type	Description	Plc.ModDiagnose, 2000,1,0,1,2001,1,0,3
0,0	WORD	Result flag (to be 0)	0
1,0	DWORD	Bus node ID	3
2,0	DWORD	Error ID	344
3,0	WORD	Error info flag	1
4,0	WORD	Array length of the error info	100
5,n	COM	Error info as string (`VT_BSTR`) or array (`VT_UI1`)	255

Plc.ModStatus

Description Determines the current DP master status.

Item syntax **Plc.ModStatus,(1),(2),(3),(4)**

1. Master ID (currently 2000)
2. Master type ID (currently 1)
3. Master type index (currently 0)
4. Master bus address

Read *Provides the DP master state and the slaves connected to the master:*

- Status 1 = OK
- Status 2 = Faulty
- Status 3 = Not communicating
- Status 4 = Stopped

Data format Number of columns: N = 130 for line 3 and line 4 (one column for each possible bus participant)

Plc.ModStatus, read

row, col	Type	Description	Plc.ModStatus,2000,1,0,1
0,0	WORD	Result flag (to be 0)	0
1,0	WORD	Number of bus participants	2
2,0	WORD	Bus master status	1
3,n	WORD	Node ID of the nth possible bus participant	2
4,n	WORD	Status of the nth possible bus participant	5

Plc.State2

Description State information of the device component and direct forwarding of the PLCHandler state.

Remark The call "AsyncRead" with "SCPCacheChanged" provides on a result if the state changes.

Item syntax **Plc.State2**

Read **Data format**

row/col	Type	Description
0,0	long	Status
0,1	string	Textual status description

Tab. 4-30: Read data format

Connecting states The following table shows the states of the PLC connection:

Connecting states

State	Brief textual description	Description
-1	STATE_TERMINATE	PLCHandler completed
0	STATE_PLC_NOT_CONNECTED	No connection to the PLC (initialization state)
1	STATE_PLC_CONNECTED	Connection to the PLC
2	STATE_NO_SYMBOLS	No symbols available or change of project
3	STATE_SYMBOLS_LOADED	Connection to PLC or symbols loaded.
4	STATE_RUNNING	Connection to PLC; symbols loaded and verified.

State	Brief textual description	Description
5	STATE_DISCONNECT	Connection interruption
6	STATE_NO_CONFIGURATION	Configuration of PLCHandler invalid.
7	STATE_PLC_NOT_CONNECTED_SYMBOLS_LOADED	PLC not available, but the symbols are available "offline". The PLCHandler permanently tries to connect to the PLC.

Tab. 4-31: State Machine

4.6.4 Intervention

Plc.StartStop

Description Switches PLC to RUN or to STOP

Item syntax **Plc.StartStop**

Write 0: Stop

1: Run

Data format **Plc.StartStop, write**

row, col	Type	Description	Example
0,0	long	Stop or Run	1

4.6.5 Field bus

Plc.FeldBus

Description Tunnels drive data via PLC to the Profibus gateway

Remark The item is to supplement or replace the current procedure via a serial interface.

Item syntax **Plc.FeldBus,(1),(2),(3),(4),(5)**

1. PB (data as BSTR as HexASCII byte sequence) or PBbin (data as BSTR as binary byte sequence)
2. Profibus master ID (e.g. 0 = Profibus DP/M OnBoard)
3. Profibus slave address
4. Slot index in the slave
5. Index in the slot

Read

Write

Data format **Plc.FeldBus, read**

row, col	Type	Description	Plc.FeldBus,PB,0,3,0,3
0,0	VT_BSTR	Reference data as BSTR	123456

4.6.6 Variables

Plc.PVA

Description Reads the addresses of symbolic variables

Remark The generation of symbol information has to be active with a PLC project. This is set in the menu under **Project ▸ Options** in the selection "Symbol configuration".

Item syntax **Plc.PVA,(1),...,(n)**

1...n: Symbolic variable names

Read The address value of invalid symbolic variables is 0. No error is generated.

Data format Number of lines: Corresponds to the number of PLC variables

Plc.PVA, read

row, col	Type	Description	Example
n,0	long	Address	30120947

Plc.PVArray

Description Reads the current values of an array variable from the PLC.

Remark Array variables correspond to the PVL variables for example.

Item syntax **Plc.PVArray,(1)**

1. Symbolic variable name without index

Read

Data format Complete content byte by byte

Plc.PVArray, read

row, col	Type	Description	Plc.PVArray,myArray Example: variable myArray consists of 4 byte elements
0,0	COM	Current value of the variables myArray (byte 0)	01
0.1	COM	Current value of the variables myArray (byte 1)	02
0.2	COM	Current value of the variables myArray (byte 2)	03
0,3	COM	Current value of the variables myArray (byte 3)	04

Plc.PVBS

Description Reads variable values from the PLC. The symbolic name of the PLC variables is given as a parameter. The variables only include structures and arrays. Reading or writing the individual elements of such variables requires the use of the "Plc.PVL" item.

Remark The generation of symbol information has to be active with a PLC project. This is set in the menu under **Project ▸ Options** in the selection "Symbol configuration". All corresponding points have to be activated.

When the item is used in an OPC client, only one PLC variable per item can be processed.

☞ No direct memory access is allowed under **IndraLogic 2G**. Thus, it has to be considered when using the item under IndraLogic IL2G that **not all data are written in one PLC cycle**.

Item syntax **Plc.PVBS,PLC_PRG.myStrukt1**

myStrukt1 variable name of data type "MyStruct" - a user-defined structure, e.g.:

```
TYPE myStrukt:
STRUCT
x: UINT;
y: UINT;
END_STRUCT
END_TYPE
```

Length of the structure in bytes: n=4

Plc.PVBS,PLC_PRG.myArr1

myArr1 variable name of data type "ARRAY" – a user-defined structure with the following elements:

Read

Write

Data format Number of columns: Number of bytes in the structure

Plc.PVBS, read

row, col	Type	Description	Plc.PVBS,PLC_PRG.myStruct1 (x := 3333, y :=4444)
0,n	byte	Byte of the structure	92

Plc.PVF

Description Reads symbol information from a PLC variable.

Item syntax **Plc.PVF,(1)**

1. PLC variable name

Read

Data format **Plc.PVF, read**

row, col	Type	Description	Plc.PVF,PLC_TYPES.bool1
0,0	string	Variable name	PLC_TYPES.bool1
0.1	string	Type	BOOL
0.2	long	Ref ID	4
0,3	long	Offset	1
0,4	long	Size	1
0,5	string	Access	b

Plc.PVL

Description Reads and writes variable values. The symbolic names of the PLC variables are given as parameters.

Remark The generation of symbol information has to be active with a PLC project. This is set in the menu under **Project ▸ Options** in the selection "Symbol configuration".

The values of the symbolic variables are converted into COM variable types. Arrays and structures are not supported:

Conversion

PLC variable type	COM variable type
USINT	VT_UI1
BYTE	VT_UI1
SINT	VT_I1
INT	VT_I2
WORD	VT_UI2
UINT	VT_UI2
DINT	VT_I4
DWORD	VT_UI4
UDINT	VT_UI4
DATE	VT_UI4
DT (DATE and TIME)	VT_UI4
TOD (Time of Day)	VT_UI4
BOOL	VT_BOOL
REAL	VT_R4
LREAL	VT_R8
STRING	VT_BSTR

PLC variables not listed here are converted according to their type size into VT_UI1, VT_UI2 und VT_UI4.

Item syntax **Plc.PVL,(1),...,(n)**

1...n: Symbolic variable names

Read

Write

Data format Number of lines: Corresponds to the number of parameters

Plc.PVL, read

row, col	Type	Description	Plc.PVL,PLC_TYPEN.bool,PLC_TYPEN.real
0,0	bool	Value of the first variable given	TRUE
...			
n,0	long	Value of the nth variable given	3.14

Plc.PVLK

Description Reads variable values from the PLC. The symbolic names of the PLC variables are given as parameters. The variables include both "simple" variables as well as structures and arrays.

Remark Variable names of all data types defined in the PLC program can be specified.

Item syntax **Plc.PVLK,PLC_PRG.myStrukt1,PLC_PRG.myArr1**

myStrukt1 *variable name of data type "MyStruct" – a user-defined structure with the following elements:*

- **myArr2**: Array [1..2] OF BYTE
- **x**: BOOL variable

- y: DWORD variable

myArr1 variable name of the data type ARRAY [1..2] OF UINT

Read

Data format

Number of lines: The number of complex variables present plus the elementary data present in the complex variables.

Plc.PVLK, read

row, col	Type	Description	Plc.PVLK,PLC_PRG.myStruct1, PLC_PRG.myArr1
0,0	string	Symbolic name of the complex variables	PLC_PRG.myStruct1
0.1	COM	Data type	DATA
0.2	long	Byte length of the variables	7
0,3	COM	Data value	DATA
1,0	string	symbolic name of the first element of the complex variables (0,0)	PLC_PRG.myStruct1.myArr2
1,1	COM	Data type	ARRAY
1,2	long	Byte length of the element	2
1,3	COM	Data value	ARRAY
2,0	string	symbolic name of the first element of the first variable element (1,0)	PLC_PRG.myStrukt1.myArr2[1]
2,1	COM	Data type	BYTE
2,2	long	Byte length of the element	1
2.3	COM	Data value	201
3,0	string	symbolic name of the second element of the first variable element (1,0)	PLC_PRG.myStrukt1.myArr2[2]
3,1	COM	Data type	BYTE
3,2	long	Byte length of the element	1
3.3	COM	Data value	202
4,0	string	symbolic name of the second element of the complex variables (0,0)	PLC_PRG.myStruct1.x
4,1	COM	Data type	BOOL
4,2	long	Byte length of the element	1
4,3	COM	Data value	TRUE
5,0	string	Symbolic name of the third element of the complex variables (0,0)	PLC_PRG.myStrukt1.y
5,1	COM	Data type	DWORD
5,2	long	Byte length of the element	4
5.3	COM	Data value	???

Plc.PVT

Description

Reads symbol information from a PLC project.

Remark The generation of symbol information has to be active with a PLC project. This is set in the menu under **Project ▸ Options** in the selection "Symbol configuration".

Item syntax **Plc.PVT**

Read

Data format Number of lines: Corresponds to the number of PLC variables

Plc.PVT

row, col	Type	Description	Example
n,0	string	Name	PLC_TYPES.bool1
n,1	string	Type	BOOL
n,2	long	Reference ID	4
n,3	long	Offset	1
n,4	long	Size	1
n,5	string	Access	b

4.7 NC - Global

4.7.1 General information

NC.AxisNamesPhys

Description Provides the physical axis name of all axes to be parameterized

Item syntax **NC.AxisNamesPhys**

Call Synchronous/asynchronous

Access Read

Cyclic (opt.) Yes

Data format **NC.AxisNamesPhys - read**

row, col	Type	Description	Example
0,0	string	Name of the first axis	K1X

NC.BitInterface

Description Provides access to the signals of the bit interface inputs/outputs. In addition to the current state of the inputs and outputs, the function and the respective signal name are provided.

Parameters specify the type of interface (global interface, channel interface, axis interface, spindle interface) as well as the desired instance (channel number, axis number, spindle number).

Remark The number of provided entries varies according to the interface range requested. The number of outputs and inputs is thus returned in the first line of the read data. The other lines contain the PLC outputs, followed by the PLC inputs.

The signal descriptions are provided in the preset language.

To specify the interface range, use the "LDX_AcbIfType" enumerator:

Enumerator LDX_AcbIfType

LDX_AcbIfType	Description
LDX_ACB_IFTYPE_CHAN_C	Channel interface
LDX_ACB_IFTYPE_AXIS_C	Axis interface
LDX_ACB_IFTYPE_SPINDLE_C	Spindle interface
LDX_ACB_IFTYPE_GENERAL_C	Global interface

Item syntax NC.BitInterface,(1),(2)

1. Type - Interface range (0 = channel, 1 = axis, 2 = spindle, 3 = global interface)

2. Index - for channel, axis or spindle

Call Synchronous/asynchronous

Access Read

Cyclic (opt.) Yes

Data format NC.BitInterface, read

row, col	Type	Description	NC.BitInterface,0,3 Request channel interface of channel 3
0,0	long	Number of PLC outputs	112
0.1	long	Number of PLC inputs	112
1,0	string	Function of the first output	Operation mode selection B0
1,1	string	Signal name	qCh_OpModeSel_00
1,2	long	Current state of the first output	1
...	...	Information on all further outputs	...
x,0	string	Function of the first input	Active operation mode of Bit0
x,1	string	Signal name	iCh_OpMode_00
x,2	long	Current state of the first input	1
...	...	Information on all further inputs	...

NC.ConnectionQuality

Description Provides the current connection state (good/bad) as well as the CPU load in percent.

Remark *The "LDX_ConnectionQuality" enumerator is available to evaluate the provided connection state and can accept the following values:*

- LDXConnectionQualityBad - No connection to the MTX kernel
- LDXConnectionQualityGood - Connection to MTX kernel exists

Item syntax NC.ConnectionQuality

Call Synchronous/asynchronous

Access Read

Cyclic (opt.) Yes

Data format NC.ConnectionQuality - read

row, col	Type	Description	Example
0,0	long	Connection state 2 = Communication connection 0 = No communication	2
0.1	long	Processor load in percent	15

NC.CpuUsage2

Description Provides the current CPU load of the NC

Remark *The following information is provided for different tasks or task groups:*

- ID
- Current CPU load in percent
- Evaluation of the CPU load
- CPU core ID (1..n)
- English short term

The enumerator "LDX_CpuUsageEntryId" provides the ID of the task or the task group:

Enumerator LDX_CpuUsageEntryId

LDX_CpuUsageEntryId	Description
LDX_CPU_USAGE_IPO_C	Interpolator
LDX_CPU_USAGE_PLC_C	PLC user tasks and PLC system tasks
LDX_CPU_USAGE_SAV_CPL_C	Block processing and CPL
LDX_CPU_USAGE_SYS_C	Other tasks (communication, file server, database)
LDX_CPU_USAGE_IDLE_C	Idle task (free capacity)

The enumerator "LDX_CpuUsageEval" provides the evaluation of the CPU load by the task or the task group:

Enumerator LDX_CpuUsageEval

LDX_CpuUsageEval	Description
LDX_CPU_USAGE_EVAL_OK_C	Normal load
LDX_CPU_USAGE_EVAL_HIGHLOAD_C	High load
LDX_CPU_USAGE_EVAL_OVERLOAD_C	Overload
LDX_CPU_USAGE_EVAL_INVALID_C	No evaluation possible (compatibility)

☞ Controls with older software releases provide for:

- Evaluation of the CPU load >>> **LDX_CPU_USAGE_EVAL_INVALID_C**
- CPU core ID >>> **0**

Item syntax NC.CpuUsage2

Call Synchronous/asynchronous

Access Read

Cyclic (opt.) Yes

Data format Number of lines: Number of tasks or task group

NC.CpuUsage2 - read

row, col	Type	Description	Example
n,0	long	ID	LDX_CPU_USAGE_IPO_C
n,1	long	CPU load in percent	5
n,1	long	Evaluation of the CPU load	LDX_CPU_USAGE_EVAL_OK_C
n,1	long	CPU core ID	1
n,2	string	English short term	IPO

NC.DryRunMode

Description Indicates the status of the complete lock.

Item syntax **NC.DryRunMode**

Call Synchronous/asynchronous

Access Read

Cyclic (opt.) Yes

Data format **NC.DryRunMode, read**

row, col	Type	Description	Example
0,0	bool	Status of the complete lock	TRUE

NC.Filesystem

Description *Provides information on internal and integrated file systems of the control:*

- Number of integrated file systems
- Type of mounted file system (PxFile,NFS)
- Mount conditions of these file systems
- Directory name (mount point)
- *Integrated file system name:*
  - for a local file system: Device name
  - for NFS: host name of the NFS server and the directory name exported by the NFS server

Remark The "LDX_MountType" enumerator provides the type of the mounted file system:

Enumerator LDX_MountType

LDX_MountType	Description
LDX_FSR_FSMOUNT_UNKNOWN_C	Unknown file system
LDX_FSR_FSMOUNT_LOCAL_C	Local PxFile file system
LDX_FSR_FSMOUNT_NFS_C	NFS file system
LDX_FSR_FSMOUNT_NOFOUND_C	No file system found

The "LDX_MountMode" enumerator provides the type of the file system integration.

Enumerator LDX_MountMode

LDX_MountMode	Description
LDX_FSR_FSMOUNT_RDONLY_C	Read-only
LDX_FSR_FSMOUNT_CACHABLE_C	Access is "cached"
LDX_FSR_FSMOUNT_FEPROMWRITEABLE_C	Writable FEPROM file system
LDX_FSR_FSMOUNT_RDWR_C	Read and write

Item syntax NC.Filesystem

Call Synchronous/asynchronous

Access Read

Cyclic (opt.) no

Data format Number of lines: Number of integrated file systems

NC.Filesystem, read

row, col	Type	Description	Reading all file system information
n,0	long	Mounted file system type	LDX_FSR_FSMOUNT_NFS_C
n,1	long	Mount conditions of the file system	LDX_FSR_FSMOUNT_ RDONLY_C
n,2	string	Directory name	/mnt
n,3	string	File system name	

NC.CheckFile

Description Delivers information on a file. The second parameter (mode) controls which information is requested.

Remark The following modes are supported:

0 - Is the file encoded ?

1 -What is the encoding comment?

Item syntax NC.CheckFile,(1),(2)

1. File name with path

2. Mode

Call Synchronous/asynchronous

Access Read

Cyclic (opt.) no

Data format NC.CheckFile, read

row, col	Type	Description	Example: NC.CheckFile,/usr/user/sekt.cnc,0
0,0	long	0 = File is not encoded 1 = File is encoded	1

NC.CheckFile, read

row, col	Type	Description	Example: NC.CheckFile,/usr/user/sekt_crypt.cnc,1
0,0	string	Comment from the file	** File version 1.1.1 **

NC.FIS

Description Provides an unknown pointer to the instance of the Fis server (IID_IFIS).

Remark This pointer allows the access to the 3 interfaces of the Fis server "IID_IFISDirectory", "IID_IFISFile" and "IID_IFISFileSystem".

Item syntax NC.FIS

Call Synchronous/asynchronous

Access Read

Cyclic (opt.) no

Data format NC.FIS

row, col	Type	Description	Example
0,0	string	Unknown indicator	LPUNKNOWN

NC.MtxCommand

Description The item provides status information or causes state changes in the control. This depends on the command specification.

☞ The item only applies for CMLxx controls

Item syntax NC.MtxCommand

Call Asynchronous

Access Read

- Reading different status information as XML string

☞ **Restriction:** The read function is optimized for a cyclic query. Thus, the read access to the item may only be carried out within a group. The first read accesses can provide "SCPQualityUncertain".

Write

- Writing different commands for the control functions that can be varied via an additional command

Cyclic (opt.) Read only

Data format

NC.MtxCommand, read

row, col	Type	Description of possible status information	Example
0,0	string (XML stream)	StartupPhase - Current startup phase StartupMode - Set startup mode Ready 0 = no Ready 1 = Ready active -1 = Undefined PlcState - PLC state 0 = PLC is at rest 1 = PLC is running -1 = Undefined FieldBus - Field bus state 0 = Field bus inactive 1 = Field bus active -1 = Undefined ScsState - Current Sercos phase CritSysState - State of critical system errors 0 = No error 1 = Error -1 = Undefined ShutdownState - Shutdown state of the control der Steuerung 0 = No shutdown 1 = Shutdown active 2 = Shutdown completed -1 = Undefined BootPanicState - State of boot panic error 0 = No error 1 = Error 2 = Error of boot loader -1 = Undefined BootPanicError - Boot panic error no. Battery - Buffer battery state 0 = Defective 1 = OK -1 = Undefined FanState - Fan state 0 = Defective or missing 1 = OK -1 = Undefined TargetTimeS - UTC time in the control	<SysCtrl> <SysState> <StartupPhase>0</StartupPhase> <StartupMode>0</StartupMode> <Ready>1</Ready> <PlcState>1</PlcState> <FieldBus>0</FieldBus> <ScsState>4</ScsState> <CritSysState>0</CritSysState> <ShutdownState>0</ShutdownState> <BootPanicState>0</BootPanicState> <BootPanicError>0</BootPanicError> <Battery>1</Battery> <FanState>1</FanState> <TargetTimeS>1334736235</TargetTimeS> </SysState> </SysCtrl>

NC.MtxCommand, write

row, col	Type	Description of possible commands and additional options	Example: Command "4" + "6" "Startup with bootstrapping"
0,0	long	The following **commands** are possible: 1 or 2 = Control reset 3 = Control shutdown 4 = Setting the startup mode 10 = Starting and stopping the PLC 11 = Create core dump	4
1,0	long	**This specification is omitted for the commands 1, 2 and 11** **Options for command 3:** 0 = Only shutdown (default) 1 = Shutdown with subsequent reset **Options for command 4:** Setting value range [0..15] of the startup mode: 0 = Normal mode 1 = PLC stop 2 = Reloading the PLC boot project 3 = Safe startup 4 = Deleting permanent CPL variables and system data 5 = Cold start 6 - Bootstrapping 7 = Creating the user FEPROM file system 8 - 15 = internal **Options for command 10:** 1 = PLC start 0 = PLC stop	6

NC.NofChannels

Description Provides the number of parameterized channels.

Item syntax **NC.NofChannels**

No parameters

Call Synchronous/asynchronous

Access Read

Cyclic (opt.) no

Data format **NC.NofChannels**

row, col	Type	Description	Example
0,0	long	Number of channels	4

NC.OsciUtil

Description The item allows the access to the oscilloscope functionality.

Remark 22 functions provide initialization options and allow reading initialization data as well as recording values. Also status information on the current oscilloscope state can be retrieved.

Item syntax **NC.OsciUtil**

No parameters

Call Synchronous/asynchronous

Access Read

Reading the currently active status information.

The following five states are defined:

- 0: NotReadyForRecording
- 1: ReadyForRecording
- 2: RecordingActive
- 3: RecordingTriggered
- 4: RecordingCompleted

Call

Reading the oscilloscope data requested by the commands

☞ The response data always provides an error number in the first line. All following lines provide user data.

Cyclic (opt.) Yes

Data format **NC.OsciUtil, call - Command**

row, col	Type	Description	Example: "Connect"
0,0	string	Establishing connection to the control and reserving resources	Connect

NC.OsciUtil, call - Response data

row, col	Type	Description	Example: "Connect" - Possible response
0,0	long	Possible acknowledgement messages	0 = Connection established -3 = No connection - Memory reservation failed -4 = Connection already established

NC.OsciUtil, call - Command

row, col	Type	Description	Example: "Disconnect"
0,0	string	Closing connection and sharing resources	Disconnect

NC.OsciUtil, call - Response data

row, col	Type	Description	Example: "Disconnect" - Possible response
0,0	long	Possible acknowledgement messages	0 = No error

NC.OsciUtil, call - Command

row, col	Type	Description	Example: "Start"
0,0	string	Starting a recording	Start

NC.OsciUtil, call - Response data

row, col	Type	Description	Example: "Start" - Possible response
0,0	long	Possible acknowledgement messages	0 = 1 = File could not be created 2 = No memory available 3 = Memory reserved 4 = File server conflict 5 = Start already initiated 6 = Incorrect parameters 7 = Trigger condition incorrect 8 = SCS communication error 10 = Invalid status for start specification 11 = At least one channel has to be selected 12 = Selected channels have to be active 13 = Incorrect recording time 14 = Incorrect mode

NC.OsciUtil, call - Command

row, col	Type	Description	Example: "Stop"
0,0	string	Stopping a recording	Stop

NC.OsciUtil, call - Response data

row, col	Type	Description	Example: "Stop" - Possible response
0,0	long	Possible acknowledgement messages	0 = Recording correctly stopped For negative values according to the error description, see below

NC.OsciUtil, call - Command

row, col	Type	Description	Example: "Trigger"
0,0	string	Initiating trigger	Trigger

NC.OsciUtil, call - Response data

row, col	Type	Description	Example: "Trigger" - Possible response
0,0	long	Possible acknowledgement messages	0 = Trigger executed For negative values according to the error description, see below

NC.OsciUtil, call - Command

row, col	Type	Description	Example: "GetSampleTime"
0,0	string	Reading recording period	GetSampleTime

NC.OsciUtil, call - Response data

row, col	Type	Description	Example: "GetSampleTime"
0,0	long	Acknowledgement message	0 = No error
1,0	double	Recording time	4.0

NC.OsciUtil, call - Command

row, col	Type	Description	Example: "SetSampleTime"
0,0	string	Setting recording period	SetSampleTime
1,0	double	Recording time	16.0

NC.OsciUtil, call - Response data

row, col	Type	Description	Example: "SetSampleTime" - Possible response
0,0	long	Acknowledgement message	0 = No error

NC.OsciUtil, call - Command

row, col	Type	Description	Example: "GetSignalIds"
0,0	string	Reading ID from selected signals	GetSignalIds

NC.OsciUtil, call - Response data

row, col	Type	Description	Example: "GetSignalIds"
0,0	long	Acknowledgement message	0 = No error
0.1	long	ID of signal 1	0
0,n	long	ID of signal 2	2

NC.OsciUtil, call - Command

row, col	Type	Description	Example: "SetSignalIds"
0,0	string	Selecting recording signals	SetSignalIds
0.1	long	ID of signal 1	0
0,n	long	Signal n ID	3

NC.OsciUtil, call - Response data

row, col	Type	Description	Example: "SetSignalIds" - Possible response
0,0	long	Possible acknowledgement messages	0 = No error 1 = Measurement point list empty 2 = File server error For negative values according to the error description, see below

NC.OsciUtil, call - Command

row, col	Type	Description	Example: "GetSampleData"
0,0	string	Reading recording data	GetSampleData

NC.OsciUtil, call - Response data

row, col	Type	Description	Example: "GetSampleData"
0,0	long	Possible acknowledgement messages	0 = No error For negative values according to the error description, see below
1,0..n	long	Data of recording channel 1	
n,0..n	long	Data of further recording channels	

NC.OsciUtil, call - Command

row, col	Type	Description	Example: "GetSampleBuffer"
0,0	string	Reading size of recording buffer	GetSampleBuffer

NC.OsciUtil, call - Response data

row, col	Type	Description	Example: "GetSampleBuffer"
0,0	long	Acknowledgement message	0 = No error
1,0	long	Size of recording buffer	12000

NC.OsciUtil, call - Command

row, col	Type	Description	Example: "SetSampleBuffer"
0,0	string	Setting size of recording buffer	SetSampleBuffer
1,0	long	Size of recording buffer	20000

NC.OsciUtil, call - Response data

row, col	Type	Description	Example: "SetSampleBuffer" - Possible response
0,0	long	Possible acknowledgement messages	0 = No error 2 = File server error For negative values according to the error description, see below

NC.OsciUtil, call - Command

row, col	Type	Description	Example: "GetPreTriggerPosition"
0,0	string	Reading trigger position	GetPreTriggerPosition

NC.OsciUtil, call - Response data

row, col	Type	Description	Example: "GetPreTriggerPosition"
0,0	long	Acknowledgement message	0 = No error
1,0	long	Trigger position (0...100%)	10

NC.OsciUtil, call - Command

row, col	Type	Description	Example: "SetPreTriggerPosition"
0,0	string	Setting trigger position	SetPreTriggerPosition
1,0	long	Trigger position (0...100%)	30

NC.OsciUtil, call - Response data

row, col	Type	Description	Example: "SetPreTriggerPosition"
0,0	long	Acknowledgement message	0 = No error

NC.OsciUtil, call - Command

row, col	Type	Description	Example: "GetTriggerKind"
0,0	string	Reading trigger type	GetTriggerKind

NC.OsciUtil, call - Response data

row, col	Type	Description	Example: "GetTriggerKind"
0,0	long	Acknowledgement message	0 = No error
1,0	long	Possible trigger types	0 = Unknown 2 = No trigger 4 = Manual trigger 64 = Program block trigger

NC.OsciUtil, call - Command

row, col	Type	Description	Example: "SetTriggerKind"
0,0	string	Setting trigger type	SetTriggerKind
1,0	long	Setting manual trigger	4

NC.OsciUtil, call - Response data

row, col	Type	Description	Example: "SetTriggerKind"
0,0	long	Acknowledgement message	0 = No error

NC.OsciUtil, call - Command

row, col	Type	Description	Example: "GetMinSampleTime"
0,0	string	Reading minimum recording time	GetMinSampleTime

NC.OsciUtil, call - Response data

row, col	Type	Description	Example: "GetMinSampleTime"
0,0	long	Acknowledgement message	0 = No error
1,0	double	Minimum recording time	4000

NC.OsciUtil, call - Command

row, col	Type	Description	Example: "GetProgramTrigger"
0,0	string	Reading program trigger	GetProgramTrigger

NC.OsciUtil, call - Response data

row, col	Type	Description	Example: "GetProgramTrigger"
0,0	long	Possible acknowledgement messages	0 = No error -1 = Program trigger not active
1,0	string	Program name	Test.npg
1,1	string	Program line	470
1,2	long	Channel number	1

NC.OsciUtil, call - Command

row, col	Type	Description	Example: "SetProgramTrigger"
0,0	string	Setting program trigger	SetProgramTrigger
1,0	string	Program name	Test.npg
1,1	string	Program line	470
1,2	long	Channel number	1

NC.OsciUtil, call - Response data

row, col	Type	Description	Example: "SetProgramTrigger"
0,0	long	Possible acknowledgement messages	0 = No error -1 = File name > 32 characters -2 = Error when opening file -3 = Trigger condition not found -4 = Incorrect channel numbers

NC.OsciUtil, call - Command

row, col	Type	Description	Example: "GetSignalInfo"
0,0	string	Reading signal information	GetSignalInfo

NC.OsciUtil, call - Response data

row, col	Type	Description	Example: "GetSignalInfo"
0,0	long	Acknowledgement message	0 = No error
1,0	long	Number of measurement made	1
1,1	long	Length of signal data	500
1,2	long	Trigger position	20

NC.OsciUtil, call - Command

row, col	Type	Description	Example: "NumberOfMpoints"
0,0	string	Reading number of possible signal points	NumberOfMpoints

NC.OsciUtil, call - Response data

row, col	Type	Description	Example: "NumberOfMpoints"
0,0	long	Acknowledgement message	0 = No error
1,0	long	Number of possible measurement points	400

NC.OsciUtil, call - Command

row, col	Type	Description	Example: "GetAvailableSignalIds"
0,0	string	Reading list of possible signal points	GetAvailableSignalIds

NC.OsciUtil, call - Response data

row, col	Type	Description	Example: "GetAvailableSignalIds"
0,0	long	Acknowledgement message	0 = No error -1 = No lists available
1..n,0	string	Signal name	Command position
2..n,1	string	Unit	mm
3..n,2	long	Axis number	3
4..n,3	long	Identification number	1
5..n,4	double	Multiplication factor	1.5

NC.PlcData

Description The item allows read/write access on the PLC data.

Remark The possible PLC data is: inputs, outputs and flags that are selected array by array according to the type specification.

☞ **Restriction:** Writing inputs and outputs can only be executed using the Target MTX Simulation!

A parallel read/write access of up to 20 type arrays is possible with the help of the transfer parameters. If multiple arrays are accessed simultaneously, the number of arrays has to be increased and the four parameters "Type, Length, Index and Offset" have to be specified additionally when creating an item.

To reduce the system load, the cyclic calling should always be **with delay**.

Item syntax *NC.PlcData,(1),(2),(3),(4),(5),(6) [,(7),(8),(9),(10)] ... [,(79),(80),(81),(82)]*

1. Number of arrays
   - 20 arrays max.
2. Delay
   - 0: None
   - 1: 300 ms delay
3. Data type (array 1)
   - 65: Outputs
   - 69: Inputs
   - 77: Flag
4. Data length (array 1)
   - Array length in bytes
5. Index (array 1)
   - Currently always "0" (reserved for type extension)
6. Offset (array 1)
   - PLC address offset within the selected data type

Optionally more arrays with data type, data length, index and offset respectively

Call Synchronous/asynchronous

Access Read/Write

Cyclic (opt.) Yes

Data format **NC.PlcData,1,0,77,8,0,6110 read**

row, col	Type	Description	Example: Reading flag data. (8 bytes starting from flag address 6110)
0,0	string	Status message of the read array data Possible messages are: 0 = No error, data is valid -10 = Data size exceeded -11 = Data type unknown -14 = Offset value invalid	0
0.1	long	Value of the first data byte	76
0.2	long	Value of the second data byte	0
0,3	long	Value of the third data byte	4
0,4	long	Value of the fourth data byte	0

row, col	Type	Description	Example: Reading flag data. (8 bytes starting from flag address 6110)
0,5	long	Value of the fifth data byte	0
0,6	long	Value of the sixth data byte	255
0,7	long	Value of the seventh data byte	3
0,8	long	Value of the eighths byte	12

NC.PlcData,1,0,77,8,0,6110 write

row, col	Type	Description	Example: Writing 8 bytes in the flag area. The flag address 6115 with "132"; all others are overwritten with "0"
0,0	long	Value of the first data byte	0
0.1	long	Value of the second data byte	0
0.2	long	Value of the third data byte	0
0,3	long	Value of the fourth data byte	0
0,4	long	Value of the fifth data byte	0
0,5	long	Value of the sixth data byte	132
0,6	long	Value of the seventh data byte	0
0,7	long	Value of the eighths byte	0

NC.ProgramChanInfo

Description Provides information in which channels the program is active.

A program is active if it is either selected or available as subroutine in the call structure of the main program.

If **up to 32 channels** are configured in the NC, an individual bit mask is used.

- Bit 0 - The program is active in channel 1
- Bit 1 - The program is active in channel 2
- ...
- Bit 31 - The program is active in channel 32

If **more than 32 channels** are configured, an additional second bit mask is used:

- Bit 0 - The program is active in channel 33
- Bit 1 - The program is active in channel 34
- ...
- Bit 31 - The program is active in channel 64

If there are **more than 64 channels** further bit masks are used.

Remark Upon item instancing, the program name is given with path.

Item syntax NC.ProgramChanInfo,(1)

1. Program name with path

Call Synchronous/asynchronous

Access Read

Cyclic (opt.) no

Data format The number of lines depends on the number bit masks

NC.ProgramChanInfo, read

row, col	Type	Description	Example NC.ProgramChanInfo,/mnt/example.npg
0,0	ulong	Bit mask for the channels 1-32	3 Bit-coded value, i.e. the program is active in channel 1 + 2
1,0	ulong	Bit mask for the channels 33-64	1 Bit-coded value, i.e. the program is active in channel 33

NC.ReloadExportUs

Description The NFS server is requested to share the directory structure for the NC access via the item. The directory structure is located in the "export.us" file.

The item provides read/write access to the file. The read access provides the complete content of the "export.us" file. The content can be edited by the user (entry or deletion of new shared directories) and moved back into the file using the item write access.

☞ When changing the "export.us" file, ensure that **no** CR (0x0d) is transferred in the string to be written. Otherwise, the NFS server will be misinterpreted

When rewriting the new data, the data is not written into the old file but into the new "export.us". The old file is renamed before in "export.us_bak" and is then available for checking. After the writing had been completed, the new structure is reported to the NFS server.

Remark Writing is also possible without transferring new data. In such a case, the existing file "export.us" is not modified. Only the NFS server has to read the content of the file again and has to share the directories.

The access paths to read and write the "export.us" file are internally determined via the path service.

The following errors can occur when calling the item:

2 = SCPErrFailed - "Registry path cannot be read on export.us"

110 = SCPErrNACK10 - "File access or renaming is not possible"

Item syntax **NC.ReloadExportUs**

Call Asynchronous

Access Read/Write

Cyclic (opt.) no

Data format **NC.ReloadExportUs, read**

row, col	Type	Description	Example
0,0	string	Content of the file "export.us"	C:\mnt mtxctrl localhost C:\mnt1 mtxctrl

NC.ReloadExportUs, write

row, col	Type	Description	Example
0,0	string	Entry of new shared directories	C:\mnt mtxctrl localhost C:\mnt1 mtxctrl C:\mnt2 mtxctrl

NC.WaitingStates

Description Indicates the system waiting state. It is a sum signal across all channels.

Remark Enumerator "LDX_McoWaitingState" is available for waiting states. It can assume the following values:

Enumerator LDX_McoWaitingState

LDX_McoWaitingState	Description
LDX_McoDwellTime_C	Dwell time
LDX_McoAuxiliaryFunction_C	Auxiliary function subject to acknowledgement
LDX_McoBlockTransferInhibit_C	Block transfer inhibit
LDX_McoChannelFeedrate0_C	Feed in the channel is equal to 0
LDX_McoProgramStop_C	Program stop with M0/M1
LDX_McoFeedHold_C	Feed halt in the channel
LDX_McoFeedInhibit_C	Feed inhibit in the channel or in a channel axis
LDX_McoCustomerStop_C	Customized block transfer inhibit
LDX_McoSynchronizedMovementStop_C	Synchronized motion stop between channels (ASTOP, ...)
LDX_McoWaitingForAxisTransfer_C	Waiting for axis upon axis transfer (G511)
LDX_McoWaitingForPermanentVariable_C	Waiting for permanent variable (WPV)
LDX_McoWaitingForIFSignalActive_C	Waiting for interface signal at active point of time (WAITA, ...)
LDX_McoWaitingForIFSignalPreparation_C	Waiting for interface signal [WAIT(IC...)] or for a specified interval (WAIT(,TIME%)) during block preparation
LDX_McoMovementCrontrolDataService_C	Motion Control Data Service (MCODS(...))
LDX_MCoWaitingForInpos_C	Waiting for channel data inpos
LDX_MCoFeedProgramming_C	Waiting for G95 programming
LDX_MCoWaitingForSpindleG33_C	Waiting till main spindle rotates G33
LDX_MCoNcProgramRestart_C	Waiting for block pre-run
LDX_MCoWaitingForSpindle_C	Waiting till speed reached, spindle stop...
LDX_MCoWaitingForEvent_C	Waiting on event >>> WEV function
LDX_MCoWaitingForInternalResource_C	Waiting on internal resource (e.g. memory)

Item syntax NC.WaitingStates

No parameters

Call Synchronous/asynchronous

Access Read

Cyclic (opt.) Yes

Data format NC.WaitingStates, read

row, col	Type	Description	Example
0,0	long	Active system waiting state: "Feed hold"	32 - LDX_McoFeedHold_C

4.7.2 Drives

NC.OscilloscopeConfiguration

Description The item determines whether the control axes in the system support the oscilloscope function. While writing, a parameter configuration can be specified for recording purposes.

Item syntax NC.OscilloscopeConfiguration

No parameters

Call Synchronous/asynchronous

Access Read

- 0 - The drive does not support any oscilloscope function
- 1 - The drive supports the oscilloscope function

Write

While writing the configuration, it has to be reported to the control

- which drive available participates in the measurement
- whether an available drive or the NC triggers the trigger. Therefore, either the number of the drive or "-1" has to be entered for the NC

Cyclic (opt.) no

Data format The data for each drive is transmitted line by line.

Each line consists of 2 columns. Column 1 contains the number of the drive (1..n). Column 2 can assume two values (0 = does not participate in the measurement; 1 = participates in the measurement). Specifications for drives not participating in the measurement are not required.

The last line contains the trigger information. If this line is omitted, the trigger is automatically on the NC!

NC.OscilloscopeConfiguration, Read

row, col	Type	Description	Example
0,0	long	Drive 1 does not support oscilloscoping	0
1,0	long	Drive 2 supports oscilloscoping	1
n,1	long	Drive n does not support oscilloscoping	0

NC.OscilloscopeConfiguration, Write

row, col	Type	Description	Example: Axes 1-3 are configured for recording. Trigger condition triggers axis 1
0,0	long	Drive number	1
0.1	long	Measurement participant	1
1,0	long	Drive number	2
1,1	long	Measurement participant	1
2,0	long	Drive number	3
2,1	long	Measurement participant	1
n,n	long	...	...
n,0	long	Trigger information	1

NC.SercosPhase2

Description

Reading and writing the Sercos phase.

Writing the item causes phase switching.

The following data is transferred while writing:

- Target phase to be switched in
  Operation mode: 0..4
  Parameterization mode: 0x14
- Reset ZK1 Error
- Execute download of .scs files in phase 2
- Execute download of .scs files in phase 3
- Startup mode (optional)
  0 = Normal startup (default)
  1 = Startup with topology addressing
- Device number (optional)
  0 = Switchover for the complete Sercos ring (default)
  >0 = Switchover for the respective Sercos device
- Placeholder for later extensions (optional)

Item syntax

NC.SercosPhase2

No parameters

Call Synchronous/asynchronous

Access Read/Write

Cyclic (opt.) no

Data format

NC.SercosPhase2, Read

row, col	Type	Description	Example:
0,0	long	Active phase [0..4]	4
0.1	long	Status of the phase machine There are the following status messages: 0 - No error during phase switching 32771 - Phase switching not possible via serial protocol 32773 - Drive address incorrect - Phase upshift is not executed if the drive addresses in the ring do not correspond to the drives parameterized 53250 - Switching to phase 3 is not possible 53251 - Switching: from phase 3 => 4 not possible 53253 - Phase switching still active - is reported if the phase switching has not yet been completed but a new phase is specified	0
0.2	long	Phase switching state 0 - Phase switching inactive 1 - Phase switching active	0
0,3	long	Startup mode (available from the version 14V14 R3) 0 = Normal startup 1 = Startup with topology addressing	0
1,0	long	Device number 0 = Switchover for the complete Sercos ring >0 = Switchover for the respective Sercos device	0

NC.SercosPhase2, Write

row, col	Type	Description	Switching to Sercos phase 4, resetting the ZK1 error, downloading the scs files, "normal" startup
0,0	long	Target phase	4
0.1	bool	Reset ZK1	TRUE
0.2	bool	Download in phase 2	TRUE
0,3	bool	Download in phase 3	TRUE
0,4	long	Startup mode (optional)	0
0,5	Void	Placeholder (optional)	
1,0	long	Device number (optional)	0

NC.SPA1

Description

Reads and writes elements of the Sercos parameters

Remark

NC.SPA1 permits access to control parameters if they are accessible in the format of Sercos parameters. The control parameters are usually addressed via the "drive address" 0.

NC.SPA1 can also provide access to drive parameters if the drives are accessible via the controls. Drives are selected via the drive address.

Errors occurring when accessing Sercos parameters are displayed by the IwSCP error number "SCPErrNACK1".

Item syntax

NC.SPA1,(1),(2),(3)

1. Drive address

2. Ident

Ident format

X-Y-ZZZZ	16Bit ident number range	
X-Y-ZZZZ.SI[.SE]	32Bit ident number range	
X	Parameter type consisting of one letter. See also device description. Examples:	
	S	Standard parameter
	P	Product-specific parameter
Y	Parameter block. (0 ... 7)	
ZZZZ	Data block number (0000 ... 4095) Attention! The data block number is specified with 4 decimals.	
SI	Sercos instance Decimal without leading 0. (0..255)	
SE	Optional: Sercos element Decimal without leading 0. (0..255)	

Tab. 4-32: Sercos idents

3. Element coding

Element coding is used to address Sercos parameter data elements.

Element coding

Parameter element	Default format only one letter can be specified	Advanced format Values can be linked by means of "Or"
Data status	S	01H
Name	N	02H
Attribute	A	04H
Unit	U	08H
min. input value	L	10H
Max. input value	H	20H
Operating date	D	40H

Tab. 4-33: Sercos parameter element coding

Examples

NC.SPA1,1,S-0-0003,L:

Minimum input value of parameter S-0-0003 of drive 1

NC.SPA1,1,S-0-0003,D:

Operating date of parameter S-0-0003 of drive 1

NC.SPA1,1,S-0-0003,48:

Unit & operating date of parameter S-0-0003 of drive 1

Call Synchronous/asynchronous

Access Read

The elements of the Sercos parameters are generally provided as strings.

Results are coded as follows:

Requested element	Result	Example
Data status	Hexadecimal word	"0x0000"
Name	String	"NC cycle time (TNcyc)"
Attribute	Hexadecimal double word	"0x60110001"
Unit	String	"µs"
min. input value	Decimal word	"2000"
Max. input value	Decimal word	"20000"
Value	decimal Hexadecimal (max. 8 digits) Binary (max. 64 characters)	"1.5" "0x80" "1111000010101010"

Write

Only the operating date can be written if it is possible in the parameter.

Data format Number of lines corresponds to requested parameters

NC.SPA1, read

row, col	Type	Description	NC.SPA1,3,S-0-0111,08
0,0	string	Requested element	"A eff"

Lists in simple format Requested with element coding "D"

NC.SPA1, read

row, col	Type	Description	NC.SPA1,3,S-0-0017,D
0,0	string	List	S-0-0001\n S-0-0002\n ... S-0-xxxx

Lists with length specification Requested with element coding "40"

NC.SPA1, read

row, col	Type	Description	NC.SPA1,3,S-0-0017,40
0,0	string	Actual length of the list in the drive	"400"
0.1	string	Max. length of the list in the drive	"400"
0.2	string	List	"S-0-0001\n S-0-0002\n ... S-0-xxxx"

NC.SPA1, write

row, col	Type	Description	NC.SPA1,3,S-0-0111,40
0,0	string	Value to be assigned	"1.500"

NC.SPH1

Description

The item SPH1 is used to read and write the Sercos phase.

Writing the item causes phase switching.

The following data is transferred while writing:

- Target phase to be switched in
  Value range [0..4]
- ID for startup type (optional)
  0 = Normal startup
  1 = Startup with topology addressing
- Placeholder for later extensions (optional)

Item syntax

NC.SPH1

No parameters

Call

Synchronous/asynchronous

Access

Read/Write

Cyclic (opt.)

Read only

Data format

NC.SPH1 read

row, col	Type	Description	Example:
0,0	long	Active phase [0..4]	4
0.1	long	Status of the phase machine There are the following status messages: 0 - No error during phase switching 32771 - Phase switching not possible via serial protocol 32773 - Drive address incorrect - Phase upshift is not executed if the drive addresses in the ring do not correspond to the drives parameterized 53250 - Switching to phase 3 is not possible 53251 - Switching: from phase 3 => 4 not possible 53253 - Phase switching still active - is reported if the phase switching has not yet been completed but a new phase is specified	0

NC.SPH1 write

row, col	Type	Description	Example: Switching to phase 4, "normal" startup
0,0	long	Target phase [0..4]	4
0,4	long	Startup mode (optional)	0
0,5	Void	Placeholder (optional)	

4.7.3 CPL data

NC.CplPermVariable

Description

Allows read/write access to permanent CPL variables. With array variables, the indices can be used to access individual elements.

Indices are based in the CPL 1. Thus, the following applies:

- 1<=Index1<=Dimension1
- 1<=Index2<=Dimension2

☞ Reading is carried out by means of a cache updated approx. every 100 ms.

Item syntax

NC.CplPermVariable,(1),(2),(3)

1. Variable name
2. Index1 (optional)
3. Index2 (optional)

Call

Synchronous/asynchronous

Access

Read/Write

Cyclic (opt.) Read-only (consider array size)

Data format Number of lines depends on MACODA

NC.CplPermVariable, read

row, col	Type	Description	NC.CplPermVariable,@ToolNo
0,0	long	Variable name	4711

NC.CplPermVariable, write

row, col	Type	Description	NC.CplPermVariable,@Darray, 2,5
0,0	double	Variable name	2.75

NC.CplPermVariableList

Description *Provides all permanent CPL variables in a list:*

- Fixed CPL variable (@1-@100)
- All definable CPL variables

Item syntax **NC.CPLPermVariableList**

Call Asynchronous

Access Read

Cyclic (opt.) no

Data format Number of lines: Number of CPL variables defined in "wmhperm.dat" and "anwperm.dat"

NC.CplPermVariableList, read

row, col	Type	Description	Example
0,0	string	Name of the first fixed variable	@1
0,1	long	Type of the first fixed variable	1
0.2	long	Dimension1 of the first fixed variable	1
0,3	long	Dimension2 of the first fixed variable	1
...			
99,0	string	Name of the 100th fixed variable	@100
99,1	long	Type of the 100th fixed variable	1
99,2	long	Dimension1 of the 100th fixed variable	1
99,3	long	Dimension2 of the 100th fixed variable	1
100,0	string	Name of the first definable variable	@_X
100,1	long	Type of the first definable variable	1
100,2	long	Dimension1 of the first definable variable	10

row, col	Type	Description	Example
100,3	long	Dimension2 of the first definable variable	2
...			

NC.CplPermVariableWatch

Description Monitors access on permanent CPL variables and provides their current values.

Remark Upon item instancing, the variable name, a watch mode and optional indices are specified.

The monitoring of the dimensioned CPL variables reacts as soon as an element of this variable is accessed.

Two optional parameters determine the data to be provided by the item. By specifying indices, elements from an array that are provided in the response can also be specified (100 max.). If no index is specified, the item provides all values of the respective PLC variable. In case of two-dimensional CPL variables and the missing Index2, the item provides the complete line of the value specified under Index1.

Indices are based in the CPL 1. Thus, the following applies:

- 1<=Index1<=Dimension1
- 1<=Index2<=Dimension2

Apart from the current data of the permanent CPL variable, the item provides a version ID. A comparison with the version ID read before indicates whether the given CPL variable was accessed.

Item syntax NC.CplPermVariableWatch,(1),(2),(3),(4)

1. Variable name

2: Watch mode

- 0: Monitoring read access
- 1: Monitoring write access

3. Index1 (optional)

4. Index2 (optional)

Call Synchronous/asynchronous

Access Read

Cyclic (opt.) Yes

Data format NC.CplPermVariableWatch, read

row, col	Type	Description	Example: NC.CplPermVariableWatch,@ToolNmb,1
0,0	long	Provides the current value in case of write access on "@ToolNmb"	1
1,0	long	Provides the current version ID in case of write access to "@ToolNmb"	245633

Example - Monitoring an one-dimensional array

row, col	Type	Description	NC.CplPermVariableWatch,@_R,0,4,1
0,n	double	Provides the current value of the fourth element when accessing (read-only) the "@_R" elements	2.0
1,0	long	Provides the current version ID in case of read access on "@_R"	12890

Example - Monitoring a two-dimensional array

row, col	Type	Description	NC.CplPermVariableWatch,@SampleArray,1,2,9
0,0	double	Provides the current value of element 9 in line 2 in case of write access to elements of the two-dimensional CPL variable "@SampleArray".	21.4
1,0	long	Provides the current version ID in case of write access on elements of the two-dimensional CPL variable "@SampleArray".	7421

4.7.4 File-based Tables

NC.XmlRootElementInfo

Description Provides the top level schema and the extension for the specified root element (table type).

Remark The extension is only provided for file-based tables.

Item syntax NC.XmlRootElementInfo,(1)

1. Root element

Call Synchronous/asynchronous

Access Read

Cyclic (opt.) no

Data format Number of lines: 2 lines for file-based tables, otherwise 1 line

NC.XmlRootElementInfo, read

row, col	Type	Description	Example NC.XmlRootElementInfo,/ZOT
0,0	string	Information on ZO tables Top level schema	/feprom/zot_sd.xsd
0.1	string	Extension	zot

NC.XmlRootElementList

Description Provides a list of all root elements for an XML data class (e.g. file-based table, configuration data).

The number of the provided root elements differs depending on the XML data type. Moreover, it depends on other boundary conditions (e.g. utilization of user-definable tables).

Remark The XML data class is specified by means of parameters upon item instancing.

To specify the XML data class, the "LDX_XmlDataClass" enumerator is available. The XML data class of a root element is determined via the "Class" attribute in the respective schema. Between the value of the "class" attribute and the XML data class there is the following assignment:

Enumerator LDX_XmlDataClass

LDX_XmlDataClass	Description
LDX_XML_DATA_CLASS_UNKNOWN_C	Other
LDX_XML_DATA_CLASS_CFG_C	XML data class "CFG"
LDX_XML_DATA_CLASS_DBTAB_C	XML data class "DBTAB"
LDX_XML_DATA_CLASS_FILETAB_C	XML data class "FILETAB"

Item syntax **NC.XmlRootElementList,(1)**

1. Data category

Call Asynchronous

Access Read

Cyclic (opt.) no

Data format Number of lines: Number of the provided root elements

NC.XmlRootElementList, read

row, col	Type	Description	NC.XmlRootElementList,3 Root elements for the "FILETAB" XML data class
n,0	string	Root element	/DCT

NC.XmlSchemaList

Description Provides the list of all schemas used in the kernel.

Each list entry receives the complete path, the name and the extension of the schema as well as the date (time stamp) of the last change (UTC).

Item syntax **NC.XmlSchemaList**

Call Asynchronous

Access Read

Cyclic (opt.) no

Data format Number of lines: Number of all schemas used in the kernel (varies with regard to the software version)

NC.XmlSchemaList, read

row, col	Type	Description	Example
n,0	string	Path/name.extension of the schema	/feprom/schemas/basic_ty.xsd
n,1	date	Last change date	04.04.04 16:26:12

NC.XmlTable_U

Description Read and write access to a file-based tables.

File-based tables are e.g. zero offset tables, D-correction tables and placement tables. When creating the item, the table to be accessed is specified. Furthermore, the node in the table is identified.

Remark In ZO tables, it is possible to insert or to delete columns.

To insert columns, the column in which a new one is to be inserted is specified in the second parameter of the item. The insertion of columns differs from the normal write function in the following: Additionally to the first line with the data for the new column, a second line with the constant "LDX_TAD_INSERT_AXIS_ZOT_C" is added.

To delete a column, the column to be deleted is specified in the second parameter of the item. A subsequent writing without data in the first line and with the constant "LDX_TAD_DELETE_AXIS_ZOT_C" in the second line causes the column to be deleted.

The "LDX_TadWriteMode" enumerator can assume the following values:

Enumerator LDX_TadWriteMode

LDX_TadWriteMode	Description
LDX_TAD_WRITE_CACHE	Access via cache
LDX_TAD_INSERT_AXIS_ZOT_C	Insert axis/column
LDX_TAD_DELETE_AXIS_ZOT_C	Delete axis/column

See section TAD Errors on page 222for a description of errors that can occur when reading or writing tables.

Item syntax **NC.XmlTable_U,(1),(2)**

1. Table name (complete path, file name)

2. Node in the table (optional)

Call Asynchronous

Access Read

- The read data is provided as XML strings.

Write

- The length of the XML string is limited to 8 Kbytes.

Cyclic (opt.) no

Data format **NC.XmlTable_U, read**

row, col	Type	Description	NC.XmlTable_U,/database/DC1.dct,/DCT Reading a D-correction table
0,0	string	Table data	<DCT> ... </DCT>

NC.XmlTable_U, write

row, col	Type	Description	NC.XmlTable_U,/database/DC1.dct,/DCT/D/SD/L1 Writing a value of a D-correction table
0,0	string	New value	<L1>2.5</L1>

NC.XmlTable_U, write

row, col	Type	Description	NC.XmlTable_U,/database/ZO1.zot,/ZOT/Offset/Ax[4] Inserting a column into a ZO table
0,0	string	Data for the new column	<Ax><Name>B</Name><Type>Lin</Type><Set><G54>5.1</G54>...</Ax>
1,0	string	Constant	LDX_TAD_INSERT_AXIS_ZOT_C

NC.XmlTable_U, write

row, col	Type	Description	NC.XmlTable_U,/database/ZO1.zot,/ZOT/Offset/Ax[2] Deleting the second column in a ZO table
0,0	string	Data is ignored	Empty
1,0	string	Constant	LDX_TAD_DELETE_AXIS_ZOT_C

NC.XmlTableCreate

Description Creates a new table.

Remark See section TAD Errors on page 222for a description of errors that can occur when creating tables.

Item syntax NC.XmlTableEvent

Call Synchronous/asynchronous

Access Write

A version number and the modified node in Xpath syntax are provided.

Cyclic (opt.) no

Data format NC.XmlTableCreate, write

row, col	Type	Description	Creation of the D-correction table sample.dct
0,0	string	File name of the new table	/sample.dct
0.1	string	XPath syntax of the new table	/DCT
0.2	string	Template used	/mnt/template.dct
0,3	long	Channel number	Empty

NC.XmlTableCreate, write

row, col	Type	Description	Creating the axis offset table offsets1.aot
0,0	string	File name of the new table	/offsets1.aot
0.1	string	XPath syntax of the new table	/AOT
0.2	string	Template used	
0,3	long	Channel number	1

NC.XmlTableEvent

Description Reports write access to the table opened in the editor. When creating the item, identify the table for which write access is to be reported.

Remark Possibly, the modified node or the new data cannot be provided. In this case, the table or the node has to be read in again.

A continuous reporting cannot be guaranteed while writing. Thus, it is necessary to check the included version number on the client side. If the version numbers are consecutive, the reporting is continuous. Otherwise, not all write accesses are recorded and the table has to be completely read in again.

Errors occurring with this item are described section TAD Errors on page 222.

Item syntax **NC.XmlTableEvent,(1)**

1. Table name (complete path, file name)

Call Synchronous/asynchronous

Access Read

A version number and the modified node in Xpath syntax are provided.

Cyclic (opt.) Yes, required!

Data format **NC.XmlTableEvent, read**

row, col	Type	Description	NC.XmlTableEvent,/database/DC1.dct Identification of write accesses to a D-correction table
0,0	dword	Version number	213
1,0	string	Modified node	/DCT/Corr[3]/Rough/L1

4.7.5 Tables based on databases

NC.DbTableDataRec_U

Description Read access to a data block. Upon item instancing, the desired data block is addressed (element path, sector, location).

Remark The table is unambiguously identified by the element path. Besides, this parameter can be used to address the data block either as a whole or a substructure.

While writing, all transferred data is normally written to a data block. If only tool-specific or location-specific data is to be written, a corresponding mode is selected. For this purpose, the "LDX_DbdSetMode" enumerator can assume the following values:

Enumerator LDX_DbdSetMode

LDX_DbdSetMode	Description
LDX_DBD_SETMODE_TOOL_DATA_C	Only tool-specific data
LDX_DBD_SETMODE_PLACE_DATA_C	Only location-specific data

See section DBD Errors on page 223for a description of errors that can occur when searching data blocks.

Item syntax **NC.DbTableDataRec_U,(1),(2),(3)**

1. Element path (in Xpath syntax)

2. Key1 (e.g. sector)

3. Key2 (e.g. place)

Call Asynchronous

Access Read

The read data is transferred as XML string.

Write

Writes data into data block

Cyclic (opt.) no

Data format **NC.DbTableDataRec_U, read**

row, col	Type	Description	NC.DbTableDataRec_U,/DBT1/Rec,1,1 Reads database table 1
0,0	string	Content of the data block	<Rec><Hd><K1>1</K1><K2>1</K2><SKQ></SKQ><IKQ1>0</IKQ1>...

NC.DbTableDataRecLocked_U

Description Read and write access to a data block. Upon item instancing, the desired data block is addressed (element path, sector, location).

Remark The table is unambiguously identified by the element path. Besides, this parameter can be used to address the data block either as a whole or a substructure.

- The addressed data block is locked by initial reading of the item.
- The initial write access releases the data block.
- When the item is reduced without writing beforehand, the data block is released automatically.
- After write access, the data block can be locked again by another read access.

While writing, all transferred data is normally written to a data block. If only tool-specific or location-specific data is to be written, a corresponding mode is selected. For this purpose, the "LDX_DbdSetMode" enumerator can assume the following values:

Enumerator LDX_DbdSetMode

LDX_DbdSetMode	Description
LDX_DBD_SETMODE_TOOL_DATA_C	Only tool-specific data
LDX_DBD_SETMODE_PLACE_DATA_C	Only location-specific data

See section DBD Errors on page 223for a description of errors that can occur when searching data blocks.

Item syntax **NC.DbTableDataRecLocked_U,(1),(2),(3)**

1. Element path (in Xpath syntax)

2. Key1 (e.g. sector)

3. Key2 (e.g. place)

Call Asynchronous

Access Read

- The read data is transferred as XML string.

Write

- The data to be written is transferred as XML string.

Cyclic (opt.) no

Data format NC.DbTableDataRecLocked_U, read

row, col	Type	Description	NC.DbTableDataRecLocked_U,/ DBT1,1,3 Reads in a locked data block
0,0	string	Content of the data block	<DBT1><Rec><Hd><K1>1</K1><K2>3</K2><SKQ></SKQ>...

NC.DbTableDataRecLocked_U, write

row, col	Type	Description	Writes the complete data block
0,0	string	New content	<DBT1><Rec><Hd><K1>1</K1><K2>1</K2><SKQ></SKQ>...

NC.DbTableDataRecLocked_U, write

row, col	Type	Description	Writes the tool-specific data
0,0	string	New content	<DBT1><Rec><Hd><K1>1</K1><K2>3</K2><SKQ></SKQ>...
1,0	long	Mode	LDX_DBD_SETMODE_TOOL_DATA_C

NC.DbTableEvent

Description Reports write access to a DB table, a substructure or a data block of the table.

Remark If only the first parameter is specified, write access to the table as a whole is reported. If the second parameter is specified, monitoring a sector is possible for example. If the third parameter is specified as well, only write accesses to a data block are monitored.

A continuous reporting cannot be guaranteed while writing. Thus, it is necessary to check the included version number on the client side. If the version numbers are consecutive, reporting is continuous. Otherwise, not all write accesses are recorded and the DB table has to be read again.

See section DBD Errors on page 223for a description of errors that can occur when searching data blocks.

Item syntax NC.DbTableEvent,(1),[(2),[(3)]]

1. Root tag
2. Key1 (optional, e.g. sector)
3. Key2 (optional, e.g. place)

Call Synchronous/asynchronous

Access Read

Provides the latest changed data block in the table (Key1, Key2).

Cyclic (opt.) Yes, required!

Data format **NC.DbTableEvent, read**

row, col	Type	Description	NC.DbTableEvent,/DBT1,0
0,0	dword	Version number of the data block changed last	213
1,0	long	Key1 (e.g. sector)	0
2,0	long	Key2 (e.g. place)	14

NC.DbTableLoad

Description Reads in data blocks of a DB table from an XML file. The read data blocks have to be available in the DB table, no new data blocks are created.

Remark If neither Key1 nor Key2 are specified, all data blocks from the XML file are read in. If Key1 is specified, all data blocks with this Key 1 (e.g. only data blocks of one sector) are read. If Key1 and Key2 are specified, the data block defined by this value pair is read.

It is controlled by an optional mode whether data blocks are completely read or whether only location-specific or tool-specific data is read in. This is specified by means of the "LDX_DbdLoadMode" enumerator:

Enumerator LDX_DbdLoadMode

LDX_DbdLoadMode	Description
LDX_DBD_LOADMODE_TOOL_DATA_C	Only tool-specific data
LDX_DBD_LOADMODE_PLACE_DATA_C	Only location-specific data

In the XML file, each data block should be present only once.

See section DBD Errors on page 223for a description of errors that can occur when reading in data blocks.

Item syntax **NC.DbTableLoad**

Call Asynchronous

Access Write

While writing, the following data has to be transferred:

- Name of the file from which data is read in
- Root tag (unambiguously identifies the DB table)
- Key1 (optional)
- Key2 (optional)
- Mode (optional)

Cyclic (opt.) no

Data format **NC.DbTableLoad, write**

row, col	Type	Description	Reading in all data blocks (only tool-specific data)
0,0	string	File from which data is read in	/mnt/tools.xml
0.1	string	Root tag	/DBT1
0.2	long	Key1, optional	

row, col	Type	Description	Reading in all data blocks (only tool-specific data)
0,3	long	Key2, optional (data block)	
1,0	long	Mode, optional	LDX_DBD_LOADMODE_TOOL_DATA_C

NC.DbTableQuery_U

Description Searching for data blocks according to a preset search criterion. The search criterion is indicated via parameters upon item instancing.

Remark Two different formats of the search criterion are supported:

Search condition in XML format:

- Root tag of the XML string is "DBTCond".
- For the individual conditions, use the element denominators from the header of the data block.
- To specify the masks of the relevant bits, use the element denominators BQ1Mask, BQ2Mask and BQ3Mask.
- *The values for the bit arrays (e.g. BQ1) and the bit masks (e.g. BQ1Mask) are interpreted as follows:*
  - As hexadecimal number if they start with "0x" or "0X",
  - as an octal number if they start with "0",
  - As decimal number in all other cases.

Boundary condition:

Only predefined tag names may be used.

Example:

<DBTCond><K1>1</K1><BQ2>1</BQ2><BQ2Mask>1</BQ2Mask></DBTCond>

Search condition in list form:

- The list contains individual conditions. All have to be met.
- The individual conditions are separated by commas.
- A normal individual condition looks like the following: <TagName><ComparativeOperator><Value> (e.g. SKQ=4711)
- *The following relational operators are permitted:*
  - "=" equal to
  - ">" greater than
  - ">=" greater than or equal to
  - "<" less than
  - "<=" less than or equal to
  - "<>" not equal to

  Restrictions for SKQ: Only "=" and "<>" are allowed.
- A bit mask individual condition is as follows: <TagName><Value><Mask (e.g. BQ1=0x307000:0xF0FF00).
  - <Value> contains the value of the relevant bits
  - <Mask> contains the mask of the relevant bits

  Attention: Only "=" is allowed as relational operator.

- Values and masks are interpreted as follows:
  - As hexadecimal number if they start with "0x" or "0X",
  - as an octal number if they start with "0",
  - As decimal number in all other cases.
- Within <Value>, preceding or following spaces are not analyzed.

  If a string with such spaces is to be specified, the substring has to be enclosed in inverted commas (e.g. SKQ=' ')
- To use an inverted comma in a substring enclosed in inverted commas, the character "\" has to precede the inverted comma (e.g. SKQ='Achim \'s')

Boundary condition:

Only predefined tag names may be used.

Example:

K1=1,IKQ2=3,BQ2=0x1:0x1

See section DBD Errors on page 223for a description of errors that can occur when searching data blocks.

Item syntax **NC.DbTableQuery_U,(1),...(n)**

1. Root tag

2. .. n. Search condition in XML format or search condition in list format

Call Asynchronous

Access Read

Provides a list of all data blocks meeting the search condition. For each data block, the complete header is provided.

Cyclic (opt.) no

Data format **NC.DbTableQuery_U, read - Search in the XML format**

row, col	Type	Description	NC.DbTableQuery_U,/ DBT1,K1=1,IKQ2<>3005 Search of all data blocks with (K1=1) and (IKQ2<>3005)
0,0	string	Search result	<DBT1> ... </DBT1>

NC.DbTableQuery_U, read - Search in list format

row, col	Type	Description	NC.DbTableQuery_U,/ DBT1,K1=1,IKQ2=3 Search of all data blocks with (K1=1) and (IKQ2=3)
0,0	string	Search result	<DBT1> ... </DBT1>

NC.DbTableSave

Description Saves data blocks of a DB table into a XML file. The DB table as a whole, several data blocks (e.g. a sector of a tool table) or individual data blocks can

be saved. If the XML file in question already exists, the data to be saved can be attached at the end or the content of the file is deleted first.

Remark It is possible to save only the data blocks of one sector (Key1) or specifically one data block (Key1 and Key2). If neither Key1 nor Key2 are specified, the DB table is saved completely.

See section DBD Errors on page 223for a description of errors that can occur when searching data blocks.

Item syntax NC.DbTableSave

Call Asynchronous

Access Write

While writing, the files contain the following information:

- Name of the XML file (incl. path and extension)
- Root tag (unambiguously identifies the DB table)
- Attaching yes/no (attaching to an existing file or overwriting)
- Key1 (optional)
- Key2 (optional)

Cyclic (opt.) no

Data format NC.DbTableSave, write

row, col	Type	Description	Saves a data block, attaching to /mnt/tools.xml
0,0	string	Name of the XML file	/mnt/tools.xml
0.1	string	Root tag	/DBT1
0.2	long	0 = Overwrite 1 = Attach	1
0,3	long	Key1, optional (sector)	3
0,4	long	Key2, optional (data block)	2

4.7.6 Machine parameters

NC.MacodaParameter

Description Provides the date of the requested MACODA parameter of a channel or spindle.

☞ MACODA access to spindle parameters is zero-based!

Item syntax NC.MacodaParameter,(1),(2)

1. Parameter number
2. Channel or spindle number (optional)

Call Synchronous/asynchronous

Access Read

The response format varies according to the parameter type.

Cyclic (opt.) no

Data format Number of lines depends on MACODA

NC.MacodaParameter

row, col	Type	Description	NC.MacodaParameter, 600500060,1
n,0	string	Date of the MACODA parameter	G54.1

NC.CfgData_U

Description Supports the read/write access on the configuration data. A parameter determines the accessible configuration data elements.

Item syntax NC.CfgData_U,(1)

1. Configuration element(s)

Call Asynchronous

Access Read

- While reading, the item data is returned as XML string.

Write

- During the write access, the item data is transferred as XML string. The data size is limited to 8 kB.

Cyclic (opt.) no

Data format NC.CfgData_U,/NCO/Vel/Ch[2]/Fact, read

row, col	Type	Description	Reading configuration data element /NCO/Vel/Ch[2]/Fact
0,0	string	Configuration element(s)	<Fact><FactG94>1<\FactG94><FactRotG70>1<\FactRotG70><\Fact>

NC.CfgData_U,(1) write

row, col	Type	Description	Sets the configuration data element /NCO/Vel/Ch[2]/Fact/FactRotG70 to 2
0,0	string	Configuration element	<FactRotG70>2<\FactRotG70>

NC.CfgDataBackup

Description Storing and transferring configuration data in the standard file format.

Remark The "LDX_McaBackupOption" enumerator allows different types of access to and storage in files:

Enumerator LDX_McaBackupOption

LDX_McaBackupOption	Description
LDX_MCA_UNLOAD_PAR_C	Writing modified parameters into file
LDX_MCA_RELOAD_PAR_C	Loading parameter file
LDX_MCA_UNLOAD_PAR_FEPROM_C	Saving parameters in "usrfep/mcadbpar.txt"
LDX_MCA_RELOAD_PAR_IF_EMPTY_C	Load parameter file only if default setting
LDX_MCA_UNLOAD_PAR_ALL_C	Save all parameters in file

☞ While writing the item, a target file can be specified. If no target file is specified, all parameters are saved under "usrfep/mcadbpar.txt".

Item syntax NC.CfgDataBackup

Call Asynchronous

Access Write

Cyclic (opt.) no

Data format NC.CfgDataBackup, write

row, col	Type	Description	Store configuration data under "usrfep/mcadbpar.txt"
0,0	long	Target file	
0.1	string	Enumerator "LDX_McaBackupOption"	

NC.CfgDataEvent

Description Supports notification due to modification of configuration data. A parameter (in Xpath syntax) specifies the configuration data element or the node in the configuration data tree for which the messages are required.

Remark If the configuration data element or a subelement of the specified node changes, the version is incremented. The item provides the current version and a string specifying the element modified. Using the "NC.CfgData" item, the current configuration data can be read.

Item syntax NC.CfgDataEvent,(1)

1. Configuration element

Call Synchronous/asynchronous

Access Read

The current version and a string in Xpath syntax are provided.

Cyclic (opt.) Yes, required!

Data format NC.CfgDataEvent, read

row, col	Type	Description	NC.CfgDataEvent,/AXSP Message request on modified configuration data of axis or spindles
0,0	dword	Current version	213
0.1	string	Modified element or subelement	/AXSP/Dr[2]/AxFun/Pos/TravLim/PosLim

NC.CfgDataLoad

Description Loads the configuration data from an XML file. All configuration data entries of the file are stored in the configuration database.

Remark *Following values are required:*

- Name of XML file to be read (path, name, extension)
- Root tag
- Mode

The "LDX_CfgDataMode" enumerator serves the optional comparison of the XML data before loading:

Enumerator LDX_CfgDataMode

LDX_CfgDataMode	Description
0x0	Standard mode Reading in XML file in MACODA
LDX_CFG_LOADMODE_LOAD_C (= 1)	Reading into MACODA (= standard mode)
LDX_CFG_LOADMODE_TEST_C (= 2)	Checks validity of content
LDX_CFG_LOADMODE_SYNC_C (= 16)	Synchronizing binary XML data with MACODA

Item syntax NC.CfgDataLoad

Call Asynchronous

Access Write

Loads configuration data from XML file.

Cyclic (opt.) no

Data format NC.CfgDataLoad, write

row, col	Type	Description	Loading the configuration data from /mnt/Ex_AX.xml
0,0	string	Name of the XML file	/mnt/Ex_AX.xml
0.1	string	Root tag	/AX
0.2	long	Mode	LDX_CFG_LOADMODE_LOAD_C

NC.CfgDataSave

Description Saves the configuration data into an XML file. If the file exists, it is overwritten.

Remark *Following values are required:*

- Name of XML file (path, name, extension)
- Root tag

Item syntax NC.CfgDataSave

Call Asynchronous

Access Write

An XML file is loaded.

Cyclic (opt.) no

Data format NC.CfgDataSave, write

row, col	Type	Description	Saves the configuration data /AX in the XML file /mnt/ax.xml
0,0	string	Name of the XML file	/mnt/ax.xml
0.1	string	Root tag	/AX

4.7.7 System data

Function

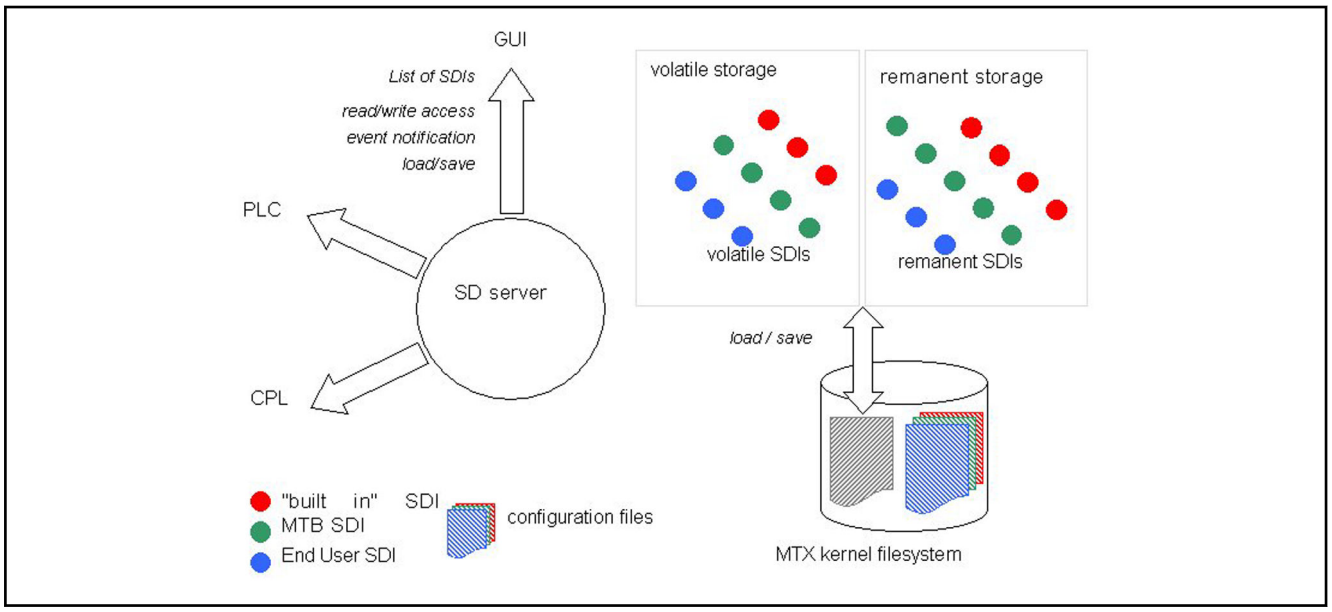

Fig. 4-1: SDI overview

The term system data means a set of data items and is characterized as follows:

- All system data items (SDIs) are managed by the system data server (SD server) within the MTX kernel.
- The SD server has a remanent memory as well as a volatile memory.
- SDIs located in the volatile storage are initialized at each startup of MTX kernel, whereas SDIs located in the remanent memory retain their values and are unaffected by a shutdown and a startup.
- The MTX kernel provides some "built in" SDIs, which do not need be configured.
- Additional SDIs can be defined by the machine tool manufacturer or the end user. The number of SDIs, their size and their structure are freely configurable within the range of available memory.
- The configuration files "SDDefMTB.xml" and "SDDefUsr.xml" are used to define the SDIs. Built-in SDIs are defined in "SDDef.xml".
- *Each entry in a configuration file specifies the following attributes of an SDI:*
  - Name (16 characters max.)
  - Number of data elements (e.g. one data element per channel)
  - Size (e.g. dimensions of arrays)
  - Structure
  - Memory class (remanent/volatile)
- SDIs are accessible from the CPL, the PLC program and the GUI.
- GUI access is based on XML. A string in the Xpath name is used to address a specific SDI. The data exchange is based on XML streams.
- The SD server provides an event mechanism for SDIs, i.e. a client may request automatic notification up to the modification of SDIs.

- A complete list of all SDIs is provided.
- SDIs can be saved into an XML file within the kernel file system or loaded from a file.
- In future, access to SDIs will be limited by access rights.

The SCP items documented support XML-based access to the system data of the MTX. The following functions are necessary to provide access to the system data.

Log in/log off As the access to the configuration data is to be protected, a log in and a log out on the MTX system data server is required. During log in, the current user level is transferred and the configuration server returns a corresponding client ID which is used for subsequent transactions.

NC.SystemData_U

Description Read and write access to a system data item

Remark While instancing, a parameter is provided which specifies the system data item or one of its subelements.

☞ **Restriction**: The item has **not** yet been optimized for a cyclic request!

Item syntax **NC.SystemData_U,(1)**

1. System data item or subelement (Xpath syntax, string length 99 characters max.)

Call Asynchronous

Access Read

- The data is read as xml stream. In doing so, the last component of the Xpath expression was used as TAG frame. If multi-dimensional arrays are read, the data is enclosed in predefined tags <SD></SD>.

Write

- The data is transferred as xml stream. Individual elements as well as a complete substructure of the system date can be written.

- Writing complete indexed structures is not possible. It is possible for reading.

Cyclic (opt.) no

Data format **NC.SystemData_U, read (single element)**

row, col	Type	Description	Example: NC.SystemData_U,/Sys_VAR[1,2,2] Reading structure element [1,2,2] from system date /Sys_VAR
0,0	string	Data of the system data item	<Sys_VAR Dimension="1,2,2">0.00000</Sys_VAR>

NC.SystemData_U, write (single element)

row, col	Type	Description	Example: NC.SystemData_U,/Sys_VAR[1,2,2] Writing structure element [1,2,2] from system date /Sys_VAR
0,0	string	Data of the system data item to be written	<Sys_VAR>1.00000</Sys_VAR>

NC.SystemData_U, read (substructure)

row, col	Type	Description	Example: NC.SystemData_U,/Sys_VARArray[1] /ArrayEx Reading the substructure Sys_VARArray[1] /ArrayEx of the system date Sys_VARArray
0,0	string	Data of the system data item	<Sys_VARArray Dimension=="1"><ArrayEx>Array Elem1>1</ ArrayElem1><ArrayElem2>2</ ArrayElem2><ArrayElem3>3</ ArrayElem3></ArrayEx></ Sys_VARArray>

NC.SystemData_U, write (substructure)

row, col	Type	Description	Example: NC.SystemData_U,/Sys_VARArray[1] /ArrayEx Writing the substructure Sys_VARArray[1] /ArrayEx of the system date Sys_VARArray
0,0	string	Data of the system data item	<Sys_VARArray Dimension=="1"><ArrayEx><ArrayElem1>44</ ArrayElem1><ArrayElem2>55</ ArrayElem2><ArrayElem3>66</ ArrayElem3></ArrayEx></ Sys_VARArray>

NC.SystemData_U, read (complete system date)

row, col	Type	Description	Example: NC.SystemData_U,/Sys_VARArray Reading the complete system date / Sys_VARArray
0,0	string	Data of the system data item	<Sys_VARArray Dimension=="1"><ArrayBsp><ArrayElem1>1</ArrayElem1><ArrayElem2>2</ArrayElem2><ArrayElem3>3</ArrayElem3></ArrayBsp></Sys_VARArray> <Sys_VARArray Dimension=="2"><ArrayBsp><ArrayElem1>21</ArrayElem1><ArrayElem2>22</ArrayElem2><ArrayElem3>23</ArrayElem3></ArrayBsp></Sys_VARArray> ....

NC.SystemDataEvent

Description Reports changed system data

Remark Upon instancing, a parameter is provided which specifies the system data item or a subelement for which changes are to be reported.

If the system data item or one of its subelements changes, the version is incremented. If the version was changed, the system data has to be read using the "NC.SystemData_U" item.

In order to be informed on write accesses to any system data, use the Xpath "/".

You can only use the root ("/") or a root element to specify the location to be monitored. Measures on subelements of a system data item are not supported.

Item syntax NC.SystemDataEvent,(1)

1. System data item or root (Xpath syntax, string length 99 characters max.)

Call Synchronous/asynchronous

Access Read

The current version is returned as well as a string specifying the system data item or its modified subelements.

Cyclic (opt.) Yes, required!

Data format NC.SystemDataEvent

row, col	Type	Description	Example: NC.SystemDataEvent,/MySDI
0,0	dword	Current version	213
1,0	string	Modified system data item or subelement	/MySDI/SubComp/IntVal

NC.SystemDataLoad

Description Reads the system data from an XML file. All system data elements of this file are loaded into the memory of the system data server.

Remark The system data elements can either be loaded completely or partially from the file into the memory of the system data server.

No parameters are specified while creating the item.

The following 4 values are necessary for writing:

- Name of XML file to be read (path, name, extension)
- Name of one or several system data as XPath specification (by individual XPath elements that can be separated with a "+" and without further space, one or several SDs can be loaded individually). If only the root character "/" is specified, all existing system data can be stored.
- Mode

  The "LDX_SDMode" enumerator allows a further selection of the desired system data:
- User ID

Enumerators LDX_SDMode

LDX_SDMode	Description
0	Default mode

Item syntax NC.SystemDataLoad

Call Asynchronous

Access Write

Loads system data of the specified .xml file

Cyclic (opt.) no

Data format NC.SystemDataLoad, write

row, col	Type	Description	Example
0,0	string	Name of the XML file	/mnt/sdis.xml
0.1	string	Name of several system data as XPath specification	/Foo+/myVel
0.2	long	Mode	0
0,3	long	User ID	0

NC.SystemDataSave

Description Saves the system data into an XML file. If the file already exists, it is overwritten.

Remark The system data can either be written completely or partially into the file.

No parameters are specified while creating the item.

The following 4 values are necessary for writing:

- Name of XML file (path, name, extension)
- Name of one or several system data and possible subelements as XPath specification (by individual XPath elements that can be separated with a "+" and without further space, one or several SDs can be saved individually). If only the root character "/" is specified, all existing system data can be stored.

- Mode

  The "LDX_SDMode" enumerator allows a further selection of the desired system data.
- User ID

Enumerators LDX_SDMode

LDX_SDMode	Description
0	Default mode

Item syntax NC.SystemDataSave

Call Asynchronous

Access Write

Writes system data into the specified XML file

Cyclic (opt.) no

Data format NC.SystemDataSave, write

row, col	Type	Description	Example
0,0	string	Name of the XML file	/mnt/bigsdi.xml
0.1	string	Name of several system data as XPath specification	/BigSDI+/Foo+/myVel
0.2	long	Mode	0
0,3	long	User ID	0

NC.SystemDataWatch_U

Description Supports the read and write access to a special system data item.

Remark To read or write an XML stream without outer start tag and end tag, the parameter mode has to be linked logically with "LDX_MODE_SD_NO_XML_TAGS_C" using an "OR" operation. This is only recommend when accessing individual elements of a system date.

Item syntax NC.SystemDataWatch_U,(1),[(2)],[(3)]

1. System data item or subelement (Xpath syntax, string length 99 characters max.)

2. Mode, controls the item's behavior

3. User ID, for future use

Call Synchronous/asynchronous

Access Read

- The data is read as XML stream.

- The data size is limited to 8 kB.

☞ **Restriction**: The read function is optimized for a cyclic query. Thus, the read access to the item may only be carried out within a group. The first read accesses can provide "SCPQualityUncertain".

Write

- The data is transferred as XML stream.

- The data size is limited to 8 kB.

Cyclic (opt.) Yes

Data format NC.SystemDataWatch_U, read

row, col	Type	Description	NC.SystemDataWatch_U,/ Sys_Simulation[1], LDX_MODE_SD_NO_XML_TAGS_C
0,0	string	Data of the system data item	FALSE

4.7.8 System Functions

System.Archive

Description Creates an archive with the data of the control or imports an existing archive.

Remark *The following specifications are required:*

- Archive name
- Log file name
- Mode (Read in archive or create archive, required partial archives)

The enumerator "LDX_TarMode" specifies if an archive is to be created or imported. Furthermore, this enumerator is used to specify the partial archives.

Enumerator LDX_TarMode

LDX_TarMode	Description
LDX_TAR_MODE_ALL_C (= 0)	all data
LDX_TAR_MODE_RAM (= 1)	RAM file system
LDX_TAR_MODE_USRFEP_C (= 2)	User FEPROM
LDX_TAR_MODE_MACODA_C (= 4)	MACODA
LDX_TAR_MODE_TOOLTABLE_C (= 8)	Tool tables
LDX_TAR_MODE_DATAMODULES_C (= 16)	Remanent PLC data function blocks
LDX_TAR_MODE_MARKERS_C (= 32)	Remanent flag of the PLC
LDX_TAR_MODE_CPLVARS_C (= 64)	Permanent CPL variables
LDX_TAR_MODE_TOOLTAB1_C (= 128)	1. Tool table (XML)
LDX_TAR_MODE_TOOLTAB2_C (= 256)	2. Tool table (XML)
LDX_TAR_MODE_SYSTEMDATA_C (= 512)	System data
LDX_TAR_MODE_CREATE_C (= 4096)	Create new archive
LDX_TAR_MODE_EXTRACT_C (= 8192)	Import existing archive

Severe errors which result in a function abort (e. g. insufficient import memory) are described in section TAR Errors on page 232.

The write process is only completed after a complete archiving. During the archiving process, errors which can be detected by reading the item "System.ArchiveLog" are recorded.

Item syntax System.Archive

Call Asynchronous

Access Read/Write

Cyclic (opt.) no

Data format System.Archive, write

row, col	Type	Description	Creating of a complete archive
0,0	string	Archive name	/mnt/complete.tar
0.1	string	Log file name	/mnt/complete.log
0.2	long	Value of the enumerator "LDX_TarMode"	LDX_TAR_MODE_CREATE_C LDX_TAR_MODE_ALL_C

System.Archive, write

row, col	Type	Description	Importing all files of the RAM file system from the archive
0,0	string	Archive name	/mnt/complete.tar
0.1	string	Log file name	/mnt/complete.log
0.2	long	Value of the enumerator "LDX_TarMode"	LDX_TAR_MODE_EXTRACT_C LDX_TAR_MODE_RAM_C

System.ArchiveLog

Description Determines errors which have occurred while creating a new archive or importing an existing archive.

The item returns a list of the archived files or partial archives. Each list entry consists of a name, an error code and a description in English.

Remark To encode the error, the enumerator "LDX_TarLogError" is used. It can assume the following values:

Enumerator LDX_TarLogError

LDX_TarLogError	Description
LDX_TAR_LOG_OK_C (= 0)	No error occurred
LDX_TAR_LOG_NOT_YET_PROCESSED_C (= 1)	The file has not yet been processed
LDX_TAR_LOG_STAT_FAILED_C (= 2)	File status failed
LDX_TAR_LOG_SKIPPED_C (= 3)	The file has been skipped (no error)
LDX_TAR_LOG_PATH_NOT_OPENED_C (= 4)	Error when opening the directory
LDX_TAR_LOG_READING_C (= 5)	Error when reading the file
LDX_TAR_LOG_EACCESS_C (= 6)	Access to file impossible
LDX_TAR_LOG_EEXIST_C (= 7)	File already exists
LDX_TAR_LOG_ENOENT_C (= 8)	File does not exist
LDX_TAR_LOG_ENOMEM_C (= 9)	No memory available
LDX_TAR_LOG_EROFS_C (= 10)	Read-only file system
LDX_TAR_LOG_MODE_DIFF_C (= 11)	Different protection
LDX_TAR_LOG_SIZE_DIFF_C (= 12)	Different size
LDX_TAR_LOG_MTIME_DIFF_C (= 13)	Different date
LDX_TAR_LOG_OTHER_DIFF_C (= 14)	Other differences

LDX_TarLogError	Description
LDX_TAR_LOG_FILE_DATA_DIFF_C (= 15)	Different file data
LDX_TAR_LOG_TIME_MODE_C (= 16)	Protection/time not restored
LDX_TAR_LOG_MKDIR_FAILED_C (= 17)	Error when creating the directory
LDX_TAR_LOG_ERR_UNKNOWN_C (= 18)	Unknown error (-> WERA)
LDX_TAR_LOG_LOADING_MACODA_C (= 19)	Error when loading MACODA
LDX_TAR_LOG_LOADING_TABLE_C (= 20)	Error when loading the tables
LDX_TAR_LOG_CORRUPTED_C (= 21)	File corrupted
LDX_TAR_LOG_WRITE_ENABLED_C (= 22)	Write protection enabled
LDX_TAR_LOG_DATAMODULES_C (= 23)	Error when loading the PLC data
LDX_TAR_LOG_MARKERS_C (= 24)	Error when loading the PLC data
LDX_TAR_LOG_CPLVARS_C (= 25)	Error when transferring CPL variables
LDX_TAR_LOG_RAMFS_NOTFOUND_C (= 26)	RAM file system missing in archive
LDX_TAR_LOG_USRFEPFS_NOTFOUND_C (= 27)	User FEPROM missing in archive
LDX_TAR_LOG_MACODA_NOTFOUND_C (= 28)	MACODA missing in archive
LDX_TAR_LOG_TABLE_NOTFOUND_C (= 29)	Tables missing in archive
LDX_TAR_LOG_DATAMODULES_NOTFOUND_C (= 30)	Remanent PLC data missing
LDX_TAR_LOG_MARKERS_NOTFOUND_C (= 31)	Remanent PLC data missing
LDX_TAR_LOG_CPLVARS_NOTFOUND_C (= 32)	Permanent CPL variables missing
LDX_TAR_LOG_PLCSTATE_RUN_C (= 33)	PLC is not in STOP mode
LDX_TAR_LOG_SYSTEMDATA_C (= 34)	Error when applying system data
LDX_TAR_LOG_SYSTEMDATA_NOTFOUND_C (= 35)	System data missing
LDX_TAR_LOG_INVALID_K1_K2_C (= 36)	File contains invalid data block
LDX_TAR_LOG_SCHEMA_CONFLICT_C (= 37)	Data does not fit to schema file
NCS_TAR_LOG_LOADING_NC_CFG_DATA_C (= 38)	NC configuration data transfer incorrect

Item syntax System.ArchiveLog

Read

Data format Number of lines: Number of partial archives

System.ArchiveLog, read

row, col	Type	Description	Error of the last archiving
n,0	string	Name of the archived file or the partial archives	/
n,1	string	Explanation on the error	OK
n,2	string	Error code	LDX_TAR_LOG_OK_C

System.ArchiveWin

Description Creates an archive with the control data or imports an existing archive where the archive and the log file are stored in the Windows file system.

Remark *The following specifications are required:*

- Archive name
- Log file name
- Mode (Read in archive or create archive, required partial archives)

The enumerator "LDX_TarMode" specifies if an archive is to be created or imported. Furthermore, this enumerator is used to specify the partial archives.

Enumerator LDX_TarMode

LDX_TarMode	Description
LDX_TAR_MODE_ALL_C (= 0)	all data
LDX_TAR_MODE_RAM (= 1)	RAM file system
LDX_TAR_MODE_USRFEP_C (= 2)	User FEPROM
LDX_TAR_MODE_MACODA_C (= 4)	MACODA
LDX_TAR_MODE_TOOLTABLE_C (= 8)	Tool tables
LDX_TAR_MODE_DATAMODULES_C (= 16)	Remanent PLC data function blocks
LDX_TAR_MODE_MARKERS_C (= 32)	Remanent flag of the PLC
LDX_TAR_MODE_CPLVARS_C (= 64)	Permanent CPL variables
LDX_TAR_MODE_TOOLTAB1_C (= 128)	1. Tool table (XML)
LDX_TAR_MODE_TOOLTAB2_C (= 256)	2. Tool table (XML)
LDX_TAR_MODE_SYSTEMDATA_C (= 512)	System data
LDX_TAR_MODE_CREATE_C (= 4096)	Create new archive
LDX_TAR_MODE_EXTRACT_C (= 8192)	Import existing archive

Severe errors which result in a function abort (e. g. insufficient import memory) are described in TAR Errors on page 232.

The write process is only completed after a complete archiving. During the archiving process, errors which can be detected by reading the item "System.ArchiveWinLog" are recorded.

Item syntax **System.ArchiveWin**

Call Asynchronous

Access Read/Write

Cyclic (opt.) no

Data format System.ArchiveWin, write

row, col	Type	Description	Creating of a complete archive
0,0	string	Archive name	c:\mnt\Archive\all.tar
0.1	string	Log file name	c:\mnt\Archive\all.log
0.2	string	Value of the enumerator "LDX_TarMode"	LDX_TAR_MODE_CREATE_C LDX_TAR_MODE_ALL_C

System.ArchiveWinLog

Description Determines errors which have occurred in the Windows file system while creating a new archive or importing an existing archive.

The item returns a list of the archived files or partial archives. Each list entry consists of a name, an error code and a description in English.

Remark To encode the error, the enumerator "LDX_TarLogError" is used. It can assume the following values:

Enumerator LDX_TarLogError

LDX_TarLogError	Description
LDX_TAR_LOG_OK_C (= 0)	No error occurred
LDX_TAR_LOG_NOT_YET_PROCESSED_C (= 1)	The file has not yet been processed
LDX_TAR_LOG_STAT_FAILED_C (= 2)	File status failed
LDX_TAR_LOG_SKIPPED_C (= 3)	The file has been skipped (no error)
LDX_TAR_LOG_PATH_NOT_OPENED_C (= 4)	Error when opening the directory
LDX_TAR_LOG_READING_C (= 5)	Error when reading the file
LDX_TAR_LOG_EACCESS_C (= 6)	Access to file impossible
LDX_TAR_LOG_EEXIST_C (= 7)	File already exists
LDX_TAR_LOG_ENOENT_C (= 8)	File does not exist
LDX_TAR_LOG_ENOMEM_C (= 9)	No memory available
LDX_TAR_LOG_EROFS_C (= 10)	Read-only file system
LDX_TAR_LOG_MODE_DIFF_C (= 11)	Different protection
LDX_TAR_LOG_SIZE_DIFF_C (= 12)	Different size
LDX_TAR_LOG_MTIME_DIFF_C (= 13)	Different date
LDX_TAR_LOG_OTHER_DIFF_C (= 14)	Other differences
LDX_TAR_LOG_FILE_DATA_DIFF_C (= 15)	Different file data
LDX_TAR_LOG_TIME_MODE_C (= 16)	Protection/time not restored
LDX_TAR_LOG_MKDIR_FAILED_C (= 17)	Error when creating the directory
LDX_TAR_LOG_ERR_UNKNOWN_C (= 18)	Unknown error (-> WERA)

LDX_TarLogError	Description
LDX_TAR_LOG_LOADING_MACODA_C (= 19)	Error when loading MACODA
LDX_TAR_LOG_LOADING_TABLE_C (= 20)	Error when loading the tables
LDX_TAR_LOG_CORRUPTED_C (= 21)	File corrupted
LDX_TAR_LOG_WRITE_ENABLED_C (= 22)	Write protection enabled
LDX_TAR_LOG_DATAMODULES_C (= 23)	Error when loading the PLC data
LDX_TAR_LOG_MARKERS_C (= 24)	Error when loading the PLC data
LDX_TAR_LOG_CPLVARS_C (= 25)	Error when transferring CPL variables
LDX_TAR_LOG_RAMFS_NOTFOUND_C (= 26)	RAM file system missing in archive
LDX_TAR_LOG_USRFEPFS_NOTFOUND_C (= 27)	User FEPROM missing in archive
LDX_TAR_LOG_MACODA_NOTFOUND_C (= 28)	MACODA missing in archive
LDX_TAR_LOG_TABLE_NOTFOUND_C (= 29)	Tables missing in archive
LDX_TAR_LOG_DATAMODULES_NOTFOUND_C (= 30)	Remanent PLC data missing
LDX_TAR_LOG_MARKERS_NOTFOUND_C (= 31)	Remanent PLC data missing
LDX_TAR_LOG_CPLVARS_NOTFOUND_C (= 32)	Permanent CPL variables missing
LDX_TAR_LOG_PLCSTATE_RUN_C (= 33)	PLC is not in STOP mode
LDX_TAR_LOG_SYSTEMDATA_C (= 34)	Error when applying system data
LDX_TAR_LOG_SYSTEMDATA_NOTFOUND_C (= 35)	System data missing
LDX_TAR_LOG_INVALID_K1_K2_C (= 36)	File contains invalid data block
LDX_TAR_LOG_SCHEMA_CONFLICT_C (= 37)	Data does not fit to schema file
NCS_TAR_LOG_LOADING_NC_CFG_DATA_C (= 38)	NC configuration data transfer incorrect

Item syntax System.ArchiveWinLog

Read

Data format Number of lines: Number of partial archives

System.ArchiveWinLog, read

row, col	Type	Description	Error of the last archiving
n,0	string	Name of the archived file or the partial archives	/
n,1	string	Explanation on the error	OK
n,2	string	Error code	LDX_TAR_LOG_OK_C

System.HardwareDescription

Description Provides the hardware name of the control used as well as their circuit board number, hardware number and serial number.

Item syntax System.HardwareDescription

Call Synchronous/asynchronous

Access Read

Cyclic (opt.) no

Data format System.HardwareDescription, read

row, col	Type	Description	Example Data of the target system
0,0	string	Designation	CMP60_2
0.1	long	PCB number	170645
0.2	long	Hardware version	101
0,3	long	Serial number	5969810

System.Language

Description Determines the set language or changes the language.

Item syntax System.Language,(1),(2)

1. Numerical country code (3 digits) e.g. 049 for "German"
2. Country code in plain text (e.g. "German", "English"

☞ When writing the item, the specification in parameter 2 **has to** correspond to a two-digit country code (ISO 639-1)!

Examples:

de - German

en - English

fr - French

Call Synchronous/asynchronous

Access Read/Write

Cyclic (opt.) no

Data format System.Language, read

row, col	Type	Description	Determining current language
0,0	string	Current language, number	049
0.1	string	Current language, name	German

System.Language, write

row, col	Type	Description	Language selection
0,0	string	New language, number	033
0.1	string	New language, name	fr

System.MainReleases

Description Returns information on the general software version of the MTX kernel as well as a copyright message (version of the basic software and the customer upgrades).

Item syntax System.MainReleases

Call Synchronous/asynchronous

Access Read

Cyclic (opt.) no

Data format System.MainReleases, read

row, col	Type	Description	Example
0,0	string	Software version	sys Release 100.0.3, 10-03-03
0.1	string	Information on the copyright	(C) Copyright Bosch Rexroth AG 1992 – 2003
1,0	string	Software version	kns release 100.0.3, 10-03-03
1,1	string	Information on the copyright	(C) Copyright Bosch Rexroth AG 1992 – 2003
2,0	string	Software version	knb release 100.0.3, 10-03-03
2,1	string	Information on the copyright	(C) Copyright Bosch Rexroth AG 1992 – 2003

System.PcReleases

Description Provides version information on the Windows software components which (under Windows) provide the communication interface for the MTX kernel.

This includes the NCS interface (ncstyp3.dll), the text management (tsvtyp3.dll), the logical device MTX (SCPBricDevice.dll, SCPBricComponent.dll) and FIS (FisServer.dll).

The following data is provided for each component:

- Subsystem name
- Release numbers (main release, release, subrelease, patch release)
- Time stamp (UTC)

Item syntax System.PcReleases

Call Synchronous/asynchronous

Access Read

Cyclic (opt.) no

Data format Number of lines: Number of components

System.PcReleases, read

row, col	Type	Description	Example
0,0	string	Component name	ncstyp3.dll
0.1	string	Release numbers	100.0.3.0
0.2	date	Time stamp	10.03.03 23:51:12

System.ProVi.KaFreeze

Description Criteria analysis for the frozen status of a ProVi message.

Item syntax System.ProVi.KaFreeze,(1)

1 - Unique message ID - Value returned in column 8 for System.MessageStore

Call Asynchronous

Access Read

Cyclic (opt.) no

Data format The number of lines depend on the size of the code. The number of columns differs in the individual lines.

System.ProVi.KaFreeze, read

row, col	Type	Description	Example
0,0	long	Type of POU in which the message is programmed 1 = Program 2 = Function block	
0.1	string	Name of POU in which the message is programmed	
0.2	long	Type of detail in which the messages are programmed 1 = Implementation 2 = Action 3 = Transition	
0,3	string	Name of detail, empty for implementation	
0,4	long	Network number or line in which the message is programmed	
0,5	string	Variable used as assignment for the ProVi message	
0,6	string	POE instance name	
0,7	long	Type of ProVi message 1 = Not negated, not saving 2 = Negated, not saving 3 = Not negated, saving 4 = Not negated, saving	

row, col	Type	Description	Example
0,8	long	IEC language 1 = ST 2 = IL	
0,9	bool	TRUE - Criteria analysis could not be computed FALSE - Error during criteria analysis computation	
0,10	bool	TRUE - IndraLogic 2G FALSE - IndraLogic 1G	

The other lines depend on the provided values:

0,8 = 2 (IL) and 0,9 = TRUE

row, col	Type	Description	Example
1+x,0	string	Label	
1+x,1	string	Operator	
1+x,2	string	Operand	
1+x,3	string	Status	
1+x,4	string	I/O address	
1+x,5	string	Comment	
1+x,6	long	Variable type	
1+x,7	bool	TRUE - Relevant, show in case of reduced display FALSE - Not relevant, only shown in case of total display	
1+x,8	string	Substitute operator	
1+x,9	bool	TRUE - Shown in case of total display FALSE - Not shown in case of total display	

0,8 = 2 (IL) and 0,9 = FALSE

row, col	Type	Description	Example
1+x,0	string	Label	
1+x,1	string	Operator	
1+x,2	string	Operand	
1+x,3	string	Status	
1+x,4	string	I/O address	
1+x,5	string	Comment	
1+x,6	long	Variable type	

row, col	Type	Description	Example
1+x,7	bool	TRUE - This line contains an error causing that the computation could not be executed FALSE - This line has no error	
1+x,8	string	If column 7 = TRUE error description, otherwise empty	
1+x,9	bool	TRUE - Shown in case of total display FALSE - Not shown in case of total display	

0,8 = 1 (ST)

row, col	Type	Description	Example
1+x,0	string	Code in ST	
1+x,1	string	Error description	

System.ProVi.KaOnline

Description Determination of the criteria analysis data of the transferred ProVi message for the online state It is calculated upon each change of the involved variables of the ProVi message.

Item syntax System.ProVi.KaOnline,(1)

1 - Unique message ID - Value returned in column 8 for System.MessageStore

Call Asynchronous

Access Read

Cyclic (opt.) no

Data format The number of lines depend on the size of the code. The number of columns differs in the individual lines.

System.ProVi.KaOnline, read

row, col	Type	Description	Example
0,0	long	Type of POU in which the message is programmed 1 = Program 2 = Function block	
0.1	string	Name of POU in which the message is programmed	
0.2	long	Type of detail in which the messages are programmed 1 = Implementation 2 = Action 3 = Transition	

row, col	Type	Description	Example
0,3	string	Name of detail, empty for implementation	
0,4	long	Network number or line in which the message is programmed	
0,5	string	Variable used as assignment for the ProVi message	
0,6	string	POE instance name	
0,7	long	Type of ProVi message 1 = Not negated, not saving 2 = Negated, not saving 3 = Not negated, saving 4 = Not negated, saving	
0,8	long	IEC language 1 = ST 2 = IL	
0,9	bool	TRUE - Criteria analysis could not be computed FALSE - Error during criteria analysis computation	
0,10	bool	TRUE - IndraLogic 2G FALSE - IndraLogic 1G	

The other lines depend on the provided values:

0,8 = 2 (IL) and 0,9 = TRUE

row, col	Type	Description	Example
1+x,0	string	Label	
1+x,1	string	Operator	
1+x,2	string	Operand	
1+x,3	string	Status	
1+x,4	string	I/O address	
1+x,5	string	Comment	
1+x,6	long	Variable type	
1+x,7	bool	TRUE - Relevant, show in case of reduced display FALSE - Not relevant, only shown in case of total display	
1+x,8	string	Substitute operator	
1+x,9	bool	TRUE - Shown in case of total display FALSE - Not shown in case of total display	

0,8 = 2 (IL) and 0,9 = FALSE

row, col	Type	Description	Example
1+x,0	string	Label	
1+x,1	string	Operator	
1+x,2	string	Operand	
1+x,3	string	Status	
1+x,4	string	I/O address	
1+x,5	string	Comment	
1+x,6	long	Variable type	
1+x,7	bool	TRUE - This line contains an error causing that the computation could not be executed FALSE - This line has no error	
1+x,8	string	If column 7 = TRUE error description, otherwise empty	
1+x,9	bool	TRUE - Shown in case of total display FALSE - Not shown in case of total display	

0,8 = 1 (ST)

row, col	Type	Description	Example
1+x,0	string	Code in ST	
1+x,1	string	Error description	

System.ProVi.Active

Description Specifies for the selected ProVi message whether the message is pending

Item syntax System.ProVi.Active,(1)

1 - Unique message ID - Value returned in column 8 for System.MessageStore

Call Asynchronous

Access Read

Cyclic (opt.) no

Data format One line, one column.

System.ProVi.Active, read

row, col	Type	Description	Example
0,0	bool	TRUE - Still present FALSE - Not present	

System.Sfc.KaFreeze

Description Criteria analysis for the frozen status of an SFC message.

For time errors, several details can be responsible for the error. If no detail is specified, the detail, in which the error occurs, is automatically calculated. Otherwise, the criteria analysis is calculated for the transferred detail.

Item syntax System.Sfc.KaFreeze,(1)[,(2),(3)]

1 - Unique message ID - Value returned in column 8 for System.MessageStore.

2 - [optional] type of detail for which the criteria analysis is requested. 2 = Action, 3 = transition. Only for time errors.

3 - [optional] name of detail for which the criteria analysis is requested. Only for time errors.

Call Asynchronous

Access Read

Cyclic (opt.) no

Data format The number of lines depend on the size of the code. The number of columns differs in the individual lines.

System.Sfc.KaFreeze, read

row, col	Type	Description	Example
0,0	long	Type of POU in which the message is programmed 1 = Program 2 = Function block	
0.1	string	Name of POU in which the message is programmed	
0.2	long	Type of detail in which the messages are programmed 1 = Implementation 2 = Action 3 = Transition	
0,3	string	Name of detail, empty for implementation	
0,4	long	Network number or line in which the message is programmed Only for monitor errors, otherwise empty	
0,5	long	Message type 3 = Monitor error 4 = Time error 5 = Start prerequisite message 6 = Initial position message	
0,6	string	Name of the step, in which the message is programmed, only in case of time error, otherwise empty	
0,7	string	Instance name of the SFC	

row, col	Type	Description	Example
0,8	long	IEC language 1 = ST 2 = IL	
0,9	bool	TRUE - Criteria analysis could not be computed FALSE - Error during criteria analysis computation	
0,10	long	Type of monitor error 0 = Error if logic result = false 1 = Error if logic result = true Only for monitor errors, otherwise undefined	
0,11	bool	TRUE - IndraLogic 2G FALSE - IndraLogic 1G	
0,12	long	Only for time errors and if the type of detail is “Action”, otherwise undefined 1 = Action is neither called with ON nor with OFF 2 = Call of action with ON 3 = Call of action with OFF	

The other lines depend on the provided values:

0,8 = 2 (IL) and 0,9 = TRUE

row, col	Type	Description	Example
1+x,0	string	Label	
1+x,1	string	Operator	
1+x,2	string	Operand	
1+x,3	string	Status	
1+x,4	string	I/O address	
1+x,5	string	Comment	
1+x,6	long	Variable type	
1+x,7	bool	TRUE - Relevant, show in case of reduced display FALSE - Not relevant, only shown in case of total display	
1+x,8	string	Substitute operator	
1+x,9	bool	TRUE - Shown in case of total display FALSE - Not shown in case of total display	

0,8 = 2 (IL) and 0,9 = FALSE

row, col	Type	Description	Example
1+x,0	string	Label	
1+x,1	string	Operator	
1+x,2	string	Operand	
1+x,3	string	Status	
1+x,4	string	I/O address	
1+x,5	string	Comment	
1+x,6	long	Variable type	
1+x,7	bool	TRUE - This line contains an error causing that the computation could not be executed FALSE - This line has no error	
1+x,8	string	If column 7 = TRUE error description, otherwise empty	
1+x,9	bool	TRUE - Shown in case of total display FALSE - Not shown in case of total display	

0,8 = 1 (ST)

row, col	Type	Description	Example
1+x,0	string	Code in ST	
1+x,1	string	Error description	

System.Sfc.KaOnline

Description Criteria analysis for the online status of an SFC message.

For time errors, several details can be responsible for the error. In this case, specify the detail (parameter 2,3) for which the CA is to be calculated.

Item syntax System.Sfc.KaOnline,(1)[,(2),(3)]

1 - Unique message ID - Value returned in column 8 for System.MessageStore

2 - [Mandatory for time errors] type of detail for which the criteria analysis is requested 2 = Action, 3 = Transition

3 - [Mandatory for time errors] name of detail for which the criteria analysis is requested

Call Asynchronous

Access Read

Cyclic (opt.) no

Data format The number of lines depend on the size of the code. The number of columns differs in the individual lines.

System.Sfc.KaOnline, read

row, col	Type	Description	Example
0,0	long	Type of POU in which the message is programmed 1 = Program 2 = Function block	
0.1	string	Name of POU in which the message is programmed	
0.2	long	Type of detail in which the messages are programmed 2 = Action 3 = Transition Only for monitor errors or time errors, otherwise empty	
0,3	string	Name of detail, empty for implementation	
0,4	long	Network number or line in which the message is programmed Only for monitor errors, otherwise empty	
0,5	long	Message type 3 = Monitor error 4 = Time error 5 = Start prerequisite message 6 = Initial position message	
0,6	string	Name of the step, in which the message is programmed, only in case of time error, otherwise empty	
0,7	string	Instance name of the SFC	
0,8	long	IEC language 1 = ST 2 = IL	
0,9	bool	TRUE - Criteria analysis could not be computed FALSE - Error during criteria analysis computation	
0,10	long	Type of monitor error 0 = Error if logic result = false 1 = Error if logic result = true Only for monitor errors, otherwise undefined	

row, col	Type	Description	Example
0,11	bool	TRUE - IndraLogic 2G FALSE - IndraLogic 1G	
0,12	long	Only for time errors and if the type of detail is “Action”, otherwise undefined 1 = Action is neither called with ON nor with OFF 2 = Call of action with ON 3 = Call of action with OFF	

The other lines depend on the provided values:

0,8 = 2 (IL) and 0,9 = TRUE

row, col	Type	Description	Example
1+x,0	string	Label	
1+x,1	string	Operator	
1+x,2	string	Operand	
1+x,3	string	Status	
1+x,4	string	I/O address	
1+x,5	string	Comment	
1+x,6	long	Variable type	
1+x,7	bool	TRUE - Relevant, show in case of reduced display FALSE - Not relevant, only shown in case of total display	
1+x,8	string	Substitute operator	
1+x,9	bool	TRUE - Shown in case of total display FALSE - Not shown in case of total display	

0,8 = 2 (IL) and 0,9 = FALSE

row, col	Type	Description	Example
1+x,0	string	Label	
1+x,1	string	Operator	
1+x,2	string	Operand	
1+x,3	string	Status	
1+x,4	string	I/O address	
1+x,5	string	Comment	
1+x,6	long	Variable type	

row, col	Type	Description	Example
1+x,7	bool	TRUE - This line contains an error causing that the computation could not be executed FALSE - This line has no error	
1+x,8	string	If column 7 = TRUE error description, otherwise empty	
1+x,9	bool	TRUE - Shown in case of total display FALSE - Not shown in case of total display	

0,8 = 1 (ST)

row, col	Type	Description	Example
1+x,0	string	Code in ST	
1+x,1	string	Error description	

System.Sfc.Mode

Description Provides the operation mode of an SFC.

Item syntax **System.Sfc.Mode,(1)**

1 - Instance of the SFC

Call Asynchronous

Access Read

Cyclic (opt.) no

Data format The number of lines depend on the size of the code. The number of columns differs in the individual lines.

System.Sfc.Mode, read

row, col	Type	Description	Example
0,0	long	Operating mode of the SFC 0 = Unknown 1 = Manual 2 = Auto 3 = Jog 4 = SemiAuto 5 = Stop 6 = Error 7 = Disabled	

System.Sfc.StepInfo

Description Specifies the incorrect actions and transitions for the selected step.

Item syntax **System.Sfc.StepInfo,(1),(2)**

1 - Instance of the SFC

2: Name of step

Call Asynchronous

Access Read

Cyclic (opt.) no

Data format The number of lines corresponds to the number of actions and transitions.

System.Sfc.StepInfo, read

row, col	Type	Description	Example
x,0	long	Type of detail in which the messages are programmed 2 = action, 3 = transition	
x,1	string	Name of the detail	
x,2	bool	TRUE - Action or transition incorrect FALSE - Action or transition correct	

System.Sfc.Active

Description Specifies for the selected SFC message whether the message is still pending

Item syntax **System.Sfc.Active,(1)**

1 - Unique message ID - Value returned in column 8 for System.MessageStore

Call Asynchronous

Access Read

Cyclic (opt.) no

Data format The number of lines depend on the size of the code. The number of columns differs in the individual lines.

System.Sfc.Active, read

row, col	Type	Description	Example
0,0	bool	TRUE - Still present FALSE - Not present	

System.Sfc.Execute

Description Computing feasibility display.

Item syntax **System.Sfc.Execute,(1) {,(2),(3),(4)}**

1 - Instance of the SFC

2 - 1 = Computing feasibility display for one step, 2 = Computing feasibility display for one action

3 - Name of step or action

4 - 1 = Force, 2 = Force on, 3 = Force off >>> Only “1” is permitted for feasibility display for one step.

Note: The parameter triple 2, 3 and 4 can be transferred any number of times. The feasibility display is computed for each one.

Call Asynchronous

Access Read

Cyclic (opt.) no

Data format The number of lines depends on the number of transferred parameters. There is one line for each requested computation

System.Sfc.Execute, read

row, col	Type	Description	Example
x,0	bool	TRUE - The motion can be executed FALSE - The motion cannot be executed	
x,1	bool	TRUE - Error when calculating the feasibility. In this case, column 0 returns TRUE in any case or it was not yet computed. FALSE - No error when calculating the feasibility	
x,2	string	Incorrect variables (separated by comma), only if column 2 returns TRUE, otherwise empty	

The first read call for this item is always provided by:

Column 0: TRUE, Column 1: TRUE, Column 2: Empty

Only after the computation has been completed, the correct results were provided.

System.Sfc.KaExecute

Description Computing criteria analysis for the feasibility display.

Item syntax **System.Sfc.KaExecute,(1),(2),(3),(4)**

1 - Instance of the SFC

2 - 1 = Computing feasibility display for one step, 2 = Computing feasibility display for one action

3 - Name of step or action

4 - 1 = Force, 2 = Force on, 3 = Force off >>> Only “1” is permitted for feasibility display for one step.

Call Asynchronous

Access Read

Cyclic (opt.) no

Data format The number of lines depend on the size of the code. The number of columns differs in the individual lines.

System.Sfc.KaExecute, read

row, col	Type	Description	Example
0,0	long	Type of POU in which the SFC is programmed 1 = Program 2 = Function block	
0.1	string	Name of POU in which the message is programmed	
0.2	long	IEC language 1 = ST 2 = IL	
0,3	bool	TRUE - Criteria analysis could not be computed FALSE - Error during criteria analysis computation	

The other lines depend on the provided values:

0,2 = 2 (IL) and 0,3 = TRUE

row, col	Type	Description	Example
1 + x,0	string	Label	
1 + x,1	string	Operator	
1 + x,2	string	Operand	
1 + x,3	string	Status	
1 + x,4	string	I/O address	
1 + x,5	string	Comment	
1 + x,6	long	Variable type	
1 + x,7	bool	TRUE - Relevant, show in case of reduced display FALSE - Not relevant, only shown in case of total display	
1 + x,8	string	Substitute operator	
1 + x,9	bool	TRUE - Shown in case of total display FALSE - Not shown in case of total display	

0,2 = 2 (IL) and 0,3 = FALSE

row, col	Type	Description	Example
1 + x,0	string	Label	
1 + x,1	string	Operator	
1 + x,2	string	Operand	
1 + x,3	string	Status	

row, col	Type	Description	Example
1 + x,4	string	I/O address	
1 + x,5	string	Comment	
1 + x,6	long	Variable type	
1 + x,7	bool	TRUE - This line contains an error causing that the calculation could not be executed FALSE - This line has no error	
1 + x,8	string	If column 7 = TRUE error description, otherwise empty	
1 + x,9	bool	TRUE - Shown in case of total display FALSE - Not shown in case of total display	

0,2 = 1 (ST)

row, col	Type	Description	Example
1 + x,0	string	Code in ST	
1 + x,1	string	Error description	

System.SubsystemReleases

Description Provides information on software versions of the subsystems in the MTX kernel.

The following data is provided for each subsystem:

- Subsystem name
- Release numbers (main release, release, subrelease, patch release)
- Time stamp (UTC)

Item syntax System.SubsystemReleases

Call Synchronous/asynchronous

Access Read

Cyclic (opt.) no

Data format Number of lines: Number of subsystems

System.SubsystemReleases, read

row, col	Type	Description	Example
0,0	string	Subsystem name	cmp
0.1	string	Release numbers	100.1.2.0
0.2	date	Time stamp	10.03.03 23:49:12

System.SupportedLanguages

Description Determining the available foreign languages. The item returns the list of all supported languages.

Item syntax System.SupportedLanguages

Call Synchronous/asynchronous

Access Read

Cyclic (opt.) no

Data format Number of lines: Number of available foreign languages

System.SupportedLanguages, read

row, col	Type	Description	Example
0,0	string	1. Foreign language, number	049
0.1	string	1. Foreign language, name	German
1,0	string	2. Foreign language, number	044
1,1	string	2. Foreign language, name	English

System.SystemTime

Description Provides or changes the current time.

Item syntax **System.SystemTime**

No parameters

Call Synchronous/asynchronous

Access Read/Write

Cyclic (opt.) Read only

Data format **System.SystemTime, read**

row, col	Type	Description	Example
0,0	date	Current time	03.03.07 11:50:12

System.TimeSync

Description Reads the time synchronization status and configures the time synchronization.

If the time synchronization is active, the control synchronizes its time cyclically using a PC running as time source.

Item syntax **System.TimeSync**

Call Synchronous/asynchronous

Access Read

- Reads the status of the time synchronization

Write

- Configures the time synchronization (incl. enabling and disabling)
- Enables/disables the time synchronization

Cyclic (opt.) no

Data format **System.TimeSync, read**

row, col	Type	Description	Example
0,0	bool	Time synchronization active	TRUE
1,0	string	IP Address of the time source (max. 39 characters)	10.110.248.230

row, col	Type	Description	Example
1,1	long	Port of the time source (0 = default port)	0
1,2	long	Timeout (milliseconds)	20
1,3	long	UpdateRate	3600
2,0	date	Current time (UTC)	31.07.2012 12:37:12
2,1	date	Latest synchronization point in time (UTC)	31.07.2012 12:00:00
2,2	long	Number of timeouts	0

System.TimeSync, write

row, col	Type	Description	Configuring the time synchronization
0,0	bool	Time synchronization active	TRUE
1,0	string	IP Address of the time source (max. 39 characters)	10.110.248.230
1,1	long	Port of the time source (0 = default port)	0
1,2	long	Timeout (milliseconds)	20
1,3	long	UpdateRate	3600

System.TimeSync, write

row, col	Type	Description	Activating the time synchronization
0,0	bool	Time synchronization active	TRUE

System.TimeZone

Description Provides or changes the time zone settings.

Item syntax System.TimeZone

Call Synchronous/asynchronous

Access Read/Write

Cyclic (opt.) no

Data format System.TimeZone, write

row, col	Type	Description	Example
0,0	string	Name of the time zone for standard time	CET
1,0	long	Standard time before Greenwich time in seconds	3600
2,0	string	Name of the time zone for summer time	CEST
3,0	long	Summer time before Greenwich time in seconds	7200

row, col	Type	Description	Example
4,0	long	Summer time starts on the Sunday (00) in the last week (05) of March (03)	0x00030500
5,0	long	Moment of time adjust calculated from midnight in seconds	7200
6,0	long	Summer time ends on the Sunday (00) in the last week (05) of October (0a)	0x000a0500
7,0	long	Moment of time adjust calculated from midnight in seconds	10800

4.8 NC - channel-dependent

4.8.1 General Channel Information

NC.Chan.BalancedConditionState

Description Informs whether axes or spindles are adjusted.

Remark The bit-coded value is "1" during adjustment. Otherwise, the item "0" is reported.

Item syntax **NC.Chan.BalancedConditionState,(1)**

1. Channel number - Integer format, valid channel number

Call Synchronous/asynchronous

Access Read

Cyclic (opt.) Yes

Data format **NC.Chan.BalancedConditionState, read**

row, col	Type	Description	Example
0,0	long	Adjustment status	1

NC.Chan.ChanHidden

Description Indicates whether the channel is hidden in the display. Set via MACODA (600500010).

Item syntax **NC.Chan.ChanHidden,(1)**

1. channel number

Call Synchronous/asynchronous

Access Read

Cyclic (opt.) no

Data format **NC.Chan.ChanHidden, read**

row, col	Type	Description	NC.Chan.ChanHidden,2
0,0	bool	Channel hidden	TRUE

NC.Chan.ChanState

Description Provides the current channel state.

Remark Apart from the channel state, the states of the block processing and of the interpolator are reported. To code all states, the enumerators "LDX_ChanState", "LDX_SavState" and "LDX_IpoState" are used. They can assume the following values:

Channel state: Enumerator LDX_ChanState

LDX_ChanState	Channel state
LDXChanInactive	Inactive
LDXChanReady	Ready
LDXChanRunning	Running
LDXChanStopped	Stopped
LDXChanResetting	Currently resetting
LDXChanCreating	A part program is compiled and checked
LDXChanReadyForNextBlock	Ready for next specification under "MDI"
LDXChanError	Errors

Block processing state: Enumerator LDX_SavState

LDX_SavState	Block processing state
LDX_SAV_OP_STATE_INACTIV_C	Inactive - Program can be selected
LDX_SAV_OP_STATE_READY_C	Ready - Program can be started
LDX_SAV_OP_STATE_RUN_C	A program or an NC block is processed.
LDX_SAV_OP_STATE_WAIT_C	Reserved
LDX_SAV_OP_STATE_HOLD_C	Reserved
LDX_SAV_OP_STATE_ERROR_C	Error - Can only be cleared by a reset or program deselection
LDX_SAV_OP_STATE_SUSPEND_C	Reserved for internal purposes
LDX_SAV_OP_STATE_RESETTING_C	Currently resetting - No jobs may be accepted until the state switches to "inactive".
LDX_SAV_OP_STATE_CREATING_C	A program is selected and prepared (e.g. linked).
LDX_SAV_OP_STATE_CANCEL_DIST_PREP_C	"Cancel distance to go" was triggered, cleaning work in progress
LDX_SAV_OP_STATE_CANCEL_DIST_RUN_C	
LDX_SAV_OP_STATE_READY_FOR_PROGSTART_C	Ready - Program is at program beginning and can be started
LDX_SAV_OP_STATE_READY_FOR_NEXT_BLOCK_C	All blocks at the buffered NC block specification are processed. Waiting for the next specification

Interpolator state: Enumerator LDX_IpoState

LDX_IpoState	Interpolator state
LDX_IPO_OP_STATE_RUN_C	Running
DX_IPO_OP_STATE_SLOPEDOWN_C	Moves down due to feed hold
LDX_IPO_OP_STATE_STOPPED_C	Has stopped axes

Item syntax NC.Chan.ChanState,(1)

1. channel number

Call Synchronous/asynchronous

Access Read

The state of the channel, the block processing and the interpolator in the channel are provided.

Cyclic (opt.) Yes

Data format NC.Chan.ChanState, read

row, col	Type	Description	Example
0,0	long	Channel state	LDXChanReady
0.1	long	Block processing state	LDX_SAV_OP_STATE_READY_FOR_PROGSTART_C
0.2	long	Interpolator state	LDX_IPO_OP_STATE_STOPPED_C

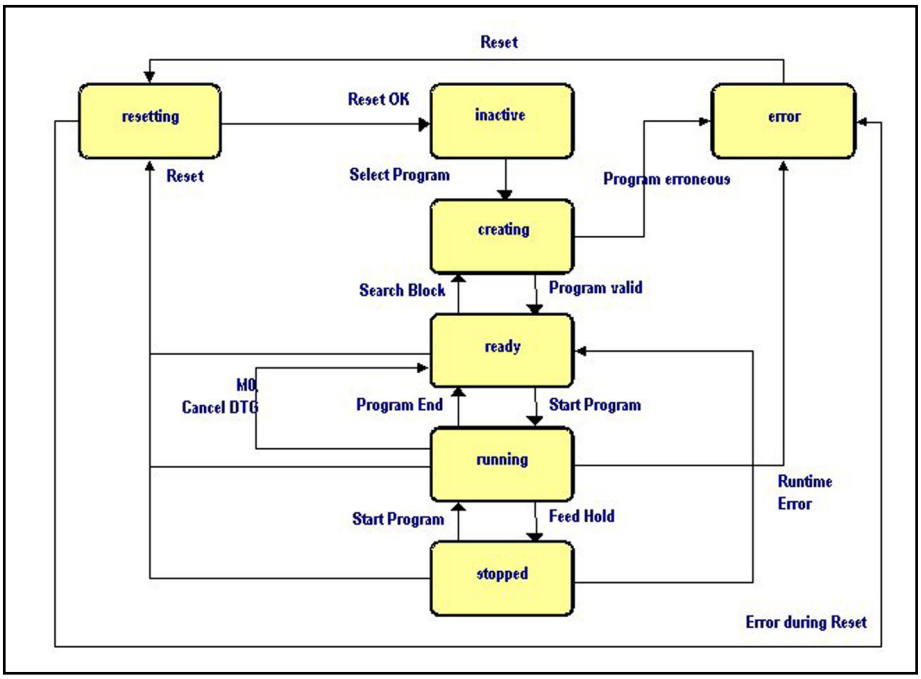

Fig. 4-2: State diagram during execution of part program

Operation mode MDI

- When manual data input blocks are specified, the state passes from "inactive" to "ready".
- After "Start", the state passes from "ready" to "running".
- After completing the manual input blocks, the state passes from "running" to "ready for next block" or "error".

NC.Chan.DryRunFeed

Description Provides the signal states of the control present on input and output side for the test feed.

Item syntax NC.Chan.DryRunFeed,(1)

1. channel number

Call Synchronous/asynchronous

Access Read

Cyclic (opt.) Yes

Data format NC.Chan.DryRunFeed, read

row, col	Type	Description	Example
0,0	long	NC output reports control state (message to the PLC) 0 = Test feed inactive 1 = Tests feed active	0
0.1	long	State of the NC input (Enabling specification of the PLC) 0 = No test feed 1 = Activate test feed	0

NC.Chan.DryRunMode

Description Provides the current status of the channel lock.

Remark Currently, the NC kernel only supports a global complete lock effective across all channels Thus, the value of the "NC.Chan.DryRunMode" item corresponds to that of the "NC.DryRunMode".

Item syntax NC.Chan.DryRunMode,(1)

1. channel number

Call Synchronous/asynchronous

Access Read

Cyclic (opt.) Yes

Data format NC.Chan.DryRunMode, read

row, col	Type	Description	Example
0,0	long	Status of the channel lock 0 = No test mode 1 = Test mode	1

NC.Chan.DryRunRapid

Description Provides the signal states of the control present on input and output side for the test rapid traverse.

Item syntax NC.Chan.DryRunRapid,(1)

1. channel number

Call Synchronous/asynchronous

Access Read

Cyclic (opt.) Yes

Data format NC.Chan.DryRunRapid, read

row, col	Type	Description	Example
0,0	long	NC output reports control state (message to the PLC) 0 = Test rapid traverse inactive 1 = Test rapid traverse active	0
0.1	long	State of the NC input (Enabling specification of the PLC) 0 = No test rapid traverse 1 = Activate test rapid traverse	0

NC.Chan.FunctionBaseConfiguration

Description Provides a list of all groups knows and their respective functions in the control.

Item syntax NC.Chan.FunctionBaseConfiguration,(1)

1. channel number (optional)

☞ The item provides information without specification of a parameter. This information was determined during the startup of the logical device. The specification of a channel number causes a new reading of this control information for each call.

Call Synchronous/asynchronous

Access Read

Cyclic (opt.) Without parameter yes

With parameter no

Data format The list has currently a length of 129 lines.

The data of column 1 is of type "long"

The data of column 2 is of type "string"

The blank is used as separator between several NC functions.

NC.Chan.FunctionBaseConfiguration, read

row, col	Type	Description	Example
0,0	long	Group number	0
0.1	string	Functions of group 0	CTA
1,0	long	Group number	1
1,1	string	Functions of group 1	SME SMD
2,0	long	Group number	2
2,1	string	Functions of group 2	G80 G81 G82 G83 G84 G85 G86
n,0	long	...	...
n,1	string	...	...
128,0	long	Group number	3
128,1	string	Functions of group 128	KNBG1 KNBG0

NC.Chan.ProgramLinkInfo

Description Provides the active program with path names and its selection criteria from the requested channel.

Remark Under "LDX_RFlags", the selection criteria are available as bit-coded enumerators. This enumeration type can assume the following values:

Enumerator LDX_Sav_Flag

LDX_Sav_Flag	Description
LDX_SAV_FLAG_LINK_BEFORE_READY_C	If this flag is set, a linker run is performed during selection. During this run, a linker table is created for the main program. This table is required if subroutine calls or CPL elements exist in the program.
LDX_SAV_FLAG_RETURN_BY_READY_C	Usually, upon selection, it is checked via the operation mode and the NC state if the selection is allowed. Afterwards, the selection is acknowledged. If this flag is set, the acknowledgement is made only after the completion of the selection, i.e. when the NC status is "ready".
LDX_SAV_FLAG_DA_C	Reserved
LDX_SAV_FLAG_REMOTE_C	Reserved
LDX_SAV_FLAG_SUPERVISOR_C	Reserved
LDX_SAV_FLAG_MDI_C	This flag has to be set if a block is to be processed in "Manual data input" operation mode.
LDX_SAV_FLAG_PROCESSDELETE_C	An active program or an active MDI is deselected first.
LDX_SAV_FLAG_AUTOSTART_C	Together with LDX_SAV_FLAG_MDI_C: The input block is started immediately. If the specified channel is not active, the block is started in this channel. All NC blocks are then permitted. If the specified channel is active, the block is handled as a switching function. Restrictions -> LDX_SAV_FLAG_MACHINE_C
LDX_SAV_FLAG_JOG_C	This flag has to be set if axes are to be traveled in the operation mode "jog mode" or if manual workpiece coordinate motions are to be carried out in the "setup" mode.
LDX_SAV_FLAG_REFPT_C	This flag has to be set if axes are to be started in the "travel to reference point" mode.
LDX_SAV_FLAG_MACHINE_C	Switching function together with LDX_SAV_FLAG_MDI_C. The block is executed parallely to the specified channel. However, only auxiliary functions and asynchronous axis motions permitted with switching function.

LDX_Sav_Flag	Description
LDX_SAV_FLAG_PROCESSCHANGE_C	If this flag is set, a program already active at program begin is not aborted by deselection, rather replaced by the newly selected one, i.e. the modal states remain. In manual data input, the old character sequence is replaced by the new one.
LDX_SAV_FLAG_BUFFERED_INPUT_C	Buffered NC specification together with LDX_SAV_FLAG_MDI_C. While blocks are still executed, others can be specified.
LDX_SAV_FLAG_NO_CHAN_CHANGE_C	The specified channel is kept. No implicit change to the Auto start channel.
LDX_SAV_FLAG_RSP_IMMEDIATELY_C	The job is acknowledged after processing and not only after the block became active.

Item syntax NC.Chan.ProgramLinkInfo,(1)

1. channel number

Call Synchronous/asynchronous

Access Read

Cyclic (opt.) Yes

Data format NC.Chan.ProgramLinkInfo, read

row, col	Type	Description	Example
0,0	string	Active program	/mnt/startup-testphase/Ex.npg
0.1	long	Call status	LDX_SAV_FLAG_LINK_BEFORE_READY_C

NC.Chan.WaitingStates

Description Provides the current waiting states of a channel.

Remark Enumerator "LDX_McoWaitingState" is available for waiting states. It can assume the following values:

Enumerator LDX_McoWaitingState

LDX_McoWaitingState	Description
LDX_McoDwellTime_C	Dwell time
LDX_McoAuxiliaryFunction_C	Auxiliary function subject to acknowledgement
LDX_McoBlockTransferInhibit_C	Block transfer inhibit
LDX_McoChannelFeedrate0_C	Feed in the channel is equal to 0
LDX_McoProgramStop_C	Program stop with M0/M1
LDX_McoFeedHold_C	Feed halt in the channel
LDX_McoFeedInhibit_C	Feed inhibit in channel or channel axis
LDX_McoCustomerStop_C	Customized block transfer inhibit

LDX_McoWaitingState	Description
LDX_McoSynchronizedMovementStop_C	Synchronized motion stop between channels (ASTOP, ...)
LDX_McoWaitingForAxisTransfer_C	Waiting for axis upon axis transfer (G511)
LDX_McoWaitingForPermanentVariable_C	Waiting for permanent variable (WPV)
LDX_McoWaitingForIFSignalActive_C	Waiting for interface signal at active point of time (WAITA, ...)
LDX_McoWaitingForIFSignalPreparation_C	Waiting for interface signal [WAIT(IC...)] or for a specified interval (WAIT(,TIME%)) during block preparation
LDX_McoMovementCrontrolDataService_C	Motion Control Data Service (MCODS(...))
LDX_MCoWaitingForInpos_C	Waiting for channel data inpos
LDX_MCoFeedProgramming_C	Waiting for G95 programming
LDX_MCoWaitingForSpindleG33_C	Waiting till main spindle rotates G33
LDX_MCoNcProgramRestart_C	Waiting for block pre-run
LDX_MCoWaitingForSpindle_C	Waiting till speed reached, spindle stop...
LDX_MCoWaitingForEvent_C	Waiting on event >>> WEV function
LDX_MCoWaitingForInternalResource_C	Waiting on internal resource (e.g. memory)

Item syntax **NC.Chan.WaitingStates,(1)**

1. channel number

Call Synchronous/asynchronous

Access Read

Cyclic (opt.) Yes

Data format **NC.Chan.WaitingStates, read**

row, col	Type	Description	Example
0,0	long	Active waiting state in the queried channel: "Feed hold"	32 - LDX_MCoFeedHold_C

4.8.2 Axis positions

NC.Chan.AxisPosAcs

Description Provides the current axis positions in respect to the axis coordinate system.

Item syntax **NC.Chan.AxisPosAcs,(1)**

1. channel number

Call Synchronous/asynchronous

Access Read

Additional information is provided in the first seven columns for each axis. The eighth column contains the position value of the axis.

Cyclic (opt.) Yes

Data format Number of lines: Corresponds to the number of axes

NC.Chan.AxisPosAcs, read

row, col	Type	Description	Example
n,0	long	Phys. axis or channel number	2
n,1	long	Channel assignment	1
n,2	string	Logical axis name or coordinate name	Y
n,3	long	Axis in position 0 = FALSE 1 = TRUE	1
n,4	long	Reference point known	1
n,5	long	Measuring unit	LDX_McoUnitMetric_C
n,6	long	Diameter programming active	0
n,7	double	Position value	

NC.Chan.AxisPosAcsCoupl

Description Provides the current axis positions with respect to the axis coordinate system with system axis coupling.

Item syntax NC.Chan.AxisPosAcsCoupl,(1)

1. channel number

Call Synchronous/asynchronous

Access Read

Additional information is provided in the first seven columns for each axis. The eighth column contains the position value of the axis.

Cyclic (opt.) Yes

Data format Number of lines: Corresponds to the number of axes

NC.Chan.AxisPosAcsCoupl, read

row, col	Type	Description	Example
n,0	long	Phys. axis or channel number	2
n,1	long	Channel assignment	1
n,2	string	Logical axis name or coordinate name	Y
n,3	long	Axis in position 0 = FALSE 1 = TRUE	1
n,4	long	Reference point known	1
n,5	long	Measuring unit	LDX_McoUnitMetric_C
n,6	long	Diameter programming active	0
n,7	double	Position value	

NC.Chan.AxisPosAcsDrv

Description Provides the current actual axis positions with respect to the axis coordinate system. The positions are directly provided from the drive data.

Item syntax NC.Chan.AxisPosAcsDrv,(1)

1. channel number

Call Synchronous/asynchronous

Access Read

Additional information is provided in the first seven columns for each axis. The eighth column contains the position value of the axis.

Cyclic (opt.) Yes

Data format Number of lines: Corresponds to the number of axes

NC.Chan.AxisPosAcsDrv, read

row, col	Type	Description	Example
n,0	long	Phys. axis or channel number	2
n,1	long	Channel assignment	1
n,2	string	Logical axis name or coordinate name	Y
n,3	long	Axis in position 0 = FALSE 1 = TRUE	1
n,4	long	Reference point known	1
n,5	long	Measuring unit	LDX_McoUnitMetric_C
n,6	long	Diameter programming active	0
n,7	double	Position value	

NC.Chan.AxisPosAcsDrvAdj

Description Provides the current actual axis positions with respect to the axis coordinate system corrected by Lsec, CComp and TempComp. The actual positions are directly provided from the drive data.

Item syntax NC.Chan.AxisPosAcsDrvAdj,(1)

1. channel number

Call Synchronous/asynchronous

Access Read

Additional information is provided in the first seven columns for each axis. The eighth column contains the position value of the axis.

Cyclic (opt.) Yes

Data format Number of lines: Corresponds to the number of axes

NC.Chan.AxisPosAcsDrvAdj, read

row, col	Type	Description	Example
n,0	long	Phys. axis or channel number	2
n,1	long	Channel assignment	1
n,2	string	Logical axis name or coordinate name	Y

row, col	Type	Description	Example
n,3	long	Axis in position 0 = FALSE 1 = TRUE	1
n,4	long	Reference point known	1
n,5	long	Measuring unit	LDX_McoUnitMetric_C
n,6	long	Diameter programming active	0
n,7	double	Position value	4.123

NC.Chan.AxisPosBcs

Description Provides the current axis positions with respect to the basic coordinate system.

Item syntax NC.Chan.AxisPosBcs,(1)

1. channel number

Call Synchronous/asynchronous

Access Read

Additional information is provided in the first seven columns for each axis. The eighth column contains the position value of the axis.

Cyclic (opt.) Yes

Data format Number of lines: Corresponds to the number of axes

NC.Chan.AxisPosBcs, read

row, col	Type	Description	Example
n,0	long	Phys. axis or channel number	2
n,1	long	Channel assignment	1
n,2	string	Logical axis name or coordinate name	Y
n,3	long	Axis in position 0 = FALSE 1 = TRUE	1
n,4	long	Reference point known	1
n,5	long	Measuring unit	LDX_McoUnitMetric_C
n,6	long	Diameter programming active	0
n,7	double	Position value	

NC.Chan.AxisPosBcsDrv

Description Provides the current actual axis positions with respect to the basic coordinate system. The positions are directly provided from the drive data.

Item syntax NC.Chan.AxisPosBcsDrv,(1)

1. channel number

Call Synchronous/asynchronous

Access Read

Additional information is provided in the first seven columns for each axis. The eighth column contains the position value of the axis.

Cyclic (opt.) Yes

Data format Number of lines: Corresponds to the number of axes

NC.Chan.AxisPosBcsDrv, read

row, col	Type	Description	Example
n,0	long	Phys. axis or channel number	2
n,1	long	Channel assignment	1
n,2	string	Logical axis name or coordinate name	Y
n,3	long	Axis in position 0 = FALSE 1 = TRUE	1
n,4	long	Reference point known	1
n,5	long	Measuring unit	LDX_McoUnitMetric_C
n,6	long	Diameter programming active	0
n,7	double	Position value	

NC.Chan.AxisPosDistAcs

Description Provides the distance to go with respect to the axis coordinate system.

Item syntax **NC.Chan.AxisPosAcsCoupl,(1)**

1. channel number

Call Synchronous/asynchronous

Access Read

Additional information is provided in the first seven columns for each axis. The eighth column contains the position value of the axis.

Cyclic (opt.) Yes

Data format Number of lines: Corresponds to the number of axes

NC.Chan.AxisPosDistAcs, read

row, col	Type	Description	Example
n,0	long	Phys. axis or channel number	2
n,1	long	Channel assignment	1
n,2	string	Logical axis name or coordinate name	Y
n,3	long	Axis in position 0 = FALSE 1 = TRUE	1
n,4	long	Reference point known	1
n,5	long	Measuring unit	LDX_McoUnitMetric_C
n,6	long	Diameter programming active	0
n,7	double	Distance to go	

NC.Chan.AxisPosDistMcs

Description Provides the distance to go with respect to the machine coordinate system.

Item syntax **NC.Chan.AxisPosDistMcs,(1)**

1. channel number

Call Synchronous/asynchronous

Access Read

Additional information is provided for each axis in the first columns. The last column contains the distance to go.

Cyclic (opt.) Yes

Data format Number of lines: Corresponds to the number of axes

NC.Chan.AxisPosDistMcs, read

row, col	Type	Description	Example
n,0	long	Phys. axis or channel number	2
n,1	long	Channel assignment	1
n,2	string	Logical axis name or coordinate name	Y
n,3	long	Axis in position 0 = FALSE 1 = TRUE	1
n,4	long	Reference point known	1
n,5	long	Measuring unit	LDX_McoUnitMetric_C
n,6	long	Diameter programming active	0
n,7	double	Distance to go	123.456

NC.Chan.AxisPosDistWcs

Description Provides the distance to go with respect to the workpiece coordinate system.

Item syntax **NC.Chan.AxisPosDistWcs,(1)**

1. channel number

Call Synchronous/asynchronous

Access Read

Additional information is provided in the first seven columns for each axis. The eighth column contains the distance to go.

Cyclic (opt.) Yes

Data format Number of lines: Corresponds to the number of axes

NC.Chan.AxisPosDistWcs, read

row, col	Type	Description	Example
n,0	long	Phys. axis or channel number	2
n,1	long	Channel assignment	1
n,2	string	Logical axis name or coordinate name	Y

row, col	Type	Description	Example
n,3	long	Axis in position 0 = FALSE 1 = TRUE	1
n,4	long	Reference point known	1
n,5	long	Measuring unit	LDX_McoUnitMetric_C
n,6	long	Diameter programming active	0
n,7	double	Distance to go	

NC.Chan.AxisPosEndWcs

Description Provides the axis end positions with respect to the workpiece coordinate system. The positions comprise corrections active to the programmed end value.

Item syntax NC.Chan.AxisPosEndWcs,(1)

1. channel number

Call Synchronous/asynchronous

Access Read

Additional information is provided in the first seven columns for each axis. The eighth column contains the position value of the axis.

Cyclic (opt.) Yes

Data format Number of lines: Corresponds to the number of axes

NC.Chan.AxisPosEndWcs, read

row, col	Type	Description	Example
n,0	long	Phys. axis or channel number	2
n,1	long	Channel assignment	1
n,2	string	Logical axis name or coordinate name	Y
n,3	long	Axis in position 0 = FALSE 1 = TRUE	1
n,4	long	Reference point known	1
n,5	long	Measuring unit	LDX_McoUnitMetric_C
n,6	long	Diameter programming active	0
n,7	double	End position	

NC.Chan.AxisPosLag

Description Provides the lag of axes.

Item syntax NC.Chan.AxisPosLag,(1)

1. channel number

Call Synchronous/asynchronous

Access Read

Additional information is provided in the first seven columns for each axis. The eighth column contains the lag of axis.

Cyclic (opt.) Yes

Data format Number of lines: Corresponds to the number of axes

NC.Chan.AxisPosLag, read

row, col	Type	Description	Example
n,0	long	Phys. axis or channel number	2
n,1	long	Channel assignment	1
n,2	string	Logical axis name or coordinate name	Y
n,3	long	Axis in position 0 = FALSE 1 = TRUE	1
n,4	long	Reference point known	1
n,5	long	Measuring unit	LDX_McoUnitMetric_C
n,6	long	Diameter programming active	0
n,7	double	Lag	

NC.Chan.AxisPosMcs

Description Provides the current axis positions with respect to the machine coordinate system of the axes.

☞ For an active axis transformation (point of action 1), coordinate values are provided instead of axis positions! The negative numerical value (corresponds to machine parameter 7080 00020 from -6 to -1) in columns n,0 is used for differentiation purposes.

Item syntax **NC.Chan.AxisPosMcs,(1)**

1. channel number

Call Synchronous/asynchronous

Access Read

Additional information is provided in the first seven columns for each axis. The eighth column contains the position value of the axis.

Cyclic (opt.) Yes

Data format Number of lines: Corresponds to the number of axes

NC.Chan.AxisPosMcs, read

row, col	Type	Description	Example
n,0	long	Phys. axis number or channel number (1 to 64) or the MCS coordinates (-6 to -1) for an axis transformation	2
n,1	long	Channel assignment	1
n,2	string	Logical axis name or coordinate name	Y

row, col	Type	Description	Example
n,3	long	Axis in position 0 = FALSE 1 = TRUE	1
n,4	long	Reference point known	1
n,5	long	Measuring unit	LDX_McoUnitMetric_C
n,6	long	Diameter programming active	0
n,7	double	Position value	5.462

NC.Chan.AxisPosProg

Description Provides the programmed axis positions.

Item syntax NC.Chan.AxisPosProg,(1)

1. channel number

Call Synchronous/asynchronous

Access Read

Additional information is provided in the first seven columns for each axis. The eighth column contains the position value of the axis.

Cyclic (opt.) Yes

Data format Number of lines: Corresponds to the number of axes

NC.Chan.AxisPosProg, read

row, col	Type	Description	Example
n,0	long	Phys. axis or channel number	2
n,1	long	Channel assignment	1
n,2	string	Logical axis name or coordinate name	Y
n,3	long	Axis in position 0 = FALSE 1 = TRUE	1
n,4	long	Reference point known	1
n,5	long	Measuring unit	LDX_McoUnitMetric_C
n,6	long	Diameter programming active	0
n,7	double	Position value	

NC.Chan.AxisPosWcs

Description Provides the current axis positions with respect to the workpiece coordinate system of the axes.

☞ For an active axis transformation (point of action 1 or 2), coordinate values are provided instead of axis positions! The negative numerical value (corresponds to machine parameter 7080 00010 from -12 to -7) in columns n,0 is used for differentiation purposes.

Item syntax NC.Chan.AxisPosWcs,(1)

1. channel number

Call Synchronous/asynchronous

Access Read

Additional information is provided in the first seven columns for each axis. The eighth column contains the position value of the axis.

Cyclic (opt.) Yes

Data format Number of lines: Corresponds to the number of axes

NC.Chan.AxisPosWcs, read

row, col	Type	Description	Example
n,0	long	Phys. axis number or channel number (1 to 64) or the WCS coordinates (-12 to -7) for an axis transformation	2
n,1	long	Channel assignment	1
n,2	string	Logical axis name or coordinate name	Y
n,3	long	Axis in position 0 = FALSE 1 = TRUE	1
n,4	long	Reference point known	1
n,5	long	Measuring unit	LDX_McoUnitMetric_C
n,6	long	Diameter programming active	0
n,7	double	Position value	3.456

NC.Chan.AxisPosWcsDrv

Description Provides the Cartesian axis positions with respect to the workpiece coordinate system of the axes.

Item syntax NC.Chan.AxisPosWcsDrv,(1)

1. channel number

Call Synchronous/asynchronous

Access Read

Additional information is provided in the first seven columns for each axis. The eighth column contains the position value of the axis.

Cyclic (opt.) Yes

Data format Number of lines: Corresponds to the number of axes

NC.Chan.AxisPosWcsDrv, read

row, col	Type	Description	Example
n,0	long	Phys. axis number or channel number (1 to 64) or the WCS coordinates (-12 to -7) for an axis transformation	1
n,1	long	Channel assignment	1
n,2	string	Logical axis name or coordinate name	X

row, col	Type	Description	Example
n,3	long	Axis in position 0 = FALSE 1 = TRUE	1
n,4	long	Reference point known	1
n,5	long	Measuring unit	LDX_McoUnitMetric_C
n,6	long	Diameter programming active	0
n,7	double	Position value	12.456

4.8.3 Axis corrections

NC.Chan.ActAxisOffsetTable

Description Provides the name and the path of the active axis offset table.

Item syntax **NC.Chan.ActAxisOffsetTable,(1)**

1. channel number

Call Synchronous/asynchronous

Access Read

Cyclic (opt.) Yes

Data format **NC.Chan.ActAxisOffsetTable, read**

row, col	Type	Description	Example
0,0	string	Name and path of the table	/database/axisoffsets.aot

NC.Chan.ActDCorrectionNmb

Description Provides the active D-correction number.

Item syntax **NC.Chan.ActDCorrectionNmb,(1)**

1. channel number

Call Synchronous/asynchronous

Access Read

Cyclic (opt.) Yes

Data format **NC.Chan.ActDCorrectionNmb, read**

row, col	Type	Description	Example
0,0	long	Active D-correction number	2

NC.Chan.ActDCorrectionTable

Description Provides name and path of the active D-correction table.

Item syntax **NC.Chan.ActDCorrectionTable,(1)**

1. channel number

Call Synchronous/asynchronous

Access Read

Cyclic (opt.) Yes

Data format NC.Chan.ActDCorrectionTable, read

row, col	Type	Description	Example
0,0	string	Name and path of the table	/database/sample.dct

NC.Chan.ActDCorrectionValues

Description Provides the active D-correction values, i.e. the active values for length and radius correction, as well as the tool edge position (orientation).

Item syntax NC.Chan.ActDCorrectionValues,(1)

1. channel number

Call Synchronous/asynchronous

Access Read

Cyclic (opt.) Yes

Data format NC.Chan.ActDCorrectionValues, read

row, col	Type	Description	Example
0,0	double	Length correction 1	2.65
0.1	double	Length correction 2	20.5
0.2	double	Length correction 3	0.109
0,3	double	Radius correction	9.04
0,4	double	Orientation	0

NC.Chan.ActOnlineCorrValue

Description Provides the active online correction values (WCS) for all axes of the channel.

Remark Additional data is provided in the first seven columns. The eighth column contains the correction value.

Item syntax NC.Chan.ActOnlineCorrValue,(1)

1. channel number

Call Synchronous/asynchronous

Access Read

Cyclic (opt.) Yes

Data format Number of lines: Corresponds to the number of axes

NC.Chan.ActOnlineCorrValue, read

row, col	Type	Description	Example
n,0	long	Physical axis or coordinate number	2
n,1	long	Channel assignment	1
n,2	string	Axis name or coordinate name	Y
n,3	bool	Axis in position	TRUE
n,4	bool	Reference point known	TRUE
n,5	long	Measuring unit	LDX_MCoUnitMetric_C

row, col	Type	Description	Example
n,6	bool	Diameter programming	FALSE
n,7	double	Active online correction value(s)	12.345

NC.Chan.ActPlacementBcrCorr

Description Provides the active offset values and angles values to the workpiece position correction.

Remark If no workpiece position correction is active, 0 is always returned.

Item syntax NC.Chan.ActPlacementBcrCorr,(1)

1. channel number

Call Synchronous/asynchronous

Access Read

Cyclic (opt.) Yes

Data format NC.Chan.ActPlacementBcrCorr, read

row, col	Type	Description	Example
0,0	double	X offset	40
0.1	double	Y offset	70.6
0.2	double	Z offset	-5.1
0,3	double	Phi angle	75
0,4	double	Theta angle	45
0,5	double	Psi angle	35

NC.Chan.ActPlacementData

Description Provides the active placement data of five banks with the related three coordinate values and three angle values.

Remark If no placement is active for a bank (plane) anymore, 0 is returned in each case.

Item syntax NC.Chan.ActPlacementData,(1)

1. channel number

Call Synchronous/asynchronous

Access Read

Cyclic (opt.) Yes

Data format Number of lines: 5 (corresponds to the number of banks)

NC.Chan.ActPlacementData, read

row, col	Type	Description	Example
n,0	double	X coordinate	4
n,1	double	Y coordinate	0.6
n,2	double	Z coordinate	1
n,3	double	Phi angle	90

row, col	Type	Description	Example
n,4	double	Theta angle	45
n,5	double	Psi angle	135

NC.Chan.ActPlacementSumData

Description Provides the active sum values of the placements and the workpiece position correction.

Remark If no placements and no workpiece position correction is active, 0 is returned.

Item syntax **NC.Chan.ActPlacementSumData,(1)**

1. channel number

Call Synchronous/asynchronous

Access Read

Cyclic (opt.) Yes

Data format **NC.Chan.ActPlacementSumData, read**

row, col	Type	Description	Example
0,0	double	X sum	25
0.1	double	Y sum	10.6
0.2	double	Z sum	0.0
0,3	double	Phi sum	45
0,4	double	Theta sum	12
0,5	double	Psi sum	135

NC.Chan.ActPlacementTable

Description Provides the active table name of the placements with path specification.

Remark The maximum character length is 100 characters (incl. path).

If no table is active, 3 spaces are provided.

Item syntax **NC.Chan.ActPlacementTable,(1)**

1. channel number

Call Synchronous/asynchronous

Access Read

Cyclic (opt.) Yes

Data format **NC.Chan.ActPlacementTable, read**

row, col	Type	Description	Example
0,0	string	Name and path of the table	/mnt/example.pmt

NC.Chan.ActProgCoordATrans

Description Provides the additive program coordinate offset values of all the axes of the channel selected.

Remark Additional information is provided in the first seven columns for each axis. The eighth column always contains the offset value of the axis.

Item syntax **NC.Chan.ActProgCoordATrans,(1)**

1. Channel number - Integer format, valid channel number

Call Synchronous/asynchronous

Access Read

Cyclic (opt.) Yes

Data format Number of lines corresponds to the number of axes

NC.Chan.ActProgCoordATrans, read

row, col	Type	Description	Example
n,0	long	Phys. axis or channel number	1
n,1	long	Channel assignment	1
n,2	string	Logical axis name or coordinate name	X
n,3	long	Axis in position 0 = No 1 = Yes	0
n,4	long	Reference point known 0 = No 1 = Yes	1
n,5	long	Measuring unit	LDX_McoUnitMetric_C
n,6	long	Diameter programming active 0 = No 1 = Yes	0
n,7	double	Program coordinate offset value	5.4

NC.Chan.ActProgCoordTrans

Description Provides the program coordinate offset values of all the axes of the channel selected.

Remark Additional information is provided in the first seven columns for each axis. The eighth column always contains the offset value of the axis.

Item syntax **NC.Chan.ActProgCoordTrans,(1)**

1. Channel number - Integer format, valid channel number

Call Synchronous/asynchronous

Access Read

Cyclic (opt.) Yes

Data format see NC.Chan.ActProgCoordATrans

NC.Chan.ActProgCoordTransSum

Description Provides the sum of the program coordinate offset values from TRANS and ATRANS of all the axes of the channel selected.

Remark Additional information is provided in the first seven columns for each axis. The eighth column always contains the sum offset value of the axis.

Item syntax **NC.Chan.ActProgCoordTransSum,(1)**

1. Channel number - Integer format, valid channel number

Call Synchronous/asynchronous

Access Read

Cyclic (opt.) Yes

Data format see NC.Chan.ActProgCoordATrans

NC.Chan.ActToolCorrectionValues

Description Provides the active tool correction values.

Remark The values for length correction and radius correction are provided as well as edge position, Euler angle (orientation in space) and edge number.

Item syntax **NC.Chan.ActToolCorrectionValues,(1)**

1. channel number

Call Synchronous/asynchronous

Access Read

Cyclic (opt.) Yes

Data format **NC.Chan.ActToolCorrectionValues, read**

row, col	Type	Description	Example
0,0	double	Length correction 1	2.65
0.1	double	Length correction 2	20.5
0.2	double	Length correction 3	0.109
0,3	double	Radius correction	9.04
0,4	double	Orientation	0.0
0,5	double	Φ	0.0
0,6	double	υ	0.0
0,7	double	ψ	0.0
0,8	double	Tool edge number	4.0

NC.Chan.ActZeroOffsetSummary

Description Provides the sum of the active zero offset values of all ZO groups for all axes in the channel.

Remark Additional data is provided in the first seven columns. The eighth column contains the sum of the offset values.

Item syntax **NC.Chan.ActZeroOffsetSummary,(1)**

1. channel number

Call Synchronous/asynchronous

Access Read

Cyclic (opt.) Yes

Data format Number of lines: Corresponds to the number of axes

NC.Chan.ActZeroOffsetSummary, read

row, col	Type	Description	Example
n,0	long	Physical axis or coordinate number	2
n,1	long	Channel assignment	1
n,2	string	Axis name or coordinate name	Y
n,3	bool	Axis in position	TRUE

row, col	Type	Description	Example
n,4	bool	Reference point known	TRUE
n,5	long	Measuring unit	LDX_MCoUnitMetric_C
n,6	bool	Diameter programming	FALSE
n,7	double	Sum of the offset values	13.5678

NC.Chan.ActZeroOffsetValues

Description Provides the active zero offset values of all ZO groups for all axes in the channel.

Remark Additional data is provided in the first seven columns.

The eighth to twelfth columns contain the values for the basic group and the values of the first to fourth additive group.

Item syntax NC.Chan.ActZeroOffsetValues,(1)

1. channel number

Call Synchronous/asynchronous

Access Read

Cyclic (opt.) Yes

Data format Number of lines: Corresponds to the number of axes

NC.Chan.ActZeroOffsetValues, read

row, col	Type	Description	Example
n,0	long	Physical axis or coordinate number	2
n,1	long	Channel assignment	1
n,2	string	Axis name or coordinate name	Y
n,3	bool	Axis in position	TRUE
n,4	bool	Reference point known	TRUE
n,5	long	Measuring unit	LDX_MCoUnitMetric_C
n,6	bool	Diameter programming	FALSE
n,7	double	Offset value of the basic group	0.04
n,8	double	Offset value of the first additive group	0.5
n,9	double	Offset value of the second additive group	0.44
n,10	double	Offset value of the third additive group	0.0
n,11	double	Offset value of the fourth additive group	0.0

NC.Chan.AxisShift

Description The item reads the values containing the offset of a programmed contour parallely displaced to the axes of the program coordinate system.

Remark Additional information is provided in the first seven columns for each axis. The eighth column always contains the offset value of the axis.

Item syntax NC.Chan.AxisShift,(1)

1. Channel number - Integer format, valid channel number

Call Synchronous/asynchronous

Access Read

Cyclic (opt.) Yes

Data format NC.Chan.AxisShift, read

row, col	Type	Description	Example
n,0	long	Phys. axis or channel number	2
n,1	long	Channel assignment	1
n,2	string	Logical axis name or coordinate name	Y
n,3	long	Axis in position 0 = No 1 = Yes	1
n,4	long	Reference point known 0 = No 1 = Yes	1
n,5	long	Measuring unit 0 = inch 1 = metric 2 = degree 4= location	1
n,6	long	Diameter programming active 0 = No 1 = Yes	0
n,7	double	Offset value	

NC.Chan.CompCrossData

Description Provides the active cross compensation values (WCS) for all axes of the channel.

Remark Additional information is provided in the first seven columns for each axis. The eighth column always contains the compensation value of the axis.

Item syntax NC.Chan.CompCrossData,(1)

1. Channel number - Integer format, valid channel number

Call Synchronous/asynchronous

Access Read

Cyclic (opt.) Yes

Data format NC.Chan.CompCrossData, read

row, col	Type	Description	Example
n,0	long	Phys. axis or channel number	2
n,1	long	Channel assignment	1

row, col	Type	Description	Example
n,2	string	Logical axis name or coordinate name	Y
n,3	long	Axis in position 0 = No 1 = Yes	1
n,4	long	Reference point known 0 = No 1 = Yes	1
n,5	long	Measuring unit 0 = inch 1 = metric 2 = degree 4= location	1
n,6	long	Diameter programming active 0 = No 1 = Yes	0
n,7	double	Active cross compensation values	0.123

NC.Chan.CompLeadScrewErrData

Description Provides the active screw (spindle) lead error compensation for all axes of the channel.

Remark Additional information is provided in the first seven columns for each axis. The eighth column always contains the compensation value of the axis.

Item syntax NC.Chan.CompLeadScrewErrData,(1)

1. Channel number - Integer format, valid channel number

Call Synchronous/asynchronous

Access Read

Cyclic (opt.) Yes

Data format NC.Chan.CompLeadScrewErrData, read

row, col	Type	Description	Example
n,0	long	Phys. axis or channel number	2
n,1	long	Channel assignment	1
n,2	string	Logical axis name or coordinate name	Y
n,3	long	Axis in position 0 = No 1 = Yes	1
n,4	long	Reference point known 0 = No 1 = Yes	1

row, col	Type	Description	Example
n,5	long	Measuring unit 0 = inch 1 = metric 2 = degree 4= location	1
n,6	long	Diameter programming active 0 = No 1 = Yes	0
n,7	double	Active screw lead error compensation	4.123

NC.Chan.CompOffsetData

Description Provides the sum value of all axis compensations for all axes of the channel.

Remark Additional information is provided in the first seven columns for each axis. The eighth column always contains the compensation value of the axis.

Item syntax **NC.Chan.CompOffsetData,(1)**

1. Channel number - Integer format, valid channel number

Call Synchronous/asynchronous

Access Read

Cyclic (opt.) Yes

Data format **NC.Chan.CompOffsetData, read**

row, col	Type	Description	Example
n,0	long	Phys. axis or channel number	2
n,1	long	Channel assignment	1
n,2	string	Logical axis name or coordinate name	Y
n,3	long	Axis in position 0 = No 1 = Yes	1
n,4	long	Reference point known 0 = No 1 = Yes	1
n,5	long	Measuring unit 0 = inch 1 = metric 2 = degree 4= location	1
n,6	long	Diameter programming active 0 = No 1 = Yes	0
n,7	double	Sum value of all axis compensations	24.089

NC.Chan.CompTempData

Description Provides the active temperature-dependent compensation values (WCS) for all axes of the channel.

Remark Additional information is provided in the first seven columns for each axis. The eighth column always contains the compensation value of the axis.

Item syntax NC.Chan.CompTempData,(1)

1. Channel number - Integer format, valid channel number

Call Synchronous/asynchronous

Access Read

Cyclic (opt.) Yes

Data format NC.Chan.CompTempData, read

row, col	Type	Description	Example
n,0	long	Phys. axis or channel number	2
n,1	long	Channel assignment	1
n,2	string	Logical axis name or coordinate name	Y
n,3	long	Axis in position 0 = No 1 = Yes	1
n,4	long	Reference point known 0 = No 1 = Yes	1
n,5	long	Measuring unit 0 = inch 1 = metric 2 = degree 4= location	1
n,6	long	Diameter programming active 0 = No 1 = Yes	0
n,7	double	Active temperature-dependent compensation value	4.589

4.8.4 Adjustment Functions

NC.Chan.AuxActual

Description Provides the list of the current modal auxiliary functions.

Item syntax NC.Chan.AuxActual,(1)

1. channel number

Call Synchronous/asynchronous

Access Read

Cyclic (opt.) Yes

Data format Each auxiliary function is output as string in an individual line.

NC.Chan.AuxActual, read

row, col	Type	Description	Example
0,0	string	Current modal auxiliary function 1	M05
1,0	string	Current modal auxiliary function 2	M08
n,0	string	Current modal auxiliary function n	M48

NC.Chan.AuxAdjustment

Description Provides the adjustment information of modal auxiliary functions after block pre-run.

Remark One byte is provided for each modal function group. The value 0 stands for an adjusted group. 1 stands for a group that is not adjusted.

Item syntax NC.Chan.AuxAdjustment,(1)

1. channel number

Call Synchronous/asynchronous

Access Read

Cyclic (opt.) Yes

Data format NC.Chan.AuxAdjustment, read

row, col	Type	Description	Example
0,0	long	Adjustment information	1

NC.Chan.AuxReference

Description Provides the list of command states of modal auxiliary functions after block pre-run.

Item syntax NC.Chan.AuxReference,(1)

1. channel number

Call Synchronous/asynchronous

Access Read

Cyclic (opt.) Yes

Data format Each auxiliary function is output as string in an individual line.

NC.Chan.AuxReference, read

row, col	Type	Description	Example
0,0	string	Current modal auxiliary function 1	M05
1,0	string	Current modal auxiliary function 2	M14
n,0	string	Current modal auxiliary function n	M40

NC.Chan.FctActual

Description Provides the list of the current modal NC functions.

Item syntax **NC.Chan.FctActual,(1)**

1. channel number

Call Synchronous/asynchronous

Access Read

Cyclic (opt.) Yes

Data format Each NC function is output as string in an individual line.

NC.Chan.FctActual, read

row, col	Type	Description	Example
0,0	string	Current NC function	G80
1,0	string	Current NC function	G01
2,0	string	Current NC function	G17
3,0	string	Current NC function	G91
4,0	string	Current NC function	G94
n,0	string	Current NC function	OVE

NC.Chan.FctActualSorted

Description Provides the list of the selected current modal NC functions.

Item syntax **NC.Chan.FctActualSorted,(1)**

1. channel number

Call Synchronous/asynchronous

Access Read

Cyclic (opt.) Yes

Data format Each NC function is output as string in an individual line.

NC.Chan.FctActualSorted, read

row, col	Type	Description	Example
0,0	string	Current NC function	G80
1,0	string	Current NC function	G17
2,0	string	Current NC function	G91
n,0	string	Current NC function	G8

NC.Chan.FctAdjustment

Description Provides the adjustment information on all modal NC functions after block pre-run.

Remark One byte is provided for each modal function group. The value "0" stands for an adjusted group. "1" stands for a group that is not adjusted.

Item syntax **NC.Chan.FctAdjustment,(1)**

1. channel number

Call Synchronous/asynchronous

Access Read

Cyclic (opt.) Yes

Data format **NC.Chan.FctAdjustment, read**

row, col	Type	Description	Example
0,0	long	Current value	0
1,0	long	Current value	1
2,0	long	Current value	1
n,0	long	Current value	0

NC.Chan.FctAdjustmentSorted

Description Provides the adjustment information on all selected modal NC functions after block pre-run.

Remark One byte is provided for each modal function group. The value "0" stands for an adjusted group. "1" stands for a group that is not adjusted.

Item syntax **NC.Chan.FctAdjustmentSorted,(1)**

1. channel number

Call Synchronous/asynchronous

Access Read

Cyclic (opt.) Yes

Data format **NC.Chan.FctAdjustmentSorted, read**

row, col	Type	Description	Example
0,0	long	Current value	1
1,0	long	Current value	0
2,0	long	Current value	1
n,0	long	Current value	0

NC.Chan.FctReference

Description Provides the list of command states of modal NC functions after block pre-run.

Item syntax **NC.Chan.FctReference,(1)**

1. channel number

Call Synchronous/asynchronous

Access Read

Cyclic (opt.) Yes

Data format Each NC function is output as string in an individual line.

NC.Chan.FctReference, read

row, col	Type	Description	Example
0,0	string	Current NC function	G80
1,0	string	Current NC function	G00
2,0	string	Current NC function	G17
3,0	string	Current NC function	G90

row, col	Type	Description	Example
4,0	string	Current NC function	G94
n,0	string	Current NC function	OVE

NC.Chan.FctReferenceSorted

Description Provides the list of command states of selected modal NC functions after block pre-run.

Item syntax **NC.Chan.FctReferenceSorted,(1)**

1. channel number

Call Synchronous/asynchronous

Access Read

Cyclic (opt.) Yes

Data format Each NC function is output as string in an individual line.

NC.Chan.FctReferenceSorted, read

row, col	Type	Description	Example
0,0	string	Current NC function	G80
1,0	string	Current NC function	G00
2,0	string	Current NC function	G17
3,0	string	Current NC function	G90
n,0	string	Current NC function	OVE

4.8.5 Spindle Data

NC.Chan.ActSpindleAutoGears

Description Indicates for which spindle an automatic gear selection is active in the selected channel.

Remark *Gear selection state:*

- 0 = Manual gear selection
- 1 = Automatic gear selection active

The spindles indicated in the channel are configured via MACODA parameter 6005 0030. Specifically for the channel, this parameter indicates which spindle is shown at which position in the spindle display.

Item syntax **NC.Chan.ActSpindleAutoGears,(1)**

1. channel number

Call Synchronous/asynchronous

Access Read

As additional information, system spindle number, spindle name and axis number are provided.

Cyclic (opt.) Yes

Data format Number of lines: Depends on the spindles displayed, 8 max.

NC.Chan.ActSpindleAutoGears, read

row, col	Type	Description	Example
n,0	long	Gear selection state	1
n,1	long	System number	2
n,2	string	Spindle name	SSP02
n,3	long	Axis number	6

NC.Chan.ActSpindleGears

Description Provides the active gear step of a spindle.

Remark The spindles indicated in the channel are configured via MACODA parameter 6005 0030. Specifically for the channel, this parameter indicates which spindle is shown at which position in the spindle display.

To decode the gear steps, the enumerator "LDX_McoSpindleGearRange" can be used. It is defined as follows:

Enumerator LDX_McoSpindleGearRange

LDX_McoSpindleGearRange	Description
LDX_McoSpindleGearRange1_C	Gear stage 1
LDX_McoSpindleGearRange2_C	Gear stage 2
LDX_McoSpindleGearRange3_C	Gear stage 3
LDX_McoSpindleGearRange4_C	Gear stage 4
LDX_McoSpindleGearIdleMode_C	Gear step disengaged

Item syntax NC.Chan.ActSpindleGears,(1)

1. channel number

Call Synchronous/asynchronous

Access Read

As additional information, system spindle number, spindle name and axis number are provided.

Cyclic (opt.) Yes

Data format Number of lines: Depends on the spindles displayed, 8 max.

NC.Chan.ActSpindleGears, read

row, col	Type	Description	Example
n,0	long	Active spindle gear step	LDX_McoSpindleGearRange1_C
n,1	long	System number	2
n,2	string	Spindle name	SSP02
n,3	long	Axis number	6

NC.Chan.ActSpindleMovements

Description Returns the motion functions of the spindles.

Remark The spindles indicated in the channel are configured via MACODA parameter 6005 0030. Specifically for the channel, this parameter indicates which spindle is shown at which position in the spindle display.

To code the gear steps, use the "LDX_McoSpindleMovements" enumerator which can assume the following values:

Enumerator LDX_McoSpindleMovements

LDX_McoSpindleMovements	Description
LDX_McoSpindleNotDefined_C	Spindle not defined
LDX_McoSpindleRight_C	Clockwise rotation without cooling agent
LDX_McoSpindleRightCool_C	Clockwise rotation with cooling agent
LDX_McoSpindleLeft_C	Counterclockwise rotation without cooling agent
LDX_McoSpindleLeftCool_C	Counterclockwise rotation with cooling agent
LDX_McoSpindleStop_C	Spindle stop
LDX_McoSpindleOrientate_C	Orient spindle

Item syntax NC.Chan.ActSpindleMovements,(1)

1. channel number

Call Synchronous/asynchronous

Access Read

As additional information, system spindle number, spindle name and axis number are provided.

Cyclic (opt.) Yes

Data format Number of lines: Depends on the spindles displayed, 8 max.

NC.Chan.ActSpindleMovements, read

row, col	Type	Description	Example
n,0	long	Spindle motion function	LDX_McoSpindleStop_C
n,1	long	System number	2
n,2	string	Spindle name	SSP02
n,3	long	Axis number	6

NC.Chan.ActSpindleOverride

Description Provides the current value of the spindle override.

Item syntax NC.Chan.ActSpindleOverride,(1)

1. channel number

Call Synchronous/asynchronous

Access Read

The result includes the override value in percent.

Moreover, the max. override value to be set, the system number, the spindle name and the axis number are provided.

Cyclic (opt.) Yes

Data format Number of lines: Depends on the spindles displayed, 8 max.

NC.Chan.ActSpindleOverride, read

row, col	Type	Description	Example
n,0	double	Active value in %	40.0
n,1	double	Maximum value	50.0
n,2	long	System number	1
n,3	string	Spindle name	S1 (SSP1)
n,4	long	Axis number	6

NC.Chan.ActSpindlePosition

Description Provides the active spindle position in degree.

Item syntax NC.Chan.ActSpindlePosition,(1)

1. channel number

Call Synchronous/asynchronous

Access Read

As additional information, system spindle number, spindle name and axis number are provided.

Cyclic (opt.) Yes

Data format Number of lines: Depends on the spindles displayed, 8 max.

NC.Chan.ActSpindlePosition, read

row, col	Type	Description	Example
n,0	double	Active actual value of the spindle	246.3700
n,1	long	System number	1
n,2	string	Spindle name	S1
n,3	long	Axis number	6

NC.Chan.ActSpindlePower

Description Provides the active and maximum spindle power in watts.

Item syntax NC.Chan.ActSpindlePower,(1)

1. channel number

Call Synchronous/asynchronous

Access Read

As additional information, system spindle number, spindle name and axis number are provided.

Cyclic (opt.) Yes

Data format Number of lines: Depends on the spindles displayed, 8 max.

NC.Chan.ActSpindlePower, read

row, col	Type	Description	Example
n,0	long	Active value in watts	80
n,1	long	Maximum value in watts	120
n,2	long	System number	1

row, col	Type	Description	Example
n,3	string	Spindle name	S1
n,4	long	Axis number	6

NC.Chan.ActSpindleSpeed

Description Provides the active command spindle speed(s).

Item syntax **NC.Chan.ActSpindleSpeed,(1)**

1. channel number

Call Synchronous/asynchronous

Access Read

As additional information, system spindle number, spindle name and axis number are provided. If "G196" is active, the cutting velocities are provided in m/min or feet/min.

Cyclic (opt.) Yes

Data format Number of lines: Depends on the spindles displayed, 8 max.

NC.Chan.ActSpindleSpeed, read

row, col	Type	Description	Example
n,0	double	Command spindle speed	450.0
n,1	long	System number	1
n,2	string	Spindle name	S1 (SSP1)
n,3	long	Axis number	6

NC.Chan.ActSpindleSpeedDrv

Description provides the active actual spindle speed(s).

Item syntax **NC.Chan.ActSpindleSpeedDrv,(1)**

1. channel number

Call Synchronous/asynchronous

Access Read

As additional information, system spindle number, spindle name and axis number are provided. If "G196" is active, the cutting velocities are provided in m/min or feet/min.

Cyclic (opt.) Yes

Data format Number of lines: Depends on the spindles displayed, 8 max.

NC.Chan.ActSpindleSpeedDrv, read

row, col	Type	Description	Example
n,0	double	Spindle list speed	2000.0
n,1	long	System number	1
n,2	string	Spindle name	S1 (SSP1)
n,3	long	Axis number	6

NC.Chan.ActSpindleSwitchGears

Description Indicates whether gears are currently shifted.

Remark *Gear step shifting state:*

- 0 = No gear shifting
- 1 = Gear shifted

The spindles indicated in the channel are configured via MACODA parameter 6005 0030. Specifically for the channel, this parameter indicates which spindle is shown at which position in the spindle display.

Item syntax NC.Chan.ActSpindleSwitchGears,(1)

1. channel number

Call Synchronous/asynchronous

Access Read

As additional information, system spindle number, spindle name and axis number are provided.

Cyclic (opt.) Yes

Data format Number of lines: Depends on the spindles displayed, 8 max.

NC.Chan.ActSpindleSwitchGears, read

row, col	Type	Description	Example
n,0	long	Gear shifting	0
n,1	long	System number	2
n,2	string	Spindle name	SSP02
n,3	long	Axis number	6

NC.Chan.ActSpindleTorque

Description Provides the active and maximum spindle torque in percent.

Item syntax NC.Chan.ActSpindleTorque,(1)

1. channel number

Call Synchronous/asynchronous

Access Read

As additional information, system spindle number, spindle name and axis number are provided.

Cyclic (opt.) Yes

Data format Number of lines: Depends on the spindles displayed, 8 max.

NC.Chan.ActSpindleTorque, read

row, col	Type	Description	Example
n,0	double	Active value in %	5.5
n,1	double	Maximum value in %	8.3
n,2	long	System number	1
n,3	string	Spindle name	S1
n,4	long	Axis number	6

NC.Chan.ProgrammedSpindleSpeed

Description Provides the programmed spindle speed(s).

Remark The spindles indicated in the channel are configured via MACODA parameter 6005 0030. Specifically for the channel, this parameter indicates which spindle is shown at which position in the spindle display.

Item syntax NC.Chan.ProgrammedSpindleSpeed,(1)

1. channel number

Call Synchronous/asynchronous

Access Read

As additional information, system spindle number, spindle name and axis number are provided.

Cyclic (opt.) Yes

Data format Number of lines: Depends on the spindles displayed, 8 max.

NC.Chan.ProgrammedSpindleSpeed, read

row, col	Type	Description	Example
n,0	double	Programmed spindle speed	2000.0
n,1	long	System number	1
n,2	string	Spindle name	S1 (SSP1)
n,3	long	Axis number	6

4.8.6 Process Data

NC.Chan.ActCallChain

Description Provides the current subroutine nesting.

Remark In each case, the program name, including the path and the block number, are provided.

The subroutine nesting is provided relatively to the active block.

Currently, the subroutine nesting relatively to the prepared block is not available.

Item syntax NC.Chan.ActCallChain,(1)

1. channel number

Call Synchronous/asynchronous

Access Read

Cyclic (opt.) Yes

Data format Number of lines: Depends on the number of nestings

NC.Chan.ActCallChain, read

row, col	Type	Description	Example
n,0	string	Program name with path	/mnt/Testprogram.cnc
n,1	string	Block number	N235

NC.Chan.ActMainProgram

Description Provides the name and the path of the main program.

Item syntax NC.Chan.ActMainProgram,(1)

1. channel number

Call Synchronous/asynchronous

Access Read

Cyclic (opt.) Yes

Data format NC.Chan.ActMainProgram, read

row, col	Type	Description	Example
0,0	string	Name and path of the main program	/usr/user/sekt.cnc

NC.Chan.ActNcBlock

Description Provides the name of the current block.

Remark If no block number is programmed, an empty string is returned.

Item syntax NC.Chan.ActNcBlock,(1)

1. channel number

Call Synchronous/asynchronous

Access Read

Cyclic (opt.) Yes

Data format NC.Chan.ActNcBlock, read

row, col	Type	Description	Example
0,0	string	Name of the current block	N150

NC.Chan.ActSubProgram

Description Provides the name of the current subroutine.

Remark The result comprises the name and the path of the subroutine.

With read-protected or encoded subroutines and cycles, the name of the calling program is provided.

Item syntax NC.Chan.ActSubProgram,(1)

1. channel number

Call Synchronous/asynchronous

Access Read

Cyclic (opt.) Yes

Data format NC.Chan.ActSubProgram, read

row, col	Type	Description	Example
0,0	string	Name and path of the subroutine	/usr/user/sub01.cnc

NC.Chan.AxisMirror

Description This item can be used to determine the current state of the mirroring. If the mirroring is activated for an axis within the channel, the item returns the value 1 for this axis. If the mirroring is switched off, 0 is returned. Apart from the mirror states, item 7 provides further additional values shown in the columns 0 - 6.

Remark Additional information is provided in the first seven columns for each axis. The eighth column contains the current state value "Mirroring active" of the axis.

Item syntax NC.Chan.AxisMirror,(1)

1. channel number

Call Synchronous/asynchronous

Access Read

Cyclic (opt.) Yes

Data format NC.Chan.AxisMirror, read

row, col	Type	Description	Example
0,0	long	Phys. axis or channel number	2
0.1	long	Channel assignment	1
0.2	string	Logical axis name or coordinate name	Y
0,3	long	Axis in position 0 = No 1 = Yes	1
0,4	long	Reference point known 0 = No 1 = Yes	1
0,5	long	Measuring unit 0 = inch 1 = metric 2 = degree 4= location	1
0,6	long	Diameter programming active 0 = No 1 = Yes	0
0,7	double	Mirror state value 0 = switched off 1 = switched on	1

NC.Chan.AxisPoleSet

Description This function specifies the absolute position of the pole for the input tools mirroring, scaling and rotating. Apart from the rotation point of the mirror, item 7 provides further additional values shown in the columns 0 - 6.

Remark Additional information is provided in the first seven columns for each axis. Each eighth columns contains the absolute position of the polar of an axis.

Item syntax NC.Chan.AxisPoleSet,(1)

1. channel number

Call Synchronous/asynchronous

Access Read

Cyclic (opt.) Yes

Data format NC.Chan.AxisPoleSet, read

row, col	Type	Description	Example
n,0	long	Phys. axis or channel number	2
n,1	long	Channel assignment	1
n,2	string	Logical axis name or coordinate name	X
n,3	long	Axis in position 0 = No 1 = Yes	1
n,4	long	Reference point known 0 = No 1 = Yes	1
n,5	long	Measuring unit 0 = inch 1 = metric 2 = degree 4 = Place	1
n,6	long	Diameter programming active 0 = No 1 = Yes	0
n,7	double	Absolute position value of the pole	12

NC.Chan.AxisRotate

Description The item provides the angle a programmed contour rotated in the active plane.

Remark

Item syntax NC.Chan.AxisRotate,(1)

1. Channel number - Integer format, valid channel number

Call Synchronous/asynchronous

Access Read

Cyclic (opt.) Yes

Data format NC.Chan.AxisRotate, read

row, col	Type	Description	Example
0,0	double	Angle value	5.0

NC.Chan.AxisScale

Description This item can be used to read scaling values specified for a programmed contour. Apart from the scaling values, item 7 provides further additional values shown in the columns 0 - 6.

Remark Additional information is provided in the first seven columns for each axis. The eighth column always contains the scaling value of an axis.

Item syntax NC.Chan.AxisScale,(1)

1. Channel number - Integer format, valid channel number

Call Synchronous/asynchronous

Access Read

Cyclic (opt.) Yes

Data format NC.Chan.AxisScale, read

row, col	Type	Description	Example
n,0	long	Phys. axis or channel number	2
n,1	long	Channel assignment	1
n,2	string	Logical axis name or coordinate name	X
n,3	long	Axis in position 0 = No 1 = Yes	1
n,4	long	Reference point known 0 = No 1 = Yes	1
n,5	long	Measuring unit 0 = inch 1 = metric 2 = degree 4 = Place	1
n,6	long	Diameter programming active 0 = No 1 = Yes	0
n,7	double	Scaling value	1.2

NC.Chan.EqualizeState

Description The item provides the status of the block pre-run adjustment. After the initialization, the event mechanism is activated in the kernel. This mechanism is addressed as soon as a state change of the block pre-run adjustment takes place.

Remark Valid data is only displayed after the control sent the first event. The provided data is invalid before the first event and the quality flag shows "Uncertain".

The first data line contains the version number of the event. Each event increases the number by 1.

The second data line contains a bit mask in which the corresponding adjustment states are marked. Bit 1 shows the status of the block pre-run adjustment. If this bit is set, an adjustment is carried out.

Item syntax NC.Chan.EqualizeState,(1)

1. Channel number - Integer format, valid channel number

Call Synchronous/asynchronous

Access Read

Cyclic (opt.) Yes

Data format NC.Chan.EqualizeState, read

row, col	Type	Description	Example
0,0	ulong	Version number of the event	5
1,0	ulong	Bit mask of modal block status Bit 0 = indicates whether the block in the interpolator was created by a block pre-run Bit 1 = indicates whether an adjustment is executed	3

NC.Chan.MessageWindow

Description Provides the currently programmed notes.

Remark Channel-specific and non-channel-specific notes as well as CPL outputs are considered.

The window size, i.e. the number of provided notes can be specified. However, it is limited by NC-internal resources (no more than 5 channel-specific and 5 non-channel-specific notes).

☞ **Restriction:** Up to now, the window size parameter has no effect!

Item syntax NC.Chan.MessageWindow,(1),(2)

1. channel number
2. Window size

Call Synchronous/asynchronous

Access Read

The following is provided:

- non-channel-specific notes
- channel-specific notes (MSG ...)
- CPL outputs PRINT#(0, ...

Cyclic (opt.) Yes

Data format Number of lines: Depends on the number of notes

NC.Chan.MessageWindow, read

row, col	Type	Description	Example: NC.Chan.MessageWindow,1,20
n,0	string	Note	Thread is drilled

NC.Chan.ModalAuxFunctions

Description Provides a list of the modal auxiliary functions. The sequence and number of the provided functions can be configured via MACODA (6005 00050).

Remark The list may have blank strings if no auxiliary function is active in a modal group.

With faulty inputs in the configuration (MACODA parameter obtains an invalid value), entries are marked as invalid. The list entry has the value "?".

Item syntax NC.Chan.ModalAuxFunctions,(1)

1. channel number

Call Synchronous/asynchronous

Access Read

Cyclic (opt.) Yes

Data format Number of lines: Depends on MACODA

NC.Chan.ModalAuxFunctions, read

row, col	Type	Description	Example
n,0	string	Active NC function	M10

NC.Chan.ModalAxisOffsetFunctions

Description Provides a list of the modal NC functions for axis zero point offset. The sequence and number of the provided modal functions can be configured via MACODA (6005 00070).

Remark The list may have blank strings if no NC function is active in a modal group.

With faulty inputs in the configuration (MACODA parameter obtains an invalid value), entries are marked as invalid. The list entry has the value "?".

Item syntax **NC.Chan.ModalAxisOffsetFunctions,(1)**

1. channel number

Call Synchronous/asynchronous

Access Read

Cyclic (opt.) Yes

Data format Number of lines: Depends on MACODA

NC.Chan.ModalAxisOffsetFunctions, read

row, col	Type	Description	Example
n,0	string	Active NC function	G55.3

NC.Chan.ModalFunction

Description Reads out a single modal group.

The modal group to be read out is specified by any modal function which may be effective in this group.

Item syntax **NC.Chan.ModalFunction,(1),(2)**

1. channel number
2. Modal function

Call Asynchronous

Access Read

Cyclic (opt.) no

Data format **NC.Chan.ModalFunction, read**

row, col	Type	Description	NC.Chan.ModalFunction,2,G70
0,0	string	Result of a modal group	G71

NC.Chan.ModalNcFunctions

Description Provides a list of the modal NC functions. The sequence and number of the provided modal functions can be configured via MACODA (6005 00040).

Remark The list may have blank strings if no NC function is active in a modal group.

With faulty inputs in the configuration (MACODA parameter obtains an invalid value), entries are marked as invalid. The list entry has the value "?".

Item syntax **NC.Chan.ModalNcFunctions,(1)**

1. channel number

Call Synchronous/asynchronous

Access Read

Cyclic (opt.) Yes

Data format Number of lines: Depends on MACODA

NC.Chan.ModalNcFunctions, read

row, col	Type	Description	Example
n,0	string	Active NC function	G18

NC.Chan.ModalTransformationFunctions

Description Provides a list of the modal NC functions for workpiece transformations. The sequence and number of the provided modal functions can be configured via MACODA (6005 00060).

Remark The list may have blank strings if no NC function is active in a modal group.

With faulty inputs in the configuration (MACODA parameter obtains an invalid value), entries are marked as invalid. The list entry has the value "?".

Item syntax **NC.Chan.ModalTransformationFunctions,(1)**

1. channel number

Call Synchronous/asynchronous

Access Read

Cyclic (opt.) Yes

Data format Number of lines: Depends on MACODA

NC.Chan.ModalTransformationFunctions, read

row, col	Type	Description	Example
n,0	string	Active NC function	G154.3

NC.Chan.ModeOfOperation

Description Provides the operation mode and the secondary operation mode of a channel.

Remark Use the enumerators "LDX_SavOpSubModes" and "LDX_SavOpModes" to code the named operation modes. Theses can assume the following values:

Enumerator LDX_SavOpModes

LDX_SavOpModes	Description
LDX_SAV_OP_MODE_EMPTY_C	No operation mode active
LDX_SAV_OP_MODE_JOGGING_C	Jog mode
LDX_SAV_OP_MODE_REFERENCE_C	Traveling to reference point
LDX_SAV_OP_MODE_SEARCHBLOCK_C	Block pre-run

LDX_SavOpModes	Description
LDX_SAV_OP_MODE_MDI_C	Manual data input
LDX_SAV_OP_MODE_AUTOMATIC_C	Automatic: Part programs can be processed in the continuous block
LDX_SAV_OP_MODE_SINGLE_BLOCK_C	Program block: With "NC start", individual blocks are processed as in the part program. Block preparation only starts upon "NC start".
LDX_SAV_OP_MODE_SINGLE_STEP_C	Single step: With "NC start", an individual block is passed on to the interpolator for processing.
LDX_SAV_OP_MODE_TEACH_C	Reserved
LDX_SAV_OP_MODE_SUSPEND_C	Reserved
LDX_SAV_OP_MODE_SINGLE_ACT_BLOCK_C	Single block: With "NC start", all the blocks created and completely prepared due to an individual NC block in the part program are passed on to the interpolator for processing.
LDX_SAV_OP_MODE_RETURN_TO_PATH_C	Return to contour: Axes can leave the contour manually and return to the contour automatically or manually.
LDX_SAV_OP_MODE_DEBUG_STEP_C	CPL debugger: Individual blocks are processed as they are in the part program.
LDX_SAV_OP_MODE_DEBUG_CONT_C	CPL debugger: All the blocks are processed up to the next break point.
LDX_SAV_OP_MODE_JOGGING_COORDS_C	Set manually: Motion in workpiece coordinates

Enumerator LDX_SavOpSubModes

LDX_SavOpSubModes	Description
LDX_SAV_OP_SUBMODE_EMPTY_C	No secondary mode active
LDX_SAV_OP_SUBMODE_OVERSTORE_C	Overwriting active. That means that traversing blocks can be transferred in the Automatic modes.

Item syntax NC.Chan.ModeOfOperation,(1)

1. channel number

Call Synchronous/asynchronous

Access Read

Cyclic (opt.) Yes

Data format NC.Chan.ModeOfOperation, read

row, col	Type	Description	Example: NC.Chan.ModeOfOperation,1
0,0	long	Active operation mode in channel 1	LDX_SAV_OP_MODE_AUTOMATIC_C
0.1	long	Active secondary operation mode in channel 1	LDX_SAV_OP_SUBMODE_OVERSTORE_C

NC.Chan.OptionalStop

Description Provides the current status of "Optional stop" as it is present at the NC interface input and the NC interface output.

Item syntax NC.Chan.OptionalStop,(1)

1. channel number

Call Synchronous/asynchronous

Access Read

Cyclic (opt.) Yes

Data format NC.Chan.OptionalStop, read

row, col	Type	Description	Example: NC.Chan.OptionalStop,1
0,0	long	Status of "Optional stop" at the input	0
0.1	long	Status of "Optional stop" at the output	0

NC.Chan.ProgramPosition

Description Provides the current file position in the respective main program or subroutine.

Remark If an error occurred in the channel, the file position, at which the error occurred, is provided.

☞ This item is not suitable for the cyclic query within a group as during each read access, there is a communication with the NC.

Item syntax NC.Chan.ProgramPosition,(1)

1. channel number

Call Synchronous/asynchronous

Access Read

The file name and the complete path of the program as well as the file position are provided.

Cyclic (opt.) no

Data format NC.Chan.ProgramPosition, read

row, col	Type	Description	Example
0,0	string	File name incl. path	/usr/user/sekt.cnc
0.1	long	File position	34

NC.Chan.ProgramWindow

Description Provides a section from the current main program or the subroutine.

Remark The window size of the program section as well as the number of NC blocks in front of the active block can be parameterized. With read-protected or encoded subroutines and cycles, the calling program provides the section.

☞ **Restriction:** The information on the block prepared last is currently not provided. A corresponding field is provided in the response data but does not yet contain a valid value.

Item syntax NC.Chan.ProgramWindow,(1),(2),(3)

1. channel number

2, Window size (number of blocks in the result, 200 max.)

3. Desired number of preceding blocks (40 max.)

Call Synchronous/asynchronous

Access Read

Program section (per line one BSTR)

Moreover, the following is provided:

- the desired number of preceding blocks
- The actual number of preceding blocks
- The reference of the block prepared last

Cyclic (opt.) Yes

Data format The line number depends on the number of blocks in the result. A maximum of 200 lines after line 0

NC.Chan.ProgramWindow, read

row, col	Type	Description	Example
0,0	long	The desired number of preceding blocks	10
0.1	long	The actual number of preceding blocks	0
0.2	long	The reference of the block prepared last	30
n,0	string	Line n of the program section	N030 X30 Y30 Z30

NC.Chan.ReentryChain

Description The item can read or reset the "call chain" saved in a channel. If the item returns a null string while reading, there is no "call chain" in the selected channel.

Remark

Item syntax NC.Chan.ReentryChain,(1)

1. Channel number - Integer format, valid channel number

Call Synchronous/asynchronous

Access Read/Write

Cyclic (opt.) no

Data format **NC.Chan.ReentryChain, read**

row, col	Type	Description	Example
0,0	string	Saved call chain	

NC.Chan.ReentryDelete

Description After calling the item, the internally saved "call chain" in the control is deleted for the specified channel.

Remark

Item syntax **NC.Chan.ReentryDelete,(1)**

1. Channel number - Integer format, valid channel number

Call Synchronous/asynchronous

Access Call

Cyclic (opt.) no

NC.Chan.ReentryEvent

Description This item reports an event in the NC occurring each time the command sequence RCB(1) or RCB(0) is reached within the running NC program. When detecting the event, the item provides the event version number in line 1. This number is a consecutive value which is incremented by 1 for each event in the NC kernel. Line 2 reports the command sequence triggered by the event. With these commands, the engineer marks the locations in the NC program where an uncomplicated return to the program is possible after an abort.

Remark The RCB(1) command marks the beginning and the RCB(0) command marks the end of the return area.

Item syntax **NC.Chan.ReentryEvent,(1)**

1. Channel number - Integer format, valid channel number

Call Synchronous/asynchronous

Access Read

Cyclic (opt.) Yes, required!

Data format **NC.Chan.ReentryEvent, read**

row, col	Type	Description	Example
0,0	long	Version ID	4
1,0	bool	Command sequence RCB(0) = FALSE RCB(1) = TRUE	TRUE

NC.Chan.ReturnToContour

Description Provides the current signal state of "Return to contour" as it is present at the NC interface input and at the NC interface output.

Remark *The signal status reports the active variant of the returning to contour:*

- 0 = Direct returning to contour
- 1 = Returning to contour via path

Item syntax **NC.Chan.ReturnToContour,(1)**

1. channel number

Call Synchronous/asynchronous

Access Read

Cyclic (opt.) Yes

Data format NC.Chan.ReturnToContour, read

row, col	Type	Description	Example: NC.Chan.ReturnToContour,1
0,0	long	"Return to contour" status at the NC input	0
0.1	long	"Return to contour" status at the NC output	0

NC.Chan.ReturnToPathStrategy

Description *It provides information on the current return to path strategy:*

- Return to contour operation mode
- Point of return
- State whether leaving conditions are logged

Remark The return to contour mode is coded by means of the "LDX_McoRetPath_Mode" enumerator which can assume the following values:

Enumerator LDX_McoRetPath_Mode

LDX_McoRetPath_Mode	Description
LDX_McoRetPathAutomatic_C	Automatic return to contour
LDX_McoRetPathSingleBlock_C	Return to contour with single block
LDX_McoRetPathManual_C	Manual return to contour

The return to contour point is coded by means of the "LDX_McoRetPath_Point" enumerator which can assume the following values:

Enumerator LDX_McoRetPath_Point

LDX_McoRetPath_Point	Description
LDX_McoRetPathStartOfBlock_C	Return to starting point
LDX_McoRetPathEndOfBlock_C	Return to end point
LDX_McoRetPathPointOfInterruption_C	Return to point of interruption

If leaving conditions are logged, the function provides "1" as third value, otherwise "0".

Item syntax NC.Chan.ReturnToPathStrategy,(1)

1. channel number

Call Synchronous/asynchronous

Access Read

Cyclic (opt.) Yes

Data format **NC.Chan.ReturnToPathStrategy, read**

row, col	Type	Description	Example
0,0	long	Return to contour operation mode	LDX_McoRetPathAutomatic_C
0.1	long	Point of return	LDX_McoRetPathStartOfBlock_C
0.2	long	Recording leaving conditions	0

NC.Chan.SkipBlockMode

Description Provides the state of the NC input signal "Skip block" and the NC output signal "Activate skip block".

Item syntax **NC.Chan.SkipBlockMode,(1)**

1. channel number

Call Synchronous/asynchronous

Access Read

Cyclic (opt.) Yes

Data format **NC.Chan.SkipBlockMode, read**

row, col	Type	Description	Example: NC.Chan.SkipBlockMode,1
0,0	long	NC input signal status	0
0.1	long	NC output signal status	0

NC.Chan.UpCallFromNcBlock

Description Determines subroutine calls in a specified NC block.

Item syntax **NC.Chan.UpCallFromNcBlock,(1),(2),(3)**

1. channel number
2. Path of the main program (without subsequent slash)
3. NC block

Call Synchronous/asynchronous

Access Read

If the block contains a subroutine call, the subroutine name with path and the string length are provided. If this is not the case, a blank string is provided.

Cyclic (opt.) no

Data format **NC.Chan.UpCallFromNcBlock, read**

row, col	Type	Description	Example: NC.Chan.UpCallFromNcBlock, 1,/usr/user,"10 CALL /usrfep/ subprogram.cnc"
0,0	string	Subroutine name with path	/usr/user/subprogram.cnc
0.1	long	String length	24

4.8.7 Feed Values and Speed Values

NC.Chan.ActAxisVelocity

Description Provides the current axis velocity.

Remark Additional information is provided in the first seven columns for each axis. The eighth column contains the velocity of the axis.

Item syntax NC.Chan.ActAxisVelocity,(1)

1. channel number

Call Synchronous/asynchronous

Access Read

Cyclic (opt.) Yes

Data format NC.Chan.ActAxisVelocity, read

row, col	Type	Description	Example
n,0	long	Phys. axis or channel number	2
n,1	long	Channel assignment	1
n,2	string	Logical axis name or coordinate name	Y
n,3	long	Axis in position 0 = FALSE 1 = TRUE	1
n,4	long	Reference point known	1
n,5	long	Measuring unit	LDX_McoUnitMetric_C
n,6	long	Diameter programming active	0
n,7	double	Axis velocity	1450.000

NC.Chan.ActAxisVelocityMax

Description Provides the maximum velocity in the axis coordinate system.

Remark Only values for the asynchronous axes and axes in the jog mode are provided. Otherwise, this value is 0.

Additional information is provided in the first seven columns for each axis. The eighth column contains the respective velocity value of the axis.

Item syntax NC.Chan.ActAxisVelocityMax,(1)

1. channel number

Call Synchronous/asynchronous

Access Read

Cyclic (opt.) Yes

Data format NC.Chan.ActAxisVelocityMax, read

row, col	Type	Description	Example
n,0	long	Phys. axis or channel number	2
n,1	long	Channel assignment	1
n,2	string	Logical axis name or coordinate name	Z
n,3	long	Axis in position 0 = FALSE 1 = TRUE	1
n,4	long	Reference point known	1

row, col	Type	Description	Example
n,5	long	Measuring unit	LDX_McoUnitMetric_C
n,6	long	Diameter programming active	0
n,7	double	Maximum axis velocity	980.000

NC.Chan.ActAxisVelocityPeak

Description Provides peak value, maximum value and nominal value of the current axis velocity.

Remark Additional information is provided in the first seven columns for each axis.

The columns eight to ten contain the peak value, maximum value and nominal value of the axis velocity. Divide the peak and the maximum value by the nominal value to determine a value in percent for the peak value and the maximum value.

☞ The control first provides the values for the velocity axis load diagnostics if these diagnostics are activated via system data entry (/SysNcDgnCtrl/AxesLoadDgn/VelLoadDgn/DgnMode = 2).

Switch off the velocity load diagnostics using the system data entry (.../VelLoadDgn/DgnMode = 0) if these diagnostics are not required anymore! This impedes unnecessary loads of the interpolator interrupt.

Item syntax NC.Chan.ActAxisVelocityPeak,(1)

1. channel number

Call Synchronous/asynchronous

Access Read

Cyclic (opt.) Yes

Data format NC.Chan.ActAxisVelocityPeak, read

row, col	Type	Description	Example
n,0	long	Phys. axis or channel number	2
n,1	long	Channel assignment	1
n,2	string	Logical axis name or coordinate name	Z
n,3	long	Axis in position 0 = FALSE 1 = TRUE	1
n,4	long	Reference point known	1
n,5	long	Measuring unit	LDX_McoUnitMetric_C
n,6	long	Diameter programming active	0
n,7	double	Peak value of the axis velocity	10500.000
n,8	double	Maximum value of the axis velocity	12000.000
n,9	double	Nominal value of the axis velocity	10000.000

NC.Chan.ActAxisAccelerationPeak

Description Provides peak value, maximum value and nominal value of the current axis velocity.

Remark Additional information is provided in the first seven columns for each axis.

The columns eight to ten contain the peak value, maximum value and nominal value of the axis acceleration. Divide the peak and the maximum value by the nominal value to determine a value in percent for the peak value and the maximum value.

☞ The control first provides the values for the acceleration axis load diagnostics if these diagnostics are activated via system data entry (/SysNcDgnCtrl/AxesLoadDgn/AccLoadDgn/DgnMode = 2).

Switch off the acceleration load diagnostics using the system data entry (.../AccLoadDgn/DgnMode = 0) if these diagnostics are not required anymore! This impedes unnecessary loads of the interpolator interrupt.

Item syntax NC.Chan.ActAxisAccelerationPeak,(1)

1. channel number

Call Synchronous/asynchronous

Access Read

Cyclic (opt.) Yes

Data format NC.Chan.ActAxisAccelerationPeak, read

row, col	Type	Description	Example
n,0	long	Phys. axis or channel number	2
n,1	long	Channel assignment	1
n,2	string	Logical axis name or coordinate name	Z
n,3	long	Axis in position 0 = FALSE 1 = TRUE	1
n,4	long	Reference point known	1
n,5	long	Measuring unit	LDX_McoUnitMetric_C
n,6	long	Diameter programming active	0
n,7	double	Peak value of the axis acceleration	1100.000
n,8	double	Maximum value of the axis acceleration	1500.000
n,9	double	Nominal value of the axis acceleration	1300.000

NC.Chan.ActFeedOverride

Description Provides the current value of the feed override in percent.

Remark In addition to the current value, the parameterized maximum value is provided as well.

Item syntax NC.Chan.ActFeedOverride,(1)

1. channel number

Call Synchronous/asynchronous

Access Read

Cyclic (opt.) Yes

Data format **NC.Chan.ActFeedOverride, read**

row, col	Type	Description	Example
0,0	double	Active value in %	85.0
0.1	double	Maximum value in %	120.0

NC.Chan.ActFeedrate

Description Provides the current feed value.

Remark *3 path feed values are provided. For each feed value, the respectively active override is considered:*

1. **External command interpolator velocity**:
   is assigned to the interpolator (programmed feed)
2. **Actual interpolator velocity**:
   current path velocity
3. **Internal command interpolator velocity**:
   can be changed - compared to path feed 1 - by an application such as the FeedAdapt function. During feed programming in mm/rev (G95), it provides the path feed in mm/min for example.

Item syntax **NC.Chan.ActFeedrate,(1)**

1. channel number

Call Synchronous/asynchronous

Access Read

Velocity is specified in mm/min.

Cyclic (opt.) Yes

Data format **NC.Chan.ActFeedrate, read**

row, col	Type	Description	Example
0,0	double	External command interpolator velocity	85
0.1	double	Actual interpolator velocity	85
0.2	double	Internal command interpolator velocity	85
1,0	long	Decimal places, metric	1
1,1	long	Decimal places, inch	2
2,0	long	mm/min	1
2,1	double	mm/rev	1.000

NC.Chan.ActRotAxisProgrammedSpeed

Description Provides the programmed speed of asynchronous, endlessly rotating rotary axes.

Remark Additional information is provided in the first seven columns for each axis. The eighth column contains the position value of the axis.

Item syntax NC.Chan.ActRotAxisProgrammedSpeed,(1)

1. channel number

Call Synchronous/asynchronous

Access Read

Cyclic (opt.) Yes

Data format NC.Chan.ActRotAxisProgrammedSpeed, read

row, col	Type	Description	Example
n,0	long	Phys. axis or channel number	2
n,1	long	Channel assignment	1
n,2	string	Logical axis name or coordinate name	Y
n,3	long	Axis in position 0 = FALSE 1 = TRUE	1
n,4	long	Reference point known	1
n,5	long	Measuring unit	LDX_McoUnitMetric_C
n,6	long	Diameter programming active	0
n,7	double	Speed value	

NC.Chan.ActRotAxisSpeed

Description Provides the current speed of asynchronous, endlessly rotating rotary axes.

Remark Additional information is provided in the first seven columns for each axis. The eighth column contains the position value of the axis.

Item syntax NC.Chan.ActRotAxisSpeed,(1)

1. channel number

Call Synchronous/asynchronous

Access Read

Cyclic (opt.) Yes

Data format NC.Chan.ActRotAxisSpeed, read

row, col	Type	Description	Example
n,0	long	Phys. axis or channel number	2
n,1	long	Channel assignment	1
n,2	string	Logical axis name or coordinate name	Y
n,3	long	Axis in position 0 = FALSE 1 = TRUE	1
n,4	long	Reference point known	1
n,5	long	Measuring unit	LDX_McoUnitMetric_C

row, col	Type	Description	Example
n,6	long	Diameter programming active	0
n,7	double	Speed value	

NC.Chan.ProgrammedFeedrate

Description Provides the programmed feed values F1 and F2.

Remark If the NC kernel software does not know the F2 value, it only returns the F1 value. In this case, the F2 value is "0".

Item syntax **NC.Chan.ProgrammedFeedrate,(1)**

1. channel number

Call Synchronous/asynchronous

Access Read

Cyclic (opt.) Yes

Data format **NC.Chan.ProgrammedFeedrate, read**

row, col	Type	Description	Example
0,0	double	programmed feed value F1	150
0.1	double	Programmed feed value F2	200
1,0	long	Decimal places, metric	0
1,1	long	Decimal places, inch	1
2,0	long	mm/min	1
2,1	double	mm/rev	1.000

NC.Chan.ProgrammedFeedrateIndex

Description Provides the information whether the programmed feed F1 or F2 is currently active.

Item syntax **NC.Chan.ProgrammedFeedrateIndex,(1)**

1. channel number

Call Synchronous/asynchronous

Access Read

1 = Feed F1 is active

2 = Feed F2 is active

Cyclic (opt.) Yes

Data format **NC.Chan.ProgrammedFeedrateIndex, read**

row, col	Type	Description	Example
0,0	long	Active feed	2

4.9 NC Commands

4.9.1 Overview

Item overview

- **NC.Cmd.DryRunMode**
  Setting of the test mode of channel, axis, spindle or overall system.

- **NC.Cmd.MessageStoreClr**
  Clears errors, warnings and notes in the error databases of the NC.

4.9.2 Global commands

NC.Cmd.DryRunMode

Description Reads or changes the setting of the test mode of channel, axis, spindle or overall system.

Remark *The data contains*

- the state of the test mode
- the instance (overall system, channel, axis or spindle)
- a number (channel number, axis number or spindle number)

The "LDX_Acb_DryRunMode" enumerator specifies the state of the test mode, it can obtain the following values:

Enumerator LDX_Acb_DryRunMode

LDX_SDMode	Description
LDX_Acb_DryRunModeOff	Test mode OFF
LDX_Acb_DryRunModeOn	Test mode on
LLDX_Acb_TestFeedrateOff	Test feed off
LDX_Acb_TestFeedrateOn	Test feed on
LDX_Acb_TestRapidTraverseOff	Test rapid traverse off
LDX_Acb_TestRapidTraverseOn	Test rapid traverse on

The "LDX_Acb_DryRunItem" enumerator specifies the instance that will be set. It can assume the following values:

Enumerator LDX_Acb_DryRunItem

LDX_SDMode	Description
LDX_Acb_AllAxis	All axes
LDX_Acb_ChannelAxis	All axes assigned to channels
LLDX_Acb_SingleAxis	Any single axis
LDX_Acb_AsyncAxis	All asynchronous axes
LDX_Acb_SingleSpindle	One spindle

Restriction: To date, only activation and deactivation of the complete lock (overall system) as well as activation and deactivation of test feed and test rapid traverse (channel) are supported.

See section SAV Errors on page 220for a description of errors potentially occurring while switching.

Item syntax NC.Cmd.DryRunMode

Call Synchronous/asynchronous

Access Read

- Reads the test mode setting

Write

- Changes the test mode setting

Cyclic (opt.) no

Data format NC.Cmd.DryRunMode, write

row, col	Type	Description	Activation of the test mode in channel 2
0,0	long	State of the test mode	LDX_Acb_DryRunModeOn
0.1	string	Instance	LDX_Acb_ChannelAxis
0.2	long	Number	2

NC.Cmd.MessageStoreClr

Description Clears errors, warnings and notes in the error databases of the NC.

Remark A clear mode determines which messages are cleared. In addition, it can be specified in which channel the messages is to be cleared.

Enumerator "LDX_WeaClrMode" is available for the clear mode. It can assume the following values:

Enumerator LDX_WeaClrMode

LDX_McaBackupOption	Description
LDX_WEA_RESETCHAN1_C	Clear due to channel 1 reset
LDX_WEA_RESETCHAN2_C	Clear due to channel 2 reset
...	
LDX_WEA_RESETCHAN12_C	Clear due to channel 12 reset
LDX_WEA_CLRWARNG_C	Clearing warnings
LDX_WEA_CLRRUNTIMEANCMT_C	Clearing programmed notes
LDX_WEA_ALLCHANRESET_C	Clear due to system reset
LDX_WEA_WARMSTART_C	Clear due to warm start
LDX_WEA_COLDSTART_C	Clear due to cold start

The "LDX_WeaClrChan" enumerator specifies the channels in which data is cleared:

Enumerator LDX_WeaClrChan

LDX_McaBackupOption	Description
LDX_WEA_CHAN1_C	Clearing in channel 1
LDX_WEA_CHAN2_C	Clearing in channel 2
...	
LDX_WEA_CHAN12_C	Clearing in channel 12
LDX_WEA_ALLCHAN_C	Clearing system-wide

Item syntax NC.Cmd.MessageStoreClr

Call Synchronous/asynchronous

Access Write

Cyclic (opt.) no

Data format NC.Cmd.MessageStoreClr, write

row, col	Type	Description	Example: Clearing all warnings
0,0	long	Clear mode	LDX_WEA_CLRWARNG_C
0.1	Empty		
0.2	long	Channels in which data is deleted	LDX_WEA_ALLCHAN_C

NC.Cmd.Shutdown

Description The item executes a shutdown of the control.

Remark All tasks are reset to their initial position and all file systems are set to a consistent state.

The file system handling before the shutdown is defined via the write parameter:

- 1 = Backup RAM , backup UsrFep
- 2 = Delete RAM , backup UsrFep
- 3 = Backup RAM, delete UsrFep

Item syntax NC.Cmd.Shutdown

Call Synchronous/asynchronous

Access Write

Cyclic (opt.) no

Data format NC.Cmd.Shutdown, write

row, col	Type	Description	Example: Backup RAM , backup UsrFep
0,0	long	Activating shutdowns after backing up RAM and UsrFep	1

NC.Cmd.Unlock

Description Unlocking the control with an “Unlock” code.

Remark The applied “Unlock” code is checked for validity. If it is valid, the control is unlocked for the period specified in the code. The version of the “Unlock” code is optional. The default value is 0.

Item syntax NC.Cmd.Unlock

Call Synchronous/asynchronous

Access Write

Cyclic (opt.) no

Data format NC.Cmd.Unlock, write

row, col	Type	Description	Example: (Individually determined coding)
0,0	string	Unlock code	nT2t-NjFy-2FpB-VjRA-JESu-EA52-gs6D-42sD
0.1	long	Optional: Version of the “Unlock” code	0

System.Cmd.Reset

Description Addressing the system reset. A reset is executed in all channels. Possibly active programs are deselected.

Item syntax System.Cmd.Reset

Call Synchronous/asynchronous

Access Write

Cyclic (opt.) no

System.Cmd.Restart

Description Activates a control startup. If the restart can be performed on the side of the control, the control is started.

Item syntax System.Cmd.Restart

Call Synchronous/asynchronous

Access Write

Cyclic (opt.) no

4.9.3 Channel-dependent commands

NC.Chan.Cmd.MDI

Description Transfer of MDI blocks to a specified channel.

Remark The transferred NC blocks are converted from UNICODE to ANSI.

Additional parameters are specified via flags. Therefore, the "LDX_Sav_Flag" enumerator is available with the following values:

Enumerator LDX_Sav_Flag

LDX_Sav_Flag	Description
LDX_SAV_FLAG_LINK_BEFORE_READY_C	If this flag is set, a linker run is performed during selection. During this run, a linker table is created for the main program. This table is required if subroutine calls or CPL elements exist in the program.
LDX_SAV_FLAG_RETURN_BY_READY_C	Usually, upon selection, it is checked via the operation mode and the NC state if the selection is allowed. Afterwards, the selection is acknowledged. If this flag is set, the acknowledgement is made only after the completion of the selection, i.e. when the NC status is "ready".
LDX_SAV_FLAG_DA_C	Reserved
LDX_SAV_FLAG_REMOTE_C	Reserved
LDX_SAV_FLAG_SUPERVISOR_C	Reserved
LDX_SAV_FLAG_MDI_C	This flag has to be set if a block is to be processed in "Manual data input" operation mode.
LDX_SAV_FLAG_PROCESSDELETE_C	An active program or an active MDI is deselected first.

LDX_Sav_Flag	Description
LDX_SAV_FLAG_AUTOSTART_C	Together with LDX_SAV_FLAG_MDI_C: The input block is started immediately. If the specified channel is not active, the block is started in this channel. All NC blocks are then permitted. If the specified channel is active, the block is handled as a switching function. Restrictions -> LDX_SAV_FLAG_MACHINE_C
LDX_SAV_FLAG_JOG_C	This flag has to be set if axes are to be traveled in the operation mode "jog mode" or if manual workpiece coordinate motions are to be carried out in the "setup" mode.
LDX_SAV_FLAG_REFPT_C	This flag has to be set if axes are to be started in the "travel to reference point" mode.
LDX_SAV_FLAG_MACHINE_C	Switching function together with LDX_SAV_FLAG_MDI_C. The block is executed parallely to the specified channel. However, only auxiliary functions and asynchronous axis motions permitted with switching function.
LDX_SAV_FLAG_PROCESSCHANGE_C	If this flag is set, a program already active at program begin is not aborted by deselection, rather replaced by the newly selected one, i.e. the modal states remain. In manual data input, the old character sequence is replaced by the new one.
LDX_SAV_FLAG_BUFFERED_INPUT_C	Buffered NC specification together with LDX_SAV_FLAG_MDI_C. While blocks are still executed, others can be specified.
LDX_SAV_FLAG_NO_CHAN_CHANGE_C	The specified channel is kept. No implicit change to the Auto start channel.
LDX_SAV_FLAG_RSP_IMMEDIATELY_C	The job is acknowledged after processing and not only after the block became active.
LDX_SAV_FLAG_SET_SEARCHRUN_C	This flag can be used to make settings for the block pre-run that have to be specified as XML string. Examples of the input of the target position 100: <SearchRun><DestPos>100</DestPos></SearchRun>

LDX_Sav_Flag	Description
LDX_SAV_FLAG_MORE_FOLLOW_C	More messages with NC blocks follow. The block preview waits until more blocks can be prepared. If the flag is not set and there is a communication delay, the MTX automatically creates a down slope and stops in a defined way.
LDX_SAV_FLAG_OVERSTORE_C	Overwriting in the operation modes Automatic, Program Block, Single Block and Single Step. After stopping an active program with feed hold, a block can be transferred using the NC block specification and this flag. The channel state becomes ready. The channel operation mode remains on Automatic. The specified block is processed with the next NC start.

Item syntax NC.Chan.Cmd.MDI,(1)

1. channel number

Call Synchronous/asynchronous

Access Write

Writing is only allowed after the last MDI has been completed (successfully or faulty). If errors occur, the write process is negatively acknowledged.

☞ *Restrictions for manual input data:*

- Several blocks are transferred collectively. To process single blocks, each block has to be transferred separately.
- CPL elements in the manual data are not allowed. However, a subroutine may be called in the manual input blocks.
- The kernel does not support linking before processing the manual input blocks.
- The length of an NC block is limited to 511 characters.
- When several NC blocks are transferred, the total length is limited to 8 Kbytes (equals 4 kB in ANSI).
- Only LF (\n) is permissible as end identifier.

Condition for block pre-run:

- All block pre-run data is transmitted in a string enclosed in the xml tags **<SearchRun>** and **</SearchRun>**.

Cyclic (opt.) no

Data format NC.Chan.Cmd.MDI, write

row, col	Type	Description	Example
0,0	long	Flag	LDX_SAV_FLAG_MDI_C
n,0	string	MDI block	N10 G1 F100 X35

NC.Chan.Cmd.MidProgramStartup

Description Parameterization of the block pre-run in the specified channel.

Remark Additional parameters can be specified via flags. Therefore, the "LDX_Sav_Flag_Set" enumerator is available with the following values:

Enumerator LDX_Sav_Flag_Set

LDX_Sav_Flag_Set	Description
LDX_SAV_FLAG_SET_INTERRUPTFCT_C	Controls asynchronous subroutines. In the StartBlock, the number (1 to 8) is specified as string ("1"), in the EndBlock, the job is specified. **Jobs for asynchronous subroutines:** "SETINT" log in "DISABLE" switch off "ENABLE" switch on again "CLRINT" delete "START" start
LDX_SAV_FLAG_SET_SEARCHRUN_C	Settings for block pre-run The block pre-run data is enclosed by <SearchRun> ... /<SearchRun>

Item syntax **NC.Chan.Cmd.MidProgramStartup,(1)**

1. channel number

Call Synchronous/asynchronous

Access Write

The name of the main program is specified.

Apart from that, the block pre-run data and flags are specified.

Cyclic (opt.) no

Block pre-run data The block pre-run data is transmitted to a string enclosed in **<SearchRun>** ... **</SearchRun>**.

Position of the target block in the part program

Syntax:

<DestPos>[Position]**</DestPos>**

Defines the position of the target block in a main program or subroutine.

Position is the offset of the first character of the target block to the program start. The first block has the position 0.

The position is deleted with:

- NC start for further processing after block pre-run,
- a change of the operation mode,
- reset,
- new specification,
- empty tag **<DestPos></DestPos>**

☞ If the block pre-run is repeated by **<StartCmd>**, the processing in a loop can be continued at any position.

Subroutine of the target position

Syntax:

<DestFName>[PartsProgram]**</DestFName>**

Specifies the part program in which the target position is located. If no part program is specified, the target position is searched in the main program.

If the name of the part program does not contain an absolute path, it is searched in the path of the main program.

The name is deleted with:

- NC start for further processing after block pre-run,
- a change of the operation mode,
- reset,
- new specification,
- Empty tag **<DestFName></DestFName>**

Start block pre-run

Syntax:

<StartCmd>[Commands]**</StartCmd>**

Starts a block pre-run and controls various strategies for returning to contour.

The commands are added or linked bit by bit (e.g. flags = LDX_PRsStart-SearchRun_C | LDX_PRsAsupAfterRepos_C):

- "**LDX_PRsStartSearchRun_C**" starts a block pre-run at the start of program or a repeated start after a target block has been reached. In loops, the target block can be reached several times.

  Start is only effective

  – if the target position is set,

  – if the subroutine of the target position is set or the machining position is located in the main program, and

  – if "Block pre-run" is the active operation mode.
- "**LDX_PRsActionBlkWhileRepos1_C**" activates the first action block. It is executed after an NC start for further processing or after the first asynchronous subroutine (asup).
- "**LDX_PRsActionBlkWhileRepos2_C**" activates the second action block. It is executed after the first action block.
- "**LDX_PRsActionBlkAfterRepos_C**" activates the third action block. It is executed after the end position of the target block has been reached.
- "**LDX_PRsAsupBeforeRepos_C**" activates the first asynchronous subroutine. It is started for further processing after the NC start.
- "**LDX_PRsAsupWhileRepos_C**" activates the second asynchronous subroutine. It is started after the completion of the second action block (if none is activated after the first asynchronous subroutine).
- "**LDX_PRsAsupAfterRepos_C**" activates the third asynchronous subroutine. It is started after the third action block (if it does not exist then after reaching the end position of the target block).
- "**LDX_PRsRcbeCallChainEnable_C**" unlocks the call chain generation of the RCBE command.
- "**LDX_PRsRunToEndOfProgram_C**" The block pre-run should completely process the program to generate call chain marks with RCBE or to check the program for example.

- "LDX_PRsUseNcCallChain_C" uses the stored state structure to find the target block.

The commands are deleted with:

- the completion of all action blocks and asynchronous subroutines,
- a change of the operation mode,
- reset,
- Tag **<ResetCmd>...</ResetCmd>**

Deleting block pre-run commands

Syntax:

<ResetCmd>[commands]**</ResetCmd>**

Deletes block pre-run commands. The commands are set as for <StartCmd>.

Saving the subroutine nesting

Syntax:

<StoreCallChain>1**</StoreCallChain>**

Stores a state structure consisting of a call chain in a channel. It can be set for the next block pre-run after processing has been stopped.

It is deleted with

- a new specification,
- empty tag **<StoreCallChain></StoreCallChain>**

Logging on different action blocks

Syntax:

<ActionBlkWhile1>[NBblock]**</ActionBlkWhile1>**

Syntax:

<ActionBlkWhile2>[NCblock]**</ActionBlkWhile2>**

Syntax:

<ActionBlkAfter>[NcBlock]**</ActionBlkAfter>**

Logs in on action blocks.

Parameter:

NC block with the maximum length "LDX_McoMaxNcBlockSize_C"

They are deleted with

- a new specification,
- an empty tag **<ActionBlk...></ActionBlk...>**

Logging in on different asynchronous subroutines

Syntax:

<AsupBefore>[PathName] [SelectionFlags]**</AsupBefore>**

Syntax:

<AsupWhile>[PathName] [SelectionFlags]**</AsupWhile>**

Syntax:

<AsupAfter>[PathName] [SelectionFlags]**</AsupAfter>**

Logs in on asynchronous subroutines. They can be called before, during, or after returning to contour.

Parameter:

- Path name of the asynchronous subroutine

If no path is specified, the system searches in the path of the main program.

- Selection flags of the program selection "LDX_SavProcessCreate".

  "LDX_SAV_FLAG_LINK_BEFORE_READY_C" for linking and "LDX_SAV_FLAG_AUTOSTART_C" for automatic start of the asynchronous subroutines are important.

They are deleted with

- a new specification,
- an empty tag **<Asup...></Asup...>**

Example "/usrfep/sekt.cnc" is to be selected. The target is block "N20". The block pre-run is to be started in the "Block pre-run" mode. After NC start for further processing, an action block is to be started, and after reaching the target position, the Asup "sektAsup.cpl" is to be started from "/usrfep".

Explanations on the XML stream (block pre-run data):

```
<SearchRun>
<DestPos>25</DestPos>
<DestFName>/usrfep/sekt.cnc</DestFName>
// not required in this case
// Start (1) with activation of action block 1 (2) and Asup 3 (64)(=67)
<StartCmd>67</StartCmd>
// Asup is to be selected with linking (1) and Auto start (128).
<AsupAfter>sektAsup.cpl>129</AsupAfter>
<ActionBlkWhile1>M99 S1000</ActionBlkWhile1>
</SearchRun>
```

Data format **NC.Chan.Cmd.MidProgramStartup, write**

row, col	Type	Description	Example
0,0	string	Program name with path	/mnt/sekt.npg
0.1	string (XML stream	Block pre-run data	<SearchRun> <DestPos>106</DestPos> <StartCmd>1<\StartCmd> <\SearchRun>
0.2	Empty	Reserved ?	
0,3	long	Flags	LDX_SAV_FLAG_SET_SEARCHRUN_C

NC.Chan.Cmd.ModeOfOperation

Description Is used to change the operation mode in the channel.

Remark Use the "LDX_SavOpModes" enumerator to code the desired operation modes. This enumeration type can assume the following values:

Enumerator LDX_SavOpModes

LDX_SavOpModes	Description
LDX_SAV_OP_MODE_EMPTY_C	No operation mode active
LDX_SAV_OP_MODE_JOGGING_C	Jog mode
LDX_SAV_OP_MODE_REFERENCE_C	Traveling to reference point
LDX_SAV_OP_MODE_BLOCKSEARCH_C	Block pre-run
LDX_SAV_OP_MODE_MDI_C	Manual data input
LDX_SAV_OP_MODE_AUTOMATIC_C	Automatic: Part programs can be processed in the continuous block
LDX_SAV_OP_MODE_SINGLE_BLOCK_C	Program block: With "NC start", individual blocks are processed as in the part program. Block preparation only starts upon "NC start".
LDX_SAV_OP_MODE_SINGLE_STEP_C	Single step: With "NC start", an individual block is passed on to the interpolator for processing.
LDX_SAV_OP_MODE_TEACH_C	Reserved
LDX_SAV_OP_MODE_SUSPEND_C	Reserved
LDX_SAV_OP_MODE_SINGLE_ACT_BLOCK_C	Single block: With "NC start", all the blocks generated and prepared due to an individual NC block in the part program are passed on to the interpolator for processing.
LDX_SAV_OP_MODE_RETURN_TO_PATH_C	Return to contour: Axes can leave the contour manually and return to the contour automatically or manually.
LDX_SAV_OP_MODE_DEBUG_STEP_C	CPL debugger: Single blocks are processed as in the part program.
LDX_SAV_OP_MODE_DEBUG_CONT_C	CPL debugger: All the blocks are processed up to the next break point.
LDX_SAV_OP_MODE_JOGGING_COORDS_C	Set manually: Motion in workpiece coordinates.

Errors that can occur while switching the operation modes, are described in section 0.

Item syntax NC.Chan.Cmd.ModeOfOperation,(1)

1. channel number

Call Synchronous/asynchronous

Access Write

Cyclic (opt.) no

Data format **NC.Chan.Cmd.ModeOfOperation, write**

row, col	Type	Description	Example
n,0	long	Operating mode	LDX_SAV_OP_MODE_AUTOMATIC_C

NC.Chan.Cmd.OptionalStop

Description Switches "Optional stop" for the specified channel.

Item syntax **NC.Chan.Cmd.OptionalStop,(1)**

1. channel number

Call Synchronous/asynchronous

Access Write

Cyclic (opt.) no

Data format **NC.Chan.Cmd.OptionalStop, write**

row, col	Type	Description	Example
0,0	bool	Optional stop	TRUE

NC.Chan.Cmd.Reset

Description Resetting a channel.

Remark The "LDX_Sav_RFlag" enumerator is available with the following values:

Enumerator LDX_Sav_RFlag

LDX_Sav_RFlag	Description
LDX_SAV_RFLAG_NORESELECT_C	Upon reset, suppresses the implicit automatic program restart

Item syntax **NC.Chan.Cmd.Reset,(1)**

1. channel number

Call Synchronous/asynchronous

Access Write

Cyclic (opt.) no

Data format **NC.Chan.Cmd.Reset, write**

row, col	Type	Description	Example: NC.Chan.Cmd.Reset, 3 Reset in channel 3
0,0	long	Trigger reset	1

NC.Chan.Cmd.ReturnToPath

Description Completes leaving the contour, i.e. logging of the leaving motions is terminated. Further jog motions are no longer recorded.

Remark A subsequent "NC start" causes a return to contour with the active return strategy.

Item syntax **NC.Chan.Cmd.ReturnToPath,(1)**

1. channel number

Call Synchronous/asynchronous

Access Write

Cyclic (opt.) no

NC.Chan.Cmd.ReturnToPathStrategy

Description Defines the return strategy by specifying the return to contour mode and the return to contour point.

Remark The return to contour mode is determined by means of the "LDX_McoRetPath_Mode" enumerator which can assume the following values:

Enumerator LDX_McoRetPath_Mode

LDX_McoRetPath_Mode	Description
LDX_McoRetPathAutomatic_C	Automatic return to contour
LDX_McoRetPathSingleBlock_C	Return to contour with single block
LDX_McoRetPathManual_C	Manual return to contour

The return to contour point is determined by means of the "LDX_McoRetPath_Point" enumerator which can assume the following values:

Enumerator LDX_McoRetPath_Point

LDX_McoRetPath_Point	Description
LDX_McoRetPathStartOfBlock_C	Return to starting point
LDX_McoRetPathEndOfBlock_C	Return to end point
LDX_McoRetPathPointOfInterruption_C	Return to point of interruption

Item syntax NC.Chan.Cmd.ReturnToPathStrategy,(1)

1. channel number

Call Synchronous/asynchronous

Access Write

Cyclic (opt.) no

Data format NC.Chan.Cmd.ReturnToPathStrategy, write

row, col	Type	Description	Example
0,0	long	Return to contour operation mode	LDX_McoRetPathAutomatic_C
0.1	long	Point of return	LDX_McoRetPathStartOfBlock_C

NC.Chan.Cmd.SearchBlock

Description Positions on the specified NC block in the selected main program.

Remark "Search block" is only possible in the main program.

The "start block" and "end block" parameters (IDs) are transmitted as string.

It is searched according to the following convention:

- Spaces at the start of an NC block are ignored,
- If an ID starting with a number is not found, the search looks for the N<ID>.

  e.g. ID "50" finds NC block "N50X100"

- If the ID ends with a number, no other numbers may follow in the found NC block.

  e.g. ID "G1X10" does not find NC block "G1X100"
- If the ID ends with a letter, a space character has to follow in the found NC block.

  e.g. ID "50A" finds NC block "50A = 1", but not "50A=1".
- Otherwise, an NC block is found if ID and start of the NC block match.

  e.g. ID "G1" does not find "G1F1000"

When the IDs are found, processing will start with the block of the start ID and will end with the block with the end ID. If there is information missing, or if an ID was not found, processing will start at the start of the program and ends at the end of the program. After the end of search, the NC state will change to "ready" (LDX_SAV_OP_STATE_READY_C).

Item syntax **NC.Chan.Cmd.SearchBlock,(1)**

1. channel number

Call Synchronous/asynchronous

Access Write

The program name is specified with path.

Apart from that, start block, end block and flags can be specified.

Cyclic (opt.) no

Data format **NC.Chan.Cmd.SearchBlock, write**

row, col	Type	Description	Example
0,0	string	Start block	N545
0.1	string	End block	

NC.Chan.Cmd.SelectProgram

Description Selects a program in the specified channel.

Remark Additional parameters can be set via flags. Therefore, the "LDX_SavFlags" enumerator is available with the following values:

Enumerator LDX_SavFlags

LDX_SavFlags	Description
LDX_SAV_FLAG_LINK_BEFORE_READY_C	If this flag is set, a linker run is performed during selection. During this run, a linker table is created for the main program. This table is required if subroutine calls or CPL elements exist in the program.
LDX_SAV_FLAG_RETURN_BY_READY_C	Usually, upon selection, it is checked via the operation mode and the NC state if the selection is allowed. Afterwards, the selection is acknowledged. If this flag is set, the acknowledgement is made only after the completion of the selection, i.e. when the NC status is "ready".
LDX_SAV_FLAG_DA_C	Reserved
LDX_SAV_FLAG_REMOTE_C	Reserved

LDX_SavFlags	Description
LDX_SAV_FLAG_SUPERVISOR_C	Reserved
LDX_SAV_FLAG_MDI_C	This flag has to be set if a block is to be processed in "Manual data input" operation mode.
LDX_SAV_FLAG_PROCESSDELETE_C	An active program or an active MDI is deselected first.
LDX_SAV_FLAG_AUTOSTART_C	Together with LDX_SAV_FLAG_MDI_C: The input block is started immediately. If the specified channel is not active, the block is started in this channel. All NC blocks are then permitted. If the specified channel is active, the block is handled as a switching function. Restrictions -> LDX_SAV_FLAG_MACHINE_C
LDX_SAV_FLAG_JOG_C	This flag has to be set if axes are to be traveled in the operation mode "jog mode" or if manual workpiece coordinate motions are to be carried out in the "setup" mode.
LDX_SAV_FLAG_REFPT_C	This flag has to be set if axes are to be started in the "travel to reference point" mode.
LDX_SAV_FLAG_MACHINE_C	Switching function together with LDX_SAV_FLAG_MDI_C. The block is executed parallely to the specified channel. However, only auxiliary functions and asynchronous axis motions permitted with switching function.
LDX_SAV_FLAG_PROCESSCHANGE_C	If this flag is set, a program already active at program begin is not aborted by deselection, rather replaced by the newly selected one, i.e. the modal states remain. In manual data input, the old character sequence is replaced by the new one.
LDX_SAV_FLAG_BUFFERED_INPUT_C	Buffered NC specification together with LDX_SAV_FLAG_MDI_C. While blocks are still executed, others can be specified.
LDX_SAV_FLAG_NO_CHAN_CHANGE_C	The specified channel is kept. No implicit change to the Auto start channel.
LDX_SAV_FLAG_RSP_IMMEDIATELY_C	The job is acknowledged after processing and not only after the block became active.
LDX_SAV_FLAG_CALLCHAIN_SELECT_C	A program is selected using the internally saved call chain. It is only effective if a empty string was transferred for "program name with path".

Item syntax NC.Chan.Cmd.SelectProgram,(1)

1. channel number

Call Synchronous/asynchronous

Access Write

The program name is specified with path.

Apart from that, start block, end block and flags can be specified.

Cyclic (opt.) no

Data format **NC.Chan.Cmd.SelectProgram, write**

row, col	Type	Description	Example
0,0	string	Program name with path	/mnt/myprogram.npg
0.1	string	Start block	""
0.2	string	End block	""
0,3	long	Flags	LDX_SAV_FLAG_LINK_BEFORE_READY_C | LDX_SAV_FLAG_PROCESSDELETE_C

NC.Chan.Cmd.SkipBlockMode

Description Switches "Skip block" for the specified channel.

Item syntax **NC.Chan.Cmd.SkipBlockMode,(1)**

1. channel number

Call Synchronous/asynchronous

Access Write

Cyclic (opt.) no

Data format **NC.Chan.Cmd.SkipBlockMode, write**

row, col	Type	Description	Example
0,0	bool	Skip block	TRUE

4.10 Error messages

4.10.1 Error handling

To provide the user with accurate information on errors, the standard SCP error handling supports. Error information is provided via the "ISCPError" interface. If an error occurs during the execution of a requested service, an SCP error code is returned. If the SCP interface is used incorrectly (e.g. by incorrect parameterization when creating an item), standard SCP error codes are returned.

If errors occur during execution of a requested function (e.g. program selection, reading of a table), the device-specific error codes are returned. Errors that can be assigned to one device-specific server, provide the individual error code. All others are reported as general SCPErrNACK(x) messages.

Via the ISCPError interface, the Logical Device MTX provides more detailed information on the error in question:

- the assignment to the respective NCS server
- a brief description of the error as text

- an unambiguous error code which can be called as error constant and allows an exact analysis of the error
- a detailed description of the error as text.

The following table provides an overview on the error information.

Error code	Description
SCPErrTimeOut - 1	See SCP documentation
SCPErrFailed - 2	
...	
SCPErrDriver - 17	
Reserved (up to 100)	
SCPErrNACK1 - 101	Sercos error
SCPErrNACK2 - 102	ISP error
SCPErrNACK3 - 103	NCS error
SCPErrNACK4 - 104	Error of the NCS server (SAV, MCA)
SCPErrNACK5 - 105	General MTX error
SCPErrNACK6 - 106	Firmware monitor error
SCPErrNACK7 - 107	Profibus error
SCPErrNACK8 - 108	Error MTX FIS server
SCPErrNACK9 - 109	MTX error during file operations
SCPErrNACK10 - 110	Win32 error
SCPErrNACK11 - 111	Sercos error (SIP)
...	Reserved
SCPErrNACK17 - 117	CoDeSys gateway error
SCPErrNACK18 - 118	Management error
SCPErrNACK19 - 119	Reserved
SCPErrMtxKrnPxr - 120	Basic kernel error
SCPErrMtxKrnSCS - 121	Sercos server error
SCPErrMtxKrnACB - 122	Axis motion server error
SCPErrMtxKrnDIV - 123	Error of several kernel groups (configuration, DB, system, XML and tables)
SCPErrMtxKrnWEA - 124	Wera server error
SCPErrMtxKrnFII - 125	File system info error
SCPErrMtxKrnCPL - 126	CPL server error
SCPErrMtxKrnTAR - 127	Archiving server error
SCPErrMtxKrnTSV - 128	Text server error
...	Reserved for MTX
SCPErrNACK40 - 140	Reserved for MTX

Tab. 4-34: Error Information

Error Code SCPErrNACK3

The error code SCPErrNACK3 indicates that an NCS error has occurred.

The brief description contains the name of the NCS function in which the error has occurred. The detailed error code and the detailed error description describe the occurred NCS error more detailed. In normal operation, these errors should not occur.

Error Code SCPErrNACK4

The error code SCPErrNACK4 indicates that an error of the SAV server or of the MCA server was reported when executing an NCS function or that the access on modal NCS functions was incorrect.

A brief description, a detailed error code and a detailed error description are provided. The error codes are described as defined error constants. The documentation of the items refers to the error relevant for the respective item. These are described in the following.

Error codes 120 to 128

The error codes are assigned to the described NCS servers of the MTX.

The assignment to the NCS server, a brief description, the detailed error code of the server as well as a detailed error description are reported. The error constants assigned to the individual error codes are described in the following chapters.

4.10.2 SAV Errors

The following table shows the error codes used by the SAV server. These codes can be provided in case of operation mode switching, program selection or reset:

Error code	Description
LDX_SAV_SERVICEOK_C	No error occurred
LDX_SAV_ILLCHANNUMBER_C	Addressed channel does not exist
LDX_SAV_CHANUSED_C	Service cannot be provided as the channel is busy.
LDX_SAV_CHANRESETTING_C	Service cannot be provided as the reset of channel is not completed.
LDX_SAV_FILENAMETOOLONG_C	Program name is too long (currently not used)
LDX_SAV_NOREFPKT_C	The service requests approached reference points
LDX_SAV_NOPROGRAM_C	The selected program does not exist or cannot be executed.
LDX_SAV_SERVICEINTERRUPTED_C	Writing in the buffer is interrupted in case of buffered NC block specification. A second instance tried to write to the buffer at the same time.
LDX_SAV_ILLMODE_C	Service cannot be provided as it is not permitted in the current operation mode.
LDX_SAV_STATENOTREADY_C	Channel cannot be started as the channel state is not "ready".
LDX_SAV_NOTSELECTED_C	Service cannot be provided as no program is selected.
LDX_SAV_EXECUTEFEEDHOLD_C	Program cannot be deselected as this is not allowed due to the channel state (e.g. Sav and Ipo "is running").

Error code	Description
LDX_SAV_ILLREMLOC_C	Currently not in use
LDX_SAV_ILLSTATE_C	Operation mode cannot be changed as it is not permitted due to the channel state.
LDX_SAV_UNKNOWNLABEL_C	Target of "Find block" was not found
LDX_SAV_STATENOTREADYFORPROGSTART_C	"Find block" is not possible as the channel state is "ready", but the main program processing has already been started (e.g. program is on M0)
LDX_SAV_NMBOFAXISEXCCEEDED_C	In the specification of external ZOs, the number of the axes is too high
LDX_SAV_NMBOFEXTZOFFSGRPSEXCCEEDED_C	In the specification of external ZOs, the number of the ZO groups is too high
LDX_SAV_UNKNOWNSYNTAX_C	Indicated syntax unknown
LDX_SAV_ILLTOOLCOMPINDEX_C	Invalid index with specification of an external Tool correction
LDX_SAV_NMBOFEXTTOOLCOMPSEXCCEEDED_C	Number of corrections too high when specifying an external tool correction (possibly together with the correction index)
LDX_SAV_ILLTOOLCOMPTYPE_C	Invalid format specification with specification of an external Tool correction
LDX_SAV_ILLTOOLCOMPORIENTATION_C	Invalid edge position with specification of an external Tool correction
LDX_SAV_ILLTOOLCOMPGROUP_C	Invalid correction group
LDX_SAV_ILLAXISNUMBER_C	Addressed axis is not available
LDX_SAV_RUNTIMEERROR_C	Runtime error during NC block specification with automatic start, e.g. syntax error
LDX_SAV_BUFFER_OVERFLOW_C	Buffer overflow in a buffered NC block specification
LDX_SAV_ILLCOORDFILTER_C	Coordinate filter specification incorrect
LDX_SAV_VNCILLSTATE_C	Specification for the VNC is not allowed as the VNC state is not frozen
LDX_SAV_NOMEM_C	Lack of memory in the server. Requested data could not be stored.
LDX_SAV_ILLPARAM_C	An invalid value has been specified for the NCS function. The server cannot provide the data.
LDX_SAV_ILLREQUEST_C	Internal error
LDX_SAV_ILLRESPONSE_C	Internal error

Error code	Description
LDX_SAV_FUNCTIONNOTIMPLEMENTED_C	Addressed service is not available in this software version
LDX_SAV_EVDATAFAIL_C	Internal error

Tab. 4-35: SAV error codes

4.10.3 MACODA Errors

The table below shows the error codes used by the MACODA server. They can occur when MACODA parameters are read:

Error code	Description
LDX_MCA_OK_C	No error
LDX_MCA_SQL_NOTFOUND_C	Parameter not available in the current software version
LDX_MCA_INSUFFICIENT_MEMORY_C	Insufficient storage area for requested data
LDX_MCA_NO_GROUP_OR_AREA_VERSION_C	No group version or function area version exists for the specified block ID
LDX_MCA_CHANNEL_SPECIFIC_PARAMETER_EXIST_C	
LDX_MCA_NO_CHANNEL_SPECIFIC_PARAMETER_C	
LDX_MCA_NO_CONCURRENT_EDIT_C	

Tab. 4-36: MACODA error codes

4.10.4 TAD Errors

The table below shows the error codes used by the table server. They can occur during read access or write access on D-correction tables for example.

The table server can also report the xml errors listed in the section XML errors, page 227.

Error code	Description
LDX_TAD_ERR_OK_C	No error
LDX_TAD_ERR_INTERNAL_ERROR_C	Internal error (see Wera errors)
LDX_TAD_ERR_SERVER_NOT_READY_C	Server not ready (see Wera errors)
LDX_TAD_ERR_NAME2LONG_C	Invalid name length
LDX_TAD_ERR_UNKNOWN_TABLETYPE_C	Unknown table type
LDX_TAD_ERR_TABLE_NOT_FOUND_C	Table does not exist
LDX_TAD_ERR_TABLE_EXISTS_C	Table already exists
LDX_TAD_ERR_BAD_DATASIZE_C	Data buffer too small
LDX_TAD_ERR_BAD_PATHSIZE_C	Path buffer too small

Error code	Description
LDX_TAD_ERR_BAD_SEGMENTPATH SIZE_C	Segment path too small (internal)
LDX_TAD_ERR_SCHEMA_INIT_FAILED _C	Error during initialization of the schema
LDX_TAD_ERR_BAD_NCS_MSG_C	Invalid NCS message (used only by .._X functions) (internal)
LDX_TAD_ERR_NO_MORE_MOD_EVE NTS_C	Maximum number of the ModEvents is reached, no more ModEvents can be created
LDX_TAD_ERR_UNKNOWN_MOD_EVE NT_C	Unknown event ID
LDX_TAD_ERR_EVENT_ALREADY_RE Q_C	ModEvent has already been requested
LDX_TAD_ERR_EVENT_DELETED_C	Error code of an event request after deletion of the respective ModEvent (no error)
LDX_TAD_ERR_CHAN_NUMB_TOO_BI G_C	Number of channels too high
LDX_TAD_ERR_EXTENSION_C	Error in file extension
LDX_TAD_ERR_INSERT_AXIS_NOT_AL LOWED_C	Inserting an axis is only allowed for ZO tables
LDX_TAD_ERR_DELETE_AXIS_NOT_A LLOWED_C	Deleting an axis is only allowed for ZO tables
LDX_TAD_ERR_PATH_BLOCK_POS_IN VALID_C	Invalid Xpath for this function
LDX_TAD_ERR_MAX_NUMBER_AXES_ REACHED_C	Maximum number of axes reached
LDX_TAD_ERR_AXIS_POSITION_NOT_ EXIST_C	Axis not available
LDX_TAD_ERR_INPUT_PARAMETER_ C	Invalid parameters
LDX_TAD_ERR_INVALID_SCHEMA_C	Schema contains unexpected elements
LDX_TAD_ERR_NO_READ_ACCESS_F OR_FILE_C	No read access to the file
LDX_TAD_ERR_NO_WRITE_ACCESS_ FOR_FILE_C	No write access to the file
LDX_TAD_ERR_TABLE_REC_TOO_BIG _C	Table record is too large. It has to be transferable with an NCS response

Tab. 4-37: TAD error codes

4.10.5 DBD Errors

The following table shows the error codes which may occur during database table access.

The xml errors listed in the section XML errors, page 227 can also be reported.

Error code	Description
LDX_DBD_ERR_OK_C	No error
LDX_DBD_ERR_INTERNAL_ERROR_C	Internal error (see Wera errors)
LDX_DBD_ERR_SERVER_NOT_READY_C	Server not ready (see Wera errors)
LDX_DBD_ERR_NAME2LONG_C	Invalid name length
LDX_DBD_ERR_UNKNOWN_TABLETYPE_C	Unknown table type
LDX_DBD_ERR_TABLE_NOT_FOUND_C	Table does not exist
LDX_DBD_ERR_TABLE_EXISTS_C	Table already exists
LDX_DBD_ERR_BAD_DATASIZE_C	Data buffer too small
LDX_DBD_ERR_BAD_PATHSIZE_C	Path buffer too small
LDX_DBD_ERR_BAD_SEGMENTPATHSIZE_C	Segment path too small (internal)
LDX_DBD_ERR_SCHEMA_INIT_FAILED_C	Error during initialization of the schema
LDX_DBD_ERR_BAD_NCS_MSG_C	Invalid NCS message (used only by .._X functions) (internal)
LDX_DBD_ERR_NO_MORE_MOD_EVENTS_C	Maximum number of the ModEvents is reached, no more ModEvents can be created
LDX_DBD_ERR_UNKNOWN_MOD_EVENT_C	Unknown event ID
LDX_DBD_ERR_MOD_EVENT_ALREADY_REQ_C	ModEvent has already been requested
LDX_DBD_ERR_MOD_EVENT_DELETED_C	Error code of an event request after deletion of the respective ModEvent (no error)
LDX_DBD_ERR_REC_ALREADY_LOCKED_C	Data block is currently locked by a client
LDX_DBD_ERR_TABLE_REC_TOO_BIG_C	Table record is too large. It has to be transferable with an NCS response

Tab. 4-38: DBD error codes

4.10.6 CFG Errors

The following table shows the error codes which can occur while accessing the configuration data.

The xml errors listed in the section XML errors, page 227 can also be reported.

Error code	Description
LDX_CFG_ERR_OK_C	No error
LDX_CFG_ERR_INTERNAL_ERROR_C	Internal error (see Wera errors)

Error code	Description
LDX_CFG_ERR_SERVER_NOT_READY_C	Server not ready (see Wera errors)
LDX_CFG_ERR_NAME2LONG_C	Invalid name length
LDX_CFG_ERR_BAD_DATASIZE_C	Data buffer too small
LDX_CFG_ERR_BAD_PATHSIZE_C	Path buffer too small
LDX_CFG_ERR_BAD_SEGMENTPATHSIZE_C	Segment path too small (internal)
LDX_CFG_ERR_SCHEMA_INIT_FAILED_C	Error during initialization of the schema
LDX_CFG_ERR_BAD_NCS_MSG_C	Invalid NCS message (used only by .._X functions) (internal)
LDX_CFG_ERR_NO_MORE_MOD_EVENTS_C	Maximum number of the ModEvents is reached, no more ModEvents can be created
LDX_CFG_ERR_UNKNOWN_MOD_EVENT_C	Unknown event ID
LDX_CFG_ERR_MOD_EVENT_ALREADY_REQ_C	ModEvent has already been requested
LDX_CFG_ERR_MOD_EVENT_DELETED_C	Error code of an event request after deletion of the respective ModEvent (no error)
LDX_CFG_ERR_INVALID_MODE_C	Invalid mode
LDX_CFG_ERR_INVALID_ROOT_ELEM_C	Invalid root element
LDX_CFG_ERR_CORRUPT_DATA_GROUP_C	Data group is initialized incorrectly
LDX_CFG_ERR_FILE_ACCESS_ERROR_C	Error when accessing file
LDX_CFG_ERR_INVALID_USER_ID_C	Invalid user ID
LDX_CFG_ERR_NO_MCA_INFO_C	Element does not contain MACODA information in the schema
LDX_CFG_ERR_INVALID_MCA_INFO_C	Element contains invalid MACODA information in the schema
LDX_CFG_ERR_NO_CONVERSION_CLASS_DEFINE D_C	No MACODA CFG conversion class is defined for the configuration data element
LDX_CFG_ERR_CONVERSION_CLASS_NOT_VALID _C	No valid MACODA CFG conversion class is defined for the configuration data element
LDX_CFG_ERR_NO_CONVERSION_FUNC_DEFINE D_C	No MACODA CFG conversion function is defined for the configuration data element

Error code	Description
LDX_CFG_ERR_CONVERSION_CLASS _1_NOT_ALL OWED_C	MACODA CFG conversion class 1 (no conversion) is not allowed for the configuration data element
LDX_CFG_ERR_MACODA_ID_NOT_IN_ CONVERSIO N_TABLE_C	No entry in the MACODA CFG conversion table for the specified MACODA block number
LDX_CFG_ERR_OPEN_MACODA_BLO CK_FAILED_ C	Opening a MACODA block failed
LDX_CFG_ERR_XMLSTRING_EMPTY_ C	Transferred XML string is empty
LDX_CFG_ERR_NO_STARTTAG_IN_X MLSTRING_ C	Transferred XML string has no start tag
LDX_CFG_ERR_NO_ENDTAG_IN_XML STRING_ C	Transferred xml string has no end tag
LDX_CFG_ERR_READ_MACODA_PAR A_FAILED_ C	Reading a MACODA block failed
LDX_CFG_ERR_WRITE_MACODA_PAR A_FAILED_ C	Writing a MACODA block failed
LDX_CFG_ERR_CLOSE_MACODA_BL OCK_FAILED_ C	Closing a MACODA block failed
LDX_CFG_ERR_XMLSTRING_BUFFER _TOO_SMALL_ C	Insufficient buffer for the xml data
LDX_CFG_ERR_NO_CONVERSION_PA TH_DEFINED_ C	No conversion path is defined in the conversion table for the conversion of a MACODA parameter into CFG data
LDX_CFG_ERR_INVALID_CONVERSIO N_PATH_ C	Faulty conversion path is defined in the conversion table for the conversion of a MACODA parameter into CFG data
LDX_CFG_ERR_UNEXPECTED_XPATH _ELEMENT_ C	Transferred XPath or an XPath in the MACODA CFG conversion table contains an unexpected input

Tab. 4-39: CFG error codes

4.10.7 SD Errors

The following table shows the error codes which may occur during access to system data.

The table server can also report the xml errors listed in the section XML errors, page 227.

Error code	Description
LDX_SD_ERR_OK_C	No error
LDX_SD_ERR_INTERNAL_ERROR_C	Internal error (see Wera errors)
LDX_SD_ERR_SERVER_NOT_READY_ C	Server not ready (see Wera errors)
LDX_SD_ERR_NAME2LONG_C	Invalid name length

Error code	Description
LDX_SD_ERR_BAD_DATASIZE_C	Data buffer too small
LDX_SD_ERR_BAD_PATHSIZE_C	Path buffer too small
LDX_SD_ERR_BAD_SEGMENTPATHSIZE_C	Segment path too small (internal)
LDX_SD_ERR_SCHEMA_INIT_FAILED_C	Error during initialization of the schema
LDX_SD_ERR_BAD_NCS_MSG_C	Invalid NCS message (used only by .._X functions) (internal)
LDX_SD_ERR_NO_MORE_MOD_EVENTS_C	Maximum number of the ModEvents is reached, no more ModEvents can be created
LDX_SD_ERR_UNKNOWN_MOD_EVENT_C	Unknown event ID
LDX_SD_ERR_MOD_EVENT_ALREADY_REQ_C	ModEvent has already been requested
LDX_SD_ERR_MOD_EVENT_DELETED_C	Error code of an event request after deletion of the respective ModEvent (no error)
LDX_SD_ERR_INVALID_MODE_C	Invalid mode
LDX_SD_ERR_INVALID_ROOT_ELEM_C	Invalid root element
LDX_SD_ERR_CORRUPT_DATA_GROUP_C	Data group is initialized incorrectly
LDX_SD_ERR_FILE_ACCESS_ERROR_C	Error accessing a file
LDX_SD_ERR_INVALID_USER_ID_C	Invalid user ID

Tab. 4-40: SD error codes

4.10.8 XML Errors

The following table shows XML error codes which are used when file-based and database-based tables are accessed.

A separate enumerator is provided for xml errors. Furthermore, the xml errors are included in the enumerators for file-based tables and database-based tables.

Error code	Description
LDX_XML_SERVICEOK_C	No error
LDX_XML_ERR_NOT_IMPLEMENTED_C	Function not yet implemented
LDX_XML_ERR_INTERNAL_ERROR_C	Internal error (see Wera errors)
LDX_XML_ERR_SCHEMA_PARSE_FAILED_C	Error when parsing the schema file (see Wera errors)
LDX_XML_ERR_UNKNOWN_LIBXML_ERROR_C	Unknown error from libxml

Error code	Description
LDX_XML_ERR_FOPEN_FAILED_C	Error while opening the file.
LDX_XML_ERR_MAX_OCCURS_EXCEEDED_C	Parser error: There are more elements than permitted according to maxOccurs
LDX_XML_ERR_BAD_ELEM_CONTENT_C	Parser error: Element content does not correspond to data type according to the schema
LDX_XML_ERR_PATH_SYNTAX_ERROR_C	Syntax error in path string
LDX_XML_ERR_PATH_CONFLICT_C	Segment path is not located in the root path (internal)
LDX_XML_ERR_PATH_ELEM_POS_INVALID_C	Element position is not permitted according to schema (path)
LDX_XML_ERR_PATH_ELEM_CREATE_FAILED_C	An element specified via path cannot be created
LDX_XML_ERR_PATH_ELEM_DELETE_FAILED_C	The element specified via path cannot be deleted
LDX_XML_ERR_PATH_ELEM_NOT_EXISTING_C	The element specified via path is not included in the xml files
LDX_XML_ERR_PATH_ELEM_NOT_DECLARED_C	The element specified via path is not declared in the schema
LDX_XML_ERR_PATH_ELEM_MISMATCH_C	The last element in path is not identical with the first element in Data
LDX_XML_ERR_ELEM_NOT_DECLARED_C	The element of the xml file or xml string is not declared in the schema
LDX_XML_ERR_INVALID_DATA_SIZE_C	The data size does not fit to the specified element
LDX_XML_ERR_INVALID_TYPE_ID_C	The data type does not fit to the specified element
LDX_XML_ERR_TOO_MUCH_DATA_REQUESTED_C	Too much binary data requested
LDX_XML_LIBERR_INTERNAL_ERROR_C	
LDX_XML_LIBERR_NO_MEMORY_C	
LDX_XML_LIBERR_DOCUMENT_START_C	
LDX_XML_LIBERR_DOCUMENT_EMPTY_C	
LDX_XML_LIBERR_DOCUMENT_END_C	
LDX_XML_LIBERR_INVALID_HEX_CHARREF_C	
LDX_XML_LIBERR_INVALID_DEC_CHARREF_C	

Error code	Description
LDX_XML_LIBERR_INVALID_CHARREF_C	
LDX_XML_LIBERR_INVALID_CHAR_C	
LDX_XML_LIBERR_CHARREF_AT_EOF_C	
LDX_XML_LIBERR_CHARREF_IN_PROLOG_C	
LDX_XML_LIBERR_CHARREF_IN_EPILOG_C	
LDX_XML_LIBERR_CHARREF_IN_DTD_C	
LDX_XML_LIBERR_ENTITYREF_IN_PROLOG_C	
LDX_XML_LIBERR_ENTITYREF_IN_EPIOLOG_C	
LDX_XML_LIBERR_ENTITYREF_IN_DTD_C	
LDX_XML_LIBERR_PEREF_AT_EOF_C	
LDX_XML_LIBERR_PEREF_IN_PROLOG_C	
LDX_XML_LIBERR_PEREF_IN_EPIOLOG_C	
LDX_XML_LIBERR_PEREF_IN_INT_SUBSET_C	
LDX_XML_LIBERR_ENTITYREF_NO_NAME_C	
LDX_XML_LIBERR_PEREF_NO_NAME_C	
LDX_XML_LIBERR_PEREF_SEMICOL_MISSING_C	
LDX_XML_LIBERR_UNDECLARED_ENTITY_C	
LDX_XML_LIBWAR_UNDECLARED_ENTITY_C	
LDX_XML_LIBERR_UNPARSED_ENTITY_C	
LDX_XML_LIBERR_ENTITY_IS_EXTERNAL_C	
LDX_XML_LIBERR_ENTITY_IS_PARAMETER_C	
LDX_XML_LIBERR_UNKNOWN_ENCODING_C	

Error code	Description
LDX_XML_LIBERR_UNSUPPORTED_ENCODING_C	
LDX_XML_LIBERR_STRING_NOT_STARTED_C	
LDX_XML_LIBERR_STRING_NOT_CLOSED_C	
LDX_XML_LIBERR_NS_DECL_ERROR_C	
LDX_XML_LIBERR_ENTITY_NOT_STARTED_C	
LDX_XML_LIBERR_ENTITY_NOT_FINISHED_C	
LDX_XML_LIBERR_LT_IN_ATTRIBUTE_C	
LDX_XML_LIBERR_ATTRIBUTE_NOT_STARTED_C	
LDX_XML_LIBERR_ATTRIBUTE_NOT_FINISHED_C	
LDX_XML_LIBERR_ATTRIBUTE_WITHOUT_VALUE_C	
LDX_XML_LIBERR_ATTRIBUTE_REDEFINED_C	
LDX_XML_LIBERR_LITERAL_NOT_STARTED_C	
LDX_XML_LIBERR_LITERAL_NOT_FINISHED_C	
LDX_XML_LIBERR_COMMENT_NOT_FINISHED_C	
LDX_XML_LIBERR_PI_NOT_STARTED_C	
LDX_XML_LIBERR_PI_NOT_FINISHED_C	
LDX_XML_LIBERR_NOTATION_NOT_STARTED_C	
LDX_XML_LIBERR_NOTATION_NOT_FINISHED_C	
LDX_XML_LIBERR_ATTLIST_NOT_STARTED_C	
LDX_XML_LIBERR_ATTLIST_NOT_FINISHED_C	
LDX_XML_LIBERR_MIXED_NOT_STARTED_C	
LDX_XML_LIBERR_MIXED_NOT_FINISHED_C	

Error code	Description
LDX_XML_LIBERR_ELEMCONTENT_NOT_STARTED_C	
LDX_XML_LIBERR_ELEMCONTENT_NOT_FINISHED_C	
LDX_XML_LIBERR_XMLDECL_NOT_STARTED_C	
LDX_XML_LIBERR_XMLDECL_NOT_FINISHED_C	
LDX_XML_LIBERR_CONDSEC_NOT_STARTED_C	
LDX_XML_LIBERR_CONDSEC_NOT_FINISHED_C	
LDX_XML_LIBERR_EXT_SUBSET_NOT_FINISHED_C	
LDX_XML_LIBERR_DOCTYPE_NOT_FINISHED_C	
LDX_XML_LIBERR_MISPLACED_CDATA_END_C	
LDX_XML_LIBERR_CDATA_NOT_FINISHED_C	
LDX_XML_LIBERR_RESERVED_XML_NAME_C	
LDX_XML_LIBERR_SPACE_REQUIRED_C	
LDX_XML_LIBERR_SEPARATOR_REQUIRED_C	
LDX_XML_LIBERR_NMTOKEN_REQUIRED_C	
LDX_XML_LIBERR_NAME_REQUIRED_C	
LDX_XML_LIBERR_PCDATA_REQUIRED_C	
LDX_XML_LIBERR_URI_REQUIRED_C	
LDX_XML_LIBERR_PUBID_REQUIRED_C	
LDX_XML_LIBERR_LT_REQUIRED_C	
LDX_XML_LIBERR_GT_REQUIRED_C	
LDX_XML_LIBERR_LTSLASH_REQUIRED_C	
LDX_XML_LIBERR_EQUAL_REQUIRED_C	
LDX_XML_LIBERR_TAG_NAME_MISMATCH_C	

Error code	Description
LDX_XML_LIBERR_TAG_NOT_FINISHED_C	
LDX_XML_LIBERR_STANDALONE_VALUE_C	
LDX_XML_LIBERR_ENCODING_NAME_C	
LDX_XML_LIBERR_HYPHEN_IN_COMMENT_C	
LDX_XML_LIBERR_INVALID_ENCODING_C	
LDX_XML_LIBERR_EXT_ENTITY_STANDALONE_C	
LDX_XML_LIBERR_CONDSEC_INVALID_C	
LDX_XML_LIBERR_VALUE_REQUIRED_C	
LDX_XML_LIBERR_NOT_WELL_BALANCED_C	
LDX_XML_LIBERR_EXTRA_CONTENT_C	
LDX_XML_LIBERR_ENTITY_CHAR_ERROR_C	
LDX_XML_LIBERR_ENTITY_PE_INTERNAL_C	
LDX_XML_LIBERR_ENTITY_LOOP_C	
LDX_XML_LIBERR_ENTITY_BOUNDARY_C	

Tab. 4-41: XML error codes

4.10.9 TAR Errors

The following table indicates the error codes which may occur during the archiving process.

If one of the errors occurs, archiving is stopped. In this case, the information in the log file will be invalid.

Error code	Description
LDX_TAR_ERR_OK_C	Archiving or reading in the archive has been completed successfully. Further information has been logged
LDX_TAR_ERR_NCS_C	NCS error
LDX_TAR_ERR_BAD_PARAMETERS_C	Incorrect parameters
LDX_TAR_ERR_OPENING_ARCHIVE_C	Error when opening the archive
LDX_TAR_ERR_WRITING_ARCHIVE_C	Error when writing the archive
LDX_TAR_ERR_READING_ARCHIVE_C	Error when reading the archive

Error code	Description
LDX_TAR_ERR_CHDIR_C	Error when changing to target directory
LDX_TAR_ERR_LOGFILE_C	Error when creating or writing the log file
LDX_TAR_ERR_MACODA_C	Error when editing the MACODA parameters
LDX_TAR_ERR_TABLES_C	Error when editing the tables
LDX_TAR_ERR_CPLVARS_C	Error when editing permanent CPL variables
LDX_TAR_ERR_DATAMODULES_C	Error when editing remanent data
LDX_TAR_ERR_MARKERS_C	Error when editing remanent data
LDX_TAR_ERR_FORMAT_C	Incorrect archive format
LDX_TAR_ERR_USRFEP_C	Error when programming the user FEPROM
LDX_TAR_ERR_ENOMEM_C	Insufficient memory
LDX_TAR_ERR_ALREADY_ACTIVE_C	Archives already running
LDX_TAR_ERR_NOLOGIN_C	Incorrect login on the file server
LDX_TAR_ERR_PLCSTATE_RUN_C	PLC not in STOP state

Tab. 4-42: TAR error codes during archiving

During archiving, a log file is written which lists all saved or restored files as well as possible errors.

The table below contains the error codes possible in the log file:

Error code	Description
LDX_TAR_LOG_OK_C	File processed without error
LDX_TAR_LOG_NOT_YET_PROCESSED_C	File not (yet) processed
LDX_TAR_LOG_STAT_FAILED_C	Error when calling the "stat" function
LDX_TAR_LOG_SKIPPED_C	File or directory skipped (no error!)
LDX_TAR_LOG_PATH_NOT_OPENED_C	Directory could not be opened
LDX_TAR_LOG_READING_C	Error when reading the file
LDX_TAR_LOG_EACCESS_C	Access to file impossible
LDX_TAR_LOG_EEXIST_C	File already exists
LDX_TAR_LOG_ENOENT_C	File does not exist
LDX_TAR_LOG_ENOMEM_C	No memory
LDX_TAR_LOG_EROFS_C	Read-only file system
LDX_TAR_LOG_MODE_DIFF_C	Different protection
LDX_TAR_LOG_SIZE_DIFF_C	Different size
LDX_TAR_LOG_MTIME_DIFF_C	Different date
LDX_TAR_LOG_OTHER_DIFF_C	Different file data

Error code	Description
LDX_TAR_LOG_FILE_DATA_DIFF_C	Other differences
LDX_TAR_LOG_TIME_MODE_C	Date/protection not restored
LDX_TAR_LOG_MKDIR_FAILED_C	Error when creating a directory
LDX_TAR_LOG_ERR_UNKNOWN_C	Unknown error
LDX_TAR_LOG_LOADING_MACODA_C	Error when transferring MACODA data
LDX_TAR_LOG_LOADING_TABLE_C	Error when transferring of tool table
LDX_TAR_LOG_CORRUPTED_C	File corrupted
LDX_TAR_LOG_WRITE_ENABLED_C	Write protection enabled
LDX_TAR_LOG_DATAMODULES_C	Error when transferring remanent PLC data
LDX_TAR_LOG_MARKERS_C	Error when transferring remanent PLC data
LDX_TAR_LOG_CPLVARS_C	Error when transferring CPL variables
LDX_TAR_LOG_RAMFS_NOTFOUND_C	RAM file system not in the archive
LDX_TAR_LOG_USRFEPFS_NOTFOUND_C	Usrfep file system not in the archive
LDX_TAR_LOG_MACODA_NOTFOUND_C	MACODA data not in the archive
LDX_TAR_LOG_TABLE_NOTFOUND_C	Tool table not in the archive
LDX_TAR_LOG_DATAMODULES_NOTFOUND_C	Remanent PLC data not in the archive
LDX_TAR_LOG_MARKERS_NOTFOUND_C	Remanent PLC data not in the archive
LDX_TAR_LOG_CPLVARS_NOTFOUND_C	CPL variables not in the archive
LDX_TAR_LOG_SYSTEMDATA_NOTFOUND_C	System data not in the archive
LDX_TAR_LOG_PLCSTATE_RUN_C	PLC not in STOP state
LDX_TAR_LOG_SYSTEMDATA_C	Error when applying system data

Tab. 4-43: TAR error codes in LOG file

4.10.10 Unlock error

The following table shows the error codes which can occur when unlocking the control.

Error code	Description
LDX_Unlock_Ok_C	No error - Unlock code is valid
LDX_Unlock_TargetNotSupported_C	The functionality "CtrlLocking" is not supported by this target
LDX_Unlock_NotEnabled_C	The functionality "CtrlLocking" is not activated at this control

Error code	Description
LDX_Unlock_WrongFormat_C	The format of the UnlockCode is incorrect. It is possible that characters are missing or its number is exceeded
LDX_Unlock_Corruped_C	The UnlockCode is damaged. It is possible that individual characters were transmitted incorrectly. Capitalization is irrelevant
LDX_Unlock_Invalid1_C	The UnlockCode is invalid. It is possible that data was incorrectly transmitted from the control to the OEM
LDX_Unlock_Invalid2_C	The UnlockCode is invalid. This “UnlockCode” has probably already been used. It is possible that data was incorrectly transmitted from the control to the OEM
LDX_Unlock_Outdated_C	The UnlockCode is too old. After transmitting UnlockData from the control to the OEM, a generated UnlockCode has to be used within 48 hours. Otherwise, it becomes invalid.
LDX_Unlock_UnknownVersion_C	The version of the UnlockCode is unknown to the control. It is possible that the firmware is too old.
LDX_Unlock_WrongMachine_C	The unlock code is not for the machine. It is only valid for the control from which the UnlockData originates.
LDX_Unlock_NoOEMKey_C	The key of the OEM is not available on the control. The respective file was either not found or corrupt!

Tab. 4-44: Unlock error codes

5 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

A
About this documentation..... 1
Validity of the documentation..... 1
AutoItemAccess..... 13

C
CNF.AddComponent..... 14
CNF.CMD..... 15
CNF.RemoveComponent..... 15
Complaints..... 6
ComPortList..... 17
Configuration
CNF.CMD..... 15
GDL..... 18
Criticism..... 6
Customer Feedback..... 6

D
Device-dependent items
DeviceTypeTest2..... 13
Diagnose items
System.MessageStore..... 47
Diagnostic items
System.MessageStatus..... 52
System.MessageStoreEx..... 50
Dialog items
System.ErrorStatus..... 53
System.InfoStatus..... 54
System.MessageReset..... 56
System.SetupStatus..... 55
System.StartupStatus..... 55
System.WarningStatus..... 54
DTY..... 17

E
Error codes..... 30

F
Feedback..... 6
Firmware - Download
Firmware.CheckDownloadRequirements..... 20
Firmware.ComponentFirmware..... 21
Firmware.DeviceFirmware..... 21
Firmware.Download..... 22
Firmware.ModuleFirmware..... 24
Firmware.Reboot..... 25
Firmware.ResumeConnection..... 25
Firmware.Status..... 26
Firmware.StatusReset..... 28
Firmware.SubmoduleFirmware..... 28
NC.LanguagesDownload..... 29
Firmware.CheckDownloadRequirements..... 20
Firmware.ComponentFirmware..... 21
Firmware.DeviceFirmware..... 21
Firmware.Download..... 22
Firmware.ModuleFirmware..... 24
Firmware.Reboot..... 25
Firmware.ResumeConnection..... 25
Firmware.Status..... 26
Firmware.StatusReset..... 28
Firmware.SubmoduleFirmware..... 28

G
GDL..... 18
General items
AutoItemAccess..... 13
CNF.AddComponent..... 14
CNF.RemoveComponent..... 15
DTY..... 17
ListOfItems..... 18
LOI..... 18
SCL..... 18
SDL..... 19

H
Helpdesk..... 237
Hotline..... 237

I
IndraLogic item
Plc.FeldBus..... 67
Plc.FirmwareConnection..... 64
Plc.FirmwareReboot..... 64
Plc.FirmwareStatus..... 64
Plc.FirmwareVersion..... 64
Plc.ModDiagnose..... 65
Plc.ModStatus..... 65
Plc.PVA..... 67
Plc.PVArray..... 68
Plc.PVBS..... 68
Plc.PVF..... 69
Plc.PVL..... 69
Plc.PVLK..... 70
Plc.PVT..... 71
Plc.StartStop..... 67
Plc.State2..... 66
Intended use
Areas of Application..... 11
Introduction..... 11
Use cases..... 11

L
ListOfItems..... 18
Logbook items
System.LogbookClear..... 61
System.LogbookDevices..... 62
System.LogbookStart..... 63
System.LogbookStore..... 57
System.LogbookStoreEx..... 59

N

NC - channel dependent (axis data)
NC.Chan.AxisPosMcs..... 160
NC.Chan.AxisPosProg..... 161
NC.Chan.AxisPosWcs..... 161
NC.Chan.AxisPosWcsDrv..... 162
NC - channel-dependent (adjustment data)
NC.Chan.AuxActual..... 173
NC.Chan.AuxAdjustment..... 174
NC.Chan.AuxReference..... 174
NC.Chan.FctActual..... 175
NC.Chan.FctActualSorted..... 175
NC.Chan.FctAdjustment..... 175
NC.Chan.FctAdjustmentSorted..... 176
NC.Chan.FctReference..... 176
NC.Chan.FctReferenceSorted..... 177
NC.Chan.FunctionBaseConfiguration..... 150
NC - Channel-dependent (axis data)
NC.Chan.AxisPosAcs..... 153
NC.Chan.AxisPosAcsCoupl..... 154
NC.Chan.AxisPosAcsDrv..... 154
NC.Chan.AxisPosAcsDrvAdj..... 155
NC.Chan.AxisPosBcs..... 156
NC.Chan.AxisPosBcsDrv..... 156
NC.Chan.AxisPosDistAcs..... 157
NC.Chan.AxisPosDistMcs..... 158
NC.Chan.AxisPosDistWcs..... 158
NC.Chan.AxisPosEndWcs..... 159
NC.Chan.AxisPosLag..... 159
NC - channel-dependent (correction data)
NC.Chan.ActAxisOffsetTable..... 163
NC.Chan.ActDCorrectionNmb..... 163
NC.Chan.ActDCorrectionTable..... 163
NC.Chan.ActDCorrectionValues..... 164
NC.Chan.ActOnlineCorrValue..... 164
NC.Chan.ActPlacementBcrCorr..... 165
NC.Chan.ActPlacementData..... 165
NC.Chan.ActPlacementSumData..... 166
NC.Chan.ActPlacementTable..... 166
NC.Chan.ActProgCoordATrans..... 166
NC.Chan.ActProgCoordTrans..... 167
NC.Chan.ActProgCoordTransSum..... 167
NC.Chan.ActToolCorrectionValues..... 168
NC.Chan.ActZeroOffsetSummary..... 168
NC.Chan.ActZeroOffsetValues..... 169
NC.Chan.AxisShift..... 169
NC.Chan.CompCrossData..... 170
NC.Chan.CompLeadScrewErrData..... 171
NC.Chan.CompOffsetData..... 172
NC.Chan.CompTempData..... 173
NC - channel-dependent (feed and speed data)
NC.Chan.ActAxisAccelerationPeak..... 199
NC.Chan.ActAxisVelocity..... 196
NC.Chan.ActAxisVelocityMax..... 197, 198
NC - channel-dependent (feed and speed values)
NC.Chan.ActFeedOverride..... 199
NC.Chan.ActFeedrate..... 200
NC.Chan.ProgrammedFeedrate..... 202
NC.Chan.ProgrammedFeedrateIndex..... 202
NC - channel-dependent (feed values and speed values)
NC.Chan.ActRotAxisProgrammedSpeed.... 200
NC.Chan.ActRotAxisSpeed..... 201
NC - channel-dependent (general information)
NC.Chan.ChanHidden..... 146
NC.Chan.ChanState..... 146
NC.Chan.DryRunFeed,..... 148
NC.Chan.DryRunMode,..... 149
NC.Chan.DryRunRapid,..... 149
NC.Chan.ProgramLinkInfo..... 151
NC.Chan.WaitingStates..... 152
NC - Channel-dependent (general information)
NC.Chan.BalancedConditionState 146
NC - channel-dependent (process data)
NC.Chan.ActCallChain..... 183
NC.Chan.ActMainProgram..... 183
NC.Chan.ActNcBlock..... 184
NC.Chan.ActSubProgram..... 184
NC.Chan.AxisMirror..... 184
NC.Chan.AxisPoleSet..... 185
NC.Chan.AxisRotate..... 186
NC.Chan.AxisScale..... 186
NC.Chan.EqualizeState..... 187
NC.Chan.MessageWindow..... 188
NC.Chan.ModalAuxFunctions..... 188
NC.Chan.ModalAxisOffsetFunctions..... 189
NC.Chan.ModalFunction..... 189
NC.Chan.ModalNcFunctions..... 189
NC.Chan.ModalTransformationFunctions.... 190
NC.Chan.ModeOfOperation..... 190
NC.Chan.OptionalStop..... 192
NC.Chan.ProgramPosition..... 192
NC.Chan.ProgramWindow..... 193
NC.Chan.ReentryChain..... 193
NC.Chan.ReentryDelete..... 194
NC.Chan.ReentryEvent..... 194
NC.Chan.ReturnToContour..... 194
NC.Chan.ReturnToPathStrategy..... 195
NC.Chan.SkipBlockMode..... 196
NC.Chan.UpCallFromNcBlock..... 196
NC - channel-dependent (spindle data)
NC.Chan.ActSpindleAutoGears..... 177
NC.Chan.ActSpindleGears..... 178
NC.Chan.ActSpindleMovements..... 178
NC.Chan.ActSpindleOverride..... 179
NC.Chan.ActSpindlePosition..... 180
NC.Chan.ActSpindlePower..... 180
NC.Chan.ActSpindleSpeed..... 181
NC.Chan.ActSpindleSpeedDrv..... 181
NC.Chan.ActSpindleSwitchGears..... 182
NC.Chan.ActSpindleTorque..... 182
NC.Chan.ProgrammedSpindleSpeed..... 183
NC - global (CPL data)
NC.CplPermVariable..... 98
NC.CplPermVariableList..... 99

NC.CplPermVariableWatch..... 100
NC - global (database-based tables)
NC.DbTableDataRec_U..... 105
NC.DbTableDataRecLocked_U..... 106
NC.DbTableEvent..... 107
NC.DbTableLoad..... 108
NC.DbTableQuery_U..... 109
NC.DbTableSave..... 110
NC - global (drives)
NC.OscilloscopeConfiguration..... 92
NC.SercosPhase2..... 93
NC.SPA1..... 94
NC.SPH1..... 97
NC - global (file-based tables)
NC.XmlRootElementInfo..... 101
NC.XmlRootElementList..... 101
NC.XmlSchemaList..... 102
NC.XmlTable_U..... 102
NC.XmlTableCreate..... 104
NC.XmlTableEvent..... 105
NC - global (general information)
NC.AxisNamesPhys..... 72
NC.BitInterface..... 72
NC.CheckFile..... 76
NC.ConnectionQuality..... 73
NC.CpuUsage2..... 74
NC.DryRunMode..... 75
NC.Filesystem..... 75
NC.FIS..... 77
NC.MtxCommand..... 77
NC.NofChannels..... 79
NC.ProgramChanInfo..... 89
NC.ReloadExportUs..... 90
NC.WaitingStates..... 91
NC - global (machine parameters)
NC.CfgData_U..... 112
NC.CfgDataBackup..... 112
NC.CfgDataEvent..... 113
NC.CfgDataLoad..... 113
NC.CfgDataSave..... 114
NC.MacodaParameter..... 111
NC - global (system data)
NC.SystemData_U..... 116
NC.SystemDataEvent..... 118
NC.SystemDataLoad..... 119
NC.SystemDataSave..... 119
NC.SystemDataWatch_U..... 120
NC - global (system functions)
System.Archive..... 121
System.ArchiveLog..... 122
System.ArchiveWin..... 124
System.ArchiveWinLog..... 125
System.HardwareDescription..... 127
System.Language..... 127
System.MainReleases..... 128
System.PcReleases..... 128
System.ProVi.Active..... 133
System.ProVi.KaFreeze..... 129
System.ProVi.KaOnline..... 131
System.Sfc.Active..... 140
System.Sfc.Execute..... 140
System.Sfc.KaExecute..... 141
System.Sfc.KaFreeze..... 133
System.Sfc.KaOnline..... 136
System.Sfc.Mode..... 139
System.Sfc.StepInfo..... 139
System.SubsystemReleases..... 143
System.SupportedLanguages..... 143
System.SystemTime..... 144
NC command (global)
NC.Cmd.DryRunMode..... 203
NC.Cmd.Shutdown..... 205
NC commands (channel-dependent)
NC.Chan.Cmd.MDI..... 206
NC.Chan.Cmd.MidProgramStartup..... 209
NC.Chan.Cmd.ModeOfOperation..... 212
NC.Chan.Cmd.OptionalStop..... 214
NC.Chan.Cmd.Reset..... 214
NC.Chan.Cmd.ReturnToPath..... 214
NC.Chan.Cmd.ReturnToPathStrategy..... 215
NC.Chan.Cmd.SearchBlock..... 215
NC.Chan.Cmd.SelectProgram..... 216
NC.Chan.Cmd.SkipBlockMode..... 218
NC commands (global)
NC.Cmd.MessageStoreClr..... 204
NC.Cmd.Unlock..... 205
NC.OsciUtil..... 80
NC.PlcData..... 87
System.Cmd.Reset..... 206
System.Cmd.Restart..... 206
NC.AxisNamesPhys..... 72
NC.BitInterface..... 72
NC.CfgData_U..... 112
NC.CfgDataBackup..... 112
NC.CfgDataEvent..... 113
NC.CfgDataLoad..... 113
NC.CfgDataSave..... 114
NC.CheckFile..... 76
NC.ConnectionQuality..... 73
NC.CplPermVariable..... 98
NC.CplPermVariableList..... 99
NC.CplPermVariableWatch..... 100
NC.CpuUsage2..... 74
NC.DbTableDataRec_U..... 105
NC.DbTableDataRecLocked_U..... 106
NC.DbTableEvent..... 107
NC.DbTableLoad..... 108
NC.DbTableQuery_U..... 109
NC.DbTableSave..... 110
NC.DryRunMode..... 75
NC.Filesystem..... 75
NC.FIS..... 77
NC.LanguagesDownload..... 29
NC.MacodaParameter..... 111
NC.MtxCommand..... 77
NC.NofChannels..... 79

NC.OscilloscopeConfiguration..... 92
NC.OsciUtil..... 80
NC.PlcData..... 87
NC.ProgramChanInfo..... 89
NC.ReloadExportUs..... 90
NC.SercosPhase2..... 93
NC.SPA1..... 94
NC.SPH1..... 97
NC.SystemData_U..... 116
NC.SystemDataEvent..... 118
NC.SystemDataLoad..... 119
NC.SystemDataSave..... 119
NC.SystemDataWatch_U..... 120
NC.WaitingStates..... 91
NC.XmlRootElementInfo..... 101
NC.XmlRootElementList..... 101
NC.XmlSchemaList..... 102
NC.XmlTable_U..... 102
NC.XmlTableCreate..... 104
NC.XmlTableEvent..... 105

R

Root..... 13

S

Scanning
ComPortList..... 17
SCL..... 18
SDL..... 19
Sercos
Attributes..... 32
Data types..... 32
DeviceAddress..... 43
DeviceAddressList..... 43
Elements..... 31
Error codes..... 36
Idents..... 31
Lists..... 33
SAC.Config..... 44
Sercos item..... 31
Sercos items
SPA1..... 38
SPA2..... 41
Service hotline..... 237
Suggestions..... 6
Support..... 237
System.Archive..... 121
System.ArchiveLog..... 122
System.ArchiveWin..... 124
System.ArchiveWinLog..... 125
System.HardwareDescription..... 127
System.Language..... 127
System.MainReleases..... 128
System.PcReleases..... 128
System.ProVi.Active..... 133
System.ProVi.KaFreeze..... 129
System.ProVi.KaOnline..... 131
System.Sfc.Active..... 140
System.Sfc.Execute..... 140
System.Sfc.KaExecute..... 141
System.Sfc.KaFreeze..... 133
System.Sfc.KaOnline..... 136
System.Sfc.Mode..... 139
System.Sfc.StepInfo..... 139
System.SubsystemReleases..... 143
System.SupportedLanguages..... 143
System.SystemTime..... 144
System.Text
System.TimeSync..... 144
System.TimeZone..... 145
System.TimeSync..... 144
System.TimeZone..... 145

T

Target groups..... 1

U

Unintended use..... 12
Unintended Use
Consequences, non-liability..... 11

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

R911342636

DOK-MTX***-OPC*COM*V14-PR03-EN-P