

MTX 14VRS

Functional Description - Special Functions

Reference Book
R911342624

Edition 07



Title	MTX 14VRS Functional Description - Special Functions
Type of Documentation	Reference Book
Document Typecode	DOK-MTX***-NC*F*SP*V14-RE07-EN-P
Internal File Reference	RS-248a4d9e0d0eb4200a6846a5002c4e5e-7-en-US-3
Change record	Edition 07, 2020-10 Refer to tab. 1-1 "Change record" on page 1
Copyright	© Bosch Rexroth AG 2020 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	Development Software and Control Firmware - Control Functions CNC, HaBu (MS/MePe)

Table of Contents

	Page
1 About this documentation.....	1
1.1 Validity of the documentation.....	1
1.2 General information.....	2
1.3 Required and supplementing documentation MTX.....	3
1.3.1 Selection/compilation.....	3
1.3.2 Configuration.....	3
1.3.3 Commissioning.....	4
1.3.4 Operation.....	5
1.3.5 OEM Engineering.....	6
1.3.6 AddOns.....	6
1.4 Information representation.....	6
1.4.1 Safety instructions.....	6
1.4.2 Symbols used.....	7
1.4.3 Names and abbreviations.....	7
1.4.4 Customer feedback.....	7
2 Important instructions on use.....	9
2.1 Intended use.....	9
2.1.1 Introduction.....	9
2.1.2 Areas of use and application.....	9
2.2 Unintended use.....	10
3 Drives (axes, spindles).....	11
3.1 Delayed error reaction in case of non-fatal drive errors.....	11
3.1.1 Description.....	11
3.1.2 Applying.....	13
3.1.3 Activating.....	13
3.1.4 Deactivating.....	13
3.2 Sercos Gateway.....	14
3.2.1 Description.....	14
3.2.2 Applying.....	15
3.2.3 Activating.....	15
3.2.4 Deactivating.....	15
3.3 Synchronous axes (channel axes).....	15
3.4 Asynchronous axes (auxiliary axes).....	15
3.5 Handling instruction: Axis reset.....	16
4 Operating functions (manual operation).....	17
4.1 Online correction with handwheel (in workpiece coordinates).....	17
4.1.1 Description.....	17
4.1.2 Handling instruction: Online correction with handwheel in tool coordinates.....	20

Table of Contents

	Page
5 Compensating mechanical inaccuracies	25
5.1 Overview.....	25
5.2 Lead screw error compensation (LSEC)	25
5.2.1 Description.....	25
5.2.2 Handling instruction: Lead screw error compensation (LSEC) - Determining compensation values.....	26
5.2.3 Handling instruction: Lead screw error compensation (LSEC), activating.....	29
5.2.4 Handling instruction: Lead Screw Error Compensation (LSEC), Deactivating.....	29
5.3 Cross compensation (CCOMP).....	30
5.3.1 Description.....	30
5.3.2 Handling instruction: Cross compensation (CCOMP), determining the compensation values.....	31
5.3.3 Handling instruction: Cross compensation (CCOMP), activating.....	35
5.3.4 Handling instruction: Cross compensation (CCOMP), deactivating.....	35
5.4 Switching compensation tables (LSEC and CCOMP).....	36
5.4.1 Description.....	36
5.4.2 Applying.....	38
5.4.3 Activating.....	39
5.4.4 Deactivating.....	39
5.5 Straightness and angle error compensation.....	39
5.5.1 Description.....	39
5.5.2 Measuring method.....	41
5.5.3 gct files.....	42
5.5.4 Switch on/off behavior, control reset.....	45
5.5.5 Handling instruction: Straightness and angle error compensation.....	45
5.6 Backlash compensation	48
5.6.1 Description.....	48
5.6.2 Handling Instruction: Backlash Compensation.....	48
5.7 Circle compensation at quadrant transition.....	50
5.7.1 Description.....	50
5.7.2 Handling instruction: Circle compensation at quadrant transition.....	52
5.8 Volumetric compensation (VCP).....	54
5.8.1 Description.....	54
6 Channel control.....	57
6.1 Retrace: Retracing on contour.....	57
6.1.1 Description.....	57
6.1.2 Applying.....	63
6.1.3 Activating.....	63
6.1.4 Deactivating.....	63
7 Safety/monitoring.....	65
7.1 Dimension of the working plane.....	65
7.1.1 Description.....	65
7.1.2 System date "/SysDimPlane".....	65
7.1.3 Handling instruction.....	68

	Page
8 Path motion.....	71
8.1 TCS travel motion.....	71
8.1.1 Description.....	71
8.2 PTP traversing motion.....	72
8.2.1 Description.....	72
9 Functions for drive parameterization.....	77
9.1 Functions for Sercos handling.....	77
9.1.1 Overview.....	77
9.2 Sercos initialization.....	77
9.2.1 Description.....	77
9.2.2 Applying.....	78
9.3 Automatic parameter download.....	79
9.3.1 Description.....	79
9.3.2 Applying.....	79
9.3.3 Activating.....	81
9.3.4 Deactivating.....	81
9.4 Logging the transmitted Sercos parameters	82
9.4.1 Description.....	82
9.4.2 Activating.....	82
9.4.3 Deactivating.....	82
9.5 Downloading Sercos files in phase 3 "Only if required".....	83
9.5.1 Description.....	83
9.6 Sercos multiplex channel	83
9.6.1 Description.....	83
9.6.2 Parameterizing.....	84
9.6.3 Transferring List Elements.....	84
10 System data	85
10.1 System data (SD).....	85
10.1.1 Description.....	85
10.1.2 Accessing system data.....	86
10.1.3 Saving defined SD.....	86
10.1.4 Execution tools.....	86
10.2 Structure types.....	87
10.2.1 Description.....	87
10.2.2 Structure type characteristics.....	87
10.3 Definition files	88
10.3.1 Description.....	88
10.3.2 Structuring a definition file (SDDef*.xml).....	88
10.3.3 Name and storage location of the definition file.....	91
10.4 Structure type files (XML schema files).....	91
10.4.1 Description.....	91
10.4.2 Structuring a structure type file.....	91
10.4.3 Name and storage location of a structure type file.....	92

Table of Contents

	Page
10.5	Initialization files..... 92
10.5.1	Description..... 92
10.5.2	Structuring an SD initialization file (SDDat*.xml)..... 92
10.5.3	Name and storage location of an initialization file..... 93
10.6	Re-initializing selected SD elements after NC restart 93
10.7	Handling Instruction: Example: Creation of four system data for a machine tool builder..... 94
11	Schema Files..... 101
11.1	General Information..... 101
11.2	Managing Schema Files..... 101
11.2.1	Description..... 101
11.2.2	Applying..... 109
12	Area monitoring with system data..... 111
12.1	Introduction..... 111
12.2	Example..... 111
12.3	Creating monitoring areas in the system data..... 111
12.4	General area properties..... 112
12.5	Geometric area properties..... 113
12.6	Deactivating all channel areas..... 114
12.7	Changing the operating data..... 114
12.8	Synchronizing the block preparation with the interpolator..... 115
12.9	Invalid area data..... 116
12.10	Time of area violation detection..... 116
12.11	Optional system data for monitoring areas..... 117
12.12	Restrictions..... 118
12.13	Comparison with the NC command Area..... 118
12.14	Handling instruction: Example..... 118
13	Macro Programming..... 121
13.1	Description..... 121
13.1.1	Function..... 121
13.1.2	Using Macros..... 121
13.1.3	Parameterizable Macros..... 122
13.1.4	Overriding NC Functions..... 123
13.1.5	Nesting Macros..... 124
13.1.6	Macro Types..... 124
13.1.7	Special Characters in the Replacement Text or Parameters..... 125
13.1.8	Execution Tools..... 125
13.1.9	Creating and Applying Macros..... 126
13.2	Macro Definition Files..... 126
13.2.1	Description..... 126
13.2.2	Structuring a Definition File..... 126
13.2.3	Names and Back Up of Definition Files..... 127
13.3	Example on XML-Based Macros..... 127

	Page
13.3.1 Specifying.....	127
13.3.2 Handling Instruction: Example.....	130
14 Fast actions MTX fa.....	133
14.1 Overview.....	133
14.1.1 Purpose.....	133
14.1.2 Terms.....	133
14.1.3 Operating principle.....	133
14.1.4 Example.....	134
14.1.5 Data exchange between fast PLC tasks, NC and drives.....	139
14.2 Configuration of the cyclic data exchange.....	140
14.2.1 Cyclic data exchange.....	140
14.2.2 General notes.....	147
14.2.3 Structure of the FA system data structure.....	148
14.2.4 Controlling fast PLC tasks.....	148
14.2.5 Setting up a transfer channel.....	149
14.2.6 Specifying data to be transmitted.....	151
14.2.7 Starting the data transfer.....	154
14.2.8 Monitoring the data transfer.....	155
14.2.9 Processing times and data volumes to be transmitted of the fast PLC tasks.....	158
14.2.10 Processing times and error counter of the fast NC task.....	159
14.3 Commissioning.....	161
14.3.1 Creating fast PLC tasks and programs.....	161
14.3.2 Creating fast PLC programs.....	162
14.3.3 Configuring the NC FA functionality.....	165
14.3.4 Impact of the PLC on the FA functionality.....	167
15 Real-time data.....	169
15.1 Real-time coordinates.....	169
15.1.1 Description.....	169
15.1.2 Handling instruction: Applies workpiece coordinates in the interpolation cycle.....	170
15.1.3 Handling instruction: Applies basic workpiece coordinates in the interpolation cycle.....	170
15.1.4 Handling instruction: Applies machine coordinates in the interpolation cycle.....	171
15.1.5 Handling instruction: Applies axis coordinates in the interpolation cycle.....	171
15.2 Programmed path velocity.....	171
15.2.1 Description.....	171
15.2.2 Handling instruction: Applies path velocities in an interpolation cycle.....	172
16 Program-controlled events.....	173
16.1 Position-dependent high-speed output.....	173
16.2 Programmable distance-dependent signal.....	173
16.2.1 Description.....	173
16.2.2 Applying.....	179
16.2.3 Activating/deactivating.....	180

	Page
17 Service and support.....	181
Glossary.....	183
Index.....	185

1 About this documentation

Editions of this documentation

Edition	Release date	Note
01	2014-05	First edition
02	2015-06	Modifications implemented
03	2015-09	New chapter "Area monitoring with system data"
04	2017-09	Chapter "Fast actions MTX fa" revised New chapter "Re-initialization of selected SD elements after NC restart"
05	2018-11	Corrections in chapter "Operating functions (manual operation)" and "Fast actions MTX fa" Revision of chapter "Channel control" - "Retrace: Retracing on the contour"
06	2020-02	Chapter "Compensation of mechanical inaccuracies" revised Chapter "Fast actions MTX fa" revised Chapter "Program-controlled events" corrected Chapter "Compensation of mechanical inaccuracies" corrected, Defdb00204822 resolved
07	2020-10	Corrections in the chapter "Retrace: Retracing on the contour", Defdb00208992 resolved

Tab. 1-1: Change record

1.1 Validity of the documentation

Overview on target groups and product phases

In the following illustration, the framed activities, product phases and target groups refer to the present documentation.

The target group "Programmer" can "parameterize", "program, configure and simulate in the product phase Engineering" using this documentation.

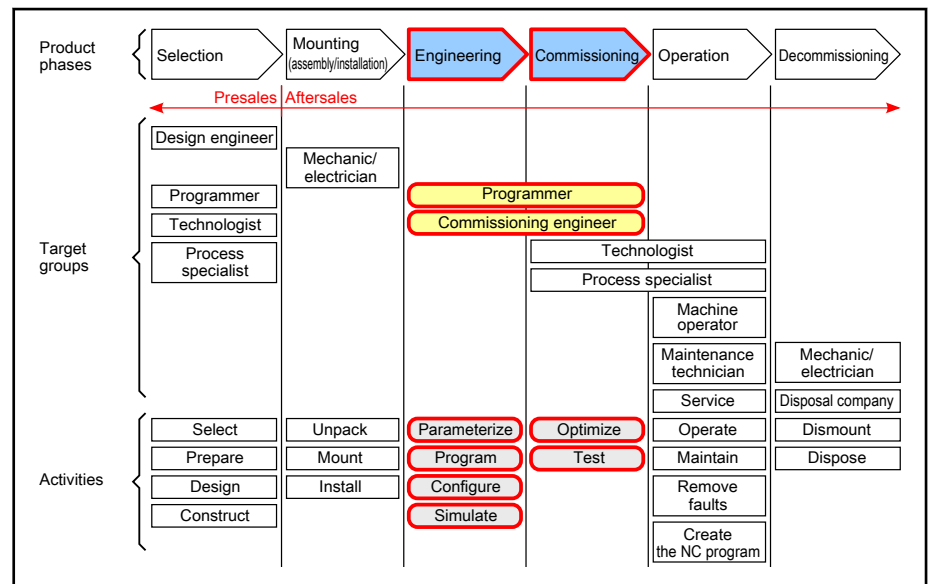


Fig. 1-1: Assigning this documentation to the target groups, product phases and target group activities

1.2 General information

The present manual supports you during the application/activation of various control functions.

It focuses on the procedures required for the parameterization.

This requires that the hardware of CNC, drives, PLC and voltage supply units are already integrated and checked according to your application documentation including wiring and accuracy. This refers especially to all safety-relevant features such as E-STOP circuits, release contacts, limit switches.

Ideally, the software of all Sercos drives is already commissioned. However, this is not imperative, since the control is able to configure any drives connected at startup via a parameter download. Thus, the fitting initialization files have to be available on the control.

To adjust the initialization files, have the corresponding documents available (such as lists with the necessary Sercos parameter settings for all drives and drive documentation).

For a better overview, all functions are assigned to different superordinate topics (see "Table of Content").

For each function, the following information is provided:

- Purpose or area of application
- Prerequisites/restrictions on use
- Interference options.

Numerous functions affect several domains at the same time (machine parameters, NC functions for programming, interface signals, etc.). Thus, many functions can also be described completely or partially in other manuals of this product.



Refer to the respective manual.

The following manuals supplement this manual:

- Machine parameters: "MTX Machine Parameters"
- NC and CPL functions: "MTX Programming Manual"
- Interface signals: "MTX PLC Interface"

Furthermore, have the drive documentation at hand.

You require good knowledge of the

- standard user interface of the control
- Windows user interface of the PC control panel
- tools for the machine parameter configuration.

These tools can modify the control configuration. Therefore, you have to be familiar with all required tools!

⚠ CAUTION

Due to an incorrect or unintended use caused by insufficiently trained or untrained staff, serious damage to the machine, loss of data and even personal injury can result!

Thus, only qualified staff that is sufficiently trained may start and operate the system and set or modify configuration parameters!

The personnel also has to be able to detect the risks resulting from changing parameters and the inherent general risks of mechanical, electrical or electronic equipment.

Incorrect or unintended use by insufficiently trained or untrained staff can cause serious damage to the equipment! Bosch Rexroth is not liable for any of those damages and consequential damages caused.

1.3 Required and supplementing documentation MTX

1.3.1 Selection/compilation

Documentation titles with type codes and part numbers

MTX 14VRS System Description DOK-MTX***-SYS*DES*V14-PRRS-EN-P, R911342640 This documentation describes the MTX control. It includes the designs, technical data, interfaces as well as the configuration of the control components.
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 MTX.

Tab. 1-2: MTX documentation overview - Selecting/composing

1.3.2 Configuration

Documentation titles with type codes and part numbers

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

MTX 14VRS Functional Description - Special Functions

DOK-MTX***-NC*F*SP*V14-RERS-EN-P, R911342624

This documentation describes the special functions of the MTX. The basic commissioning steps and the control functions are provided as description and handling instruction.

MTX Free Form Surface Milling

DOK-MTX***-FREEFORM***-APRS-EN-P, R911341435

This documentation describes the free form surface milling process with the MTX control. CNC programs generated by a CAD/CAM system are used as basis for the entire process. Overview over the following topics: Description of the MTX or CNC function for freeform surface milling, strategy during the NC parameterization process, boundary conditions when generating CNC programs with CAM software.

MTX Conversion of MTX Projects

DOK-MTX***-PROCONV****-PRRS-EN-P, R911342484

This documentation provides support during the conversion of MTX 1.x projects to MTX 2G. This project planning manual provides information on how to convert a project and identifies potential difficulties during the conversion.

Tab. 1-3: MTX documentation overview - Configuring

1.3.3 Commissioning

Documentation titles with type codes and part numbers

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.

MTX 14VRS Commissioning

DOK-MTX***-STARTUP*V14-CORS-EN-P, R911342620

This documentation describes the commissioning of the 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 Basis 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 the field bus and local periphery connections supported by the XLC, MLC and MTX systems. The focus of this documentation lies in the configuration, parameterization, commissioning and diagnostics of different periphery connections. It 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" supports the user in achieving the best results 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 engineering tools are integrated. It includes instructions on how to work with IndraWorks and how to operate the oscilloscope function.

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. The documentation 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

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.
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.
MTX 14VRS Programming Manual DOK-MTX***-NC**PRO*V14-RERS-EN-P, R911342634 This documentation describes the standard programming of the 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.
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 MTX. It includes the operation of the interface, the NC program development as well as the tool management.
MTX 14VRS Multitouch DOK-MTX***-MULTI***V14-APRS-EN-P, R911384735 The Multitouch user interface of the NC control MTX is described in this documentation.
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 the MTX control.
MTX 14VRS NC Simulation DOK-MTX***-NC*SIM**V14-APRS-EN-P, R911379629 This documentation describes the NC simulation for the MTX control.

xx Corresponding edition
Tab. 1-5: MTX documentation overview - Operating

1.3.5 OEM Engineering

Documentation titles with type codes and part numbers

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 interface of the Automation Interface. Different objects including code examples are described. The Automation Builder is also described in this manual.
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.
IndraWorks OPC UA Communication DOK-IWORKS-OPC*UA*****-APRS-EN-P, R911379309 This documentation describes the OPC UA communication of the MLC and MTX control systems.

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

1.3.6 AddOns

Documentation titles with type codes and part numbers

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.
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 MTX cta and MTX ega.
MTX Remote Condition Monitoring DOK-MTX***-RCM*****V01-APRS-EN-P, R911334383 This documentation describes the operation of the Remote Condition Monitoring System.
MTX visIREC User Documentation DOK-MTX***-VISIREC*V01-APRS-EN-P, R911344242 This documentation describes the analysis tool visIREC used to optimize the free form surface milling process. 2D or 4D display of path-related data. 2D or 4D display of coordinate-related data. Analysis of critical areas (path and orientation deviation). Comparing the programmed NC blocks to the 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 signal graphics (warning triangle with exclamation mark), added in front of the signal words Danger, Warning and Caution refers to hazards to 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 **could** occur.

⚠ CAUTION

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

NOTICE

In case of non-compliance with this safety instruction, material or property damage may 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
IWE	IndraWorks Engineering
IWO	IndraWorks Operation
OWG	Optical waveguide
NC	Numerical Control
OEM	Original Equipment Manufacturer
Profibus	Communication connection
Sercos	Communication connection
MTB	Machine Tool Builder

Tab. 1-8: Terms and abbreviations

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 Important instructions on use

2.1 Intended use

2.1.1 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 handling 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. It is not permitted to decompile software products or alter source codes.
- Do not install damaged or defective products or use them in operation.
- 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

2.1.2 Areas of use and application

The 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 documents and functional descriptions.

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

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

2.2 Unintended use

Using the CNC system MTX outside the above-referenced areas of application or under operating conditions other than the conditions described in the documentation and the specified technical data, is defined as "unintended use".

The CNC system 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.
- Furthermore, the CNC system MTX shall not be used for applications not expressly approved of by Bosch Rexroth. Therefore, please read the information given in the general safety instructions!
- The CNC system 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.

3 Drives (axes, spindles)

3.1 Delayed error reaction in case of non-fatal drive errors

3.1.1 Description

Function To avoid machine damages, it is required for some applications that the NC controls the drives for a short period and performs a coordinate standstill of the machine axes when drive errors occur.

This is possible due to this function for "non-fatal drive errors" (diagnostics F2xxx).

The prerequisite is that the "NC error reaction" function is enabled in the drive.

*If the delayed error reaction is enabled and a **non-fatal** drive error occurs:*

- The NC sets the signal iAx_DrvErrClass1 or iSp_DrvErrClass1.
- The NC sets additionally the signal iAx_DelayErrClass1 or iSp_DelayErrClass1.
- The NC sets the control and drive error 1303 "Sercos error..."
- 20 s after the error occurred, the NC internally sets a feed inhibit for the axis or spindle. This period can be shortened by setting qAx_RedDelayErrClass1 or iSp_RedDelayErrClass1.

If there is another error:

- Refer to the "standard error reaction of the NC".

If the delayed error reaction is disabled (standard error reaction of the NC):

- The NC sets the signal iAx_DrvErrClass1 or iSp_DrvErrClass1.
- *The NC internally sets a feed inhibit for the axis or spindle:*
 - In case of non-fatal errors with configured NC error reaction, the NC decelerates the axis or the spindle.
 - In case of other errors or an unconfigured NC error reaction, the drive sets itself to standstill.
- The NC sets the control and drive error 1303 "Sercos error..." with regard to the channel.

Setting Sercos parameters and machine parameters applies the function. The NC function DERDelay can affect the reaction of drives labeled as "programmable".

DERDelay(1)	Activates the delayed error reaction for all drives labeled as "programmable" in "MAIN/Dr[i]/DelayedError-Reaction" (MP 1001 00012).
DERDelay() or DERDelay(0)	Deactivates the delayed error reaction for all drives labeled as "programmable" in "MAIN/Dr[i]/DelayedError-Reaction" (MP 1001 00012).
DERDelay({ON(<DriveList>)}, {OFF(<DriveList>)})	
	Activates/deactivates the delayed error reaction for the programmed drives.

with	
<Drive list>	List of system drive numbers or system axis names separated by commas.

Tab. 3-1: *DERDelay syntax*

If the function "Delayed NC error reaction for non-fatal drive errors" is active, there has to be a suitable mechanism to perform a coordinated standstill of the drives.

Restrictions

- Only for drives of the type IndraDrive.
- When approaching the reference point, the standard error reaction of the NC is performed.
- The NC does not affect the error reaction of the drive.
If a non-fatal error follows an error of another error class, the drive sets itself to standstill.
- The error message output 1303 "Sercos error..." is shortened for the MTX micro.

Example:

Instead of "Sercos error x-axis ## 01 F2019 Motor overtemperature switch-off", only "Sercos error x-axis ## 01 F2019." is output.

Relevant machine parameters (MP)

MAIN/Dr[i]/DelayedErrorReaction

- Enables the delayed error reaction in case of non-fatal drive errors (F2xxx).
- Allows to enable the NC function "DERDelay".

Relevant Sercos parameters

S-0-1050.1.6	Configuration list AT S-0-390 has to be entered.
S-0-0095	Diagnostics Complete diagnostic message (e.g. "F2022 device temperature monitoring defective").
S-0-0390	Diagnostic number Diagnostic number in hexadecimal representation (e.g. F2022(hex)).
P-0-0117	Enables the control reaction in case of error (only IndraDrive) Bit 0 enables the NC error reaction.

Tab. 3-2: *Relevant Sercos parameters***Relevant IF signals**

iAx_DrvErrClass1	Error of state class 1 (axis)
iAx_DelayErrClass1	Delayed error reaction (axis)
qAx_RedDelayErr-Class1	Shorten delayed error reaction (axis)
iSp_DrvErrClass1	Error of state class 1 (spindle)
iSp_DelayErr-Class1	Delayed error reaction (spindle)
qSp_RedDelayErr-Class1	Shorten delayed error reaction (spindle)

Tab. 3-3: *Relevant IF signals*

Example: A coordinated retract motion is to be allowed in a certain machining section if a non-fatal error occurs.

Preparation:

1. Set the relevant drives in the machine parameter to "programmable".
2. Create an asynchronous subroutine for the retract motion.
3. Link the signals iAx_DelayErrClass1 (and iSp_DelayErrClass1) in the PLC to the triggering of the asynchronous subroutine.

Runtime:

1. Enable the delayed reaction at the beginning of the machining section by calling DERDelay(1).
2. Disable the delayed reaction at the end of the machining section by calling DERDelay(0).

3.1.2 Applying

Sercos parameters

Set the following Sercos parameters for all IndraDrive drives to be operated with a delayed error reaction:

P-0-0117, Enabling the control reaction in the event of an error	Set bit 0 to 1
S-0-1050.0.6, Config. list of the drive telegram	Enter S-0-0390

Machine parameters

- Set the machine parameter MAIN/Dr[i]/DelayedErrorReaction (1001 00012) to 2 (On) for all IndraDrive drives to be operated with a delayed error reaction by default.
- Set the machine parameter MAIN/Dr[i]/DelayedErrorReaction (1001 00012) to 1 (programmable) for all IndraDrive drives to be operated optionally with a delayed error reaction.

PLC

- Link the signals iAx_DelayErrClass1 and iSp_DelayErrClass1 additionally to iAx_DrvErrClass1 and iSp_DrvErrClass1 in the PLC program.
- A suitable reaction to set the drives to standstill has to be implemented.
- After the error reaction has been completed or if no error reaction is to be performed, the waiting time up to the NC-controlled standstill of the drive can be reduced by setting qSp_RedDelayErrClass1 and qAx_RedDelayErrClass1.

3.1.3 Activating

- The function is directly active for all drives labeled with the value 2 (On) in MAIN/Dr[i]/DelayedErrorReaction.
- The function can be enabled/disabled with the NC function "DERDelay" for all drives labeled with 1 (programmable) in MAIN/Dr[i]/DelayedErrorReaction .

3.1.4 Deactivating

Return to the machine parameter setting:

- Call the NC function "DERDelay(0)".
- Perform system reset.

Disabling completely:

- For all drives disabling the delayed error reaction in MAIN/Dr[i]/DelayedErrorReaction.

3.2 Sercos Gateway

3.2.1 Description

General information

The Sercos gateway allows a distribution of cyclic Sercos data of a Sercos axis (source axis) to one or several Sercos axes (target axes). The source data can either be position command values generated from the NC for the source axis or Sercos data read from the exchange telegram of the source axis. The data target is a cyclic Sercos parameter that has to be in the master data telegram of the target axis. The gateway can consist of several routes. Each drive can contain up to five different source data in five different ident numbers. Thus, multiple routes can be established between the individual axes and drives.

Application for the gateway is the sending of the NC command value of an (also virtual) axis to the drive of another axis. Additionally, a delayed cross communication between different Sercos II drives can be carried out by copying the content of a Sercos ident number in the drive telegram to an ID in a master data telegram.

Relevant system data

Defining a route

/SysScsGateway/SysAxIdRoute			
Name	Type	Description	Write access
SrcScsId	Int_t	Sercos ID of the source Not relevant, if the source of the NC is-command value	All
DstScsId	Int_t	Sercos ID of the target	All
SrcSysDrNmb	Int_t	Source axis number 0: Do not activate route	All
SrcTelegtype	Byte_t	Telegram type of the source: 0: Sercos drive telegram 1: Sercos master data telegram Not relevant, if the source of the NC is-command value	All
SrcAttrib	Byte_t	ID type of the source: 0: S-parameter 1: P-parameter 99: NC command value	All
DstAttrib	Byte_t	ID type of the target 0: S-parameter 1: P-parameter	All
InUse	Byte_t	1: Route in active	NC kernel
ConfErr	Int_t	Configuration error	NC kernel
Suppress	Byte_t	1: Suppress route	All

Tab. 3-4: Relevant system data "SysReentry"

3.2.2 Applying

The system data "SysScsGateway" is defined in FEPR0M.

The gateway is applied by declaring the system data "SysScsGateway". The user has to insert the definition in the SDDefMTB.xml file and to pre-assign the system data. The SDDatMTB.xml can be used for the pre-assigning.

If the declaration exists but has not been pre-assigned, the gateway is inactive.

3.2.3 Activating

The gateway is activated by entering at least one valid route and a subsequent system startup

Program:

<SysScsGateway Dimension="4">	Target axis is 4th axis
<SysAxIdRoute>	
<SrcScsId>51</SrcScsId>	Source ID
<DstScsId>1375</DstScsId>	Target ID
<SrcSysDrNmb>1</SrcSysDrNmb>	Source axis no.:
<SrcTelegtype>0</SrcTelegtype>	Source is AT
<SrcAttrib>0</SrcAttrib>	Source SERCOS-ID type
<DstAttrib>1</DstAttrib>	Target SERCOS-ID type
<InUse>0</InUse>	
<ConfErr>0</ConfErr>	
<Suppress>0</Suppress>	
</SysAxIdRoute>	
<SysAxIdRoute>	
<SrcScsId>0</SrcScsId>	Source ID
<DstScsId>1376</DstScsId>	Target ID
<SrcSysDrNmb>1</SrcSysDrNmb>	Source axis no.:
<SrcTelegtype>0</SrcTelegtype>	Source is AT
<SrcAttrib>99</SrcAttrib>	Source NC command value
<DstAttrib>1</DstAttrib>	Target SERCOS-ID type
<InUse>0</InUse>	
<ConfErr>0</ConfErr>	
<Suppress>0</Suppress>	
</SysAxIdRoute>	

3.2.4 Deactivating

The gateway is disabled by deleting the entry in SDDefMTB.xml. A single route can be disabled by the entry SrcSysDrNmb = 0. A system startup is required.

3.3 Synchronous axes (channel axes)

Synchronous axes are linear or rotary axes assigned to a channel. During control startup, assignment takes place by the machine parameter "MP 1003 00002: Channel assignment" or at runtime by an axis transfer (WAX, GAX). In the channel, up to eight axes can participate in a path motion in a coordinate manner. An NC block with a path motion is only completed, when all synchronous axes have reached their end point with the specified accuracy. The modal functions set in the channel always apply to the synchronous axes.

3.4 Asynchronous axes (auxiliary axes)

Asynchronous axes are auxiliary axes that are not assigned to any channel. Synchronous axes can become asynchronous due to axis enabled (RAX). A spindle/C-axis is also asynchronous after switching to the C-axis mode (STA). Asynchronous axes are started with the programming in an NC block. Processing in the channel is continued without waiting for the end of the motion. Asynchronous axis motions can only be canceled by system reset or axis reset. An active positioning motion blocks the axis for new specifications. In

case of an active velocity interpolator (e.g. C=S(1000)), new specifications are possible. The modal functions such as rapid traverse/feed (G0/G1), absolute dimension/ incremental (G90/G91) or the type of feed (G93/G94/G95) with the related F-word have no importance for asynchronous axes. With regard to the motion, the axis parameters as well as special details on the feed (FA), jerk limitation (ASO) and the programming attributes (IC, ACN, ACP, INCH und S) are applicable.

3.5 Handling instruction: Axis reset

Activating With "axis reset", an asynchronous axis can be reset using the signal **qAX_Reset** "Axis reset".

IW Engineering/IndraLogic: Triggering axis reset via interface signal

- Setting axis signal **qAx_Reset** "Axis reset" in the NC/PLC interface.
-
- 
 - Setting the axis-specific interface signal **qAx_Reset** "Axis reset" cancels the current motion of an asynchronous axis. The axis is decelerated up to standstill. Moreover, error states are reset.
 - The interface signal **iAx_Reset** "Axis reset" shows that the asynchronous axis was reset and that the asynchronous axis is ready again to accept new motion specifications.

	Documentation	Instruction
Instruction:		Editing PLC signal
Documentation:	MTX PLC Interface	Triggering axis reset

4 Operating functions (manual operation)

4.1 Online correction with handwheel (in workpiece coordinates)

4.1.1 Description

Function

With an active or inactive part program, the "online correction" is used to

- correct a **position** or **orientation** in a workpiece coordinates system (WCS) of a channel using the handwheel.
- the **position of the tool length axis** in the tool coordinate system (TCS) (in TCS-Z-direction of the tool) is traveled (caution: no tool correction!).



An online correction cannot be performed in the "manual setup" (jog mode), "setup approach reference point" and "setup, jogging in workpiece coordinates" modes.

CAUTION

Personal injury and damage to property by tool and workpiece breakage!

Danger due to a collision with the workpiece during online correction!

Proceed very carefully with the online correction, especially when the program is active and the axes are moving!

If a correction is made in the direction of the tool, there may be no orientation motions in the part program!

Once the correction is retracted via handwheel, it remains until it is moved out again. Thus, the correction in an active program also affects the following programmed positions. In this case, the programmed positions are displaced by the correction.

Example:

Online correction

The position of a tool is corrected online via the handwheel in TCS-Z-direction. In the next block, a position in X, Y is approached. In this block, the tip of the tool is displaced by the online correction in the longitudinal axis of the tool.

Permitted coordinates

The online correction of the handwheel can sporadically only be selected for a subset of the (x, y, z, phi, theta) coordinates in case of an active axis transformation. The subsequent table provides an overview for frequently used axis transformations. A warning is output in case of invalid coordinates.

Transformation / MP 1030 00110	Axes (Pseudo coordinates)	WCS Linear coordinates	WCS Orientation coordinates	TCS Coordinates
None	All	-	-	-
Face transformation / 2011001	All	-	-	-
Cylinder jacket transformation (handling) / 3021001	All	x, y, z,	-	-

Operating functions (manual operation)

Transformation / MP 1030 00110	Axes (Pseudo coordinates)	WCS Linear coordinates	WCS Orientation coordinates	TCS Coordinates
Cylinder jacket transformation (projection) / 3021002	All	z	-	-
5-axis transformation with rotary axis programming (LLLRR) / 3032001	All	x, y, z,	-	Z-direction
5-axis transformation with linear orientation (LLLRR) / 3232101	All	x, y, z,	phi, theta	Z-direction
5-axis transformation with vector orientation (LLLRR) / 3232201	All	x, y, z,	-	Z-direction
5-axis transformation with vector orientation (RLLLR) / 3232202	All	x, y, z,	-	Z-direction
5-axis transformation with vector orientation (RRLLL) / 3232203	All	x, y, z,	-	Z-direction
Cardanic 5-axis transformation / 3232211	All	x, y, z,	-	Z-direction

Tab. 4-1: Online correction

Special features

The active correction values of a retracted online correction are transformed when enabling, disabling or changing a(n)

- axis transformation
- coordinate transformation (inclined plane)

Any connection that is still established between coordinate/axis and handwheel is canceled with the result that the online correction is deselected.

At axis transfer, the correction memory is deleted and the online correction is deselected if the specified axis is connected to the handwheel.

The online correction memory of a recently accepted axis is always empty.

Application

For a 5-axis transformation, the online correction can for example be used to remove a damaged drilling tool stuck in the workpiece at an angle by the handwheel in TCS-z-direction. To continue from the same position with a new tool, the old online correction is "reset" via the handwheel until the new tool has the old TCS-z-value (set to zero before to consider the different tool length corrections).

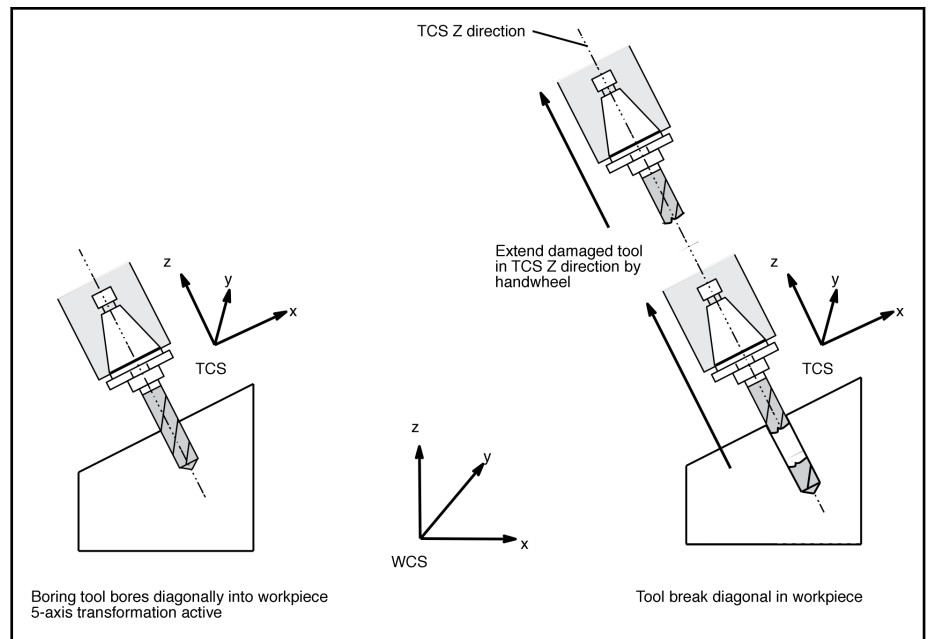


Fig. 4-1: Online correction

Restrictions

- An online correction cannot be performed in the operating modes "Manual setup" (jog mode), "setup approach reference point" and "setup, jogging in workpiece coordinates" modes.
- Machine-fixed position:
G76 approaches an offset position, i.e. the online correction is not counted back.
- The CPL functions "PPOS" und "PCSPROBE/CPROBE" do not consider the correction value of the online correction.
- Touch probe:
G75 measures the correct position.
Use the CPL function "PROBE" to read out this measured value.
- Measurement at fixed stop:
FsProbe measures the correct position.
Use the CPL function "PROBE" to read out this measured value.
- Limit switch:
A coordinate position - generated by the online correction - is **not** checked for a possible exceedance of the software limit switches in the NC.
Therefore, enable the "limit switch check" in the Sercos drive.

Relevant NC functions

HWOC(...)	Enabling online correction
HWOCDIS(...)	Disabling online correction
HWOC(0)	Disabling and deleting online correction

Tab. 4-2: Relevant NC functions

Relevant CPL functions

MCODS(89) and MCODES(90)	see "MTX Programming manual".
--------------------------	-------------------------------

Tab. 4-3: Relevant CPL functions

Relevant IF signals

At the channel interface	
qCh_OnlCorrWcs	Online correction enabled
qCh_OnlCorrWcsDir	Online correction direction (direction of rotation)

Tab. 4-4: Relevant IF signals

Relevant machine parameters

7050 00900	Online correction in workpiece coordinates (WCS) available.
7050 00910	Online correction WCS: Sensor selection, device group
7050 00910	Online correction WCS: Sensor selection, device number
7050 00914	Online correction WCS: Sensor selection, identification
7050 0920	Online correction WCS: Sensor increments → mm factor
7050 0921	Online correction WCS: Sensor increments → mm divisor
7050 00926	Online correction WCS: Incremental step, system axis number
7050 00940	Online correction WCS: Max. velocity in mm/min
7050 00941	Online correction WCS: Max. acceleration in m/s ²
With connection of the handwheel via Profibus-DP:	
9060 00001	Number of increments per revolution

Tab. 4-5: Relevant machine parameters

4.1.2 Handling instruction: Online correction with handwheel in tool coordinates

Applying: Using the **Online correction**, a position or orientation is corrected online in the workpiece coordinate system (WCS) of a channel using the handwheel irrespective of whether the part program is active or inactive, or the position of the tool length axis in the tool coordinate system (TCS) is traveled (in TCS-z-direction of the tool) (Attention: no online correction!).

IW Engineering/configuration: Editing parameters

- The online correction function has to be activated using the parameter **onl** "Online correction" in the setup (SUP).

- **EnablWcsCorr** "Activate online correction" (7050 00900)
- Right click on the node "Device sensor selection" under the node **WcsOnlCorr**
- "Change to"
- Selecting the function:
HandWh "handwheel" (7050 00910)

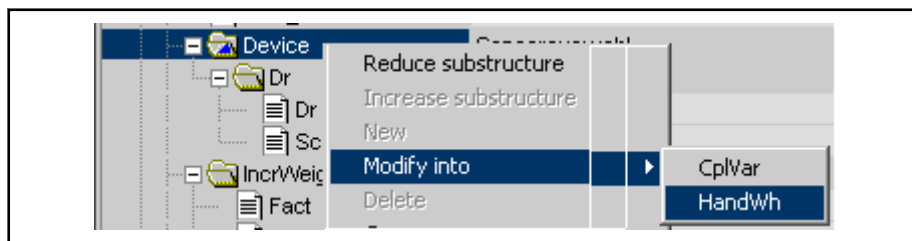


Fig. 4-2: Sensor/device selection

- **HandwWcsCorr**
"Handwheel" (7050 00912):
Selecting the desired handwheel
- **FactWcsCorr**
"Multiplier" (7050 00920)
Sensor increments → mm factor
- **DivWcsCorr**
"Divisor" (7050 00921):
Sensor increments → mm divisor
- **DrIndIncrSource**
"System axis number for increment specification" (7050 00926)
- **MaxVelWcsCorr**
"Maximum velocity" (7050 00940)
- **MaxAccWcsCorr**
"Maximum acceleration" (7050 00941)



With connection of the handwheel via Profibus-DP, the parameter **IncrPerRev** "Number of increments/revolution" (9060 00001) must be adjusted.

	Documentation	Instruction
Instruction:		Editing machine parameters
Documentation:	MTX Machine Parameters	Online correction with hand-wheel

IW Operation/IndraLogic: Linking interface signals

- **qCh_OnlCorrWcs** "Online correction enabled"
- **qCh_OnlCorrWcsDir** "direction of online correction"
1: A positive direction of rotation of the handwheel results in a negative online correction.
0: A positive direction of rotation of the handwheel results in a positive online correction.

Operating functions (manual operation)

	Documentation	Instruction
Instruction:		Editing machine parameters
Documentation:	MTX Machine Parameters	Online correction with hand-wheel

IW Operation/program: Adjust Sercos parameters if necessary

The relevant Sercos parameters have to be adjusted if the handwheel is directly connected to the drive.

	Documentation	Instruction
Instruction:		Editing machine parameters
Documentation:	MTX Machine Parameters	Online correction with hand-wheel

Activating: *IW Operation/NC programming: Activating online correction with handwheel*

- Program **HWOC(...)**: to enable the online correction.
- **qCh_OnlCorrWcs** "Online correction enabled":
Set interface signal to enable the function
- Triggering the traversing motion by rotating the handwheel.



If **G70 (Inch)** is active, the handwheel increments are to be multiplied by the factor 2.54 (0.001 mm → 0.0001 inch) for linear coordinates.

	Documentation	Instruction
Documentation:	MTX Programming Manual	Activating online correction with handwheel

Deactivating: **IW Operation/NC programming: Deactivating online correction with hand-wheel**

There are different ways to deactivate the online correction:

- **Interrupt:**
Removing the **qCh_OnlCorrWcs** "online correction enabled" interface signal.
- **Switch-off:**
The **HWOCDIS(...)** NC function switches off the online correction.
- **Delete:**
The **HWOC(0)** NC function switches off the online correction and deletes the correction memory of this channel.
- **Reset**



Does not directly affect the online correction.

*An online correction is switched off only if one of the following is performed in the **ChResetState** "State of channel reset" (7060 00020):*

- Axis transfer
- Switching to the coordinate transformation
- Switching the axis transformation

The content of the correction memory is also retained after control reset.

Online correction values are only deleted if **HWOC()** is entered into the **ChResetState** "State after channel reset" (7060 00020).

	Documentation	Instruction
Documentation:	MTX Programming Manual	Deactivating online correction with handwheel

5 Compensating mechanical inaccuracies

5.1 Overview

Including:

- Lead screw error compensation (LSEC)
- Cross compensation (CCOMP)
- Straightness and angle error compensation
- Backlash compensation
- Circle compensation at the quadrant transition
- Volumetric compensation

5.2 Lead screw error compensation (LSEC)

5.2.1 Description

Function

Compensates linearity errors at an axis occurring in the following areas:

- Axis measuring system
- Mechanical motion conversion (motor axis)

This also includes the backlash of an axis.

Restrictions

- Together with the CCOMP (cross compensation), a maximum of 24,000 compensation values is possible.
- For rotary or endless axes (compare MP 1003 00004), only the compensation values on axis positions in the interval [0 .. 360] have to be determined. For axis positions outside of this interval, the control always transforms the actual position to the interval [0...360] before it computes the respective compensation value in this interval.



From MTX 14V18, the modulo calculation for a rotary axis (SysAxType = Rotary axis, EnablModCalc = No, S-0-0076 bit 7 = 0) can be deactivated by entering the startup parameter NOMODULO.

```
#STUP
```

```
NOMODULO
```

```
#ESTUP
```

```
<StartingPoint> <StepWidth>
```

```
<P0> <N0>
```

Commissioning the LSEC

1. Determining compensation values for all relevant axes
2. Activating the entered compensation values

5.2.2 Handling instruction: Lead screw error compensation (LSEC) - Determining compensation values

The lead screw error compensation compensates for linearity errors of an axis. This also includes the backlash of an axis.

Determining LSEC compensation values of an axis

- Deactivate LSEC (lead screw error compensation) and CCOMP (cross compensation) for the axis to be measured.
- Subdivide the traversing range to be compensated into a number of sections of equal size (reference value: ca. 10-20 mm). This grid defines at which positions of the traversing range the compensation values are to be determined.

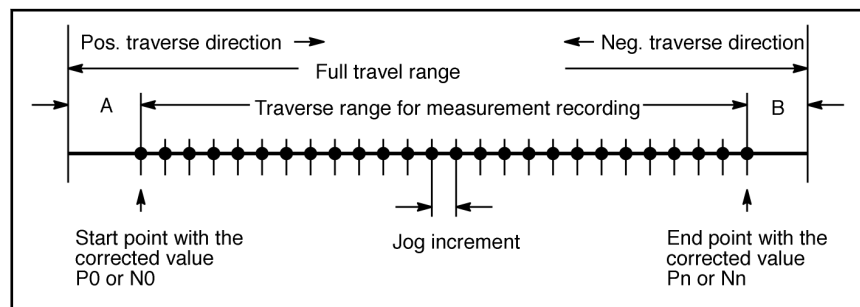


Fig. 5-1: Traversing range

Step width:

Distance between two subsequent measuring positions:

Value range (linear axis):

0.1 to 214748364.7 μm

Value range (rotary/endless axes):

0.1 to 359999.9*10⁻³ degrees

Starting point:

The smallest coordinate value for which a compensation is determined.

Value range (linear axes):

+/-214748364.7 μm

Value range (rotary axes):

0.0 to 359999.9*10⁻³ degrees

As of MTX 14V18, the modulo calculation for a rotary axis can be deactivated by entering the startup parameter `NOMODULO`. The value range is extended to

+/-214748364.7 degrees

Value range (endless axes):

0.0 to 359999.9*10⁻³ degrees

End point:

The largest coordinate value for which a compensation is determined. Value range as for **starting point**. The coordinate value of the **end point** has to be greater than the coordinate value of the **starting point**.

If the **starting point** or the **end point** is not identical with the start or the end of the total traversing range, the compensation value of the **starting point** or **end point** becomes effective in the section **A** or **B**.



For the two compensation procedures LSEC and CCOMP as well as all for axes, up 24,000 compensation values are permitted.

	Documentation	
Documentation:	MTX Functional Description	Lead Screw Error Compensation - Determining Compensation Values

Applying LSEC compensation data in positive traversing direction

- Traverse the desired axis to the beginning of the traversing range.
- Through an external measuring device (e.g. laser), determine the correction value $P[i]$ in μm (i = absolute number of the measuring position: 0 to n) for each measuring position in positive traversing direction.

The following applies:

- **Compensation value in μm = command position (NC) - actual position (machine)**
- All compensation values have to be determined in the "micrometer" unit with current a resolution of no more than $0.001 \mu\text{m}$ (1 nm) and have to be in a range of $\pm 2147483.647 \mu\text{m}$.
- After the last compensation value (end point) in positive traversing direction has been determined, traverse the axis to the end of the traversing range.

	Documentation	
Documentation:	MTX Functional Description	Lead screw error compensation - Compensation data in positive direction

Applying LSEC compensation data in negative traversing direction

- Determine the compensation value $N[i]$ in μm (i = absolute number of the measuring position: 0 to n) for each measuring position in negative traversing direction.

The following applies:

- **Compensation value in μm = command position (NC) - actual position (machine)**
- All compensation values have to be determined in the "micrometer" unit with current a resolution of no more than $0.001 \mu\text{m}$ (1 nm) and have to be in a range of $\pm 2147483.647 \mu\text{m}$.

	Documentation	
Documentation:	MTX Functional Description	Lead screw error compensation - Compensation data in negative direction

Structuring a LSEC compensation data file

- The LSEC compensation data file name is determined as follows:
lsecc<AxisNumber>.tab
 (e.g. lsecc08.tab is the name of the compensation table of the eight system axis)
- Structure of a LSEC file:

Compensating mechanical inaccuracies

```

<StartingPoint in µm> <StepWidth in µm> <Comment>
<P0> <N0> <Comment>
<P1> <N1> <Comment>
<P2> <N2> <Comment>
<P3> <N3> <Comment>
<P4> <N4> <Comment>
:
<Pn> <Nn> <Comment>

<P[i]> in µm = <Starting point> + i * <Step width> ; Compensation value
in positive traversing direction
<N[i]> in µm = <Starting point> + i * <Step width> ; Compensation value
in negative traversing direction

```



- At least one space has to be programmed between the individual entries within a line.
- Complete each line with <Enter>.

*Example:***LSEC table**

The traversing range of the x-axis (0 to 90 mm) is subdivided into a grid with the step width 10000 (10 mm).

As a starting point of the LSEC compensation, the absolute axis position of 20000 µm (20 mm) was selected.

For positive and negative traversing direction, seven axis positions were measured each. For axis positions greater than 20 mm, the compensation values of the starting point become effective. For axis positions greater than 80 mm, the compensation values of the end point become effective.

In the system, the x-axis is defined as first axis. Thus, LSEC data for this axis has to be entered into the "lsecc01.tab" file.

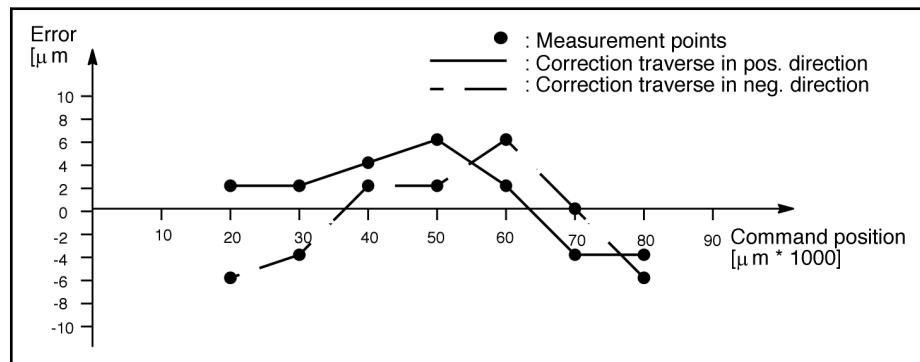


Fig. 5-2: Programming example - Entering compensation values

Content of the file "lsecc01.tab":

```

20000.0 10000.0 (starting point 20 mm; step width 10 mm;
7 measurements)
2.0 -6.0 (position 20 mm)
2.0 -4.0 (position 30 mm)
4.0 2.0 (position 40 mm)
6.0 2.0 (position 50 mm)

```

2.0 6.0 (position 60 mm)
 -4.0 0.0 (position 70 mm)
 -4.0 -6.0 (position 80 mm)

	Documentation	
Documentation:	MTX Functional Description	Lead screw error compensation - Structuring a compensation data file

5.2.3 Handling instruction: Lead screw error compensation (LSEC), activating

IW Operation/program: Read/re-activate LSEC files

- For the axis to be compensated, the compensation values of the LSEC are available in the **lsecc<axis number>.tab** file.
- Copy or read in the required LSEC files into the root directory ("/") or into the user FEPROM directory ("/usrfep").

- or -

- Activate existing, deactivated file again:
 Rename the relevant LSEC files in the root directory ("/") or in the user FEPROM directory ("/usrfep"). Therefore, change the file name extension from **"off"** to **"tab"** (e.g. lsecc03.off to lsecc03.tab).
- Trigger NC restart.



If the root directory and the user FEPROM contain files of the same name, the NC uses the file from the root directory.

	Documentation	Instruction
Documentation:	MTX Functional Description	Activating lead screw error compensation

5.2.4 Handling instruction: Lead Screw Error Compensation (LSEC), Deactivating

IW Operation/program: Deactivating LSEC files

- Removing the LSEC file from the root directory ("/") or from the user FEPROM directory ("/usrfep").

- or -

- Rename the relevant LSEC files in the root directory ("/") or in the user FEPROM directory ("/usrfep"). Therefore, change the file name extension from **"tab"** to **"off"** (e.g. lsecc03.tab to lsecc03.off).
- Trigger NC restart.

	Documentation	
Documentation:	MTX Functional Description	Deactivating lead screw error compensation

Tab. 5-1: *IW Operation/program: Deactivating LSEC files*

5.3 Cross compensation (CCOMP)

5.3.1 Description

Function

Compensates the geometric deviation between theoretical and actual position of an axis if this deviation depends on the actual positions of up to two reference axes.

In the following, the axis to be compensated is also referred to as "target axis".

Example:

Portal-type milling machine

The of the z-axis is to be compensated with regard to the position of the y-axis.

⇒ Target axis (axis to be compensated): Z

⇒ Reference axis: Y

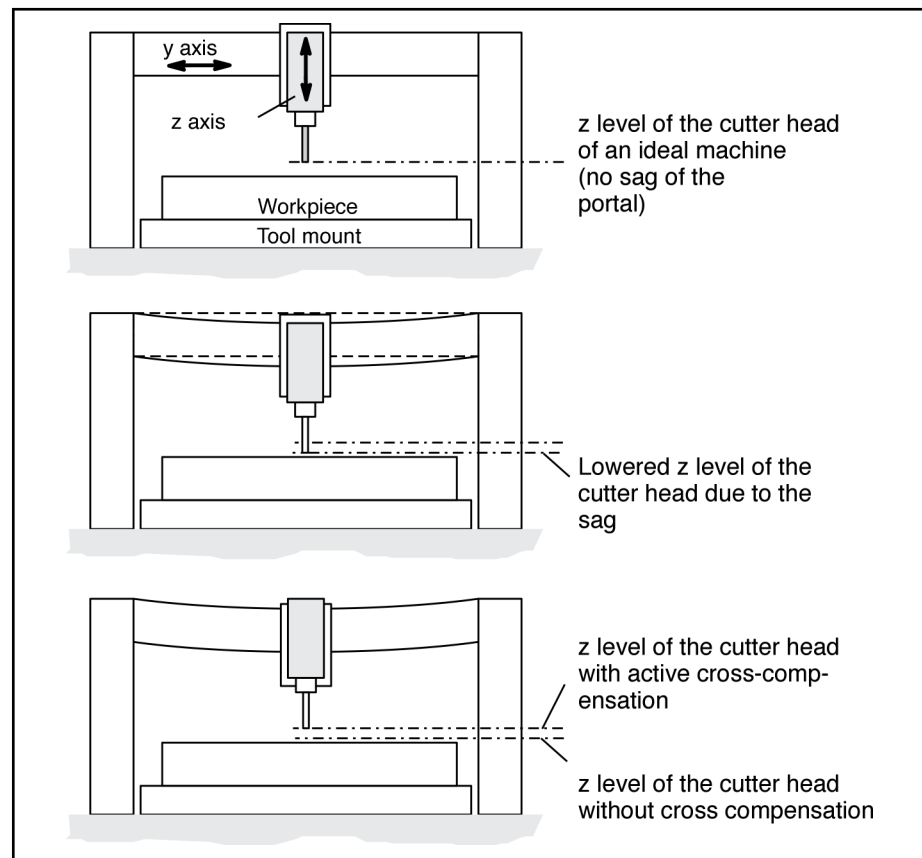


Fig. 5-3: Example of cross compensation

Restrictions

- Together with the LSEC (lead screw error compensation), there can be up to 24,000 compensation values.
- For rotary or endless axes (compare MP 1003 00004), only the compensation values on axis positions in the interval [0 .. 360] have to be determined. For axis positions outside of this interval, the control always

transforms the actual position to the interval [0...360] before it computes the respective compensation value in this interval.



From MTX 14V18, the modulo calculation for a rotary axis (`SysAxType = Rotary axis`, `EnablModCalc = No`, `S-0-0076 bit 7 = 0`) can be deactivated by entering the startup parameter `NOMODULO`.

`#STUP`

`NOMODULO`

`#ESTUP`

`<StartingPoint> <StepWidth>`

`<P0> <N0>`

- For one target axis, a maximum of two reference axes can be considered

Commissioning CCOMP

To use the LSEC function, commission it first.

This ensures that the corresponding linearity errors of the reference axis are minimized when the CCOMP compensation values are determined.

1. Determining compensation values for all relevant axes
2. Activating the entered compensation values

5.3.2 Handling instruction: Cross compensation (CCOMP), determining the compensation values

The cross compensation compensates the geometric deviation between theoretical and actual position of an axis if this deviation depends on the actual positions of not more than two reference axes (target axis = axis to be compensated).

Determining CCOMP compensation values

1. Activate LSEC (lead screw error compensation) if required for the reference system(s) to be measured.
2. Deactivate CCOMP (cross compensation) for the reference system(s) to be measured.
3. Subdivide the traversing range of the reference axis/axes to be compensated into a number of sections of equal size (reference value: ca. 10-20 mm).

This grid defines at which positions of the traversing range the compensation values are to be determined. If two reference axes have to be used (positions of the target axis depend on two axes), the grids of the two reference axes can differ from each other.

Compensating mechanical inaccuracies

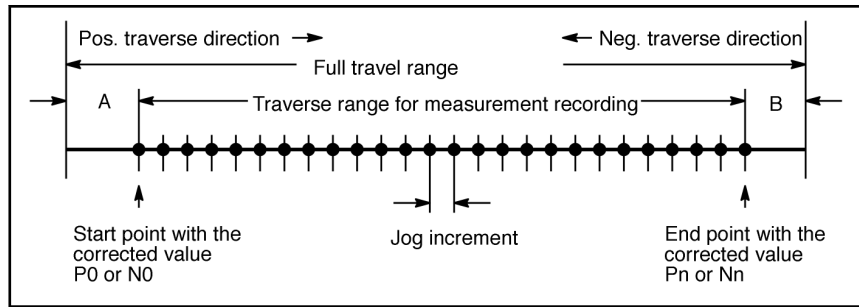


Fig. 5-4: Determining compensation values

Step width:

- Distance between two subsequent measuring positions:

Value range (linear axis):

0.1 to 214748364.7 μm

Value range (rotary/endless axes):

0.1 to 359999.9*10⁻³ degrees

Starting point:

- The smallest coordinate value for which a compensation is determined.

Value range (linear axes):

+/-214748364.7 μm

Value range (rotary axes):

0.0 to 359999.9*10⁻³ degrees

As of MTX 14V18, the modulo calculation for a rotary axis can be deactivated by entering the startup parameter `NOMODULO`. The value range is extended to

+/-214748364.7 degrees

Value range (endless axes):

0.0 to 359999.9*10⁻³ degrees

End point:

- The largest coordinates value for which a compensation is determined. Value range as for **starting point**. The coordinate value of the **end point** has to be greater than the coordinate value of the **starting point**.

If the **starting point** or the **end point** is not identical with the start or the end of the total traversing range, the compensation value of the **starting point** or **end point** becomes effective in section **A** or **B**.



For the two compensation operations CCOMP and LSEC and for all axes up to 24,000 compensation values are permitted.

	Documentation	Instruction
Documentation:	MTX Functional Description	Cross compensation - Determining compensation values

Applying CCOMP compensation data in positive traversing direction

1. Move the reference axis/axes to the beginning of the traversing range.
2. Move the target axis to the command position to be measured. During further measurements, this position **is not changed anymore for the current target axis**.
3. An external measuring device (e.g. laser) determines the correction value $P[i]$ in μm (i = absolute number of the measuring position: 0 to n) for each measuring position in positive traversing direction. Only traverse

the reference axis and measure the resulting actual position of the target axis.

The following applies:

Compensation value in μm = command position (NC) - actual position (machine)

All compensation values have to be determined in the "micrometer" unit with current a resolution of no more than $0.001 \mu\text{m}$ and have to be in a range of $\pm 2147483.647 \mu\text{m}$.

4. After the last compensation value (end point) in positive traversing direction has been determined, traverse the reference axis to the end of the traversing range.

	Documentation	Instruction
Documentation:	MTX Functional Description	Cross compensation - Compensation data in positive direction

Applying CCOMP compensation data in negative traversing direction

1. Determine the compensation value $N[i]$ in μm (i = absolute number of the measuring position: 0 to n) for each measuring position in negative traversing direction.

The following applies:

Compensation value in μm = command position (NC) - actual position (machine)

2. All compensation values have to be determined in the "micrometer" unit with current a resolution of no more than $0.001 \mu\text{m}$ and have to be in a range of $\pm 2147483.647 \mu\text{m}$.

	Documentation	Instruction
Documentation:	MTX Functional Description	Cross compensation - Compensation data in negative direction

Structuring a CCOMP compensation data file

- The CCOMP compensation data file name is determined as follows:

cnc##r\$\$tab with:

- System axis number of the target axis (01 to 250)

\$\$ - System axis number of the reference axis (01 to 250)

and **\$\$** have to differ from each other

- Structure of a CCOMP file:

<StartingPoint in μm > <StepWidth in μm > <Comment>

<P0> <N0> <Comment>

<P1> <N1> <Comment>

<P2> <N2> <Comment>

<P3> <N3> <Comment>

<P4> <N4> <Comment>

:

<Pn> <Nn> <Comment>

Compensating mechanical inaccuracies

$\langle P[i] \rangle$ in μm = $\langle \text{Starting point} \rangle + i * \langle \text{Step width} \rangle$; Compensation value in positive traversing direction

$\langle N[i] \rangle$ in μm = $\langle \text{Starting point} \rangle + i * \langle \text{Step width} \rangle$; Compensation value in negative traversing direction



- At least one space has to be programmed between the individual entries within a line.
- Complete each line with $\langle \text{Enter} \rangle$.

Example:

CCOMP table

At a portal machine, the sag of the y-axis is to be compensated as it affects the z-axis position too much.

⇒ Target axis (axis to be compensated): Z

⇒ Reference axis: Y

The traversing range of the y-axis (0 to 700 mm) is subdivided into a grid with an step width of 100000 (100 mm).

As starting point of the CCOMP compensation, the absolute axis position of 20000 μm (20 mm) has been selected.

For positive and negative traversing direction, seven axis positions were measured each. For axis positions greater than 20 mm, the compensation values of the starting point become effective. For axis positions greater than 620 mm, the compensation values of the end point become effective.

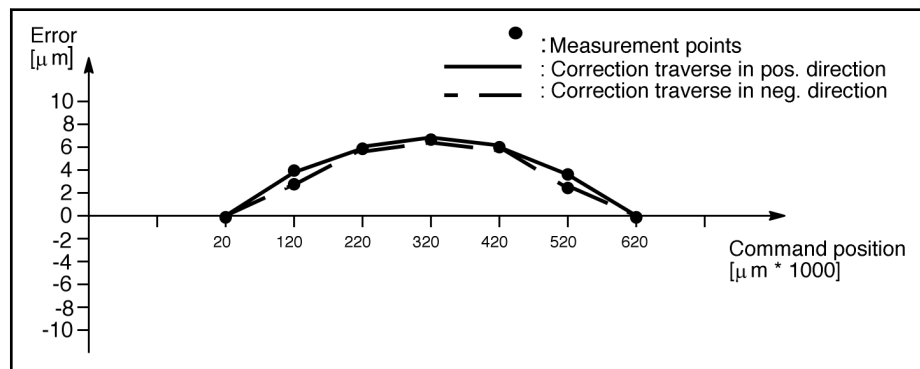


Fig. 5-5: Programming example

Target axis Z has the system axis number 3, the reference axis Y has the system axis number 2.

⇒CCOMP data have to be entered into the "cnc03r02.tab" file.

Required file content of "cnc03r02.tab":

20000.0 100000.0 (starting point 20 mm; step width 100 mm; 7 measurements)

0.0 0.0 (Position 20 mm)

4.0 3.0 (position 120 mm)

6.0 6.0 (position 220 mm)

6.5 6.5 (position 320 mm)

6.0 6.0 (position 420 mm)

4.0 3.0 (position 520 mm)

0.0 0.0 (Position 620 mm)

	Documentation	Instruction
Documentation:	MTX Functional Description	Cross Compensation - Structuring a compensation data file

5.3.3 Handling instruction: Cross compensation (CCOMP), activating

IW-Operation/program: Read/activate the CCOMP file

- For the axis pairs to be compensated, the compensation values of the CCOMP are available in the file `cnc##r$$$.tab`
 - ## - System axis number of the target axis (01 to 250)
 - \$\$ - System axis number of the reference axis (01 to 250)
- Copy or read in the required CCOMP files into the root directory ("/") or into the user FEPROM directory ("/usrfep")

or

- Activate existing, deactivated files again:
Rename the relevant CCOMP files in the root directory ("/") or in the user FEPROM directory ("/usrfep"). Therefore, change the file name extension from "off" to "tab" (e.g. `cnc03r02.off` to `cnc03r02.tab`).
- Trigger NC restart



If the root directory and the user FEPROM contain files of the same name, the NC uses the file from the root directory.

If the "CCOMP" and "LSEC" functions are active simultaneously for one target axis, the respective compensation values apply additively.

	Documentation	
Documentation:	MTX Functional Description	Activating Cross Compensation

5.3.4 Handling instruction: Cross compensation (CCOMP), deactivating

Target axis = axis to be compensated

IW Operation/program: Deactivating CCOMP files

- Remove the CCOMP file from the root directory ("/") or from the user FEPROM directory ("/usrfep").
- or -
Rename the relevant CCOMP files in the root directory ("/") or in the user FEPROM directory ("/usrfep").
- Therefore, change the file name extension from "tab" to "off" (e.g. `cnc03r02.tab` to `cnc03r02.off`).
- Trigger NC restart.

	Documentation	
Documentation:	MTX Functional Description	Activating Cross Compensation

5.4 Switching compensation tables (LSEC and CCOMP)

5.4.1 Description

Function

This function is an extension of the functions "Lead Screw Error Compensation (LSEC)" and "Cross Compensation (CCOMP)". It allows switching tables without control restart.

A modified and common LSEC or CCOMP table is required (e.g. lsecc03.tab, cnc03r02.tab).

Extending table format

To structure the compensation tables, formatting keywords are used. All formatting keywords start with a "#". They label the beginning and the end of an area. Comments in an area start with ";".

The formatting keywords are:

Start/end of the startup parameter

#STUP

#ESTUP

These keywords are used in regular compensation tables (e.g. lsecc03.tab, cnc03r02.tab).

Start/end of the configuration parameter

#CONF

#ECONF

These keywords are used in reloadable tables.

Startup parameter list (between #STUP, #ESTUP):

LOADABLE	; Compensation table can be switched via NC function
EMPTY	; Compensation table does not contain any compensation values
NOMODULO	; No modulo calculation for a rotary axis

Configuration parameter list (between #CONF, #ECONF):

CAX <Target axis>	; System axis to be compensated
RAX <reference axis>	; Reference axis for CCOMP table
EMPTY	; Compensation table does not contain any compensation values

If there is a regular table for the axis error compensation for an axis and it is named LOADABLE, an alternative compensation table can be activated using the "AxCompTab" function.

Switching a table

Reloading a table requires a regular table. The regular table (here cnc03r02.tab) has to include the startup parameter "LOADABLE". There have to be no compensation values if the parameter "EMPTY" is specified.

The reloadable table has to contain the configuration parameters CAX and RAX.

The regular table "cnc03r02.tab" contains the parameter to enable the table switching and the parameter to identify an "empty" table.

The tables "ReloadA0302.tab" and "ReloadB0302.tab" are activated via the NC function "AxCompTab" if required.

cnc03r02.tab

```
#STUP
LOADABLE
EMPTY
#ESTUP
```

ReloadA0302.tab

```
;
;Compensation table A
;
#CONF
CAX 3 ; ZA is the compensated axis
RAX 2 ; YA is the reference axis
#ECONF
-20000.0 50000.0 ; Starting value -20mm, step width 50 mm
6.0 6.0 ; Compensation for -20 mm
8.0 8.0 ; Compensation for 30 mm
9.0 9.0 ; Compensation for 80 mm
8.0 8.0 ; Compensation for 130 mm
6.0 6.0 ; Compensation for 180 mm
3.0 3.0 ; Compensation for 230 mm
0.0 0.0 ; Compensation for 280 mm
```

ReloadB0302.tab

```
;
;Compensation table B
;
#CONF
CAX 3
RAX 2 ; YA is the reference axis
#ECONF
-20000.0 50000.0 ; Starting value -20mm, step width 50 mm
8.0 8.0 ; Compensation for -20 mm
10.5 10.5 ; Compensation for 30 mm
12.0 12.0 ; Compensation for 80 mm
10.5 10.5 ; Compensation for 130 mm
```

```
8.0 8.0 ; Compensation for 180 mm
4.5 4.5 ; Compensation for 230 mm
0.0 0.0 ; Compensation for 280 mm
```

The table "cnc03r02.tab" is active after control startup.
AxCompTab(ZA, ReloadA0302.tab) activates the table "ReloadA0302.tab".
AxCompTab(ZA, ReloadB0302.tab) activates the table "ReloadB0302.tab".
AxCompTab(ZA, YA) returns to the table "cnc03r02.tab".



Switching a compensation table causes an axis motion.

Special features and restrictions

Switching a compensation table by calling the "AxComTab" function is **not executed in the block pre-run**.

Relevant NC functions

AxisCorrTab(<CAxis>, <Table>)	
	Switching a table
AxCompTab(<CAxis>, <RAxis>)	
	Switching a table back to the regular table.
AxCompTab(<CAxis>)	Switching all tables back to the regular tables.

Tab. 5-2: Relevant NC functions

Relevant CPL functions

MCODS(133)	Current lead screw error values
MCODS(134)	Current cross compensation values

Tab. 5-3: Relevant CPL functions

Relevant system data

The name of the respectively active compensation table is stored in the system date "SysAxActData[]".

SD.SysAxActData[<Drive>]	
	.LSEC_FileName
	.CCOMP_FileName1
	.CCOMP_FileName2

Tab. 5-4: Relevant system data

5.4.2 Applying

- LSEC tables and/or CCOMP tables with the extended table format have to be created.
The parameter LOADABLE has to be entered.
- Alternatively, LSEC tables and/or CCOMP tables have to be created for switching.

5.4.3 Activating

The table switching can be started any time by programming "AxCompTab".



Switching a compensation table causes an axis motion.

5.4.4 Deactivating

- Return to the regular tables using the "AxCompTab" function.
- Hiding the parameter LOADABLE, with a ";" in the front for example deactivates the option to switch the compensation table after control start-up.

5.5 Straightness and angle error compensation

5.5.1 Description

Function

The straightness and angle error compensation compensates errors caused by mechanical inaccuracies. These cannot be detected by the encoders. The position error of the coordinate to be compensated (referred to as compensation or target coordinate below) depends on the actual positions of the remaining coordinates (adjustment or reference coordinates).

Virtually, eight compensation coordinates can be compensated per channel depending on up to seven adjustment coordinates each. In practice, however, avoid mutual dependencies during the selection.

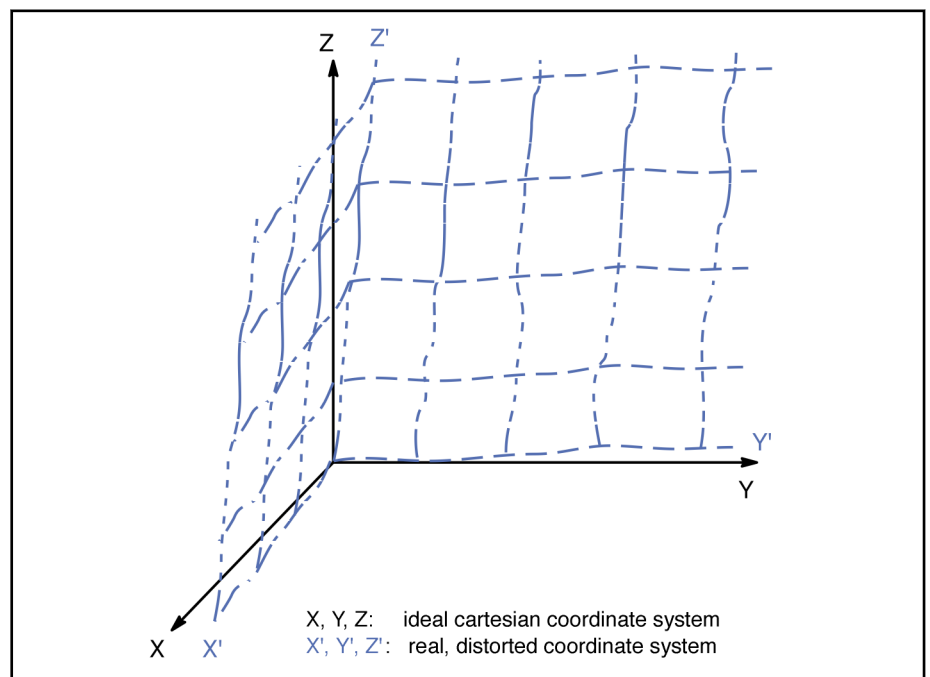


Fig. 5-6: Angle and straightness errors

The straightness and angle error compensation corresponds to an extension of the cross compensation on MCS level. In particular, it can be used with kinematics which require an axis transformation 1 (AT1).

Straightness errors are position deviations that are vertical to the traversing direction of the machine coordinates. The slide path describes an undulation.

Compensating mechanical inaccuracies

Angle errors occur when installing the axis slides. The Cartesian axes are not completely aligned vertically even if adjustment is as precise as possible.

The straightness and angle error compensation compensates machine-related geometry errors in the MCS (machine coordinate system). Thus, the user-created compensation tables are analyzed.

The straightness and angle error compensation is based on the following principle:

- Compensation coordinate:
Machine coordinate (target coordinate) to be compensated.
- Adjustment coordinate:
This machine coordinate approaches data points across the entire traversing range and measures the deviations in the direction of the compensation coordinate.
- The straightness and angle error compensation can correct each machine coordinate of a channel. There can be up to seven adjustment tables per compensation coordinate.
- There is one table for each compensation and adjustment coordinate. This table is used to determine the individual compensation value. The total of the individual compensation values of all adjustment coordinates of a specific correction coordinate results in the total compensation value. This total compensation value is added to the interpolator command value of the compensation coordinate.

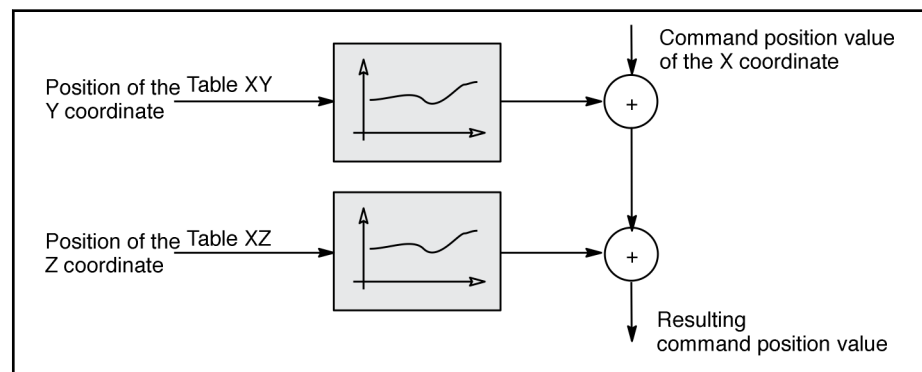


Fig. 5-7: Geometry compensation of the compensation coordinate X with regard to the two adjustment coordinates Y and Z

Restrictions

The straightness and angle error compensation can only compensate geometry errors efficiently if the straightness and angle errors in the direction of the compensation change only to a minor degree in the workspace.

Example:

At the position (X=10, Y=20, Z=100), the error of the compensation coordinate X in relation to the adjustment coordinate Z is 11 µm. The straightness error measured at the position (X=500, Y=100, Z=100) is to 11 µm as well.

5.5.2 Measuring method

Procedure

Before a machine is measured, all compensations (lead screw error compensation, cross compensation, straightness and angle error compensation) are switched off

If several compensation types are used at the same time, ensure the following order:

1. Measure and subsequently activate the lead screw error compensation.
2. Measure and subsequently activate the cross compensation.
3. Measure and subsequently activate the straightness and angle error compensation.

Measuring straightness

- Clamp in a ruler parallel to the adjustment coordinate. Move along the ruler with a dial gauge (dial gauge aligned to the direction of the compensation coordinate and provided with a data interface).
- NC program to determine the straightness: All (equidistant) data points of the adjustment coordinate are approached consecutively. Subsequently, the deviation is read via the data interface.
- The deviations are transferred to the compensation tables in $[\mu\text{m}]$. The sign of the table values results from

Compensation value = command position (NC) - actual position (machine)

All compensation values have to be determined in the "micrometer" unit with current a resolution of no more than $0.001 \mu\text{m}$ and have to be in a range of $\pm 3276.7 \mu\text{m}$.

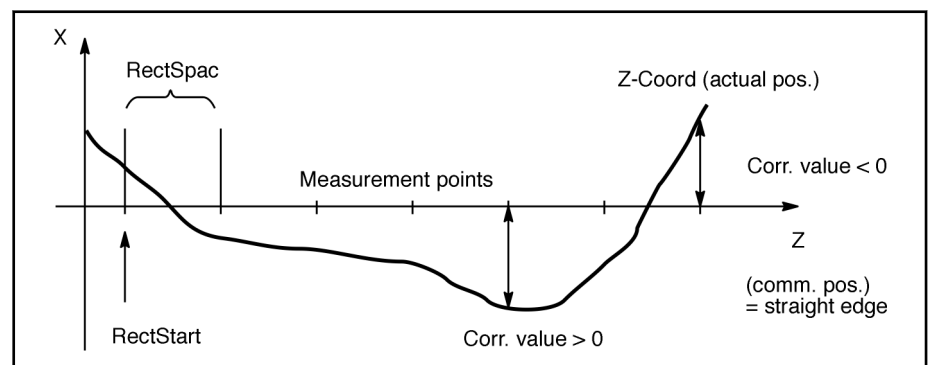


Fig. 5-8: Measuring straightness and sign definition

If the ruler is aligned exactly parallel to the adjustment coordinate (i.e. perpendicular to the compensation coordinate), both the straightness errors and the angle errors are fully compensated by means of the straightness compensation.

If the ruler is aligned at an inclined position compared to the adjustment coordinate, measure the angle error additionally.

Measuring angle error

- Clamp a measurement cube in the machine.
- Determine the angle error:

Compensating mechanical inaccuracies

Approach two data points of the adjustment coordinate. Subsequently, read the deviation in the direction of the compensation coordinate via the data interface.

- The deviations are transferred to the compensation tables in [μm]. The sign of the table values results from

Compensation value = command position (NC) - actual position (machine)

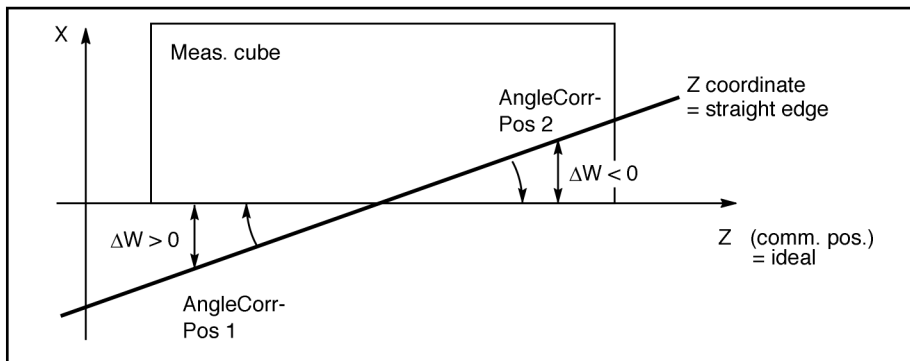


Fig. 5-9: Measuring angle error and sign definition

5.5.3 gct files

Procedure

Save the data to an XML schema. The file has to be stored in UTF-8 format.

This can be directly performed on the control. For this purpose, two CPL functions are available:

CPL function name	Description
XTABCR	Creating a new table
XTAB	Reading or writing table values

Tab. 5-5: CPL functions

The files containing the straightness and angle error compensation data are saved under the file name <GEOCOMP>.gct to the root directory or to the USRFEP directory of the control. The data is saved in ASCII format. GEOCOMP is a user-specific file name. The file extension "get" identifies the "straightness and angle error compensation" file type.

At control startup, the control searches for gct files and downloads the straightness and angle error compensation data of all available gct files. The search starts in the root directory and continues in the USRFEP directory. If there are compensation files with the same name in both directories, only the file in the root directory is analyzed.

Each gct file has a file header in which the mode of action of the compensation is defined.

Up to 50 gct files are permitted.

A new gct file or the modification of an existing gct file is applied at system restart.

Structuring a gct file (header + user data)

Header:	Short:	Chan	Channel number (1..60)
	Short:	AT1CoordNo	Machine parameter number of the axis transformation 1 (1..20 or -1 as wildcard)

Short:	SaciCorr	SACI index of the compensation coordinate (-6, -5, -4, 1..99)
Short:	SaciBal	SACI index of the adjustment coordinate (-6, -5, -4, 1..99)
double:	RectStart	Starting position of the straightness tables in [mm]
double:	RectSpac	Distance between straightness table and data point in [mm]
double:	AngleCorrPos1	1. Measuring position to determine the angle error in [mm]
double:	AngleCorrPos2	2. Measuring position to determine the angle error in [mm]
double:	AngleCorrVal1	Angle compensation value at the first measuring position in [μm]
double:	AngleCorrVal2	Angle compensation value at the second measuring position in [μm]

Compensation values:

Float:	C1	Compensation value 1 of the straightness error compensation in [μm]
Float:	C2	Compensation value 2 of the straightness error compensation in [μm]
Float:	C3	Compensation value 3 of the straightness error compensation in [μm]
:	:	:
Float:	C<n>	Compensation value n of the straightness error compensation in [μm]

Comments on the parameters:

- **Chan:**
The channel number defines the channel in which the table is applied.
- **AT1CoordNo:**
Various axis transformations 1 can become active consecutively in one channel (e.g. 4-axis bipod (3131001)/3-axis bipod (2121001)/axis transformation 1 deactivated). These are activated with the "Coord (AT1CoordNo)" command. In this case, the different SACI assignment requires that a separate "gct" table is stored for each configuration.
If no differentiation is required, "-1" can be entered as wildcard.
- **SaciCorr / SaciBal:**
Up to eight compensation coordinates can be combined per channel. Up to seven adjustment coordinates can be defined for each of these compensation coordinates. As mentioned above, avoid mutual dependencies.
Compensation and adjustment coordinates have to belong to the same channel.
- **RectStart / RectSpac:**
The radius of action of the straightness table is defined by the starting position and the distance or number of the data points. The first table value (compensation value) corresponds to the starting position. Table value "k" is assigned to the "RectStart + (k-1) * RectSpac" position.

Compensating mechanical inaccuracies

- The number of table values is limited to 200. The number is not displayed as parameter value in the file header, but results from the number of the compensation values in the file.
- The table values between the data points are obtained by linear interpolation. If values are outside the table range, the first or last table compensation value remains effective.
- The straightness table and the angle error are internally summarized to a total compensation.

Example of a gct file

Observe the following configuration:

Channel 1 contains eight axes: R1, R2, R3, YL, YR, ZA, B, SC with the corresponding consecutive Saci numbers 1...8.

The machine to be corrected is a bipod which is optionally operated as a 3-axis or a 4-axis variant.

- **(1) The 4-axis bipod (3131001)**
is defined as axis transformation no. 1 in the machine parameters. It transforms between the axes (YL, YR, ZA, SC) and the coordinates (X, Y, Z, psi).
The x-coordinate is the compensation coordinate.
X has SaciNo. -6 $\Rightarrow$ SaciCorr = -6
The z-coordinate is the adjustment coordinate.
Z has SaciNo. -4 $\Rightarrow$ SaciBal = -4
- **(2) The 3-axis bipod (2121001)**
is defined in the machine parameters as axis transformation no. 2. It transforms between the axes (YL, YR, SC) and the coordinates (X, Y, psi).
The x-coordinate is the compensation coordinate.
X has SaciNo. -6 $\Rightarrow$ SaciCorr = -6
The ZA-coordinate is the adjustment coordinate.
ZA has SaciNo. 6 $\Rightarrow$ SaciBal = 6

The following "gct" files with an identical effect result for the two configurations above:

Case (1)	Case (2)
<pre> <?xml version="1.0"?> <GCT> <Header> <Chan>1</Chan> <AT1CoordNo>1</AT1CoordNo> <SaciCorr>-6</SaciCorr> <SaciBal>-4</SaciBal> <RectStart>0.00000</RectStart> <RectSpac>1.00000</RectSpac> <AngleCorrPos1>10.00000</AngleCorrPos1> <AngleCorrPos2>70.00000</AngleCorrPos2> <AngleCorrVal1>0.00000</AngleCorrVal1> <AngleCorrVal2>17.5000</AngleCorrVal2> </Header> <Values> <C>10.100</C> <C>20.200</C> <C>25.300</C> <C>30.400</C> <C>20.500</C> </Values> </GCT> </pre>	<pre> <?xml version="1.0"?> <GCT> <Header> <Chan>1</Chan> <AT1CoordNo>2</AT1CoordNo> <SaciCorr>-6</SaciCorr> <SaciBal>6</SaciBal> <RectStart>0.00000</RectStart> <RectSpac>1.00000</RectSpac> <AngleCorrPos1>10.00000</AngleCorrPos1> <AngleCorrPos2>70.00000</AngleCorrPos2> <AngleCorrVal1>0.00000</AngleCorrVal1> <AngleCorrVal2>17.5000</AngleCorrVal2> </Header> <Values> <C>10.100</C> <C>20.200</C> <C>25.300</C> <C>30.400</C> <C>20.500</C> </Values> </GCT> </pre>

Tab. 5-6: Example of a gct file

5.5.4 Switch on/off behavior, control reset

Control startup: After control startup, the function is disabled.

Reset: After control reset, the function remains in its old state.

5.5.5 Handling instruction: Straightness and angle error compensation

Applying: The straightness and angle error compensation allows the compensation of errors which are caused by mechanical inaccuracies. These cannot be detected by the encoders. The position error of the coordinate to be compensated (referred to as compensation or target coordinate below) depends on the actual positions of the remaining coordinates (adjustment or reference coordinates).

NC: Measuring the straightness

- Clamp in a ruler parallel to the adjustment coordinate.
Move a dial gauge along the ruler (with the dial gauge aligned in the direction of the compensation coordinate and provided with a data interface).
- Write NC program to determine the straightness:
All (equidistant) data points of the adjustment coordinate are approached consecutively. Thereafter, the deviation is read via the data interface.

Compensating mechanical inaccuracies

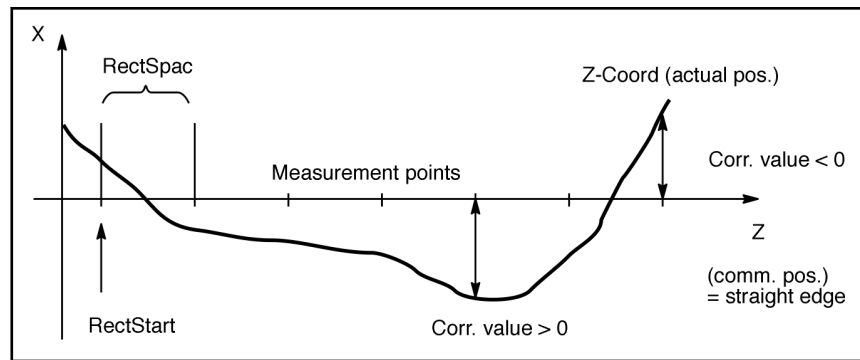


Fig. 5-10: Measuring straightness and sign definition

- The deviations are transferred to compensation tables in [μm]. The sign of the table values results from the following equation:
Correction value = command position (NC) - actual position (machine)



If the ruler is aligned exactly in parallel with the adjustment coordinate (i.e. perpendicular to the compensation coordinate), the straightness errors and the angle errors are fully compensated by means of the straightness compensation.

If the ruler, however, is aligned at a slight inclined angle compared to the adjustment coordinate, the angle error has to be measured in addition.

NC: Measuring the phase error if necessary

- Clamp a measurement cube in the machine.
- Determine the angle error:

Two data points of the adjustment coordinate are approached. Subsequently, the deviation in the direction of the compensation coordinate is read via the data interface.

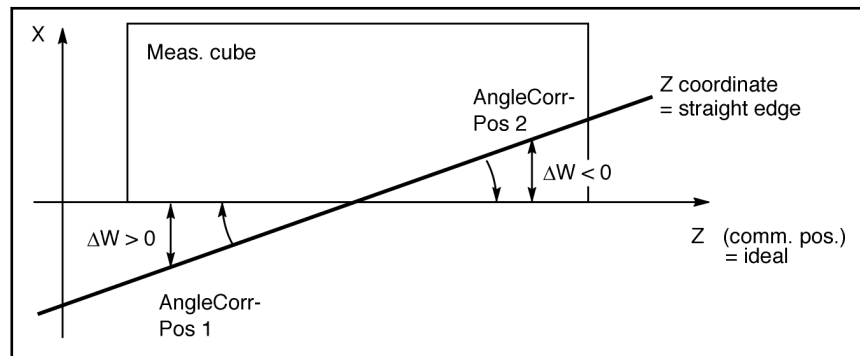


Fig. 5-11: Measuring angle error and sign definition

- The deviations are transferred to compensation tables in [μm]. The sign of the table values results from:

Correction value = command position (NC) - actual position (machine)

IW Operation/NC programming: Entering values in a *.gct file

Write the values in a *.gct file via the CPL function "General access to XML tables".

Description	Name of CPL function
Creating a new table	XTABCR(<Table>,<Tyep>[,<Table template>])
Reading or writing table values	XTAB(<Table>,<Substructure>)

Tab. 5-7: General access to XML tables

- or -

- Copy an existing *.gct file to the root directory or the /usrfep.



The files containing the straightness and angle error compensation data are saved under the file name **<GEOCOMP>.gct** to the root directory or to the **usrfep** directory of the control. The data is saved in ASCII format. **<GEOCOMP>** is a user-specific file name.

Activating: Upon system restart, the NC searches for straightness and phase error compensation files with the file name **""*.gct"**. If such files exist, the straightness and phase error compensation is parameterized for the machine coordinates specified.

If the maximum parameter values are exceeded, if the same tables are created several times, if invalid machine coordinates are used, if more tables than permitted are defined, ..., the appropriate error messages are output.

The straightness and angle error compensation can be activated using the following NC function after referencing the axes of the relevant channels.

GCT(1) (optionally: **GCT**, **GeoComp(1)**, **GEOCOMP(1)**)

When activating, the compensation values calculated for the current position are formed with a defined velocity profile. The selected velocity profile is a linear velocity ramp (with the velocity profile corresponding to a trapezoid) forming the compensation value in 100 ms.

After the starting compensation value has been generated, the new compensation values are generated in each interpolation cycle by analyzing the tables.

The compensation is effective in the MCS coordinate system. As a result, the straightness and phase error compensation can also be used with kinematics requiring axis transformation 1.

If the MCS0 coordinate system is selected for the position display, the compensation is not visible.

Jog mode of axes:

The axis is moved with compensation of the straightness and angle error. I.e. after activation, the compensation is also active during jog mode.



- We strictly recommend the activation only after all participating axes successfully traveled to the reference point.
- **GCT(1)** enables the straightness and angle error compensation for all channels.

Deactivating

The straightness and phase error compensation is deactivated with the following NC function:

GCT(0) (alternatively: **GeoComp(0)**, **GEOCOMP(0)**)

When the compensation is deactivated, the compensation values reduced with a defined velocity profile (as during the activation).



GCT(0) disables the straightness and angle error compensation for all channels.

5.6 Backlash compensation

5.6.1 Description

Function

Compensates the backlash of an axis occurring in the area of the mechanical motion conversion (motor axis). For example, the backlash is caused by a flank clearance while a motion is transmitted by a gear.

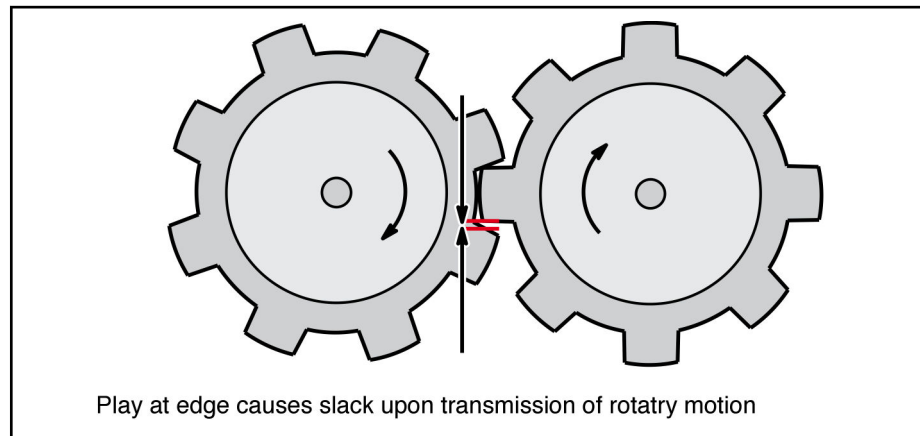


Fig. 5-12: Backlash compensation

The NC generally provides two possibilities to compensate the hysteresis of an axis:

- If the lead screw error compensation (LSEC) is used, the hysteresis of the axis is automatically compensated with the linearity errors, since two compensation values for the positive and negative traversing directions are measured per measuring point. The lead screw error compensation has the advantage that a backlash that changes across the traversing range of the axis can be considered.

Chapter 5.2 on page 25 describes the lead screw error compensation (LSEC).

- For more simple applications, it is often sufficient to compensate the backlash. The drive can compensate the backlash via the Sercos ID number S-0-0058.

The drive function is activated by entering the backlash in the parameter S-0-0058, Hysteresis. Then, the actual position values of the encoder are corrected with the value from S-0-0058 considering the traversing direction.

Relevant Sercos parameters

- S-0-0058: Backlash

5.6.2 Handling Instruction: Backlash Compensation

Applying *IW Operation / Program: Determining and adapting the SCS file of the corresponding axis*

- S-0-0058 "Hysteresis":
If not done yet, reset to 0.
- Traverse the axis in positive direction in jog mode.

- Fix the dial gauge at an appropriate point of the axis mechanics (if possible in the center of the traversing range of the axis) and reset it to zero.
- Use appropriate small increments to jog the axis in negative direction until the dial gauge indicates a change.
- The backlash results from the difference between the path specified in increments in negative direction and the distance traveled as indicated by the dial gauge.

$$S-0-0058 = X_{\text{control}} - X_{\text{dial gauge}}$$

- The following applies to the compensated actual position values of the axis:

$$x_{\text{act, comp}} = x_{\text{act}}, \text{ for positive direction}$$

$$x_{\text{act, comp}} = x_{\text{act}} + S-0-0058, \text{ for negative direction}$$

	Documentation	Instruction
Instruction:		Editing Sercos Parameters
Documentation:	Rexroth IndraDrive Parameter Description	Backlash - Hysteresis

Activating IW Operation / Program: Activating backlash compensation by adjusting SCS files

S-0-0058 "Hysteresis":

To activate the drive function, enter the backlash into the parameter of the corresponding axis. Subsequently, the actual position values of the encoder are compensated with the value from S-0-0058 considering the traversing direction.



- If the lead screw error compensation (LSEC) is active, the hysteresis compensation cannot be activated.
- By resetting the Sercos parameter **S-0-0058** "Hysteresis", the hysteresis compensation is deactivated.

	Documentation	Instruction
Instruction:		Editing Sercos Parameters
Documentation:	Rexroth IndraDrive Parameter Description	Activating Backlash Compensation

Deactivating IW Operation / Program: Deactivating the backlash compensation by adjusting the SCS file

S-0-0058 "Hysteresis":

Enter 0.

	Documentation	Instruction
Instruction:		Editing Sercos Parameters
Documentation:	Rexroth IndraDrive Parameter Description	Backlash - Deactivating Compensation

5.7 Circle compensation at quadrant transition

5.7.1 Description

Function



The "Circle compensation at the quadrant transition" function is only available with Rexroth IndraDrive drives!

The circle compensation reduces pulse-shaped contour distortions caused in circular paths behind the quadrant transitions by static friction.

Circular errors at quadrant transitions are also caused by the reversal of direction of an axis in the circular path. While one axis travels at maximum velocity at the point of transition, the second axis comes to a standstill. While accelerating into the new direction, the static friction keeps the axis in the inflection point. This results in a noticeable lag which is reduced in jerks only after exceeding the static friction limit.

Therefore, the drive is able to independently connect a triangular speed pulse at reversal of direction of an axis which - starting from a certain pulse height - is reduced after a certain number of sampling steps.

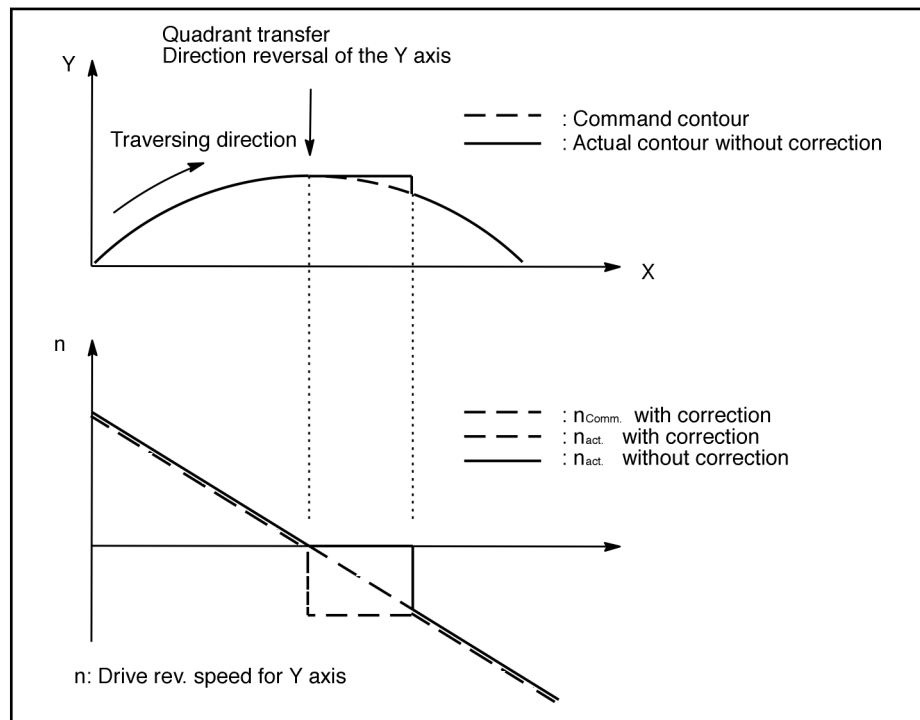


Fig. 5-13: Circle compensation at the quadrant transition

To commission this drive function,

- A reference circle is to be passed through with a radius as required for a component to be accepted for example. Store the radius in the Sercos parameter **P-0-0436 Reference radius for quadrant error compensation**.
- The velocity pulse of the quadrant error compensation is specified with regard to the circular path velocity. The pulse is added in jumps and decreases in ramps later on. The higher the path velocity, the higher the pulse and the faster it is lead towards zero, since the velocity time line of the pulse is constant and does therefore not depend on the velocity. The range corresponds to the value given in **P-0-0437 Velocity time range for quadrant error compensation**.

- To dimension P-0-0437, it is recommended to estimate the required pulse as well as its period. This is to be performed with regard to the path velocity as well as to the period required to travel along the reference circle (P-0-0436):

$$P-0-0437 = 1/2 * a / 100\% * v_{\text{path}} * b / 100\% * T_{\text{circle}}$$

P-0-0437: Velocity time line for quadrant error compensation

a: Percentage of the path velocity

v_{path} : Path velocity

b: Percentage of the period required to travel along the circular contour

T_{circle} : Period required to travel along the circular contour

- *The circular path velocities as well as the velocity pulses are assigned to each other via list parameters:*
 - P-0-0438, Table of path velocities for quadrant error compensation
 - P-0-0439, Table of velocity pulse for quadrant error compensation

20 value pairs for velocity and pulse can be stored in the mentioned parameters. Velocities have to be entered in an ascending sequence.
- *The velocity pulse is assigned to the current circular path velocity using the list parameter values:*
 - Between two value pairs, the pulse interpolates linearly depending on the current path velocity.
 - The pulse is set to zero below the smallest path velocity.
 - The pulse assigned to the highest path velocity is also valid above the highest path velocity.
- The control device calculates the current circular path velocity using the current velocity command value of the drive. This value is calculated using the position command values. The path velocity calculation is based on the reference radius (P-0-0436). To ensure sufficient resolution of the drive-internal acceleration command value, the parameter **P-0-0100, Position command value extension** is to be included in the real-time data transfer. By doing so, the decimal digits of the position command value are extended.
- The quadrant error compensation is enabled by the control master by setting bit 0 of **P-0-0435, Control word of position controller**.

Restrictions

- Active axis transformations are not allowed.
- The program coordinates (PCS) of the circular plane have to be parallel to the axis coordinate system (ACS). Offsets are allowed, spatial rotations are not allowed.



The control enables the quadrant error compensation at the beginning of a circular interpolation if the above mentioned conditions are met. It disables the compensation automatically at the end of the block. To avoid the activation of the compensation pulse when approaching, the activation is always delayed by one cycle.

Relevant Sercos parameters

P-0-0100	Position command value extension Increases the decimal digits of the position command value
P-0-0435	Control word of position controller Bit 0 activates the quadrant error compensation Is automatically set by the NC
P-0-0436	Reference radius for the quadrant error compensation Sets the quadrant error compensation (e.g. circular radius of the component to be accepted)
P-0-0437	Velocity time line for quadrant error compensation The velocity time line does not depend on the path velocity The higher the path velocity, the higher the added pulse and the faster the pulse becomes zero again
P-0-0438	Table of path velocities for quadrant error compensation
P-0-0439	Table of velocity pulse for quadrant error compensation
S-0-0015	Telegram type parameter
S-0-0024	Configuration list MDT Defines which parameters are expected in the freely configurable telegram from the drive
S-0-0027	Configuration list of the signal control word Parameter list transferred in the signal control word
S-0-0145	Signal control word
S-0-0329	Assignment list of the signal control word Specifies which bits are transferred to the signal control word according to the configured parameters

Tab. 5-8: Relevant Sercos parameters

5.7.2 Handling instruction: Circle compensation at quadrant transition

Prerequisite

- Use of IndraDrive drives (only MPH)
- Extension package "servo function" (ordering code SRV) in closed-loop characteristic
- The relevant Sercos parameters (see above) are correctly parameterized in the drive and saved or transmitted from the control to the drive during the Sercos initialization.

NOTICE

Workpiece or machine can be damaged!

Do not reset the control while machining.

1. Include the Sercos parameters P-0-0100 and S-0-0145 in the list of the cyclic telegram S-0-0024 and ensure that the "freely configurable telegram" is set in the parameter S-0-0015.

2. Enter the parameter P-0-0435 "Control word of position controller" into the configuration list for the signal control word S-0-0027 and enter bit 0 into the assignment list for the signal control word S-0-0329.
3. Select a radius typical for the machine (e.g. 100 mm) and enter the value into P-0-0436 (unit: mm, without decimal places).
4. Enter 0 as the velocity range of the quadrant error compensation P-0-0437.
5. In P-0-0438, enter all path velocities (unit: mm/min; without decimal positions) for which you want to determine the required cycle level in the course of the process (e.g. in the area of 100mm/min to 5000 mm/min).
Enter the path velocities (max. 20 values) in ascending order and separated by a comma.
Later, the respective pulse levels (likewise separated by a comma) are entered in rpm (with 3 decimal positions) in P-0-0439. Pre-initialize the values with 0.
6. Create a part program in which only a full circle is programmed with the previously specified radius (see par. 2.) and the first path feed in P-0-0438 (see point 3). Starting and end point are not to be on one quadrant transition.
7. Save all Sercos parameters in the drive or enter the Sercos parameters determined up to this point, together with their values, into the relevant Sercos files of the NC. During Sercos initialization, the control transmits the Sercos files to the drive. (For information on Sercos, refer to [chapter 9.1 Functions for Sercos handling, page 77](#)).
8. Restart control and drives.
9. Record the actual positions of the axes for the previously created part program using an axis oscilloscope.
A circular error at the quadrant transition is now shown in the display.
10. Change velocity range (P-0-0437; max. three decimal positions) and the pulse level belonging to the current path velocity (P-0-0439; unit: rpm, three decimal positions) until a new recording in the axis oscilloscope shows no significant circular offset anymore. If this is the case, do not change the determined velocity line anymore.
11. Now, change path feed (path velocity) in the part program to the next value stored in P-0-0438 and record the process of the part program once again as described under 8.
12. Change the pulse level belonging to the current path velocity (P-0-0439; unit: rpm, three decimal positions) until a new recording in the axis oscilloscope shows no significant circular offset anymore.
13. Repeat the processes described in 10 to 11 for the remaining velocity values stored in P-0-0438.
14. Save all Sercos parameters in the drive or enter the data of the parameter P-0-0439 determined up to now into the relevant Sercos files of the NC. The Sercos files are transferred from the control to the drive during the Sercos initialization (for information on Sercos refer to [chapter 9.1 Functions for Sercos handling, page 77](#)).
15. Restart control and drives.

5.8 Volumetric compensation (VCP)

5.8.1 Description

Function

Large machines have mechanical inaccuracies that counteract a required accuracy at the tool tip. Theoretically, it is possible to measure position deviations at many interpolation points in the entire operating range and to use these deviations as compensation values. The volumetric compensation assumes a rigid body model in which the errors only duplicate along the kinematic chain. Thus, decisively less error values are needed to describe the position error at the tool tip. Consequently, the accuracy of a machine can be improved by up to factor 10 (depending on the repeatability).

Limitations of the compensation model

- Only reproducible errors (repeatability)
- Only serial kinematics with Cartesian axis arrangement
- Rotary axes only at the start or the end of the kinematic chain
- Rotary axes errors not in the model
- Thermal influences not in the model
- Only the position is corrected, not the orientation

During the first step, the machine has to be measured. Thus, service work is needed. Please contact the distribution partner of Bosch Rexroth.

The measurement result is a file stored in the internal file system. The file name as well as the path are user-defined. It is recommended to assign a descriptive name as well as the extension ".ofm". Thus, the file type can later be easily identified.

Any number of compensation files can be created (file system space permitting). Different files can thus be created for different workpieces, different temperatures or specific tools.

In case of complex machine kinematics (more than 3-axis), define a compatible axis transformation. For detailed information on the axis transformation, refer to the documentation "MTX Functional Description 14VRS Extension", chapter "Axis transformation".

The following 5-axis machine kinematics are supported by the volumetric compensation:

TrafoType	Name	Restrictions
3232101	LLLRR (linear orientation)	None
3032101	LLLRR (linear orientation of the rotary axes)	
3033001	LLLRRR (linear orientation of the rotary axes)	
3232201	LLLRR (vector orientation)	
3232102	CLLLB, CLLLA, ALLLB (linear orientation)	ALLLB only in case of tool length correction in the TCS
3232202	CLLLB, CLLLA, ALLLB (vector orientation)	in case of tool length correction in the BCS, the C-axis must not be in off-center condition

TrafoType	Name	Restrictions
3232103	CBLLL, CALLL (linear orientation)	Only in case of tool length correction in the TCS
3232203	CBLLL, CALLL (vector orientation)	

Tab. 5-9: 5-axis machine kinematics for VCP

The machine kinematic type is specified in the compensation file.

Read in the correction file before enabling the compensation. Therefore, the NC function `VCPLoad` is programmed. The system reads in the specified file in an internal memory.

Subsequently, the volumetric compensation is enabled using the NC function `VCPEna`.

⚠ CAUTION

Upon switch on, the compensation is traversed and retracted upon switch off. An axis motion occurs.

Ensure that the machine traverse the compensation without any collision before switching the machine on or off.

If the compensation is not required anymore, disable it using `VCPEna`.

Restrictions and special features

- When enabling and disabling the compensation, the correction value is traveled in within 3 s. Thus, the machine moves
- The period of the motion depends on the correction value and the permitted (safe) velocity
- The compensation has an effect on the machine coordinates (MCS). Sporadically, a subordinate axis transformation AT1 can also be active
- If the required axes are not in the channel or the specified axis transformation AT1 is not active, a warning is created at activation and no compensation takes place
- If the axis transformation AT 1 is changed during an active compensation, the compensation deactivates itself and a warning is reported. The compensation is retracted
- If axes are replaced in the channel, it is checked whether all compensated axes are still in the channel. If this is the case, the compensation remains active. If required axes are not in the channel anymore, a warning is output and the compensation is retracted for axes that are still in the channel
- In case of complex kinematics, a corresponding 5-axis transformation has to be configured. This transformation does not necessarily have to be active
- As the volumetric compensation can significantly increase the tool machine accuracy, it is only contains in an MTX version requiring an export license

Relevant NC functions

<code>VCPLoad(<Compensation file> {, <Memory space>})</code>	
	Loading of a correction file in the memory

Compensating mechanical inaccuracies

VCPEna {(<MemorySpace>)}	
	Enabling the volumetric compensation
VCPEna()	
	Disabling the volumetric compensation

Tab. 5-10: Relevant NC functions

Relevant system data

A system data **SysVolComp** (per channel) can be created. The files "sdvol-comp.xsd_" and "SDDefVolComp.xml_" are stored in the "/feprom/" directory of the MTX. Copy the XSD file to "/schemes/" and copy the XML file to "/" and remove the "_" at the end of the files. Following an NC restart, the **SD.SysVolComp** system data variable has been created.

The 3D compensation (if active) cyclically (in each interpolation cycle) saves the calculated correction values in float format.

SD.SysVolComp[<Channel>]		
	.TcpPos[3]	Uncorrected TCP position (X, Y, Z) mm
	.Pivot[3]	Uncorrected pivot position (X, Y, Z) mm
	.Tool[3]	Vector of tcpPos to pivot (X, Y, Z) mm
	.Corr[3]	Correction vector (X, Y, Z) µm
	.SACI[3]	System axis indexes of the X, Y, Z axis
	.TabIdx	Active table index (0 = disabled)

Tab. 5-11: Relevant system data

If the machine is currently traversed, the system data does not have to be consistent. The TcpPos vector is directly determined from the difference "Pivot - Tool". Consequently, this position complies with the WCS-TC position at 3-axis machines and at LLLRR kinematics. In case of other kinematics, the positions deviate.

Adjustment after block pre-run

Following a block pre-run, the adjustment function **ADJ(VCP)** is used to adjust the state of the volumetric compensation. Depending on the block pre-run result, the relevant compensation files are loaded and the programmed compensation is enabled.

6 Channel control

6.1 Retrace: Retracing on contour

6.1.1 Description

Function

A part program can be stopped and can subsequently be positioned on the contour already machined via interface signals using the "Retrace" function. After completing the retrace operation, the program can be continued in normal mode.

Retrace program generation

Program points, for which "retracing" is to be provided, are labeled with the NC function "Retrace" in the part program.

Within this identifying mark, the control generates a forward and a return program when processing the part program. These programs can be processed in reverse operation.

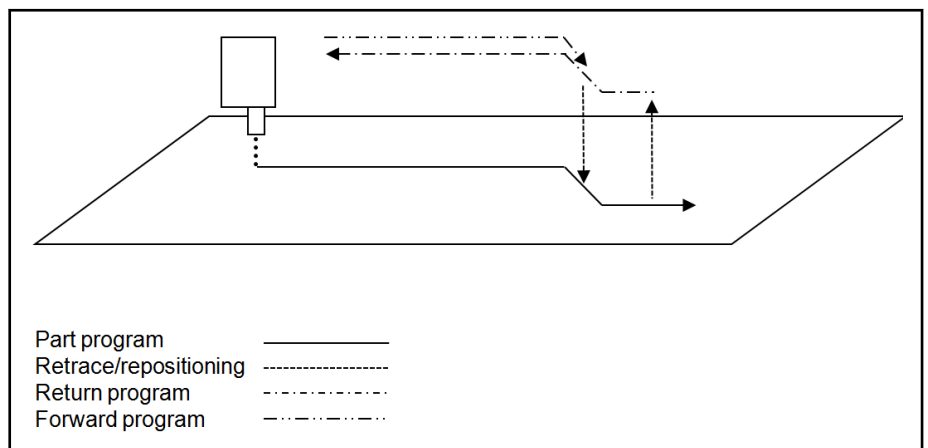


Fig. 6-1: Retrace mode for laser cutting

Line-by-line, the forward and the backward program contains the contour generating specifications of the part program. The programmed coordinates and part of the contour-relevant NC functions are automatically transmitted to the forward and the return program. Asynchronous axis motions are not transmitted. The following NC commands are available for application-specific adjustments:

- AdvancedAppend (ADVA)
Insert NC functions in forward program line
- ReversAppend (REVA)
Insert NC functions in return program line
- AdvancedSubstitute (ADVS)
NC functions replace the forward program line
- ReverseSubstitute (REVS)
NC functions replace the return program line



Please note that contour blocks of part programs, backward and forward programs have to be identical! Only then, switching between normal and return mode is possible and only then switching of machining direction is possible during retrace mode.

CPL commands and subroutine calls are not allowed.

Also note that not all NC functions can be retraced.

The [tab. 6-1 "Switchable NC functions in the retrace area" on page 58](#) describes the supported NC functions. For all other NC functions, a correct retracing is not ensured.

DANGER

Unexpected traversing motion while retracing

When retracing, only the NC functions described in [tab. 6-1 "Switchable NC functions in the retrace area" on page 58](#) may be used. If unsupported NC functions are used in a RETRACE sequence, an error or an unexpected traversing motion can result during the NC execution while retracing.

The table below shows the automatically transmitted NC function in forward and return programs:

NC function	Function group
G0, G1, G2, G3, G5, G6	Interpolation functions *) **)
G8, G9	Path slope *)
G17, G18, G19	Plane preselection *)
G40, G41, G42	Milling path correction *)
G43, G44	Transition path segment
G45, G46	Feed at milling cutter contact point/milling cutter center point
G47, G48	Tool correction *)
G53.x,...,G59.x	Zero offsets
G61, G62	Exact stop
G70, G71	Inch/metric programming
G90, G91	Absolute/relative programming:
G93, G94, G95	Feed programming
G140, G141, G142	3D tool radius correction
G153.x,...,G159.x	Placements
CHLL, CHSL	Local chamfer programming
SCO, SCOL	Modal or local corner rounding with splines
RNDL	Local rounding

LockRetrace	Retrace switching lock (on)
NcfLockRetraceOff	Retrace switching lock (off)

- *) No parameter support in the forward and the backward program
- **) For G6, only the spline programmings specified in the documentation "Functional Description - Special Functions, chapter "Channel control", "→ Retrace: Retracing on the contour", → "Description", are supported

Tab. 6-1: Switchable NC functions in the retrace area

Optionally, the retrace command can be parameterized for debugging:

- Retrace(<ForwardProg>,<BackwardProg>)
saves generated forward and return program as file for control purposes.

The following example shows how to generate the forward and the return program from the part program.

Program:

Main program	fwd.npg	bwd.npg
N0100 G90 G1 X0 Y0 G94 F10000		
N0110 RETRACE(fwd.npg,bwd.npg)		
N0120 X10 M21 ADVA(M21) REVA(M22)	N001 X10 M21	N001 X0 M22
N0130 Y10	N002 Y10	N002 G1 Y0
N0140 G0 M123 ADVS(;empty) REVS(M222)	N003 ;empty	N003 M222
N0150 RETRACE(0)		

After having machined the retrace area, the log files fwd.npg for the forward program and bwd.npg for the return program are saved in the directory of the main program.

A retrace buffer is configured for retracing via a machine parameter. This buffer can temporarily store a certain number of prepared NC blocks for the "retracing" function. The size of the buffer is limited. Thus, if the buffer is full, the oldest block always has to be deleted before a new block can be stored.

Switching from normal to retrace mode:

The function "retracing on contour" is initiated by the channel interface signal qCh_Retrace. Amongst others, the interpolation has the task to stop the channel motion. After stopping, the axes are at standstill and located at the abort position.

Processing the backward and forward program:

At the rising edge of the channel interface signal qCh_RetraceMovRev (path motion backward), the generated retracing program is started. The processing is executed as long as the interface signal is present and the NC blocks are still in the retrace buffer. The return program extends to the retrace start mark. If the start of the retrace area is reached, a warning/message is output. Only the NC functions available in the retrace program are executed during retracing.

When removing the interface signal qCh_RetraceMovRev, the interpolation gets the task to stop. The control system changes to the "NC ready" state.

At the rising edge of the channel interface signal qCh_RetraceMovFwd (path motion forward), the generated forward program is started. The processing is executed as long as the interface signal is present and the NC blocks are still in the retrace buffer. The forward program extends to the activation block of the retrace mode. If the end of the activation block is reached, a warning/message is output.

Channel control

When removing the interface signal `qCh_RetraceMovFwd`, the interpolation stops. The motion standstill is signaled by the "NC ready" state. When moving forward during retrace mode, only the NC functions in the forward program are executed.

Returning to machining

When removing the interface signal `qCh_Retrace`, the retrace mode is switched off. The motion standstill is again signaled by the "NC ready" state.

If no asynchronous subroutine is logged in, the normal program is continued with NC start.

If an asynchronous subroutine is logged in the system data, it will be called automatically after NC start. The name of the ASUPs is stored in the **SysRetrace** system data. The components are described in the document "MTX 14VRS Machine Parameters".

The asynchronous subroutine is logged into the "NameOfAsup" component, e.g. in the part program (CPL):

```
10 SD.SysRetrace[1].NameOfAsup="/usrfep/RetraceAsup.cnc"
```

At the end of `RetraceAsup`, machining is continued.

While `RetraceAsup` is running, the following system data is set to 1:

```
"SD.SysRetrace.AsupActive"
```

Leaving and returning to contour:

If retracing should not be done on the contour - see [fig. 6-1 "Retrace mode for laser cutting" on page 57](#) - use "Feed Halt" to stop the parts program before entering the retrace mode and to retract with leaving the contour and returning to the contour. Leaving and returning to the contour has to be completed at the new "height" before the retrace mode can be started. Adjust the vertical coordinate (e.g. Z) in the asynchronous subroutine (`ReentryAsup`) intended for re-entry.

```
N0100 G77 Z0
```

Additionally, it has to be prevented that the part program is continued after completing leaving the contour and returning to the contour. The system data **SysReentry/FeedHold** controls the program continuation after the re-entry. In `ReentryAsup`, the continuation of the part program can be prevented using the following CPL instruction:

```
100 SD.SysReentry.FeedHold=1
```

In retrace mode, positioning along the contour in the set distance to the part program is now possible.

By completing the retrace mode (resetting `qCH_Retrace`) and "Feed Halt", leaving the contour and returning to the contour is possible. The `ReentryAsup` can be applied for the re-entry in the part program. In this case, the `ReentryAsup` is suppressed upon the NC start and only the `ReentryAsup` is started.

The retrace interface signals are inactive during leaving the contour and returning to the contour. The retrace mode can only be switched on again by the rising edge of the interface signal "`qCh_Retrace`" after the approaching point has been reached while returning to path.

Suppressing the Infeed motion

In retrace mode, the infeed motions perpendicular to the machining plane have to be ignored. The infeed coordinate is assigned the attribute `FR(<co-ord>)`.

The attribute prevents the application of the coordinate in the forward and return program. If the retrace mode is activated in the infeed block of the highlighted coordinate, the infeed coordinate remains "frozen" on the current position.

Change in direction The control internally executes a "Distance to go" whenever changing the direction. All already prepared blocks are discarded.

A change in direction occurs:

- When enabling the retrace mode and subsequent retracing
- When changing the traversing direction in the retrace mode

"Distance to go" is also executed if the last retrace mode is exited and if the interrupted part program is continued.



If correction values (e.g. zero points or tool data) have been changed within the range of already prepared blocks, the changed values apply to **all** blocks to be prepared again. That also applies in the retrace mode.

If this behavior is not desired, a separate line has to be programmed using a "WAIT" before each influencing command (such as ZOT or DCT)!

Optionally, the refresh mechanism of tables and correction registers can be suppressed globally using the `/SysCorrRefrDis` system date. Changed correction values are applied upon the next ZO/correction programming or the next table activation in this case.



The number of blocks to be stopped in block preparation can be set through machine parameter `NCO/LookAh/Ch[k]/NofBlkPrep` (7060 00110).

Retrace switching lock The commands "LockRetrace(1)" and "LockRetrace(0)" can be used in the part programs to define any number of subareas with locked retrace switchings and which cause a warning/message. Areas labeled that way can be passed. However, it is not possible

- to start the retrace mode with subsequent retracing
- to change the direction of machining in retrace mode
- to end the retrace mode by continuing the interrupted part program

Restrictions

- The retrace area limits the retrace mode. Retracing is only possible within the active area. This also applies if several areas are defined in the part program.
- Forward and return programs contain the resolved NC block sequence without CPL statements and without subroutine calls.
- No subroutine calls must be inserted in the forward and return program.
- With active retrace buffer (retrace(1)) and deactivated retrace operation (`qCh_Retrace = 0`), it is possible to call normal asynchronous subroutines.

In retrace mode (`qCh_Retrace = 1`), asynchronous subroutines cannot be called. Only at the end after NC start for the machining, the asynchronous subroutine logged into the system date `SysRetrace` is automatically called.

- The retrace mode supports the linear and the circular interpolation as well as partially the spline interpolation.
- The SBC and EBC programming for splines is not supported.

- Modal switchings in the retrace are only supported for the functions listed in [tab. 6-1 "Switchable NC functions in the retrace area" on page 58](#).
- Leaving and returning to contour is locked in the active retrace mode.

Relevant NC functions

Retrace(1)	Labeling the beginning of an area in which retracing is possible
Retrace(0)	Labeling the end of a retrace area
LockRetrace(1)	Labeling the beginning of a program area in which retrace switchings are locked
LockRetrace(0)	Labeling the end of a program area in which retrace switchings are locked
ADVA(<NCF>...<NCF>)	Inserts the transmitted NC functions in the forward program line
ADVS(<NCF>...<NCF>)	Replaces the forward program line by the transmitted NC functions
REVA(<NCF>...<NCF>)	Inserts the transmitted NC functions in the back ward program line
REVS(<NCF>...<NCF>)	Replaces the return program line by the transmitted NC functions
FR(<Coordinate>)	Hiding an infeed motion

Tab. 6-2: Relevant NC functions

Relevant machine parameters (MP)

NCO/Mem/ ChMem/Ch[]/ NofBlkRetrace (7060 00610)	Number of blocks for retrace Defines the size of the retrace buffer. This buffer stores the blocks for which retracing is to be facilitated. Using this parameter you can specify by how many NC blocks driving back on the path is possible. Note that blocks not triggering traversing motions are counted as well e.g., subroutine calls: N100 P SR. The parameter is channel-specific.
--	--

Tab. 6-3: Relevant machine parameters

Relevant IF signals

qCh_Retrace	Activates the retrace mode
qCh_RetraceMovFwd	
	Path motion forward in retrace mode
qCh_RetraceMovRev	
	Path motion backward in retrace mode
iCh_RetraceAct	Retrace mode active

Tab. 6-4: Relevant interface signals

Supported spline programming

Spline type	Parameterization	Subtype	Degree
1	1,2,3	1,2,3	3
2	1,2,3	0	3
3 *)	1,2,3	0	1,2,3,4,5

*) Without additional PL and/or PW programming

Tab. 6-5: Supported spline programming

6.1.2 Applying

1. In the machine parameter 7060 00610
NCO/Mem/ChMem/Ch[]/NofBlkRetrace
enter the number of blocks to be provided for the corresponding retrace mode / channel for the retrace mode.
2. Link the corresponding, relevant interface signals in the PLC program
3. Instrument the corresponding part program passages for which retracing is to be allowed by "Bracketing" with Retrace(1) and Retrace(0).
4. Add missing functions in the forward and the return programs with ADVA, ADVS, REVA and REVS.
5. Use FR(<Infeed coordinate>) to program the suppressed infeed motions in retrace mode.
6. For leaving and returning to the contour, apply the height adjustment in the re-entry ASUP as well as stopping and continuing of the part program.

6.1.3 Activating

1. Activate the retrace mode by setting the interface signal **qCh_Retrace**.
2. By setting the interface signal **qCh_RetraceMovRev**, it can be traversed backward based on the blocks buffered in the retrace buffer.
3. Use the following interface signal in retrace mode
qCh_RetraceMovFwd
to traverse forward. Within the block containing the abort position, it not possible to traverse forward up to the end point in retrace mode!

6.1.4 Deactivating

1. Deactivate the reverse mode by resetting the interface signal **qCh_Retrace**. The control system then changes to the "NC ready" state.
2. Continue machining with NC start. If an asynchronous subroutine is logged into the system date "SysRetrace", it is called first.

7 Safety/monitoring

7.1 Dimension of the working plane

7.1.1 Description

Using the NC function **"DimPlane"**, the dimension (i.e. the size) of the programmed machining planes can be determined. This calculation can be done without a motion in a calculation run before the actual machining starts. In case of extremely large workpieces, it can be determined if the size of the material is sufficient to complete machining of the entire program.

The function is enabled by programming

Program:

```
N100 DimPlane(1)
```

Following the activation, all values are initialized: Minimum values = infinite / maximum values = -infinite.

As long as the function is active, the minimum and maximum positions resulting from the NC program are internally determined. This enables the calculation of the minimum and maximum values in all three coordinate systems Acs (axis coordinates), Mcs0 (machine coordinates without ZO) and Bcs (basic coordinates). The extreme values are determined individually for each channel axis or channel coordinate.

The calculation is completed with

Program:

```
N100 DimPlane(0) or DimPlane( )
```

Until the next activation (i.e. programming of "DimPlane(1)"), the determined values are displayed on the system date. After values are zeroed after the control startup.

7.1.2 System date **"/SysDimPlane"**

General information

The extreme values are internally saved on the volatile channel-specific system date **"/SysDimPlane"**. The system date has to be created by the user. The following structural elements are defined as template in the schema **"sddimp.xsd_"** (see handling instruction).

After values are zeroed after the control startup.

With the next activation (i.e. Programming of "DimPlane(1)"), all values are initialized: Minimum values = infinite / maximum values = -infinite.

The calculated values are pending on the system date until the next activation.

Structure element	Description
MinAcs	Minimum values of the channel axes
MaxAcs	Maximum values of the channel axes
MinMcs0	Minimum values of the machine coordinates in the channel (without ZO)
MaxMcs0	Maximum values of the machine coordinates in the channel (without ZO)

Structure element	Description
MinBcs	Minimum values of the basic coordinates in the channel
MaxBcs	Maximum values of the basic coordinates in the channel

Tab. 7-1: Overview on structural elements SysDimPlane

Minimum values of the channel axes

Minimum values of the channel axes		MinAcs[1..8]
	Returns the minimum axis positions per channel.	
Path	/SysDimPlane[<Channel>]/MinAcs[1..8] CPL: SD.SysDimPlane[<Channel>].MinAcs[1..8]	
Value range	-	
Pre-assignment	0	
Unit	-	
Access rights	MTB: Read (read-only) Setter: Read (read-only) User: Read (read-only) NC prog.: Read (read-only) PLC prog.: Read (read-only)	
Activation	By the NC command "DimPlane(1)"	
Update	By the NC kernel	
Dependency	-	
Data type	8 x Float_t (8 x 4 Bytes)	

Maximum values of the channel axes

Maximum values of the channel axes		MaxAcs[1..8]
	Returns the maximum axis positions per channel.	
Path	/SysDimPlane[<Channel>]/MaxAcs[1..8] CPL: SD.SysDimPlane[<Channel>].MaxAcs[1..8]	
Value range	-	
Pre-assignment	0	
Unit	-	
Access rights	MTB: Read (read-only) Setter: Read (read-only) User: Read (read-only) NC prog.: Read (read-only) PLC prog.: Read (read-only)	
Activation	By the NC command "DimPlane(1)"	
Update	By the NC kernel	
Dependency	-	
Data type	8 x Float_t (8 x 4 Bytes)	

Minimum values of the machine coordinates (without ZO)

Minimum values of the machine coordinates		MinMcs0[1..8]
Returns the minimum values of the machine coordinates (without ZO).		
Path	/SysDimPlane[<Channel>]/MinMcs0[1..8] CPL: SD.SysDimPlane[<Channel>].MinMcs0[1..8]	
Value range	-	
Pre-assignment	0	
Unit	-	
Access rights	MTB:	Read (read-only)
	Setter:	Read (read-only)
	User:	Read (read-only)
	NC prog.:	Read (read-only)
	PLC prog.:	Read (read-only)
Activation	By the NC command "DimPlane(1)"	
Update	By the NC kernel	
Dependency	-	
Data type	8 x Float_t (8 x 4 Bytes)	

Maximum values of the machine coordinates (without ZO)

Maximum values of the machine coordinates		MaxMcs0[1..8]
Returns the maximum values of the machine coordinates (without ZO) per channel.		
Path	/SysDimPlane[<Channel>]/MaxMcs0[1..8] CPL: SD.SysDimPlane[<Channel>].MaxMcs0[1..8]	
Value range	-	
Pre-assignment	0	
Unit	-	
Access rights	MTB:	Read (read-only)
	Setter:	Read (read-only)
	User:	Read (read-only)
	NC prog.:	Read (read-only)
	PLC prog.:	Read (read-only)
Activation	By the NC command "DimPlane(1)"	
Update	By the NC kernel	
Dependency	-	
Data type	8 x Float_t (8 x 4 Bytes)	

Minimum values of the basic coordinates

Minimum values of the basic coordinates		MinBcs[1..8]
Returns the minimum values of the basic coordinates per channel.		
Path	/SysDimPlane[<Channel>]/MinBcs[1..8] CPL: SD.SysDimPlane[<Channel>].MinBcs[1..8]	
Value range	-	

Pre-assignment	0
Unit	-
Access rights	MTB: Read (read-only) Setter: Read (read-only) User: Read (read-only) NC prog.: Read (read-only) PLC prog.: Read (read-only)
Activation	By the NC command "DimPlane(1)"
Update	By the NC kernel
Dependency	-
Data type	8 x Float_t (8 x 4 Bytes)

Maximum values of the basic coordinates

Maximum values of the basic coordinates		MaxBcs[1..8]
Returns the maximum values of the basic coordinates per channel.		
Path	/SysDimPlane[<Channel>]/MaxBcs[1..8] CPL: SD.SysDimPlane[<Channel>].MaxBcs[1..8]	
Value range	-	
Pre-assignment	0	
Unit	-	
Access rights	MTB: Read (read-only) Setter: Read (read-only) User: Read (read-only) NC prog.: Read (read-only) PLC prog.: Read (read-only)	
Activation	By the NC command "DimPlane(1)"	
Update	By the NC kernel	
Dependency	-	
Data type	8 x Float_t (8 x 4 Bytes)	

7.1.3 Handling instruction

Creating the system data Start "IndraWorks Engineering", use the Project Explorer to go online and open the NC file system of IndraWorks Operation.

1. Open the "feprom" directory.
2. Copy the file "SDDefDimP.xml_" to "usrfep" and remove the underscore. The "SDDefDimP.xml_" file is now contained in "usrfep".
3. Open the "feprom/schemes" directory.
4. Copy the "sddimp.xsd_" to "usrfep/schemas" and delete the underscore.



If it does not exist yet, create the "schemas" folder below "usrfep".

The "sdcta.xsd" file is now available in "usrfep/schemas".

5. Restart the NC.

The system date is now available in the control.

Determining the extreme values

Step 1: Program the the following in the NC program for the plane to be checked.

Program:

```
N100 DimPlane(1) : N200 DimPlane(0) N1000 ; <- Target block
```

Step 2: Trigger the "Block pre-run" function. The NC blocks are calculated until the "Abort block" (i.e. to the "Target block - 1"), however, no traversing motion is executed. The rough interpolation between "DimPlane(1)" and "DimPlane(0)" is active during the computation. This slows down the block pre-run but the NC can calculate the minimum and maximum values without a traversing motion.

Step 3: Example for channel 2:

In the NC program, the required expansion in the (x, y) plane in channel 2 can be checked using the following variables.

Program:

```
1010 XMIN = SD.SysDimPlane[2].MinBcs[1]  
1011 XMAX = SD.SysDimPlane[2].MaxBcs[1]  
1020 YMIN = SD.SysDimPlane[2].MinBcs[2]  
1021 YMAX = SD.SysDimPlane[2].MaxBcs[2]
```

8 Path motion

8.1 TCS travel motion

8.1.1 Description

Function

Even if the axis transformation 2 is active, this function permits traversing in the tool coordinate system TCS. The function takes effect locally, i.e. it is applicable to separate blocks. It is a relative dimension programming.

During machining, it is often necessary to traverse in tool direction (e.g. milling cutter and drill). With the described above, such a motion can also be achieved for axis transformation 2 by programming one single coordinate.

The TcsMove (in short: TCM) function permits selected linear travel motions in the tool coordinate system (TCS). This is also possible for the active axis transformation 2.

It is a local relative dimension programming (that means that the programmed position specified is interpreted incrementally irrespective of G90/G91).

The method of operation of this function depends on the current orientation movement, i.e. on the active axis transformation 2: Tensor orientation

Tensor orientation: This is the most general case. It is used with an axis transformation whose transformation type (Macoda parameter 1030 00110) is equal to 3 at the hundredth place.

In this case, all three degrees of freedom of the TCS are uniquely defined. Thus, travel motions in TCSx, TCSy, and TCSz are possible. Vector or linear orientation

Vector or linear orientation: It is used for an axis transformation whose transformation type (MACODA parameter 1030 00110) is equal to 2 (vector) or 1 (linear) at the 100th place.

In this case, only TCSz is uniquely defined. TCSx and TCSy are not defined. Thus, travel motions are only possible in TCSz. Any programming of TCSxp0 or TCSyp0 causes a runtime error.

No Orientation: It is used with an axis transformation whose transformation type (MACODA parameter 1030 00110) is equal to 0 at the 100th place.

Without plane selection, the TCS is identical to the WCS. If G17, G18, G19 or G20 is active, the TCSz is applied perpendicularly to the selected plane. Only travel motions are possible in TCSz. Any programming of TCSxp0 or TCSyp0 causes a runtime error.

Restrictions

It is not permitted to program coordinates together with the TcsMove command in one NC block.

Only a TcsMove command may be programmed in an NC block.

Syntax

TcsMove ({<xVal>}, {<yVal>}, {<zVal>})	
with	
<xVal>, <yVal>, <zVal>	Values to be traveled incrementally in respect to the tool coordinate system TCS
Short form: TCM(,,)	

Tab. 8-1: Syntax

Example:

TCM (, , -100)	Plunging 100 [mm] in tool direction
TCM (, , 200)	Retracting 200 [mm] in tool direction

Tab. 8-2: Example - syntax

8.2 PTP traversing motion

8.2.1 Description

Function

When the axis transformation is active, this function allows for a **PointToPoint** traversing motion (PTP), i.e. a motion in MCS or ACS without deactivating the axis transformation. The programmed end position can either be specified in WCS, MCS, or ACS.

For an exact term definition, the following figure briefly illustrates the interaction of the MTX-internal coordinate systems.

The individual coordinate systems are described in detail in the MTX Functional Description 14VRS Basics (DOK-MTX***-NC*F*BA*V1x-RE01-EN-P).

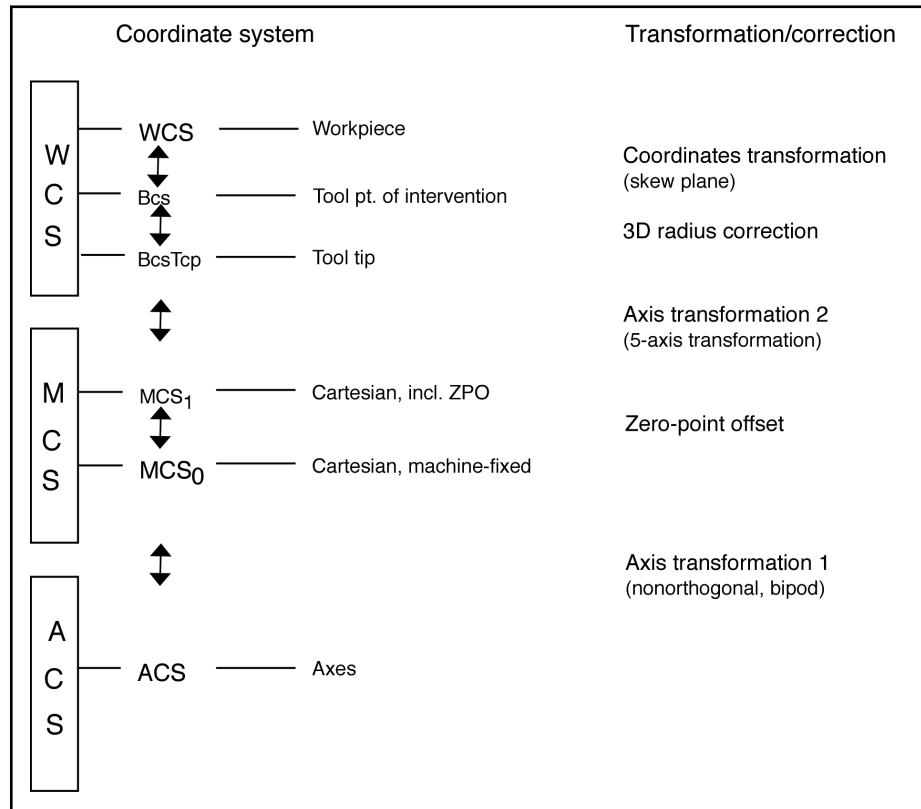


Fig. 8-1: Structure of the MTX coordinate systems

In this chapter, ACS names the axis coordinate system, MCS the Cartesian machine coordinate system MCS0 fixedly connected to the machine body, and WCS is the workpiece coordinate system fixedly connected to the workpiece.

Contrary to a pure WCS or BCS motion, a PTP traversing motion can be used advantageously to:

- Bypass or cross singularities

- Directly approach an initial position (e.g. changing the transformation branch)
- *Cycle time reduction (fastest motion possible)*
 - Each axis orients at its maximum velocity
 - The weakest axis involved defines the total motion
 - No control of TCP and path velocity
(⇒ unsuitable for machining)
 - Avoiding collisions

If the axis transformation is active, the standard motions (e.g. G0, G1, G3, G6 etc.) take place in the WCS or BCS. The end position is specified in the same coordinate system (WCS, BCS). The PtpMove function (short form PTP) additionally allows a motion in the MCS or ACS without deactivating the axis transformation. Furthermore, the position can be specified irrespective of the motion coordinate system in the WCS, MCS or ACS. The respective conversion is performed automatically in the control.

A PTP traversing motion is not suitable for machining (directly at the work-piece), as the tool center point (TCP) is not controlled on the path between the starting and the end point.

However, the PTP motion is suitable for approaching specified initial or axis positions as fast as possible. This also allows to pass singularities of the axis transformation or to switch the transformation branch on purpose.

It is a synchronous Ptp movement, i.e. all axes involved start the movement simultaneously and reach the programmed target point simultaneously. The weakest involved axis specifies the resulting velocity, as it moves at its maximum velocity over the whole distance. Thus, the motion from starting point to end point is always performed as fast as possible.

The function takes effect locally, i.e. it is applicable to separate blocks. This is an absolute programming.

Restrictions

A PTP traversing motion is always interpreted as absolute programming, i.e. G91 is ignored. Use the programming attribute "IC" (e.g. x=IC(50)) for incremental programming.

During a PTP traversing motion, the 3D radius correction has to be deactivated. Otherwise, a runtime error is generated.

Only PTP command can be programmed per line.

Syntax

PTP (<EndposSys>, <MveSys>, TB<Mode>) and programming of the end position in EndposSys (alternatively: PtpMove (,)) with	
<EndposSys>	Coordinate system in which the end position has been programmed. Valid WCS: Programmed end position in the workpiece coordinates MCS: Programmed end position in the machine coordinates AC: Programmed end position in the axis coordinates

Path motion

<MveSys>	Coordinate system in which the motion is carried out. Valid MCS: Motion in machine coordinates ACS: Motion in axis coordinates
TB<Mode>	TrafoBranch To control the transformation branch change in case of dual branch axis transformations. Only possible for the PTP(WCS,MCS) variant. "Mode" can have the following values: 0(Default): The position closer to the starting point is approached out of the two possible MCS positions MCS(+1) and MCS(-1). If required, the transformation branch is changed from +1 to -1 or vice versa. 10: Retain the transformation branch during the motion. 11: Change the transformation branch during the motion. +1: Move to transformation branch +1, i.e. move to position MCS(+1) -1: Move to transformation branch -1, i.e. move to position MCS(-1) Further properties: TB is ignored in case of one branch transformations. Only the PTP(WCS,MCS) variant supports TB programming. A syntax error is generated for other variants.

Tab. 8-3: Syntax

In each case, "MCS" is used as default value.

Example:

```

PTP (WCS, MCS) x50 y55 ; WCS coord. x=50, y=55 is approached,
                        ;Motion in the machine coordinates

PTP (WCS, ACS) x=IC(50) ;WCS coord. x is traversed incrementally by 50,
                        ;Motion in the axis coordinates

PTP (MCS, MCS) X100 Y200 ;MCS coord. X=100, Y=200 is approached,
                        ;Motion in the machine coordinates

PTP (MCS, ACS) X101 Y201 ;MCS coord. X=101, Y=201 is approached,
                        ;Motion in the axis coordinates

PTP (ACS, MCS) XA50 C45 ;ACS position. XA=50, C=45 is approached,
                        ;Motion in the machine coordinates

PTP (ACS, MCS) XA=IC(22) ;ACS position. XA incrementally 22 traversed,
                        ;Motion in the axis coordinates

```

Explanation:	PTP ()	Corresponds to PTP (MCS, MCS)
	PTP (,)	Corresponds to PTP (MCS, MCS)
	PTP (WCS)	Corresponds to PTP (WCS, MCS)
	PTP (WCS,)	Corresponds to PTP (WCS, MCS)
	PTP (, ACS)	Corresponds to PTP (MCS, ACS)
Examples for TB programming with 5-axis transformation 3232201:	B10 C0	Approaching the initial orientation Phi=0, Theta=30
	COORD(1)	
	G1 phi0 theta30	The axes move to B30, C0 (transformation branch is +1)
	PTP (WCS,,TB0) phi180 theta30	MCS(-1) is approached out of the two possible MCS positions MCS(+1) = (B30,C180) and MCS(-1) = (B-30,C0) (shortest distance with transformation branch change)
	PTP (WCS,,TB10) phi0 theta30	Transformation branch is not changed: MCS(-1) = (B-30,C180) is approached
	PTP (WCS,MCS,TB11)	Transformation branch is changed: MCS(+1) = (B30,C0) is approached. WCS position is still Phi=0, Theta=30
	PTP (WCS,,TB+1)	Approach transformation branch +1. No motion as transformation branch is already +1
	PTP (WCS,MCS,TB-1)	Approach transformation branch -1. MCS(-1) = (B-30,C180) is approached. WCS position is still Phi=0, Theta=30

9 Functions for drive parameterization

9.1 Functions for Sercos handling

9.1.1 Overview

The NC provides a number of functions which significantly simplify the handling with the connected Sercos drives:

- Sercos initialization for all connected Sercos drives during control start-up.
- Automatic parameter download for all connected Sercos drives during control startup.
- Program-controlled read-only access to the Sercos parameters of all connected drives using CPL commands SCS3(..) und SCS3L(..).
Program-controlled write access to Sercos parameters of all connected drives by the NC command "Writeld3" (WID3) and the CPL command "WSCS".
- For detailed information, refer to the "MTX Programming Manual".
- *Selectable debug functions to protocol the following during the NC start-up:*
 - the Sercos parameters transmitted in the phases 2 and 3 (see from [chapter 9.4 Logging Transmitted Sercos Parameters, page 82](#))

9.2 Sercos initialization

9.2.1 Description

Function

The tasks of the Sercos initialization of the NC are as follows:

- Closing the Sercos ring. Error diagnostics during problems on the Sercos transmission
- Structure of the Sercos timing and the communication channels to the connected drives
- Starting up the phases of all Sercos drives up to phase 4 (cyclic operation is enabled)



The Sercos initialization can be executed either with or without automatic parameter download in the drive. For information on the automatic parameter download, refer to [chapter 9.3 Automatic Parameter Download, page 79](#).

Relevant machine parameters (MP)

1050 00001	Drive type
1050 00002	Drive exists in the ring
1050 00004	Assignment of drives to the Sercos address
1050 00020	Enabling test mode
1050 00021	Time monitoring of Sercos startup
9030 00001	NC cycle time

Tab. 9-1: Relevant machine parameters

9.2.2 Applying



The tool "IndraWorks Drive" integrated in the GUI performs the basic initialization of the drives.

NOTICE

For an enabled parameter download, the current drive parameters are overwritten with parameter values in the Sercos files of the NC.

If the Sercos files are not adapted, unintended drive reactions or dangerous machine states can result.

Thus, disable the automatic parameter download for all drives for which there are presently no adapted Sercos files in the NC.

For specifications on the deactivation of the parameter download, refer to [Chapter 9.3.4 Deactivating, page 81](#).

Starting the parameterization and initialization:

1. Switch off the power supply of the drives.
2. Adapt the machine parameter group 1050 to your application.
3. Set the respective Sercos address and baud rate at the individual drives.

Use the information given in the drive documentation.

4. Trigger control reset.
5. After NC startup, check which Sercos phase is specified.
Ideally, the NC (see IndraWorks Drive) and all connected drives should indicate phase "4".

If phase 4 is not indicated, note:

- IndraWorks Drive displays phase "0":
 - The NC cannot close the ring (the transmitted MST does not return to the NC).
Verify if all Sercos devices in the ring are active.
 - The NC cannot find a specified drive.
Check whether all Sercos addresses parameterized in the NC (MP 1050 00004) correspond to the addresses of the drives
- IndraWorks Drive displays phase "1":
The NC cannot find a specified drive.
Check whether all Sercos addresses parameterized in the NC (MP 1050 00004) correspond to those of the drives.
- Drive displays phase "2":
The NC could not establish a connection or set up a correct timing to the drive or
the drive could not be switched up to phase 3 due to a parameterization problem.
Check the drive parameterization (if automatic parameter download is activated, check the respective Sercos file for phase 2 the NC. If appli-

cable, activate the logging function for transmitted Sercos parameters: see [chapter 9.3.3 Enabling, page 81](#)).

- Drive displays phase "3":

The drive could not be switched to phase 4 due to a parameterization or timing problem.

Check the drive parameterization (if automatic parameter download is activated, check the respective Sercos file for phase 3 the NC. If applicable, activate the logging function for transmitted Sercos parameters: see [chapter 9.3.3 Enabling, page 81](#)).



Do not switch on the power supply for drives not yet correctly parameterized!

9.3 Automatic parameter download

9.3.1 Description

Function

If the parameter download is active, the NC can completely parameterize connected drives at startup.

Prerequisite: Correctly adjusted Sercos files (*.scs) in the NC. Sercos files are always saved in ASCII format.

NOTICE

When the parameter download is enabled, the current drive parameters are overwritten with parameter values from the Sercos files of the control.

If the Sercos files are not adapted, unintended drive reactions or dangerous machine states can result.

Thus, disable the automatic parameter download for the drives currently without adapted Sercos files in the control.

For specifications on the deactivation of the parameter download, refer to [Chapter 9.3.4 Deactivating, page 81](#).

9.3.2 Applying

The two files "load_ph2.scs" and "load_ph3.scs" are significant for the parameter download.

Specify in these files which Sercos files are transmitted to which drives via the "use..." command.

The following files are used for transmission:

- Phase 2: Datei "load_ph2.scs"
- Phase 3: Datei "load_ph3.scs"

Functions for drive parameterization

Syntax of the "use" command:

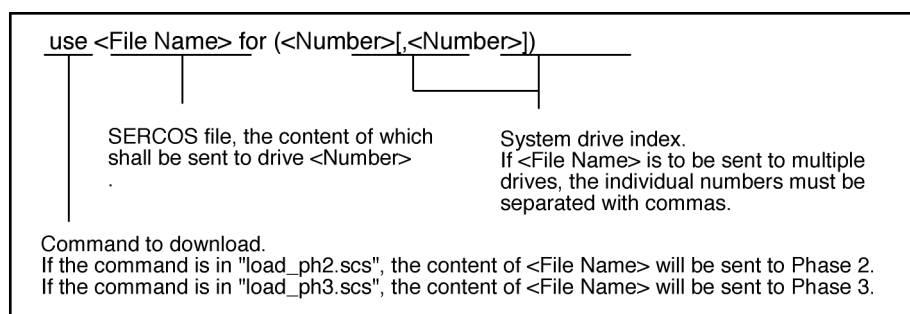


Fig. 9-1: Syntax of the "use" command



MP 1050 00022 ("SCS files Allow disabling phase 3") defines the behavior of the NC with regard to all Sercos files listed in "load_ph3.scs".

If you allow disabling, the files are sent to the drive in question only when this is required (e.g. following a modification).

For the corresponding criteria, refer to [chapter 9.5 Downloading Sercos Files in Phase 3, page 83](#).

All files which can be used for the automatic download in "load_ph2.scs" and "load_ph3.scs" via the "use" command can include:

- Comment lines
- Lines for drive parameterization

Comment lines always start with a semicolon (;). The NC interprets all characters from the semicolon up to the end of the line as comment. Comments are not transferred to the drives. They are only used to improve structuring and overview.

Lines for the drive parameterization are structured as follows:

<ID number > = <Value> [; <CommentText >]

The following applies:

<ID number>	Sercos parameter in the format S-x-xxxx or P-x-xxxx (S- or P-parameters). Refer to the drive documentation for the available Sercos parameters.	
<Value>	Parameter value in the following formats:	
	Decimal:	e.g.: 500
	Binary:	e.g.: 0b...
	String:	e.g. "Text"
Parameter list	e.g.: (S-0-0047,S-0-0189)	
<CommentText>	Comment; has to be separated from<Value> by at least one space and one semicolon	

Tab. 9-2: Lines for the drive parameterization

; This is a comment line	(Comment)
S-0-0121 = 1 ;input rev.	(Parameter with comment)
S-0-0122 = 1	(Decimal value)
S-0-0032 = 0b0011	(Binary value)

S-0-0142 = "Application (String)
type"

S-0-0016 = (S-0-0051,S-0-0189) (ID list)



If **all** adapted Sercos files (*.scs) required for your application are already available, copy all relevant Sercos files into the "root directory" ("/") of the NC (for the directory structure, refer to the manual "Operating Instructions"). To enable or disable, proceed as described in the following sections.

If **no** Sercos files (*.scs) were created for your application, create new files in the "root directory" ("/") of the NC by means of the editor or copy the example files enclosed in the delivery to the "root directory" ("/") and adapt them to your requirements.

9.3.3 Activating

1. In the "load_ph2.scs" or "load_ph3.scs" files, delete the character ";" to the left of the relevant "use" commands (see above for the syntax).

Thus, **at next startup**, the NC no longer interprets these lines as comment lines but as download commands.

2. Save the modified file(s).
3. Create backup copies of the files.

Example:

Enable

Automatic download (in phase 2) of the files

- "p2linall.scs" in the drives with the system drive numbers 1, 2 and 4
- "p2lin3.scs" in the drive with the system drive number 3.

Excerpt from the file "load_ph2.scs" (phase 2!):

```
:
use p2linall.scs for (1,2,4)
use p2lin3.scs for (3)
:
```

9.3.4 Deactivating

1. In the "load_ph2.scs" or "load_ph3.scs" files, set the character ";" to the left of the relevant "use" commands (see above for the syntax).

Thus, **at next startup**, the NC no longer interprets these lines as download commands but as comment lines.

2. Save the modified file(s).
3. Create backup copies of the files.

Example:

Disable

Deactivate the automatic download for the two files from the example shown above:

Excerpt from the file "load_ph2.scs" (phase 2!):

```
:
```

```
;use p2linall.scs for (1,2,4)
;use p2lin3.scs for (3)
:
```



Provided that there are backup copies of all relevant Sercos files in the "user FEPROM" ("/usrfep"), it is not sufficient to deactivate the automatic parameter download to just delete or rename the relevant Sercos files in the "root directory" of the NC (if the NC does not find these files in the root directory, it searches for them also in the "user FEPROM")!

9.4 Logging the transmitted Sercos parameters

9.4.1 Description

Function

All Sercos parameters transmitted in phase 2 or phase 3 can be logged for diagnostic purposes.

Program the "dbg" command in the "load_ph2.scs" file (for phase 2) or in the "load_ph3.scs" file (for phase 3).



The "dbg" command has to be programmed prior to the first "use" command.

Syntax of the "dbg" command:

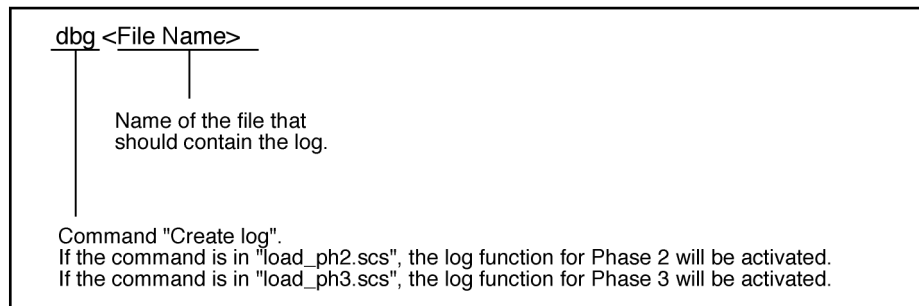


Fig. 9-2: Syntax of the "dbg" command

9.4.2 Activating

1. In the "load_ph2.scs" or "load_ph3.scs" files, delete the character ";" to the left of the relevant "dbg" command.

Following that, the NC, **at its next startup**, no longer interprets the line as comment line but as a command to log the Sercos parameters.

2. Save the modified file(s).

9.4.3 Deactivating

1. In the "load_ph2.scs" or "load_ph3.scs" files, set the character ";" to the left of the relevant "dbg" command.

Thus, **at next startup**, the NC interprets the line as comment line.

2. Save the modified file(s).

9.5 Downloading Sercos files in phase 3 "Only if required"

9.5.1 Description

Relevant machine parameters (MP)

1050 00022	Permit hiding SCS files of phase 3
------------	------------------------------------

Tab. 9-3: Relevant machine parameters

If the parameter download is active, this function reduces the required startup time of the machine.

Sercos files **always** sent to the drives in phase 3 (as defined in "load_ph3.scs") are now transmitted by the NC **only if** it is highly probable that the parameters in this file differ from the current drive configuration.

The "saveph3.scs" file in the NC is significant when identifying these differences:

- "saveph3.scs" content:
In "saveph3.scs", name and date of all Sercos files to be transmitted for phase 3 are saved. At Sercos startup, the NC compares this data to the name and date of the corresponding current Sercos files. If the data is not identical, the respective Sercos files have probably been modified (or at least loaded into an editor and saved again). The NC sends the corresponding Sercos files again to the drive and updates "saveph3.scs".
- Date of the "saveph3.scs" file:
As soon as the "Save main memory" command is triggered via a soft key, the drive writes its data from the RAM to its FEPRAM (otherwise, the current drive parameterization is lost when the drive is switched off or reset).
The drive parameter S-0-0142 (application type) containing the date of "saveph3.scs" is also taken into account.
At Sercos startup, the NC reads S-0-0142 from the drive and compares the value with the current date of "saveph3.scs".
If the dates differ, "Save main memory" was not triggered via softkey after updating "saveph3.scs". The NC transmits again all Sercos files for phase 3 to the drive.

9.6 Sercos multiplex channel

9.6.1 Description

Function

The Sercos multiplex channel allows to access the Sercos parameters of the Sercos drive. The Sercos multiplex channel is a fast alternative to the relatively slow Sercos service channel. However, the desired Sercos parameters have to be entered into a multiplex parameter list before. The parameters included in this list are the only parameters that are transferred via the Sercos multiplex channel.

If possible, the following transfer mechanisms use the multiplex channel:

- NC instructions to write Sercos parameters ("WID3")
- CPL instructions to write Sercos parameters ("WSCS")
- CPL instructions to read Sercos parameters ("SCS3")

Special features and restrictions

- PLC function block "MT_ScsData"

The Sercos multiplex channel can also be used to write or read single elements from Sercos lists.

- The multiplex parameter lists to read and write Sercos parameters are limited to 20 entries each.
- Only 2- or 4-byte parameters can be transferred via the multiplex channel. The same applies to lists. Single-element access is only allowed for lists containing 2- or 4-byte list elements.
- Simultaneous access to single list elements from multiple instances is not allowed.

9.6.2 Parameterizing



It is recommended to make the following settings in the SCS file.

1. Include the Sercos multiplex channel in the cyclic MDT data. This requires that the two parameters S-0-368 and S-0-360 be included into the list of cyclic data.
S-0-1050.0.6 = (..., 368, 360)
2. Include the Sercos multiplex channel in the cyclic MDT data. This requires that the two parameters S-0-368 and S-0-364 be included into the list of cyclic data.
S-0-1050.1.6 = (..., 368, 364)
3. Sercos parameters to be transferred via the multiplex channel must be entered in the multiplex parameter lists for command values (S-0-370) or feedback values (S-0-371), respectively, depending on the direction in which they are transferred.
Example: S-0-370 = (36, 92, 37, ...)

9.6.3 Transferring List Elements

1. Sercos list parameters to be transferred via the multiplex channel must be entered in the multiplex parameter lists for command values (S-0-370) or feedback values (S-0-371), respectively, depending on the direction in which they are transferred.
2. Before a list element is transmitted, transfer the list index:
 - Change list element: S-0-362 = list index
 - Read list element: S-0-366 = list index
3. Subsequently, the list element addressed by the list index can be modified or read. The prerequisite is that the Sercos multiplex channel transfer mechanisms are used.

10 System data

10.1 System data (SD)

10.1.1 Description

System data (SD) is a structured variable xml-based data structure used to read and influence states or functions of the control during operation. It provides specific data already structured and formatted for all system parts.

System data

- They can be stored either in the volatile or non-volatile memory. The content of non-volatile SD is retained after system restart
- They are platform-independent by its xml-structuring and can thus be processed by internal system parts (e.g. CPL commands) or xml- or html-capable applications (e.g. MS Internet Explorer)
- They are partially based on self-defined structure types described in different structure type files, also named **xml schema files** (extension ".xsd")
- They are created on xml basis in **definition files** with predefined and/or self-defined structure types (see [Chapter 10.2 Structure types, page 87](#))
- They can be pre-assigned with xml-based default values in **initialization files** (see [chapter 10.3 Definition Files, page 88](#))
- They can be defined on three levels: NC-internal, machine tool builder, end user



For a description of the system data defined by the control manufacturer, refer to the manual "MTX 14 VRS Machine Parameters" in the chapter "System data".

System data

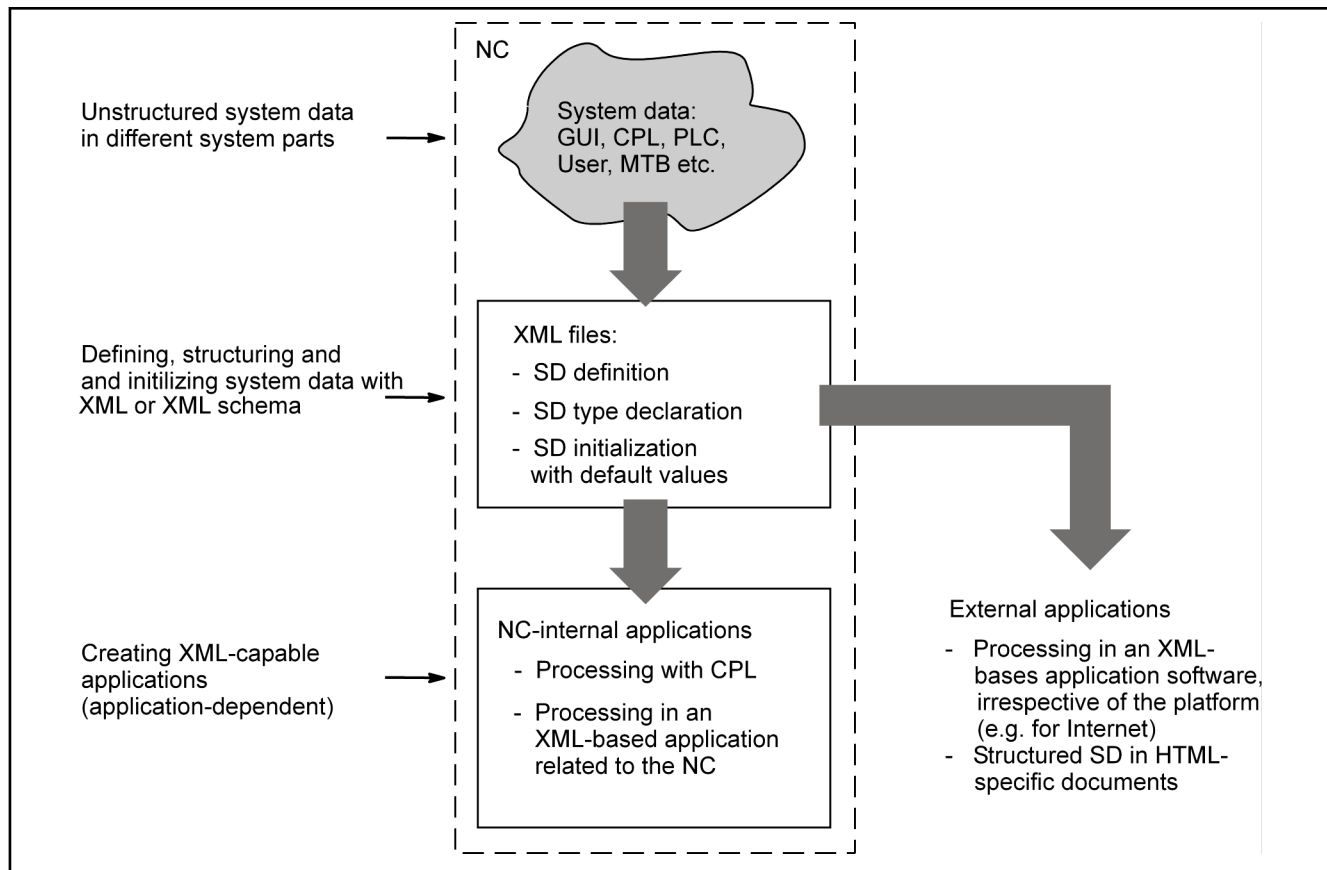


Fig. 10-1: System data

10.1.2 Accessing system data

All system parts have read and write access to SD:

- GUI: XML-based with standardized XPath
- CPL: Via SD command ("system data")
- PLC: With special function blocks on binary data
- System software or user software: Via header files directly on the SD
- Access to array elements with "[]"
- Access to structure elements with the point operator "."

10.1.3 Saving defined SD

System data is defined according to

- number (e.g. for n data elements for n channels)
- size (e.g. array size)

They are stored either volatile or non-volatile **in the NC**.

The SD is saved in the archive on XML basis. Thus, SD can be processed externally or read in again from an archive after software exchange.

Clients can be notified about the SD changes.

10.1.4 Execution tools

System data can be defined using the following execution languages:

- XML (Extensible Markup Language):

XML is a meta language to define document types (e.g. documents similar to html) and was authorized by the **World Wide Web Consortium (W3C)**.

- XML schema:

In the XML schema, SD-XML files are defined with an individual schema language. For more information on the XML schemas, go to <http://www.w3.org/2001/XMLSchema>.



The developer has to have extensive programming know-how to work with the meta languages XML and XML schema.

10.2 Structure types

10.2.1 Description

The user can define the required system data for the own application.

The NC provides the following standard structure types:

- **Basic type**
- **MTX type**

If the type classifications "Basic type" and "MTX type" are not sufficient to meet the requirements of the application, individual system data can be defined:

- **Self-defined types** (for MTB or end user)

This self-defined system data can consist of complex structure types which, in turn, can consist of a combination of the three possible types. The declaration of the self-defined types is stored in special structure type files.

10.2.2 Structure type characteristics

Basic types:

The **basic types** are derived from standard xml types. They are part of the system software and defined in an XML schema file with the name **basic_ty.xsd**.

An overview on the basic types is shown in the following table:

Name	Length in bytes	Description	C-type	IndraLogic type	CPL
String	2 maxLength+1	String (UTF-8 format)	char[]	STRING()	STRING
isoLa- tin1String	maxLength+1	String (Latin1 format)	char[]	STRING()	STRING
Byte_t	1	Signed 8-bit integer	char	SINT	INT
Short_t	2	Signed 16-bit integer	short	INT	INT
Int_t	4	Signed 32-bit integer	int	DINT	INT
Unsigned- Byte_t	1	Unsigned 8-bit integer	unsigned char	USINT	INT
Unsigned- Short_t	2	Unsigned 16-bit integer	unsigned short	UINT	INT
Unsigne- dInt_t	4	Unsigned 32-bit integer	unsigned int	UDINT	INT

Name	Length in bytes	Description	C-type	IndraLogic type	CPL
Float_t	4	32-bit real	float	REAL	REAL
Double_t	8	54-bit real	double	LREAL	DOUBLE
Boolean_t	1	true, false, 1, 0	char	BOOL	BOOLEAN

Tab. 10-1: Basic types



The basic types "string" and "isoLatin1String" cannot be used directly in the definition file SDDefxxx.xml (see [chapter 10.3 Definition files, page 88](#)), as the length of the character string is not determined. The structure type has to be determined in a schema file with the attribute "maxLength".

It has to be considered that not all non-ASCII characters (such as umlauts) consist of more than one character for strings in UTF-8 format. They cannot be processed in the NC kernel. Thus, strings in UTF-8 format should exclusively be used in the user interface.

In the CPL, only the constants TRUE and FALSE are permitted for the "Boolean" type.

Storage location:

The file "**basic_ty.xsd**" is located in the directory "/feprom/schemas".

MTX types: MTX types describe structures for existing NC functions. They are delivered together with the system software.

Storage location:

MTX types are located in the xml schema files (sd*.xsd) in the directory "/feprom/schemas".

Self-defined types: The self-defined types consist of basic types, MTX types and/or other self-defined types.

These structure types are generated by the MTB or the end user (e.g. for CPL).

Storage location:

Self-defined types (names can be freely selected) are defined in XML schema files with the extension ".xsd" and are located in the directory "/usrfep/schemas" or "/schemas".

10.3 Definition files

10.3.1 Description

System data variables are written to a definition file. In these files, the number (e.g. n data elements for n channels), the size (e.g. array size) and the structure (e.g. structured data type) are defined for the SD variable.



To create an SD definition file, a suitable text editor or XML editor (such as xmldspy) can be used.

10.3.2 Structuring a definition file (SDDef*.xml)

A definition file consists of the following elements:

- Obligatory header:

The definition file starts with the header. There, the valid XML version and the XML format are described in UTF-8 coding.

- Element **<SDDef>**:

Main element; all SDs to be defined are included in the main element.

- Element **<Variable>**:

Subelement; each SD variable is defined in a subelement **<Variable>**.

- Attribute **Dimension**:

It is only specified for arrays (up to six dimensions).

For each dimension, the number of elements has to be specified (e.g. an SD variable "Var" with the dimension "3,4" has 3 x 4 elements and is addressed with /Var[1,1] to /Var[3,4]). A dynamic number of elements can be set with the keywords **Channel**, **Drive** or **Spindle**.

In a channel-specific SD variable, the keyword **Channel** is entered instead of the number of elements of one dimension. It is then created for each applied machining channel (corresponding to the machine parameter /MAIN/NofCh).

Example:

An SD variable with the dimension "Channel,4" is created four times for each channel. It is addressed like a normal array, e.g. with /Var[1,2].

The indication of the channel number can be omitted if the proper channel is to be addressed in the CPL; for example SD.Var[,2], or - if Dimension only consists of Channel - [] can be omitted (e.g. SD.Var).

In a drive-specific SD variable, the keyword **Drive** is entered instead of the number of elements of one dimension. It is then created once for each applied drive (corresponding to the machine parameter /Main/NofDr). However, it is at least created once.

In a spindle-specific SD variable, the keyword **Spindle** is entered instead of the number of elements of a dimension. It is then created once for each applied spindle (number of elements Sp[i] existing in the machine parameters under /SP/SpFunc). However, at least one time.

- Attribute **Storage**:

Storage type

Using the attribute "Storage", a memory class can be assigned to an SD variable describing how the variable is stored in the control memory.

The memory class is defined by the following keywords:

- **permanent**

The variable is stored in the "non-volatile" memory.

The size of the memory in the memory class **permanent** is limited to 500 Kbyte.

- **volatile**

The variable is stored in the "volatile" memory.

The size of the memory in the memory class **volatile** is limited to 1 Mbyte.

- **temporary**

The variable is stored in the "volatile" memory.

The maximum size of the memory in the memory class **temporary** depends on the current control configuration (number of channels, look-ahead buffer, etc.).

If no memory class is specified, the variable is stored in the volatile memory.

- **Element Name:**

Name of the SD variables

Name specification of an SD variable with a **maximum of 14 characters**. The name can consist of letters, numbers, hyphen and underline. Upper and lower case letters are differentiated.

"SD." is placed in front of the name in the PLC. The name may not exceed 17 characters.

- **Element Type:**

Type of the SD variables

The type of an SD variable is a basic type, an MTX type or a self-defined type. These types are defined in schema files.

- **Element Comment:**

The length of the comment for further information is not restricted:

- Brief description of SD variables,
- Schema: Name of the respective schema file,
- Author: Initials of the employee that introduced the SD variable.

Example:

XML fragment of a definition file "SDDef*.xml" with the SD variables My3DimArrVar and MyChanStruct:

Program:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SDDef>
  <Variable dimension="3,4,12" Storage="volatile">
    <Name>My3DimArrVar</Name>
    <Type>MyVar_t</Type>
    <Comment>volatile three-dimensional variable</Comment>
    <Comment>Schema: sdmyvar.xsd</Comment>
    <Comment>Author: XY</Comment>
  </Variable>
  <Variable Dimension="Channel" Storage="permanent">
    <Name>MyChanStruct</Name>
    <Type>MyStruct_t</Type>
    <Comment>Permanent data structure for each applied channel</Comment>
    <Comment>Author: XY</Comment>
  </Variable>
</SDDef>
```



- **Permanent system data variables** at which the data type or subtypes of the variables were changed, are automatically created again and initialized after the next control startup. If necessary, these system data may have to be archived **before** restarting the system.
 - **Permanent system data variables** defined as array with the keywords **Channel**, **Drive** or **Spindle** are automatically created again and initialized after the next control startup if the number of channels, drives or spindles has changed due to changes in the machine parameters.
-

10.3.3 Name and storage location of the definition file

System data containing self-defined structure types of the MTB or the end user and SD defined by MTX types, are created in specially named XML-based definition files (SDDef.xml, see below):*

- System file definitions of the MTB are stored in the file **SDDefMTB.xml**. The file is searched in the system search path. The root directory "/" has a higher priority than "/usrfep" and "/feprom".
- Freely definable system file definitions of the end user are stored in the file **SDDefUsr.xml**. The file is searched in the system search path. The root directory "/" has a higher priority than "/usrfep" and "/feprom".
- Any freely definable system data definitions or such for NC diagnostic purposes are stored in the files **SDDef*.xml**. "*" has to be selected in such a way that the file name is unique. The files are searched in the system search path. The root directory "/" has a higher priority than "/usrfep" and "/feprom".
- The internal system data of the control manufacturer is defined in the file **SDDef.xml**. This file is stored in the "/feprom" directory and cannot be changed at runtime.

10.4 Structure type files (XML schema files)

10.4.1 Description

For all system data variables whose type is not an already basic type or of type "String" or "IsoLatin1String", the structure type has to be defined in an SD-specific XML schema file.



To create an XML schema file, a suitable XML editor (such as xmlspy) can be used.

10.4.2 Structuring a structure type file

The following example shows a simple structure definition.

An "Include" command follows the obligatory XML header. This instruction includes the basic types. Then, the type definitions (in this example the structure "MyStruct_t") follow. Multiple types can also be defined in one schema file. The "MyStruct_t" structure consists of a sequence of four different elements. In this case "SDIsoString", "SDByte", "SDInt" and "SDDouble". For "string" elements such as SDIsoString, the attributes "minLength" and "maxLength" have to be used to specify the minimum and maximum string length. For arrays, the number the respective element might exist, is given with the attributes "minOccurs" and "maxOccurs".

With XML, any complex, hierarchically-structured types can be defined. Thus, an element can contain a sequence of elements.

Examples of further schema files can be found in the structure definitions of the system data defined by the control manufacturer. These are located in the directory /feprom/schemas (sd*.xsd).

Program:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified" attributeFormDefault="unqualified">

  <xs:include schemaLocation="basic_ty.xsd"/>

  <xs:complexType name="MyStruct_t">
    <xs:sequence>
```

```

<xs:element name="SDIsoString">
  <xs:simpleType>
    <xs:restriction base="isoLatin1String">
      <xs:annotation>
        <xs:documentation>isoLatin1String with up to 3 characters
        </xs:documentation>
      </xs:annotation>
      <xs:minLength value="0"/>
      <xs:maxLength value="3"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
<xs:element name="SDByte" type="Byte_t" minOccurs="3" maxOccurs="3">
  <xs:annotation>
    <xs:documentation>array with 3 byte elements</xs:documentation>
  </xs:annotation>
</xs:element>
<xs:element name="SDInt" type="Int_t">
  <xs:annotation>
    <xs:documentation>single integer element</xs:documentation>
  </xs:annotation>
</xs:element>
<xs:element name="SDDouble" type="Double_t">
  <xs:annotation>
    <xs:documentation>single double float element</xs:documentation>
  </xs:annotation>
</xs:element>
</xs:sequence>
</xs:complexType>

</xs:schema>

```

10.4.3 Name and storage location of a structure type file

Structure types for system data are stored in XML schema files:

- Implemented basic types and MTX types:
These are located in the XML schema files (basic_ty.xsd, sd*.xsd) in the directory "/feprom/schemas".
They **cannot** be overwritten by schema files with the same name in the directories "/schemas" or "/usrfep/schemas".
- Self-defined types of the machine manufacturer or the user:
These are defined in XML schema files with the extension ".xsd" and stored in the directory "/usrfep/schemas" or "/schemas".
Files in the directory "/usrfep/schemas" can be overwritten in the directory "/schemas" by individual schema files.

10.5 Initialization files

10.5.1 Description

The values of all volatile system data are initialized during each control start-up. The values of all permanent system data are only initialized once. Initialization takes place at control startup when they are applied first.

Initialization values can be specified for the SD in an initialization file. For SD or SD components not entered here, the value "0" or an "empty string" applies as initialization value.



To create an SD initialization file, a suitable text editor or XML editor (such as xmlspy) can be used.

10.5.2 Structuring an SD initialization file (SDDat*.xml)

An SD initialization file is created in XML format coded as UTF-8. Thus, it starts with the obligatory XML header.

All system data to be initialized is enclosed in the main element **<SD>**. For array system data whose dimensions were defined on top level (that means in a definition file **SDDef*.xml**), the indices have to be specified for each element to be initialized with the attribute "Dimension".

For arrays in lower levels defined in a structure file type, the "Dimension" attribute is not specified. Here, **all** array elements have to be entered.

If information on system data is faulty or missing in the initialization file, the data is pre-assigned with 0 or an empty string.

Program:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SD>
  ..
  <UU>
    <Subcomponent>4711</Subcomponent>
  </UU>
  <VV Dimension="1,1">
    <Subcomponent>17</Subcomponent>
  </VV>
  <VV Dimension="1,2">
    <Subcomponent>18</Subcomponent>
  </VV>
  ..
</SD>
```

10.5.3 Name and storage location of an initialization file

Initialization files for system data are stored in specially named XML-based initialization files (SDDat.xml):*

- System data of the machine tool builder:
The initialization values are stored in the file **SDDatMTB.xml**.
- Freely definable system data of the end user:
The initialization values are stored in the file **SDDatUsr.xml**.
- Any freely definable system data or such for NC diagnostic purposes:
The initialization values are stored in the files **SDDat*.xml**. "" has to be specified in such a way that the file name is unique.
- Internal system data of the control manufacturer:
The initialization values are defined in the file **SDDat.xml**.

The files are searched in the system search path.

The root directory "/" has a higher priority than /usrfep, which, in turn, has a higher priority than /feprom.

10.6 Re-initializing selected SD elements after NC restart

Permanent system data (SD) are newly created once (upon the first restart) and initialized with "0", empty string or values from a default setting file **SDDat*.xml**.

System data (permanent and volatile SD) can be specifically re-initialized irrespective of the default setting data using the following two files (search path: "root directory --> /usrfep", templates in "/feprom"):

File	Purpose
SDForceByCmd1SnglRestart.xml	One-time re-initialization of selected SD elements after the next NC restart
SDForceByCmd2EachRestart.xml	Repetitive re-initialization of selected SD elements after each NC restart

Tab. 10-2: Files to re-initialize SD elements

The user copies the respective file template from the directory "/feprom" to the root directory, removes the character "_" at the end of the file, enters data (as in SDDat.xml) and moves the file into the "/usrfep" directory if required.

The opening of these files is controlled via a permanent SD control structure SD.SysSDForce with the following settings when created first:

Control structure element	Default value
SD.SysSDForce.SDForceCmd1SnglRestart	true
SD.SysSDForce.SDForceCmd2EachRestart	true

Tab. 10-3: Control structure SysSDForce with pre-assignment

The mentioned files are searched with these default settings upon restart in the search path "Root directory --> /usrfep" and imported when found.

After the file "SDForceByCmd1SnglRestart.xml" has been imported successfully, "SD.SysSDForce.SDForceCmd1SnglRestart" is set to false". This ensures that this file is only imported once. If it is to be imported again (e.g. after a change), it has to be triggered again first (SD.SysSDForce.SDForceCmd1SnglRestart = true).

The file "SDForceByCmd2EachRestart.xml" is imported again after each NC restart as long as "SD.SysSDForce.SDForceCmd2EachRestart" is true".

Example file *Program:*

Example file SDForceByCmd1SnglRestart.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<SD>
  <SysScsIpAdr>
    <IPAddMtx>192.168.143.253</IPAddMtx>
  </SysScsIpAdr>
</SD>
```

10.7 Handling Instruction: Example: Creation of four system data for a machine tool builder

An MTB wants to create the system data described in the following table:

MTBSimpleVar (Integer)	Simple variable, non-permanent
MTBIntArr (Integer)	Array (32 elements), permanent
MTBArrVar of the self-defined type MTBVar_t	3-dimensional array, non-permanent. "MTBVar_t" names a structure consisting of two integers.

MTBChanVar	permanent variable for each active channel. MTBChanVar_t denominates a structure with the following design:	
of the self-defined type MTBChanVar_t	Iso string of length 3	SDIsoString
	3-element byte array	SDbyte
	Simple integer variable	SDint
	Simple Double variable	SDdouble

Tab. 10-4: Specification - Creation of four system data for an MTB

In this example, the abbreviation "MTB" describes a machine tool builder to differentiate between the MTB-specific files of the system data and the other SD. For the application, a **definition file**, a **structure type file**, and an **initialization file** are generated. Access and output of the SD via CPL.

Example: Overview on SD and files

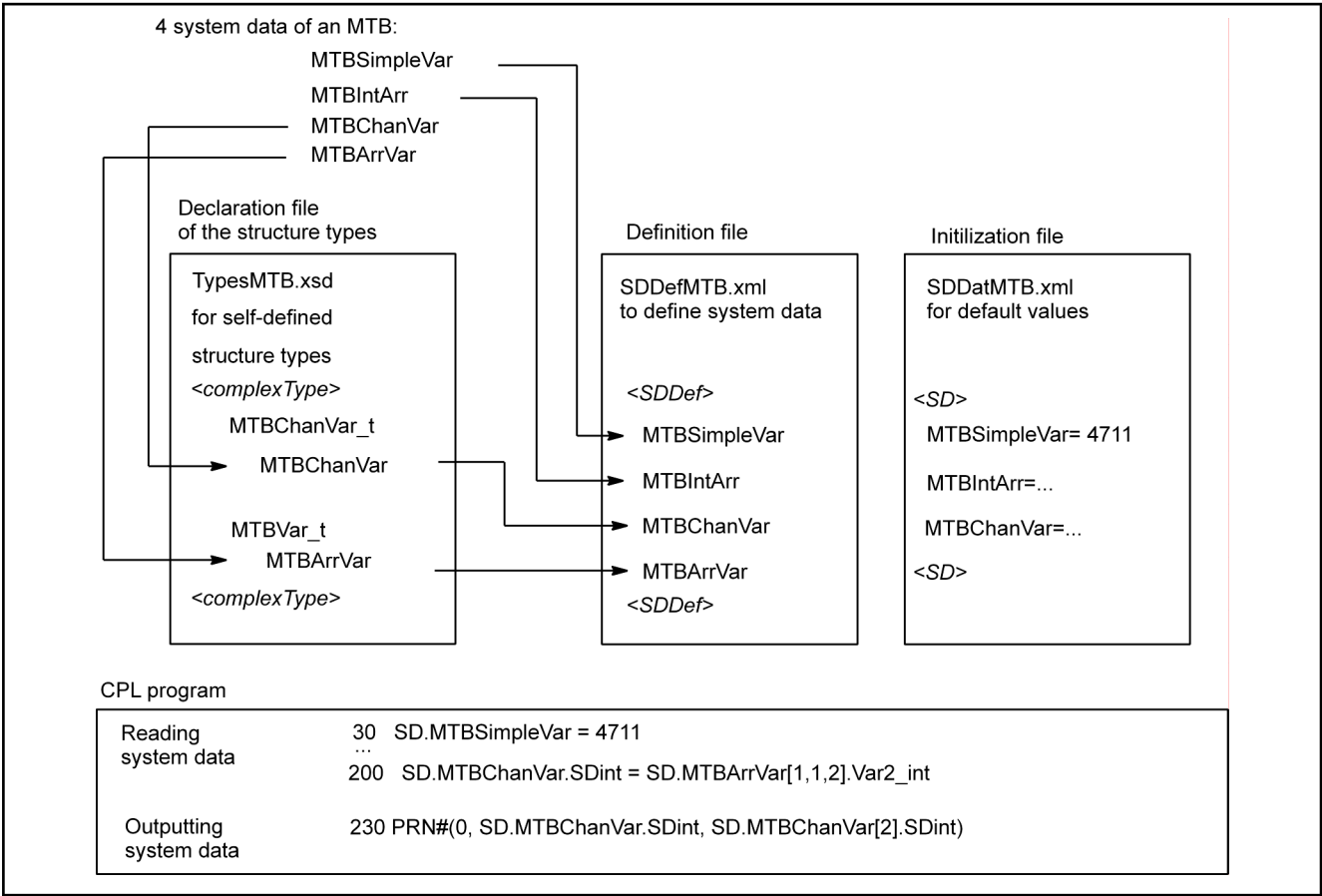


Fig. 10-2: Overview on SD files

1. Declaration of the structure types in the XML schema file "TypesMTB.xsd"

In the directory "/schemas", the file TypesMTB.xsd is generated with the complex self-defined structure variables "MTBChanVar_t" and "MTBVar_t":

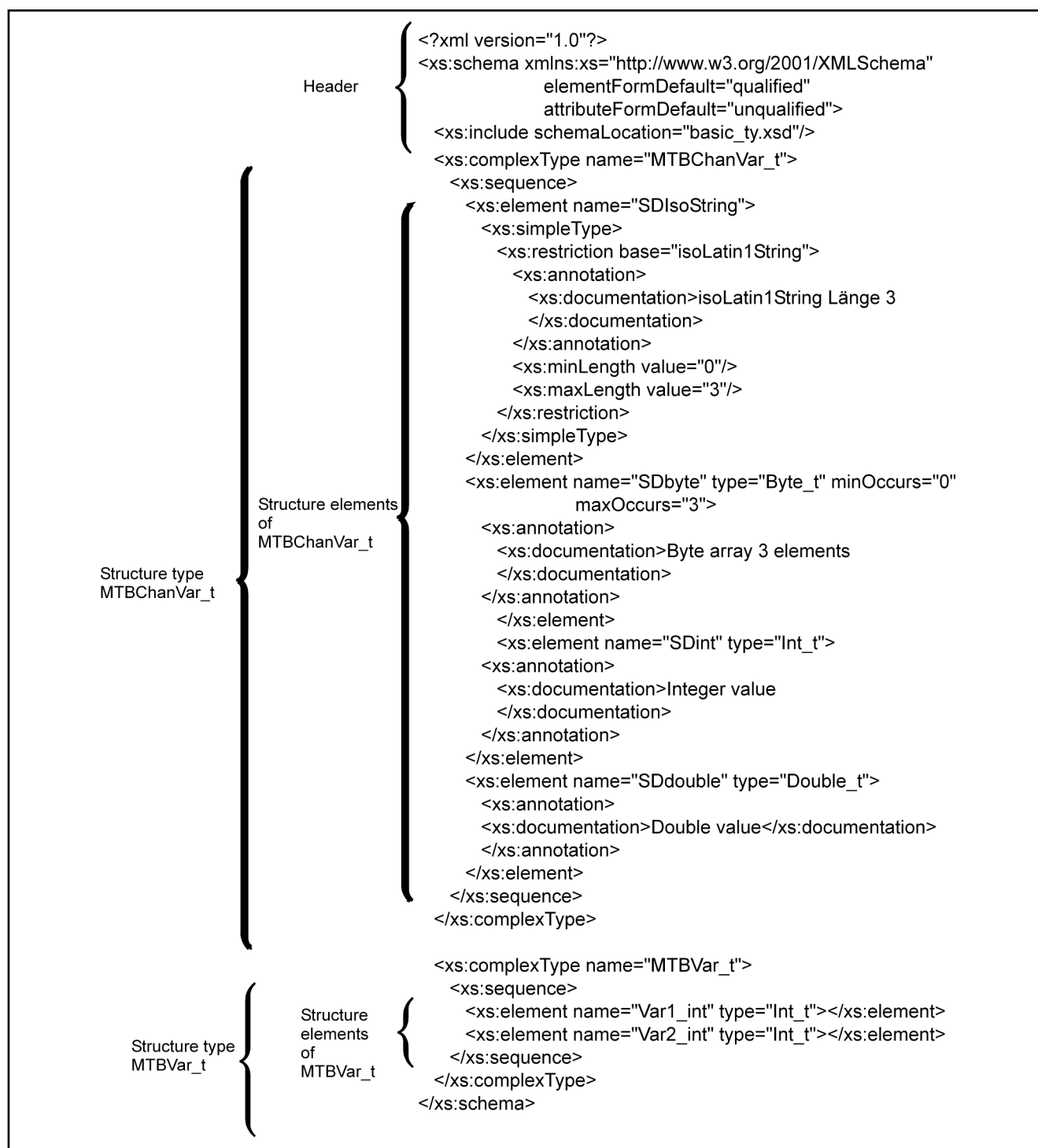


Fig. 10-3: Structure types in the file "TypesMTB.xsd"

Notes:

The header contains the XML version and the information that the file is an XML schema file.

Each of the two self-defined structure types "`MTBChanVar_t`" and "`MTBVar_t`" are created as `<complexType>`.

Individual structure elements of the structure types are generated in `<sequence>`. They consist of `<name>` and `<type>`. `<type>` can be a basic type or another type specified with `<complexType>`.

The array limits are defined with <minOccurs> and <maxOccurs>.

Comments are created in <annotation> and <documentation> and are optional.

2. Definition of the system data in the definition file "SDDefMTB.xml"

In the root directory "/", the SDDefMTB.xml file is created with the following content:

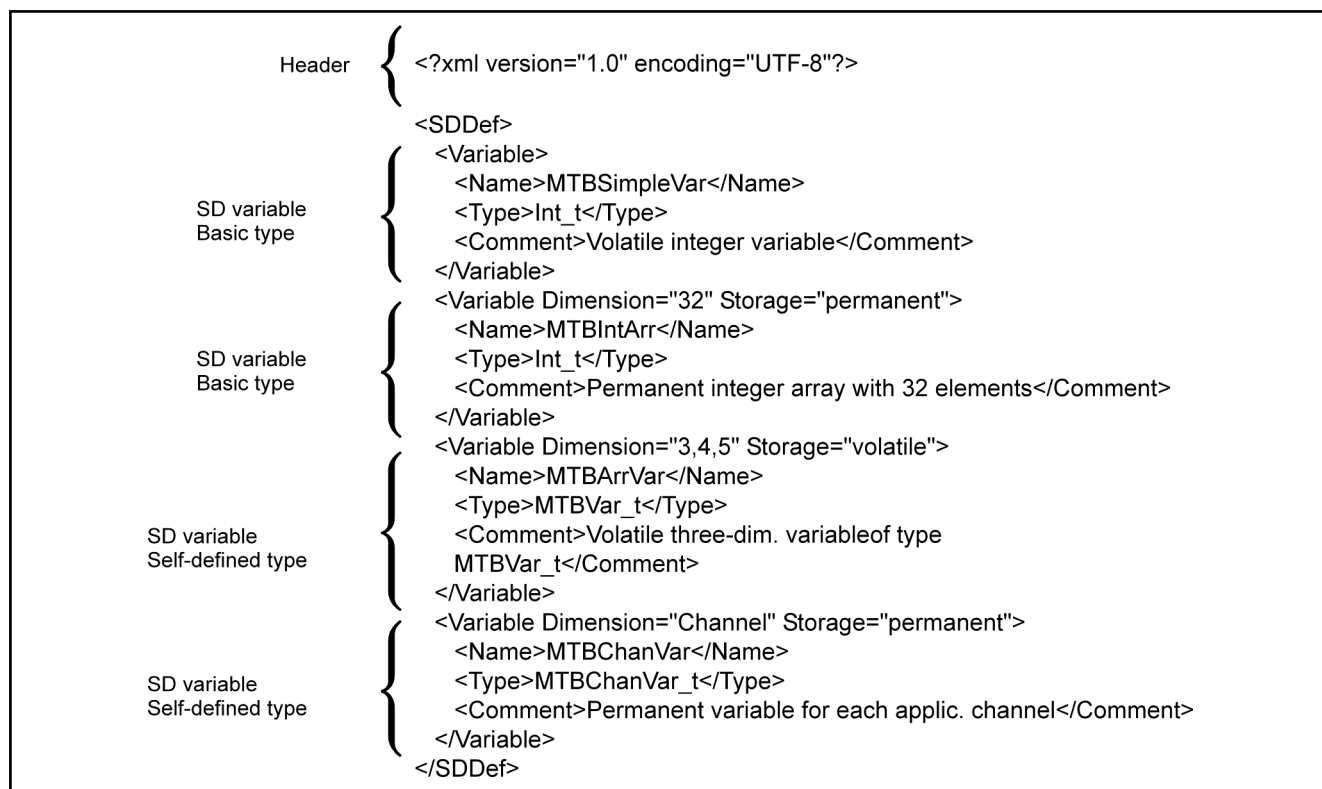


Fig. 10-4: Definition of the structure file in the file SDDefMTB.xml

Notes:

The header specifies the respective XML version and indicates the coding. All SD variables defined afterwards, are enclosed between <SDDef> and </SDDef>.

The SD variable definitions are put in brackets <Variable> and </Variable>. For each <Variable>, there are the attributes <Name>, <Type> and optionally <Comment>. The type is a basic type, an MTX type or a self-defined type.

In the attribute <Storage>, the memory attribute of the SD is set to permanent, volatile or temporary ("volatile" is the default setting).

In arrays, <Dimension> is used for the maximum number of elements for each dimension. This number of elements is either constant or dynamic if one of the keywords **Channel**, **Drive** or **Spindle** is specified.

In a channel-specific SD, the keyword **Channel** is entered instead of the number of elements of one dimension. It is then created for each applied machining channel (corresponding to the machine parameter /Main/NofCh).

In a drive-specific SD variable, the keyword **Drive** is entered instead of the number of elements of one dimension. It is then created once for each applied drive (corresponding to the machine parameter /Main/NofDr). However, it is at least created once.

In a spindle-specific SD variable, the keyword **Spindle** is entered instead of the number of elements of a dimension. It is then created once for each applied spindle (number of elements Sp[i] existing in the machine parameters under /SP/SpFunc). However, at least one time.

3. Initialization with default values in the file "SDDatMTB.xml"

In the root directory "/", the file SDDatMTB.xml can be created for example with the following content:

Header	{	<?xml version="1.0" encoding="UTF-8"?>
Initialization of the SD variable (basic type) with default value	{	<SD> <MTBSimpleVar>111</MTBSimpleVar>
Initialization of the SD array variables (basic type) default values	{	<MTBIntArr Dimension="1">11</MTBIntArr> <MTBIntArr Dimension="32">12</MTBIntArr>
Initialization of the SD variable (self-defined type) with default values	{	<MTBChanVar Dimension="1"> <SDIsoString>Str</SDIsoString> <SDbyte>1</SDbyte> <SDbyte>2</SDbyte> <SDbyte>3</SDbyte> <SDint>22</SDint> <SDdouble>1.1</SDdouble> </MTBChanVar> <MTBChanVar Dimension="2"> <SDint>44</SDint> </MTBChanVar> </SD>

Fig. 10-5: Initialization with default values in the file SDDatMTB.xml

Notes:

The header comprises information on the XML version.

The SD variables enclosed in <SD> comprise default values.

Default values can be entered for "non-permanent" (MTBSimpleVar) as well as for "permanent" SD variables. For array SD variables on the topmost level, the dimension has to be indicated. On the lower levels (SDbyte), either all or none have to be specified.



When creating, "zero" or an "empty string" is assigned to all non-defined SD variables.

4. Access with CPL

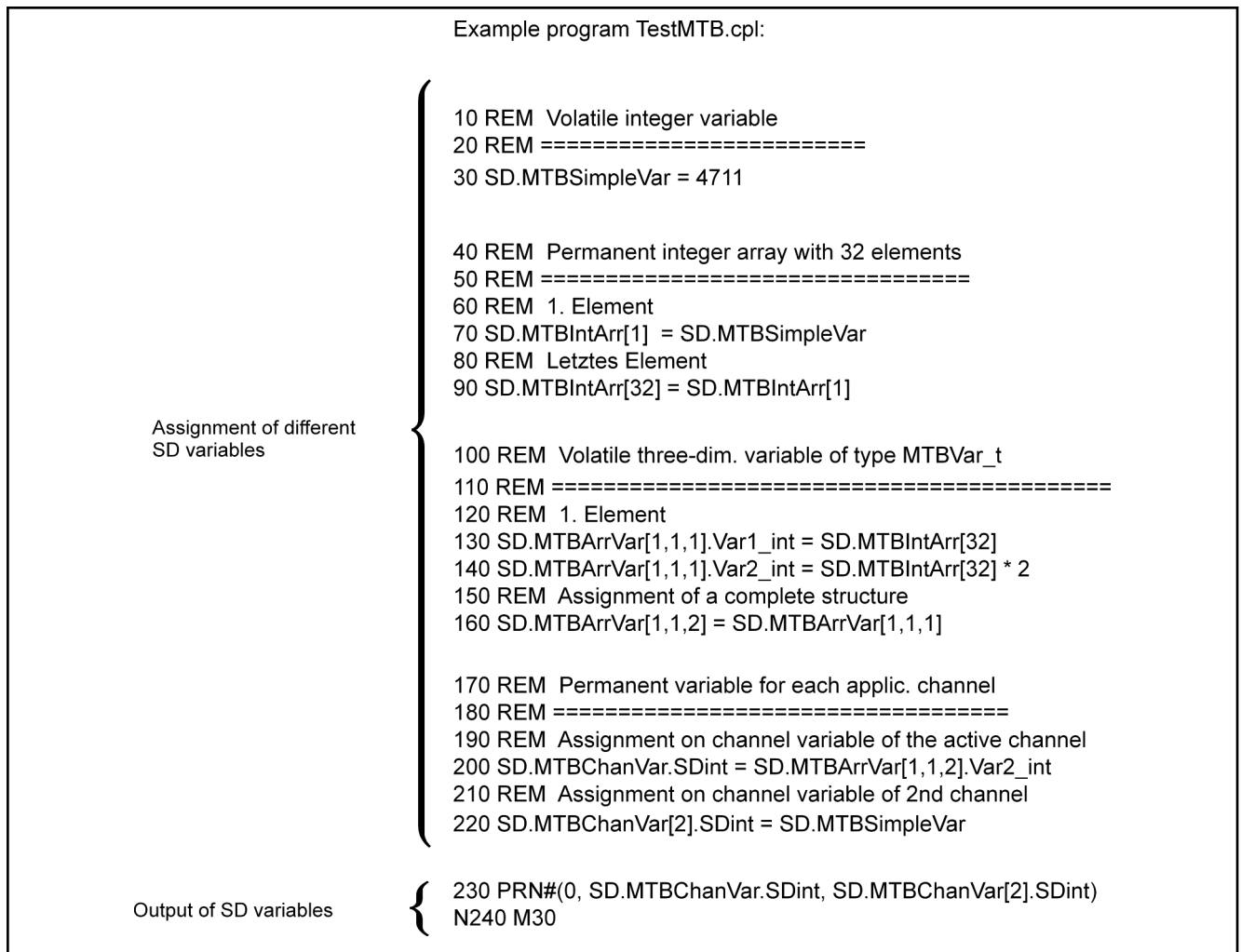


Fig. 10-6: Access with CPL

11 Schema Files

11.1 General Information

In the MTX, schema files are used to define data structures.

These data structures are used in the following parts of the control:

- XML tables (DCT, ZOT, PMT)
- Database tables (tool table)
- Configuration data
- System data

The MTB or the user can adapt data structures individually (e.g. tool table) and create system data with self-defined data structures. The syntax to define the data structures generally corresponds to the XML schema definition language standardized by the World Wide Web Consortium (W3C). Only the language elements described in this document are supported.

11.2 Managing Schema Files

11.2.1 Description

Schema directories

The storage design of the provided schema files allows that schema files are "overloaded", i.e. they can be newly defined by the user. Thus, own definitions can be integrated into existing data structures.

There is a search path for the schema files. This search path indicates the sequence the directories are searched for schema files. Only the schemas found first are read in. Schema files of the same name occurring later in the search path are not considered. System schemas are an exception, as they are only read from the directory /feprom/schemas.

The search path is defined via three directories:

- **/schemas**
User definitions in the RAM
- **/usrfep/schemas**
User definitions in the Userfep (user FEPROM)
- **/feprom/schemas**
Definitions by Rexroth in the FEPROM

Types of Schema Files

Three types of schema files are distinguished:

1. System schema files

These schemas are an integral part of the MTX firmware and cannot be overwritten by the user. They are located in the /feprom/schemas directory. (example: configuration data schemas)

2. Default user schema files

These schemas are standard schemas of the MTX which may be adjusted by the user. Rexroth provides the standard user schema files in /feprom/schemas. The user can overwrite them by files of the same name in the search path for schema files. (example: dbt1ud.xsd)

3. Other user schema files

Other user schema files created by the user are searched via the search path for schema files.

Applying Schemas

To apply newly generated schema, restart the control.

At control restart in safe mode (switch position 3), only the schemas comprised in the /feprom/schemas directories are read in.

If errors occur while parsing the schemas, a general error message is output. For a detailed error description, refer to the file /schemas/Schema.Err.log.

Example:

"SchemaErr.log" file with error messages

/schemas/dbt1ud.xsd: Schemas: element TI type TINew_t not found

Element "TI" has no type

element 'TI' has error in definition

element 'UD' has error in definition

complexType 'DBT1UD_t' has error in definition

complexType 'DBT1Rec_t' has error in definition

element 'Rec' has error in definition

/feprom/schemas/dbt1sd.xsd: element 'DBT1' has error in definition

In the "/schemas/SchemaAdm.log" file, all read in schemas are listed.

Memory Management for Schemas

For the internal representation of the schemas in the MTX, memory is required in the XML memory management. The default size of the basic memory provided is only sufficient to read in the schema files from /feprom/schemas.

Additional required memory for user schema files has to be configured. This is performed via the "startup.xml" file. This file is searched via the search path for configuration files: "/", "/usrfep", "/feprom". Via the parameter <XmlUserMem>, the size of the additional memory is specified in bytes.

The "/schemas/SchemaAdm.log" file provides information on XML memory management and on how much memory space has been actually used. The memory size to be set has to ensure that a continuous memory block of approximately 100 000 bytes is available in the memory management (free slots - biggest).

Example:

"startup.xml" file

```
<?xml version="1.0" encoding="UTF-8" ?>
```

```
<Params>
```

```
<XmlUserMem>200000</XmlUserMem>
```

```
</Params>
```

Schema Syntax

This chapter describes the permitted XML schema language elements in the MTX.

Schema Structure

The following file contains a simple schema defining the complex data type "MyEdLife_t".

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified"
attributeFormDefault="unqualified">

<xs:complexType name="MyEdLife_t">
<xs:sequence>
<xs:element name="MonOn" type="xs:boolean"/>
<xs:element name="RemLife" type="xs:float"/>
<xs:element name="WarnLim" type="xs:float"/>
<xs:element name="MaxLife" type="xs:float"/>
</xs:sequence>
</xs:complexType>

</xs:schema>
```

Each XML schema has to have a <schema> element as root element. Accordingly, the template for each schema file is as follows:

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified">
...
...
</xs:schema>
```

Simple Predefined Types

There is a large variety of predefined types in the XML schema. These types can be used in all element and attribute declarations. They have to be labeled in the schema by a preceding "xs:". The float type, for example, is identified with "xs:float".

The following table lists the predefined types supported by the MTX:

Name	Length in bytes	Description	C-type	IndraLogic type	Cpl
xs:string	2 * maxLength + 1	String (UTF-8 format)	char []	STRING()	STRING
isoLatin1String	maxLength + 1	String (Latin1 format)	char []	STRING()	STRING
xs:byte	1	Signed 8 bit integer	char	SINT	INT
xs:short	2	Signed 16 bit integer	short	INT	INT
xs:int	4	Signed 32 bit integer	int	DINT	INT
xs:unsignedByte	1	Unsigned 8 bit integer	unsigned char	USINT	INT
xs:unsigned-Short	2	Unsigned 16 bit integer	unsigned short	UINT	INT

xs:unsignedInt	4	Unsigned 32 bit integer	unsigned int	UDINT	INT
xs:float	4	32 bit real	float	REAL	REAL
xs:double	8	64 bit real	double	LREAL	DOUBLE
xs:Boolean	1	true, false, 1, 0	char	BOOL	BOOLEAN

Tab. 11-1: Predefined types of MTX



Any XML type not listed in the table must not be used in an MTX schema file.

Special features for strings:

For strings in UTF-8 format (type xs:string), all non-ASCII-characters (e.g. umlauts) consist of more than one character. They cannot be processed in the NC kernel. Thus, the "xs:string" type is only to be used if the string is used exclusively in the GUI.

The "isoLatin1String" type is not a predefined type as for the W3C, but a type derived from xs:string. The MTX provides a special treatment for strings with this type. Elements of a type derived from isoLatin1String only contain Latin1 characters. Thus, they can be processed by the NC kernel.

Defining Simple Types

New simple types are defined by derivation from existing simple types (pre-defined or derived). Especially a new simple type can be derived by restricting an existing simple type. In other words, the valid value range of the new type is a sub-range of the value range of the existing type. Use the simple-Type element to define and name the new simple type. The "restriction" element is used to indicate the existing (basic) type and to define the "facets" restricting the value range.

In the following table, all predefined simple types of XML schemas supported in the MTX are listed together with the facets that can be used:

Simple type	Facets				
	length	minLength, maxLength	pattern	enumeration	minInclusive, maxInclusive, minExclusive, maxExclusive
string	*	*	*	*	
byte			*	*	*
short			*	+	*
int			*	*	*
unsignedByte			*	*	*
unsigned-Short			*	*	*
unsignedInt			*	*	*
float			*	*	*

double			*	*	*
boolean			*		

Tab. 11-2: Simple types and facets used with them

The following example generates a new type "Float-10M_10M_t" derived from float with a value range from –10 billions to +10 billions (inclusive).

```
<xs:simpleType name="Float-10M_10M_t">
  <xs:restriction base="xs:float">
    <xs:minInclusive value="-100000000"/>
    <xs:maxInclusive value="100000000"/>
  </xs:restriction>
</xs:simpleType>
```

A special feature is applicable in the MTX when defining simple types derived from string. **To specify the memory space required internally for an element of the string type, the use of the "maxLength" facet is obligatory.**

A facet called 'pattern' together with a regular expression can be used to determine a valid sequence of numbers and digits for the type. For more detailed information to the options provided by regular expressions, refer the "Regular Expressions" chapter.

The example for the simple type "Str8_8_t" defines a string derived from isoLatin1String with a length of exactly eight characters consisting only of the characters 0 and 1.

```
<xs:simpleType name="Str8_8_t">
  <xs:restriction base="isoLatin1String">
    <xs:maxLength value="8"/>
    <xs:pattern value="[01]{8,8}"/>
  </xs:restriction>
</xs:simpleType>
```

The enumeration facet restricts a simple type to a predefined range of permitted values.

In the next example, a simple type "LifeUnit_t" is defined derived from isoLatin1String. It has to have the value "min" or "cyc".

Note that the facet "maxLength" is also required in this case.

```
<xs:simpleType name="LifeUnit_t">
  <xs:restriction base="isoLatin1String">
    <xs:maxLength value="3"/>
    <xs:enumeration value="min"/>
    <xs:enumeration value="cyc"/>
  </xs:restriction>
</xs:simpleType>
```

Defining Complex Types

Complex types contain elements and may have attributes. Attributes are not permitted for simple types.

Declaring elements: New complex types are defined by the `complexType` element.

"EdLife_t" for example is defined as complex type and four element declarations are shown in the "EdLife_t" definition. The element declarations are in a sequence. They have to appear in the XML document in their declared sequence. `<xs:complexType name="EdLife_t">`

```
<xs:sequence>
<xs:element name="MonOn" type="xs:boolean"/>
<xs:element name="RemLife" type="Float-10M_10M_t"/>
<xs:element name="WarnLim" type="Float-10M_10M_t"/>
<xs:element name="MaxLife" type="Float-10M_10M_t"/>
</xs:sequence>
</xs:complexType>
```

The "EdLife_t" definition only contains declarations of simple types. Likewise, elements can be declared in a sequence which are of a complex type as it can be seen in the following example:

```
<xs:complexType name="Ed_t">
<xs:sequence>
<xs:element name="Geo" type="Geo_t"/>
<xs:element name="Wear" type="Wear_t"/>
<xs:element name="Life" type="EdLife_t"/>
</xs:sequence>
</xs:complexType>
```

In a sequence, another sequence with additional elements can be declared:

```
<xs:complexType name="CfgPLC_Plc_t">
<xs:sequence>
<xs:sequence>
<xs:element name="IL" type="CfgPLC_PlcII_t"/>
</xs:sequence>
<xs:element name="ResetCplIf" type="CfgPLC_PlcII_t"/>
</xs:sequence>
</xs:complexType>
```

In an element declaration, the maximum number the respective element may occur is specified in the "max" attribute. This value can be a positive integer or the term "unbounded" can be assigned to indicate that there is no upper limit. In the MTX always has to be an upper limit, as memory space has to be created internally for the respective element. Accordingly, the term "unbounded" thus means in the MTX that an element may occur only once.



The `minOccurs` attribute can be used, but is not monitored by the NC kernel of the MTX!

The example below declares an "Ed" element which occurs at least one time and at most three times.

```
<xs:complexType name="DBT1UD_t">
```

```

<xs:sequence>
<xs:element name="Tl" type="Tl_t"/>
<xs:element name="Ed" type="Ed_t" minOccurs="1" maxOccurs="3"/>
</xs:sequence>
</xs:complexType>

```

Declaring attributes: Due to an element declaration in the MTX, memory space is provided for the respective element in which its data is stored. Contrarily, no memory is created in the control for an attribute declaration in the schema.



Attributes cannot be used to hold data of an XML document - this means that no attributes will be comprised in an XML file generated by the MTX.

In the MTX, attribute declarations in the schema are primarily used to specify additional information for elements such as access rights.

The next example shows a typical attribute declaration in the MTX. An attribute "L1" which may not exist in the xml document (use="prohibited") is set to a specified value (fixed="R").

```
<xs:attribute name="L1" type="xs:string" use="prohibited" fixed="R"/>
```

Attributes can only be declared within a complex type. There are several methods:

In a complex type with a sequence, attribute declarations can be included in addition to element declarations.

```

<xs:complexType name="EdLife_t">
<xs:sequence>
<xs:element name="MonOn" type="xs:boolean"/>
<xs:element name="RemLife" type="Float-10M_10M_t"/>
<xs:element name="WarnLim" type="Float-10M_10M_t"/>
<xs:element name="MaxLife" type="Float-10M_10M_t"/>
</xs:sequence>
<xs:attribute name="L1" type="xs:string" use="prohibited" fixed="R"/>
<xs:attribute name="L2" type="xs:string" use="prohibited" fixed="RW"/>
</xs:complexType>

```

To define a type is based on a simple type and also includes, a complex type with simpleContent is created. Usually, this is executed as anonymous type (see [Defining anonymous types, page 108](#)).

```

<xs:complexType>
<xs:simpleContent>
<xs:extension base="xs:short">
<xs:attribute name="L1" type="xs:string" use="prohibited" fixed="R"/>
<xs:attribute name="L2" type="xs:string" use="prohibited" fixed="RW"/>
</xs:extension>

```

```
</xs:simpleContent>
</xs:complexType>
```

A type based on a complex type and extended by attributes is defined via a complex type with complexContent. Usually, this is performed as anonymous type.

```
<xs:complexType>
<xs:complexContent>
<xs:extension base="Ed_t">
<xs:attribute name="L1" type="xs:string" use="prohibited" fixed="R"/>
<xs:attribute name="L2" type="xs:string" use="prohibited" fixed="RW"/>
</xs:extension>
</xs:complexContent>
</xs:complexType>
```

Defining Anonymous Types

Schemas can be designed by defining quantities of named types (as in the examples above) and then by declaring elements which reference these types with the "type=" construction. For types required only for one element declaration, the type can be defined directly as anonymous type. This is possible with simple and with complex types.

```
<xs:element name="Descr">
<xs:simpleType>
<xs:restriction base="xs:int">
<xs:minInclusive value="0"/>
<xs:maxInclusive value="9999"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
```

Notes

To add comments to schemas which can be read by the user as well as by applications, a "documentation" element can be used. A "documentation" is displayed as a sub-element in an "annotation". The "annotation" element may be located at the beginning of a schema construct characterized by the elements "schema", "simpleType", "complexType", "element" and "attribute".

```
<xs:simpleType name="isoStr16_t">
<xs:annotation>
<xs:documentation>Iso Latin 1 String 0 - 16</xs:documentation>
</xs:annotation>
...
</xs:simpleType>
```



Regular XML comments are also permitted:

```
<!-- This is a comment -->
```

Regular Expressions

The pattern facet of the XML schema uses a language for regular expressions. A regular expression is a pattern which is compared to a character string to check whether the latter is valid.

(Regular expressions are described in detail in "XML schema part 2: Data types".



Note that an element of the type "string" always requires the facet "maxLength" in the MTX. This also applies if the string length is already specified by the regular expression.

Expression	Suitable character strings	Explanation
a	a	Letter a
[a-z]	a, b, c ...	Lower-case letter
[0-9]	0, 1, 2 ...	Digit
[0123456789]	0, 1, 2 ...	Digit
[A-Z0-9]	A, B, ..., 0, 1 ...	Capital letter or digit
A[0-9]	A0, A1, A2 ...	A followed by a digit
[01][01][01]	000, 010, 111 ... 3	digits, but only 0 and 1
[^äöüß]	a, A, 0, 1, 3 ...	Character, but no ä, ö, ü or ß
.	a, Z, 1, 9, - ...	Any character
a*	, a, aa, aaa ...	Zero or more a
[a-z]*	, a, ab, hallo ...	Zero or more lower-case letters
a?	, a	Zero or one a
a+	a, aa, aaa ...	One or more a
[a-z]{8}	abcdefgh ...	Eight lower-case letters
[a-z]{0,8}		Zero to eight lower-case letters
[a-z]{2,8}		Two to eight lower-case letters
male female	male, female	male or female
.x	x, ax, aax, aaax ...	Any character followed by an x
a*x	x, ax, aax, aaax ...	Zero or more a, followed by an x
(a b)+x	ax, bx, aax, abx, bax, bbx, aaax, abbaabbax ...	One or more (a or b) followed by an x
ab{2}x	abbx	
(ab){2}x	ababx	

Tab. 11-3: Examples of regular expressions

11.2.2 Applying

Creating new schema file

PC / schema editor: Create new schema:

Process as follows to create a new schema file:

- Create a new schema with the schema editor (preferably Spy by Messrs. Altova) or

Schema Files

- by means of an UTF8 editor open an existing schema (EMACS (Version >2000, Word (version >2000) (e.g. from the "...\\Rexroth\\IndraWorks\\Config\\Mtx\\schemas" directory) and execute the necessary modifications.

PC / schema editor: Saving schema files:

Save the created schema file to one of the two following paths:

- /schemas - User definitions in the RAM - /usrfep/schemas
- User definitions in the Userfep (user FEPROM)

MTC kernel: Data transfer:

Transfer of data requires a control reset.



- If the error message 3513 "Not sufficient memory in the XML memory management" is signaled at startup, increase the memory space to manage the XML schemas using the "startup.xml" file.
 - If the error message 3121 "Error when reading in XML schema files" is signaled after startup, the NC kernel stores a detailed error description in the "/schemas/SchemaErr.log" file.
 - If nothing functions, start up with switch position "3". Only the schemas from "/feprom/schemas" are then used.
-

12 Area monitoring with system data

12.1 Introduction

In the ACS, MCS and BCS coordinate systems, the following types of rectangular monitoring areas, parallel to the coordinate system axes, can be created:

- **Exclusive operating ranges:**
The position must not be outside the exclusive operating area
- **Inclusive operating areas:**
The position must not be outside all inclusive operating areas of a coordinate system
- **Protected area:**
The position must not be within a protected area

The area limit is part of the area.

If these requirements are not met, the area has been violated. As soon as the control detects a area violation, an error is reported.

Thus, machine parts can be protected from collision with the tool or the work-piece due to incorrect NC programming.

12.2 Example

If in a coordinate system, the inclusive operating area □, the protected areas ■ and the exclusive operating area, indicated by a dashed line, are created, the path curve would not result in a area violation.

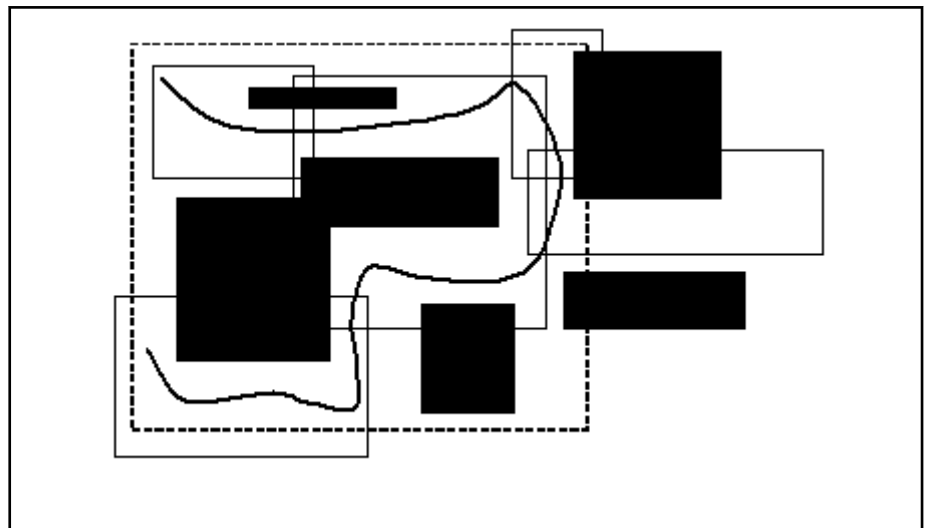


Fig. 12-1: Example of 2-dimensional monitoring areas

12.3 Creating monitoring areas in the system data

To create, for example, 17 areas, enter the following in a definition file for the system data:

Program:

```
<Variable Dimension="17" Storage="volatile">
  <Name>ColpQuad</Name>
  <Type>ColpQuad_t</Type>
  <Comment>Here you can insert a comment.</Comment>
```

Area monitoring with system data

```

    <Comment>Hier kann ein Kommentar stehen.</Comment>
  </Variable>

  <Variable Dimension="Channel" Storage="volatile">
    <Name>ColpDeact</Name>
    <Type>UnsignedByte_t</Type>
    <Comment>Here you can insert a comment.</Comment>
    <Comment>Hier kann ein Kommentar stehen.</Comment>
  </Variable>

  <Variable Storage="volatile">
    <Name>ColpValid</Name>
    <Type>UnsignedByte_t</Type>
    <Comment>Here you can insert a comment.</Comment>
    <Comment>Hier kann ein Kommentar stehen.</Comment>
  </Variable>

```

Instead of the number 17, any other number can be entered as long as the volatile system data memory can accommodate the number of monitoring areas.

Subsequently, trigger an NC restart.

For more information about the definition files for system data, please note [chapter 10 "System data" on page 85](#) as well as [chapter 11 "Schema Files" on page 101](#) in this documentation.

12.4 General area properties

SD.ColpQuad[i].ID	Any number can be entered for the area i. The name must not exceed 30 characters. If this area is violated, the entered name is output in the error message in addition to "i". However, the entry is not mandatory.
SD.ColpQuad[i].Type	Enter type of area i: • 0 - Exclusive operating area • 1 - Inclusive operating area • 2 - Protected area
SD.ColpQuad[i].Active	Enter if the area i is to be active or inactive: • 0 - Area is not active. No monitoring takes place • 1 - Area is active. Monitoring takes place
SD.ColpQuad[i].React	Enter how the control is to react in case of a violation of area i: • 0 - Error message as soon as the violation is detected by the control Another reaction is currently not planned.
SD.ColpQuad[i].CoordSys	Enter the coordinate system to which area i applies to: • 0 - Axis coordinate system (ACS) • 1 - Machine coordinate system (MCS) • 2 - Basic workpiece coordinate system (BCS) The MCS is MCS ⁽⁰⁾ . Areas for which the BCS was entered are usually monitored opposite the tool tip (TCP). This also applies to an active tool radius correction. For more information about axes, coordinates and coordinate systems as well as about tool corrections, please refer to the "MTX Functional Description - Basics" documentation.
SD.ColpQuad[i].Jog	Only the value zero is invalid:

- 0 - The area i only applies to automatic mode and to jogging in the WCS.

In a later version, the area i for jogging in the ACS and handwheel in the ACS can be enabled here. This option is currently not available.

SD.ColpQuad[i].Chan

Specify the channel number that is to apply to area i, if area i is active.

- > 0 - Number of the channel

Moreover, it can be specified that the area is to be applied to all channels if the it is active:

- -1 - Area applies to all channels

12.5 Geometric area properties

To specify the desired geometry of the area i, the parameter **SD.ColpQuad[i].Coord[1..4]** is available as array with 4 elements:

Thus, a maximum of 4 coordinates can be monitored by one area.

Enter the following for each coordinate k = 1...4 of area i:

SD.ColpQuad[i].Coord[k].SACI

SACI (system axis and coordinate index) of coordinate k of area i with the following valid values:

-12	Linear workpiece coordinate x
-11	Linear workpiece coordinate y
-10	Linear workpiece coordinate z
-9	Rotary workpiece coordinate phi
-8	Rotary workpiece coordinate theta
-7	Rotary workpiece coordinate psi
-6	Linear machine coordinate X
-5	Linear machine coordinate Y
-4	Linear machine coordinate Z
-3	Rotary machine coordinate phi
-2	Rotary machine coordinate theta
-1	Rotary machine coordinate psi
0	Coordinate k of area i should not be active
>0	System axis

Tab. 12-1: Valid values for an area coordinate

The values -12 to -7 assume an active axis transformation at position 2 and assume that BCS is the coordinate system of the area. If this not the case, all values -12 to -7 have the same effect as the value 0. The same applies if an axis transformation is active at operating position 2 but does not returned the entered SACI. For example, this would be the case if a 5-axis transformation is active and if the value -7 was entered.

The value -6 to -1 assume an active axis transformation at operating position 1 as well as the MCS as coordinate system of the area and the BCS, unless an axis transformation is active at operating point 2. If this is not the case, all values -6 to -1 have the same effect as the value 0. The same applies if an axis transformation is active at operating position 1 but does not returned the entered SACI.

For more information about axis transformation, please refer to documentation "MTX Functional Description - Extension".

By specifying the value 0, individual coordinates can be disabled so that the area only monitors 3, 2, 1 or no coordinate at all.

The values > 0 assume that the respective axis is in the channel. If this not the case, the value has the same effect as value 0.

SD.ColpQuad[i].Coord[k].Min

The lower limit of area i referring to coordinate k

SD.ColpQuad[i].Coord[k].Max

The upper limit of area i referring to coordinate k

The current position is outside area i if **both** the following conditions for **all** active coordinates k of area i are complied with:

$SD.ColpQuad[i].Coord[k].Min \leq \text{Coordinate k of the current position}$

$\text{Coordinate k of the current position} \leq SD.ColpQuad[i].Coord[k].Max$

In all other cases, the current position is outside the area. As the area limit is part of the area, a position that is up to 10 µm outside the area due to rounding errors are still identified as within the area.

An area has been violated if the current position is outside of an exclusive working area or outside all inclusive working ranges of a coordinate system at the same time or if it is within a protected area.

As soon as the control detects an area violation caused by an NC block or a jog motion in workpiece coordinates, an error message is triggered. The same applies if the control detects that an area has been violated upon the start of a motion already, e.g. upon the NC program start.

A 1-dimensional monitoring area is a section on a straight line. A 2-dimensional monitoring area is a rectangular and a 3-dimensional monitoring area is a cuboid.

12.6 Deactivating all channel areas

SD.ColpDeact

This system data is channel-specific.

- No area monitoring takes place in all channels where $SD.ColpDeact = 1$.
- Area monitoring takes place in all channels where $SD.ColpDeact = 0$, if areas the channels refer to are active.

12.7 Changing the operating data

The area data $SD.ColpQuad$ and $SD.ColpDeact$ can be changed at any time, however, note the following:

If the input in the area data has been completed and if the control is to apply the inputs to its internal area data memory, the block preparation has to be synchronized with the interpolator for all channels for which the area data has been changed and in which NC programs are currently running. How to synchronize the block preparation with the interpolator is described in the following section. Set $SD.ColpValid = 1$ to trigger the transfer. If the control has completed applying of data, it sets $SD.ColpValid = 0$. However, this happens so fast that the user does not have to wait for completion but that other operating data can be changed.

Even if the operating data is initialized upon startup, trigger the initial transfer to the internal operating data memory of the control. For more information about the system data initialization at startup, please refer to [chapter 10 "System data "](#) on page 85 in this documentation.

12.8 Synchronizing the block preparation with the interpolator

If you want to change the `SD.ColpQuad` and `SD.ColpDeact` area data while NC programs are running in the channels affected by the modifications, proceed as follows:

1. Select the channel the operating data is to be changed for and only change the operating data referring to this channel.
2. Call the following asynchronous subroutine, consisting of only one block, for this channel:

Program:

```
1 SD.ColpValid = 1
```

3. Select another channel the operating data is to be changed for and only change the operating data referring to this channel.
4. Call the same asynchronous subroutine for this channel.
5. Continue until desired operating data changes have been completed.

An asynchronous subroutine automatically synchronizes the block preparation with the interpolator upon the start. As the `SD.ColpValid = 1` command is positioned at the beginning of the subroutine, the block preparation and the interpolator are synchronized at this point.

Instead of changing the operating data of a channel before calling the asynchronous subroutine, change the channel data in the subroutine. Insert the area data changes in front of the only subroutine block to data, e.g.:

Program:

```
1 SD.ColpQuad[3].Active = 1
2 SD.ColpValid = 1
```

This is valid as the initial synchronization is not canceled by additional CPL blocks.

If you have to insert NC blocks in the asynchronous subroutine before changing the area, the initial synchronization can sporadically be lost. Thus, a `WAIT` command is required after these NC blocks. After the area data changes are completed, additional NC blocks can be inserted.

For example, the following asynchronous subroutine is valid:

Program:

```
N1 G0 X0 Y0 Z0
2 WAIT
3 SD.ColpQuad[3].Active = 1
4 SD.ColpValid = 1
N5 G1 X30 Y70 Z60
```

If area data referring to the channel this NC program is running in is to be changed in an NC program, no asynchronous subroutine is required. Insert the following blocks at the NC program positions at which the area data is to be changed:

Program:

```
Nxx WAIT
xx SD.ColpValid = 1
```

"xx" stands for any block numbers to be selected. Insert your area data changes between these two blocks, e.g.:

Program:

```
Nxx WAIT
xx SD.ColpQuad[3].Active = 1
xx SD.ColpValid = 1
```

Particular caution is required if you want to change or set the channel reference of an area:

```
SD.ColpQuad[3].Chan = 5
```

If area 3 previously referred to channel 2, this would imply that by using this command, a area is changed for channel 2 and also for channel 5.

If NC programs run in both channels, it is recommended to execute these changes via the following intermediate steps:

```
SD.ColpQuad[3].Active = 0
```

In case the change cannot be completed in one of the channels, proceed as follows:

1. Set SD.ColpQuad[3].Active = 0.
2. Set SD.ColpQuad[3].Chan = 5.
3. Call the following subroutine for channel 2:

Program:

```
1 SD.ColpValid = 1
```

4. Set SD.ColpQuad[3].Active = 1.
5. Call the following subroutine for channel 5:

12.9 Invalid area data

If invalid area data was entered in the system data, an error message is reported in all channels when applying the data to the internal area data memory of the control. Please note that no data is transferred in this case; not even for invalid area data.

12.10 Time of area violation detection

The area monitoring takes place in the block preparation as well as in each interpolator cycle. Normally, an area violation is detected in the block preparation and an error message is issued before the actual area violation. If an area violation is not detected in the block preparation, it is detected in the first interpolation cycle with the area violation. The area violation has already been taking place at this point in time. The area monitoring then issues an error message and a complete brake causing an additional braking distance. If the monitoring areas are very narrow around the machine parts to be protected, collisions can result. Such cases are described in the following.

If the user changed the area in the synchronization of the block preparation for example as described in the last section, the changed data area for the block preparation can be too late and thus causing an area violation only detected by the interpolator.

Thus, due to security reasons, the control reacts to area data changes without synchronization with full braking and an error message if at least one axis is moving. This also applies if currently no area is violated.

Corrections that only become active upon interpolation can also only be considered during interpolation by the area monitoring.

The same applies to the following corrections:

- Selective additive coordinate coupling (SCC/SCCT)
- Online correction
- Straightness and angle error compensation
- Volumetric compensation

The same applies also to jogging and the handwheel in workpiece coordinates. In this case, the distance to be traversed it is also not known upon block preparation. Thus, the area can only be monitored upon the interpolation.

For more information about selective, additive coordinate coupling (SCC/SCCT), please refer to the chapter about couplings in the "MTX Functional Description - Extension".

For more information about the online correction, please refer to [chapter 4 "Operating functions \(manual operation\)" on page 17](#) in this documentation.

For more information about straightness and phase error compensation as well as volumetric compensation, please refer to [chapter 5 "Compensating mechanical inaccuracies " on page 25](#) in this documentation.

For more information on jogging and handwheel in workpiece coordinates, please refer to the chapter on operating functions in the documentation "MTX Functional Description - Extension".

For more information on traversing motions without an area monitoring being possible, refer to [chapter 12.12 "Restrictions" on page 118](#).

12.11 Optional system data for monitoring areas

In addition to the already mentioned system data `SD.ColpQuad`, `SD.ColpDeact` and `SD.ColpValid`, the following channel-specific system data can be created.

Enter the following in a definition file for the system data:

Program:

```
<Variable Dimension="Channel" Storage="volatile">
  <Name>ColpLimit/Name>
  <Type>Int_t</Type>
  <Comment>Here you can insert a comment.</Comment>
  <Comment>Hier kann ein Kommentar stehen.</Comment>
</Variable>
```

Subsequently, trigger an NC restart.

For more information about the definition files for system data, please refer to [chapter 10 "System data " on page 85](#) in this documentation.

SD.ColpLimit

When an exclusive working area is violated, the control writes the number *i* to this working area.

When violating all inclusive working areas of a coordinate system, the control writes the number *i* of the latest working area left.

When violating a protected area, the control writes the number *i* to this protected area.

If the position is outside of all inclusive working areas of a coordinate system upon NC program start or if a jump to this position occurs, e.g. by activating a tool correction, the control writes:

- -1 if all inclusive working areas of the ACS are violated
- -2 if all inclusive working areas of the MCS are violated
- -3 if all inclusive working areas of the BCS are violated

- In case of an error message that is not caused by an area violated, the control writes the value 0.

12.12 Restrictions

Modulo axes and coordinates are not valid.

No areas are monitored when referencing with G74 (HOME).

Gantry slave axes, following axes in system coupling and following axes in channel axis coupling (AXC) are not monitored. This means, all slave or following axes in ACS couplings.

Position changes due to spindle pitch error compensation (LSEC), cross compensation (CCOMP), hysteresis compensation and circle correction at quadrant transition.

No areas are monitored during jogging in the ACS.

No areas are monitored when traveling axes with the handwheel.

If ranges are violated upon the NC program start or following the startup, use the two last mentioned functions to travel to a position that does not violate any ranges.

For more information about approaching reference points, jogging of axes as well as the handwheel for axes, please note the chapter about operating functions (manual mode) in the "MTX Functional Description - Basics" documentation.

12.13 Comparison with the NC command Area

The following applies to monitoring areas based on system data, irrespective of the monitoring areas, programmable by Area, described in the "MTX Programming Manual" documentation.

- Any number of traversing motions, also splines can be monitored
- Traversing motions outside of the active plane can also be monitored
- Any axes and coordinates in the channel can be monitored with the exception of modulo axes and modulo coordinates
- With one area, a maximum of 4 axes and coordinates can be monitored
- Inclusive working areas can also be monitored
- The number of areas to be monitored is user-defined and is only limited by the size of the volatile system data memory
- Areas can be monitored in the ACS, MCS and BCS coordinate systems
- Monitoring in jog mode is only possible for jogging in workpiece coordinates

12.14 Handling instruction: Example

For the following example, an x-axis with system axis number 1 and a y-axis with system axis number 2 is required. Both axes have to be in channel 1.

1. Take the file of the root directory or if the SDDefMTB.xml file is not available in this directory, copy the SDDefMTB.xml file from the UserFE-PROM directory to the root directory. Insert the following variables at the end of the file:

Program:

```
<xml version="1.0" encoding="UTF-8" ?>
<SDDef>
  ..
```

```

<Variable Dimension="4" Storage="volatile">
  <Name>ColpQuad</Name>
  <Type>ColpQuad t</Type>
  <Comment>My first system data based ranges</Comment>
  <Comment>Meine ersten SD-Überwachungsbereiche</Comment>
</Variable>

<Variable Dimension="Channel" Storage="volatile">
  <Name>ColpDeact</Name>
  <Type>UnsignedByte t</Type>
  <Comment>Channel Deactivation Flag</Comment>
  <Comment>Kanal-Deaktivierungs-Flag</Comment>
</Variable>

<Variable Storage="volatile">
  <Name>ColpValid</Name>
  <Type>UnsignedByte t</Type>
  <Comment>Validate Flag - Übernahme Flag</Comment>
</Variable>

</SDDef>

```

2. Trigger an NC restart.

3. Save the following subroutine under the name SD_INIT:

Program:

```

N1 WAIT
1 SD.ColpQuad[1].ID      = "1 Excl. Work"
1 SD.ColpQuad[2].ID      = "2 Incl. Work"
1 SD.ColpQuad[3].ID      = "3 Incl. Work"
1 SD.ColpQuad[4].ID      = "4 Prot."

1 SD.ColpQuad[1].Type    = 0
1 SD.ColpQuad[2].Type    = 1
1 SD.ColpQuad[3].Type    = 1
1 SD.ColpQuad[4].Type    = 2

1 SD.ColpQuad[1].Active  = 1
1 SD.ColpQuad[2].Active  = 1
1 SD.ColpQuad[3].Active  = 1
1 SD.ColpQuad[4].Active  = 1

1 SD.ColpQuad[1].CoordSys = 2
1 SD.ColpQuad[2].CoordSys = 2
1 SD.ColpQuad[3].CoordSys = 2
1 SD.ColpQuad[4].CoordSys = 2

1 SD.ColpQuad[1].Chan     = 1
1 SD.ColpQuad[2].Chan     = 1
1 SD.ColpQuad[3].Chan     = 1
1 SD.ColpQuad[4].Chan     = 1

1 SD.ColpQuad[1].Coord[1].SACI = 1
1 SD.ColpQuad[1].Coord[1].Min  = -300
1 SD.ColpQuad[1].Coord[1].Max  = 300

1 SD.ColpQuad[1].Coord[2].SACI = 2
1 SD.ColpQuad[1].Coord[2].Min  = -300
1 SD.ColpQuad[1].Coord[2].Max  = 300

1 SD.ColpQuad[2].Coord[1].SACI = 1
1 SD.ColpQuad[2].Coord[1].Min  = -200
1 SD.ColpQuad[2].Coord[1].Max  = 200

1 SD.ColpQuad[2].Coord[2].SACI = 2
1 SD.ColpQuad[2].Coord[2].Min  = -200
1 SD.ColpQuad[2].Coord[2].Max  = 200

1 SD.ColpQuad[3].Coord[1].SACI = 1
1 SD.ColpQuad[3].Coord[1].Min  = 0
1 SD.ColpQuad[3].Coord[1].Max  = 400

1 SD.ColpQuad[3].Coord[2].SACI = 2
1 SD.ColpQuad[3].Coord[2].Min  = 0
1 SD.ColpQuad[3].Coord[2].Max  = 400

1 SD.ColpQuad[4].Coord[1].SACI = 1
1 SD.ColpQuad[4].Coord[1].Min  = -400
1 SD.ColpQuad[4].Coord[1].Max  = 0

```

Area monitoring with system data

```

1  SD.ColpQuad[4].Coord[2].SACI  =  2
1  SD.ColpQuad[4].Coord[2].Min   = -400
1  SD.ColpQuad[4].Coord[2].Max   =  0

1  SD.ColpValid                   =  1

```

4. Restart the NC program in channel 1.

Program:

```

N01  G1 F5000
N02  X0 Y0.01
N03  P SD_INIT
N04  Y300
N05  X300
N06  Y0
N07  X200
N08  Y-200
N09  X0.01
N10  Y0.01
N11  X-200
N12  Y200
N13  X0
N14  Y0.01
M30

```

Completed without error message.

5. Extend the NC program by the following block:

Program:

```

N15  X-400 Y-400

```

6. Restart the NC program in channel 1.

The error message 5578 "NC block violates SD area 4 (prot.)" is reported.

7. Delete the NC block N15.

8. Extend the NC program by the following blocks:

Program:

```

N15  WAIT
16  SD.ColpDeact = 1
17  SD.ColpValid = 1
N18  X-400 Y-400
N19  X0 Y0.01

```

9. Restart the NC program in channel 1.

Completed without error message.

13 Macro Programming

13.1 Description

13.1.1 Function

Macros are one method of input help. MTX still works with the underlying syntax structures, but replaces the macros before in a separate step by the configured replacement text.

Macros in NC blocks are programmed like NC functions. Macros in CPL blocks are programmed like CPL variables.

Macros consist of three parts: Macro name, replacement text and attributes.

If the following applies for example:

Macro name:	macro_name
Replacement text:	replacement_text
Attribute:	DIN

The programmed NC block `N10 macro_name X10` then becomes `N10 replacement_text X10`.

Both, the macro name and the replacement text are single-line.

The macro name has to start with a letter and may contain letters, digits and the character "_". The length of the macro name is limited to 29 characters. Each macro name may be assigned only once and has to be unique. The macro name is case-sensitive (distinction between upper and lower-case letters).

The replacement text may contain any character and may not exceed 511 characters.

When the MTX processes a part program, each macro name is replaced by the corresponding replacement text.

Macros act globally (that is channel-comprehensive). They cannot be deactivated temporarily. Changes in the macro definition only become active upon a system restart.

13.1.2 Using Macros

Using macros, the following is possible:

1. Combining several NC functions in one syntax,
2. Shortening long variable names (e.g. in system data)
3. Simplifying complex functions

For the first case, the macro **HSC_ON** can for example combine the NC functions **SplineDef(4203, X, Y, Z) CLD(DFACT3.0, DIST5, ANG45, E0.1)**.

The block `N10 HSC_ON G1 F1000` then becomes `N10 SplineDef(4203, X, Y, Z) CLD(DFACT3.0, DIST5, ANG45, E0.1) G1 F1000`.

In the second example, the system date **SD.SysCyc.CD** is to be shortened by the macro **R11**. In this example, a CPL macro is defined for that purpose.

Thus, the block `10 R11=5.5` acts like `10 SD.SysCyc.CD=5.5`.



A CPL macro automatically overrides CPL variables with the same name. In the example, it can thus not be accessed on a local variable R11, since it is then overridden by the macro R11.

In the third example, the parameterizable macro **TravXYZ** executes a traversing motion with the replacement text **G0(NIPS) X[{1}] Y[{2}] Z[{3}]**.

If the block `N10 TravelXYZ(10,20,30)` is programmed, the MTX executes the following block: `N10 G0(NIPS) X[10] Y[20] Z[30]`.



For technical reasons, the continuous support of macros as parameter of NC functions cannot be guaranteed. Thus, syntax errors can occur due to unresolved macros. A simple solution is to define the complete NC function including the parameters as macro.

13.1.3 Parameterizable Macros

The preceding example showed that parameterizable macros are more flexible than static ones.

To use a macro parameter, the replacement text has to contain parameter placeholders. The latter have the form "{1}" to "{9}". Thus, a parameterizable macro can expect a maximum of nine parameters. The number in the placeholder specifies at which point in the programmed block the parameter is searched. One placeholder can be used multiple times in the replacement text.

Example:

Macro with parameters	
Macro name:	PMacro
Replacement text:	X={3} Y={1} Z={2}
Attribute:	DIN
Programmed block	N10 G1 PMacro(60, 50, 20) F1000
Acts like	N10 G1 X=20 Y=60 Z=50 F1000

It is not imperatively necessary (but reasonable) to start with {1} and to leave no gaps in the parameters. The following example is also possible.

Example:

Macro parameters with "gaps"	
Macro name:	PMacro2
Replacement text:	X={6} Y={3} Z={4}
Attribute:	DIN

Programmed block	N10 G1 PMacro(10,
	30, 60, 50, 0, 20)
Acts like	F1000
	N10 G1 X=20 Y=60
Z=50 F1000	

However, six parameters always have to be programmed (even if only three are in the replacement text).

When inserting the parameters, they are not translated or checked for a correct syntax, but only copied to the position of the placeholder as character strings.

If (in the programmed block) a parameter is to contain the characters "comma" or "right parenthesis", the complete parameter has to be specified in inverted commas ("").

Example:

Macro parameters with ","

Macro name:	G0
Replacement text:	AAC(X{2}, Y{2}) G0({1})
Attribute:	DIN and overlay="yes"

Programmed block	N10 G0 ("NIPS,
	POL", 3.2)
Acts like	N10 AAC(X3.2,
	Y3.2) G0(NIPS,
POL)	



Inverted commas ("") may always enclose each parameter. They are automatically removed upon insertion of the parameter character string.



If inverted commas are to be used in the parameter, use the syntax according to [the chapter 13.1.7 Special Characters in Replacement Text or Parameters, page 125](#).

13.1.4 Overriding NC Functions

In the last example, the NC function G0 was overridden. That means that if G0 has been programmed in a part program, the MTX resolves the corresponding macro instead of switching to rapid traverse.

The possibility to override certain syntax parts of the MTX is only available for NC functions. If it is tried to override the CPL syntax, NC variables, auxiliary functions or axis names, an error message is generated.

In addition, the overlay attribute has to be set when defining a macro that has to override an NC function.

NOTICE

Only use this feature in a limited number of cases and with extreme caution. If the NC syntax is changed, existing programs can suddenly behave differently way. Troubleshooting is much more complicated.

Specify unique macro names labeling them as macro.

13.1.5 Nesting Macros

Macro replacement texts may contain macros. These have to be written in curly brackets "{}". If this is not the case, an error message is output. For DIN macros overriding NC functions, it is thus distinguished whether the macro or the NC function is meant.

Example:

Nested macros		
Macro name:	G1	
Replacement text:	G1 F2000	
Macro name:	Move_X	
Replacement text:	G1 X100	
Macro name:	Move_Y	
Replacement text:	{G1} Y200	
Macro name:	Move_XZ	
Replacement text:	Move_X Z300	
Programmed block	N10 Move_X	; G1 is NC function
Acts like	N10 G1 X100	; G1 is NC function
Programmed block	N10 Move_Y	; G1 is macro
Acts like	N10 G1 F2000 Y200	; G1 is macro
Programmed block	N10 Move_XZ	
Acts like	Syntax error! Macro "Move_X" not in {}.	

The nesting depth is limited to 20 nestings. Thus, unintended recursions cannot be configured.

For nesting, the order in which the macros are defined or the files in which the definitions are contained is of no importance. Macros may be in replacement texts that are located in a completely different definition file.

13.1.6 Macro Types

The type of a macro is already determined in the configuration.

The following types are possible:

- CPL macros

- DIN macros
- CPL macros may only be programmed in the CPL context; DIN macros only in the DIN context. That means that in lines starting with a number or within square brackets "[]" only CPL macros are permitted. At all other positions, only DIN macros may be programmed.
- If there is a CPL macro in a DIN context, it is not detected and a syntax error is caused. The same applies to the opposite case.

13.1.7 Special Characters in the Replacement Text or Parameters

Every macro parameter has to be put between inverted commas if the following characters are programmed:

- Comma,
- right round bracket)

Reserved characters Inverted commas in parameters belong to the reserved characters. In the replacement text, the curly brackets { and } are reserved.

To program or configure a reserved character in the parameter or the replacement text, a backslash \ has to precede the character. A simple backslash \ is shown by two backslashes \\. The leading backslashes \ are removed upon macro insertion.

Example:

Macro parameters with special characters

Macro name:	Printf
Replacement text:	PRN#(0,{1})
Programmed block	10 Printf(\"A CPL string with \, and \\)\")
Acts like	10 PRN#(0,\"A CPL string with , and)\")

Example:

Replacement text with special characters

Macro name:	Bracket
Replacement text:	"{bracket}"
Programmed block	10 PRN#(0,bracket)
Acts like	10 PRN#(0,"{bracket}")

13.1.8 Execution Tools

In configuration files, macros are defined by XML.

XML (extensible markup language): XML is a meta language to define document types (e.g. documents similar to HTML) and was authorized by the World Wide Web Consortium (W3C).

Advantages:

Macros become independent of their platform in the xml-defined structuring and can be processed by internal system parts (e.g. CPL) or by XML or HTML-capable applications (e.g. MS Internet Explorer) or simple text editors.



The developer requires extensive programming know-how to work with the meta languages XML or XML schema.

13.1.9 Creating and Applying Macros

NC macros:

- They are XML-based in definition files with name, replacement text and attributes (see "macro definition files")
- They are programmed in CPL or DIN context (depending on the definition) in the part program
- They are replaced internally by the replacement text
- The number of DIN macros is limited to 100. CPL macros are only limited by the available memory.

The macro definitions are saved implicitly in the archive upon saving the RAM file system. Thus, the definitions can be read in again from an archive after software exchange.

The definition files can be created and modified in the editor of IndraWorks Operation and in other editors.

13.2 Macro Definition Files

13.2.1 Description

Macros are described in definition files. In these files, name (e.g. Macro1), replacement text (e.g. G0 X10) and attributes (e.g. DIN macro) of each macro are defined.

13.2.2 Structuring a Definition File

The definition file starts with the header. There, the valid XML version and the XML format are described in UTF-8 coding.

The macros to be defined, all of which are enclosed in the so-called TAG <MacroDef> (main element) follow. Each macro is defined with the TAG <Macro> (sub-element).

Example:

Definition file with macros HSC_ON and R11

```
<?xml version="1.0" encoding="UTF-8"?>
<MacroDef>
  <Macro Type="DIN">
    <Name>HSC_ON</Name>
    <Replace>SplineDef(4203,X,Y,Z) CLD(DFACT3.0, DIST5, ANG45, E0.1)</Replace>
  </Macro>
  <Macro Type="CPL">
    <Name>R11</Name>
    <Replace>SD.SysCyc.CD</Replace>
  </Macro>
</MacroDef>
```

A macro is defined with the TAG <Macro>. The declaration for its description is defined in more detail by sub-elements and attributes of the TAG <Macro>:

- *The TAG <Macro> has the following attributes:*
 - Type = "DIN" or "CPL" (mandatory)

- Overlay = "Yes" or "No" (only in DIN macros and optional, specification "No")
- *The sub-elements of <Macro> are:*
 - <Name> - Macro name (mandatory)
 - <Replace> - Replacement text (mandatory)
 - <Comment> - Comments (optional)

The definition files are read and analyzed at MTX next startup. From now onwards, all macros defined as valid are active. To modify macros, change the corresponding definition and reboot the control.

If errors occur while reading the configuration file, exactly one message is generated and all error data is written into the file "/macro/MacroErr.log". The analysis of the definition files is not canceled. All other errors are attached to this log file. The file is newly created at each system restart.

13.2.3 Names and Back Up of Definition Files

Macros created by the MTB or end user are created in specially named XML-based definition files (MacroDefxxx.xml see below):

- Macro definitions of the MTB are stored in the file **MacroDefMTB.xml**. The file is searched in the macro system search path. The root directory "/macro" has a higher priority than "/usrfep/macro" and "/feprom/macro".
- Freely definable macro definitions of the end user are stored in the file **MacroDefUsr.xml**. The file is searched in the macro system search path. The root directory "/macro" has a higher priority than "/usrfep/macro" and "/feprom/macro".
- Other topic-related macros can be stored in definition files of the type **MacroDefTopic.xml**. These files are also searched in the macro system search path. The root directory "/macro" has a higher priority than "/usrfep/macro" and "/feprom/macro".
- The internal NC system macros are defined in the file **MacroDef.xml**. This file is stored in the "/feprom/macro" directory and cannot be changed during operation.



For creating a definition file under XML, a special editor must be used (e.g. Emacs, Spy, etc.).

13.3 Example on XML-Based Macros

13.3.1 Specifying

Macros are to be defined for:

1. Changing the permitted axis acceleration at G0/G1
2. Accessing SD.SysCyc.SL as R11
3. Moving to a safe position using G0_Safe
4. Shortened programming with the substructure of SD.SysG81.Rvs via RVS
5. Hiding the underlying SD structure when checking a string

To change the configuration, create or supplement the file "/macro/MacroDefUsr.xml" (or a file according to section [Names and Back Up of Definition Files, page 127](#)).

1. Changing the permitted axis acceleration at G0/G1

Macro Programming

As an existing NC function is to be overridden using these two macros, the "overlay" attributes has to be set to "Yes" for both macros. Thus, the MTX "knows" that the operator is informed about the consequences of these macros.

Macro name:	G0
Replacement text:	ACC(X2.0, Y2.0, Z2.0) G0
Attribute:	DIN and overlay="yes"
Macro name:	G1
Replacement text:	ACC(X1.0, Y1.0, Z1.0) G1
Attribute:	DIN and overlay="yes"

Program:

```
<Macro Type="DIN" Overlay="Yes">
  <Name>G0</Name>
  <Replace>ACC(X2.0, Y2.0, Z2.0) G0</Replace>
</Macro>
<Macro Type="DIN" Overlay="Yes">
  <Name>G1</Name>
  <Replace>ACC(X1.0, Y1.0, Z1.0) G1</Replace>
</Macro>
```

Programmed block	N10 G0 (NIPS) X20
Acts like	N10 ACC (X2.0, Y2.0, Z2.0) G0 (NIPS) X20
Programmed block	N20 G1 F300 X10
Acts like	N20 ACC (X1.0, Y1.0, Z1.0) G1 F300 X10

2. Access to SD.SysCyc.SL as R11

At this point, a CPL macro has to be defined as the system data is accessed from the CPL.

Macro name:	R11
Replacement text:	SD.SysCyc.SL
Attribute:	CPL

Program:

```
<Macro Type="CPL">
  <Name>R11</Name>
  <Replace>SD.SysCyc.SL</Replace>
</Macro>
```

Programmed block	N30 X[R11]
Acts like	N30 X[SD.Sy- sCyc.SL]

3. Moving to a safe position with G0_Safe

The R11 macro can now be used again. In the replacement text, it has to be written in curly brackets "{}".

Macro name: **G0_Safe**

Replacement text: G0 X[{R11}]

Attribute: DIN

Program:

```
<Macro Type="DIN">
  <Name>G0_Safe</Name>
  <Replace>G0 X[{R11}]</Replace>
</Macro>
```

Programmed block N40 G0_Safe M5

Acts like N30 G0 X[SD.Sys-
 sCyc.SL] M5

4. Shortened programming for the substructure of SD.SysG81.Rvs via RVS

For system data, it is not necessary that the replacement text contains the whole variable name. A substructure may also be shortened by a macro while the rest of the structure tree is programmed.

Macro name: **RVS**

Replacement text: SD.SysG81.Rvs

Attribute: CPL

Program:

```
<Macro Type="CPL">
  <Name>RVS</Name>
  <Replace>SD.SysG81.Rvs</Replace>
</Macro>
```

Programmed block N50 S[RVS.Speed]

 N50

Acts like S[SD.SysG81.Rvs.Sp
 eed]

5. Hiding the underlying SD structure when checking a number

Since some characters are invalid (e.g. <, >, and &), in an XML text, they have to be replaced by corresponding character combinations. In this example, ">" is replaced by ">". Normal XML editors perform the replacement already during the character input.

Macro name: **isSPos**

Replacement text: {RVS}.StartPos >= {1}

Attribute: CPL

Program:

```
<Macro Type="CPL">
  <Name>isSPos</Name>
  <Replace>{RVS}.StartPos &gt;= {1}</Replace>
</Macro>
```

Programmed block 60 IF isS-
 Pos(#XPos) THEN

Acts like 60 IF
 SD.SysG81.Rvs.Star
 tPos >= #XPos THEN

13.3.2 Handling Instruction: Example

Applying If the file `"/macro/MacroDefUsr.xml"` does not exist, it has to be created. An empty definition file for macros only contains the opening and closing tags `<MacroDef>` and `</MacroDef>`. Afterwards, the macro definitions described above are supplemented in the `<MacroDef>` tags.

After the file has been saved, trigger a system restart to enable the MTX to read the definition files and the macros defined therein are activated.

If the configuration contains errors, exactly one message is created ("The macro configuration contains errors"). Each occurring error is written into the file `"/macro/MacroErr.log"` in detail. This file is overwritten at each control startup.

For this application, a definition file in XML format is generated. The XML files are created and edited by the integrated editor.

In this example, the abbreviation "MTB" labels a machine tool builder to differentiate the MTB-specific macros from the other macros.

The macros are accessed and used in the part program (end of the section).

The five MTB macros are:

1. Changing the permitted axis acceleration at G0/G1
2. Accessing SD.SysCyc.SL as R11
3. Moving to a safe position using G0_Safe
4. Shortened programming with the substructure of SD.SysG81.Rvs via RVS
5. Hiding the underlying SD structure when checking a string

Definition of the system data in the file `"MacroDefMTB.xml"`

Creating the file `MacroDefMTB.xml` in the directory `"/macro"` with the following content:

Program:

```
<?xml version=.1.0. encoding=.UTF-8.??>

<MacroDef>
  <Macro Type="DIN" Overlay="Yes">
    <Name>G0</Name>
    <Replace>ACC(X2.0, Y2.0, Z2.0) G0</Replace>
    <Comment>Aufgabe 1</Comment>
  </Macro>
  <Macro Type="DIN" Overlay="Yes">
    <Name>G1</Name>
    <Replace>ACC(X1.0, Y1.0, Z1.0) G1</Replace>
    <Comment>Aufgabe 1</Comment>
  </Macro>

  <Macro Type="CPL">
    <Name>R11</Name>
    <Replace>SD.SysCyc.SL</Replace>
    <Comment>Aufgabe 2</Comment>
  </Macro>

  <Macro Type="DIN">
    <Name>G0_Safe</Name>
    <Replace>G0 X[{R11}]</Replace>
    <Comment>Aufgabe 3</Comment>
  </Macro>

  <Macro Type="CPL">
    <Name>RVS</Name>
    <Replace>SD.SysG81.Rvs</Replace>
    <Comment>Aufgabe 4</Comment>
  </Macro>

  <Macro Type="CPL">
```

```

<Name>isSPos</Name>
<Replace>{RVS}.StartPos &gt;= {1}</Replace>
<Comment>Aufgabe 5</Comment>
</Macro>
</MacroDef>

```



Before <MacroDef>, the header specifies the XML version and the coding. All subsequently defined macros are enclosed in <Macro>.

The macro definitions are executed in <Macro>. For each <Macro>, there is the obligatory attribute <Type> (CPL or DIN) and the optional attribute <Overlay> (Yes or No).

The obligatory sub-elements <Name> (the macro name) and <Replace> (the replacement text) as well as the optional sub-element <Comment> (any comment) are enclosed in the <Macro> tags.

Use in the part program

Example program "TestMTB.npg"

The following has been programmed:	Acts like:
N10 G0(NIPS) X20 S3000	N10 ACC(X2.0, Y2.0, Z2.0) G0(NIPS) X20 S3000
N20 X120 Y110	N20 X120 Y110
N30 G1 F300 X100	N30 ACC(X1.0, Y1.0, Z1.0) G1 F300 X10
N40 Y[R11]	N40 Y[SD.SysCyc.SL]
N50 X80	N50 X80
N60 G0_Safe	N60 G0 X[SD.SysCyc.SL] S[SD.SysG81.Rvs.Speed]
S[RVS.Speed]	N70 WAIT
N70 WAIT	80 IF SD.SysG81.Rvs.StartPos >= #XPos THEN
80 IF isSPos(#XPos) THEN	90 GOTO N10
90 GOTO N10	100 ENDIF
100 ENDIF	N110 M30
N110 M30	

14 Fast actions MTX fa

14.1 Overview

14.1.1 Purpose

The fast actions functionality enables interpolator-synchronous PLC tasks to exchange data between the NC and the respective PLC task in an interpolator-synchronous way. Using this functionality, fast reactions to external and internal events can be implemented. For example, it is possible to immediately stop an axis (with feed hold) if an external event occurs, e.g. by using a high-speed input.

14.1.2 Terms

Fast action	A fast action is an action that is executed with a high priority. For example, high priority influence and/or overrides of processes/movements or fast switching of I/Os.
FA	FA is short for Fast Action.
Fast PLC task	This refers to a fast, interpolator-synchronous PLC task with a high priority that has a fast data connection to the NC as well as to the periphery.
Fast NC task	This is a fast NC task with a high priority that is called cyclically after every interpolator cycle. On the NC side, the task provides and forwards the required data.
MTX fa	MTX fa refers to the complete fast actions functionality.

14.1.3 Operating principle

The FA functionality consists of one or several fast PLC tasks. Contrary to other PLC tasks, these tasks are called interpolator-synchronously and have a fast data connection to the NC and to parts of the periphery (Sercos drives, high-speed I/Os). Periphery data for which no fast action items are used are processed according to the general IndraLogic configuration!

If function blocks are called in a fast PLC task, the runtime can increase considerable or complete cycles can be fail. The same applies to NC bit interface accesses as well as to I/Os if they are not accessed via cyclic data exchange by fast actions (this can result in an watchdog error!).



To ensure an interpolator-synchronous data exchange between NC and PLC task, only use the data defined under [chapter 14.2 "Configuration of the cyclic data exchange"](#) on page 140.

Data transfer is configured via a system data structure. This may either be done via the HMI, an NC program or the PLC. With this system data structure, the fast PLC tasks including the fast data connection can be started and stopped. Also, further information may be accessed for monitoring, such as synchronous running of tasks or runtime behavior of the involved instances.

- Interpolator [Interpolator functions]
- NC [NC functions]
- Sercos drives (with and without MLD) [Sercos drive functions]
- System data



The MTX fa functionality is available in the MTX 13VRS and above.

Due to the performance requirements of the "fa" functionality, a control of the performance classes "MTX performance" and "MTX advanced" are mandatory up to (including) the MTX 14VRS. Thus, controls of the performance classes "MTX standard" and "MTX micro" do not support the "fa" functionality.

14.1.4 Example

Task

Using the MTX fa functionality, the override in a machine is to be kept at a constant value when a specified limit "MaxOvr" is reached.

Solution

The required functionality is to be implemented using the MTX fa functionality.

For this purpose, an interpolator-synchronous fast PLC task is to be created that cyclically reads the path override (PathOvr) and correspondingly writes to a second override (FAPathOvr) if the specified limit is exceeded. By using the FAPathOvr, the resulting path override generated from the product of PathOvr and FAPathOvr can be modified.

Implementing the solution

Creating FA system data

Start IndraWorks Engineering, use the Project Explorer to go online and open the NC file system of IndraWorks Operation.

1. Open the "feprom" directory.
2. Copy the file "SDDeffa.xml_" to "usrfep" and remove the underscore.
The "SDDeffa.xml_" file is now available in "usrfep".
3. Open the "feprom/schemes" directory.
4. Copy the file "sdfa.xsd_" to "usrfep/schemas" and remove the underscore.



If it does not exist yet, create the "schemas" folder below "usrfep".

The "sdfa.xsd" file is now available in "usrfep/schemas".

5. Restart the NC.

Creating a PLC program for fast PLC task

IndraWorks Engineering is open and the logic is not logged in

1. Right-click on the node "Applications" (below IndraMotion/Logic).
Select "Add" in the context menu, then select POU.
2. Select the program name and insert it into the first line in the open dialog (e.g. "FastAction"). Then, select "Program", choose an implementation language and confirm the entries using the respective button.

Creating a fast PLC task

IndraWorks Engineering is open and the logic is not logged in

1. Right-click on the node "Task Configuration" (below IndraMotion/Logic/ Application).
First, select "Add" in the context menu, then select "Task".
A dialog opens to enter the task name.
2. Enter the desired name of the task in the open dialog and confirm it with the respective button.

Configuring the Fast PLC Task

IndraWorks Engineering is open and the logic is not logged in

1. Double-click on the newly created task.

The following dialog opens

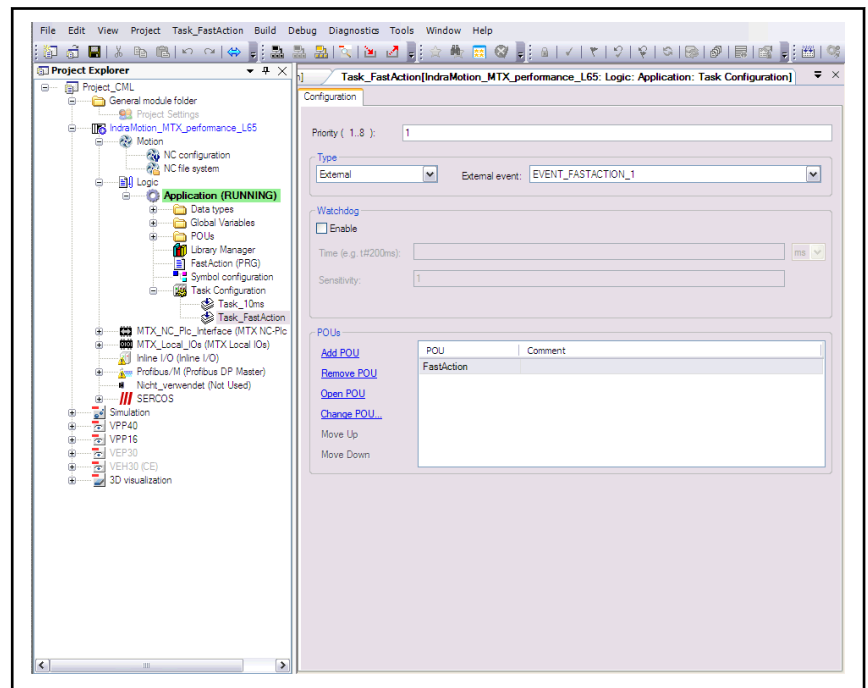


Fig. 14-1: Dialog for task configuration

2. Enter the required settings in this dialog:
 - Priority = 1
 - Type = External
 - External event = select EVENT_FASTACTION_i (i = 1, 2, 3 or 4)
 - Watchdog: Do not tick Enable!
(Using the system date "MaxFARunTime", the NC already implements a watchdog timer to monitor the data exchange between the fast PLC task and the fast NC task.)
 - Assign the required programs using AddPOU.

Click on AddPOU to open a dialog to select the desired program

Type definition for the FA input data that are to be transmitted cyclically

IndraWorks Engineering is open and the logic is not logged in

1. Right-click on the node "Application".
First select "Add" in the context menu, then select "Data types ...".

Fast actions MTX fa

A dialog opens to enter the data type name.

2. Enter the desired data type name of the FA input data structure (e.g. "FAIn_t(STRUCT)") and select "Structure" as the type.

Close the dialog.

The corresponding data type was inserted.

3. Double-click on the respective node.

Enter the data to be transmitted cyclically and their data types into the following structure. In this example, enter "PathOvr" of type "REAL".

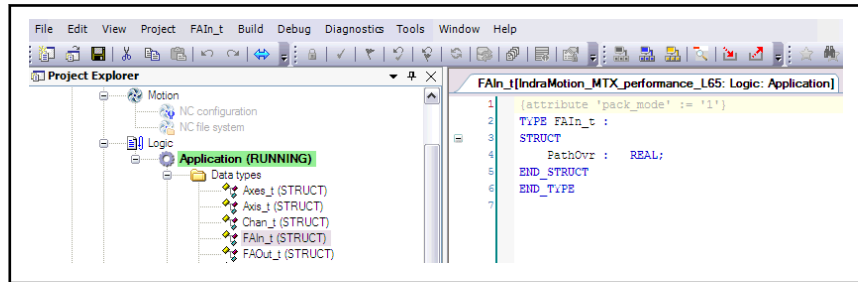


Fig. 14-2: Type definition for the FA input data that are to be transmitted cyclically

Type definition for the FA output data that are to be transmitted cyclically

IndraWorks Engineering is open and the logic is not logged in

1. Right-click on the node "Application".
First select "Add" in the context menu, then select "Data types ...".
A dialog opens to enter the data type name.
2. Enter the desired data type name of the FA output data structure (e.g. "FAOut_t(STRUCT)") and select "Structure" as the type.

Close the dialog.

The corresponding data type was inserted.

3. Double-click on the respective node.

Enter the data to be transmitted cyclically and their data types into the following structure. In this example, enter "FAPathOvr" of type "REAL".

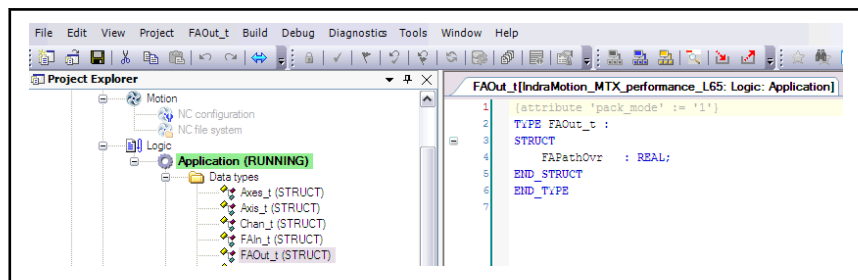


Fig. 14-3: Type definition for the FA output data that are to be transmitted cyclically

Creating a fast PLC program

IndraWorks Engineering is open and the logic is not logged in

1. Double-click on the program node ("FastAction(PRG)").

The program editor opens

2. Declare the required variables.

Important: Create the cyclic FA input data and output data using the data types created previously.

3. Insert the following PLC program.

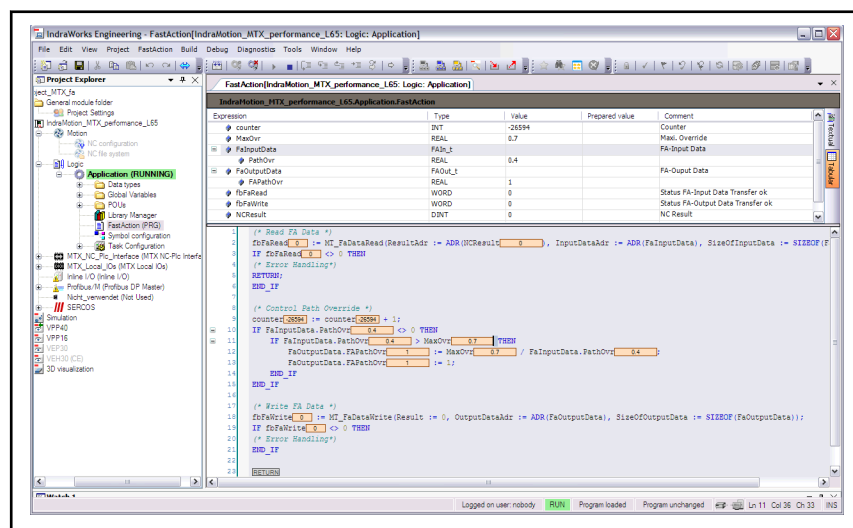


Fig. 14-4: Example of a PLC program with variable declaration

4. Click on the **Save** icon to save the program.
5. Compile the program and transfer it to the control by logging in (click on the respective **Login** icon).
6. Click on the **Start** icon to start the fast PLC task with the program FastAction.



The program FastAction is not executed yet, as the fast PLC task is not yet started by the NC.

Explanations on the fast PLC program

The PLC program called "FastAction" consists of three parts:

- First, it checks whether an error occurred while reading the required data.
- In the following, there is the actual override control logic.

The following applies: **Resulting path override = path override * FA path override**

If the path override (FaInputData.PathOvr) $\neq 0$, the path override is monitored with regard to MaxOvr. If the path override exceeds MaxOvr, the FA path override (FaOutputData.FAPathOvr) is calculated as follows: **FA path override = MaxOvr / path override**. If the path override remains below MaxOvr, the FA path override is set to "1".

- Finally, the program checks the writing of FA data.



Due to its simplicity, the "Result" input and/or the respective system data element "UsrError" was not used in the example.

Configuring the NC FA Functionality

Use the following program to carry out the required FA configuration. To make the program clearer and easier, use subroutines.

Fast actions MTX fa

Below "Reset FA data", the system data structure is deleted and a reset is carried out. In the "Set FA data" part, the basic settings for the fast PLC task including the data to be exchanged cyclically are then specified. In the "Validate" part, the MTX checks the specified data. In the final part, "Start fast PLC task", the result of the validation is read. If the check was passed, the respective fast PLC task is started before the actual application starts.

Program:

```
; -----
; Reset FA-Task
; -----
100 SD.SysFA.PLCTask[1].Control.Enable = 0
110 SD.SysFA.PLCTask[1].Config.Base.IECTaskName = ""
120 SD.SysFA.PLCTask[1].Config.Base.Cycle = 0
130 SD.SysFA.PLCTask[1].Config.Base.MaxPLCTaskTime = 0
140 FOR I% = 1 TO 20
150   SD.SysFA.PLCTask[1].Config.CycData[I%].Cmd = ""
160   SD.SysFA.PLCTask[1].Config.CycData[I%].ID = 0
170   SD.SysFA.PLCTask[1].Config.CycData[I%].Access = 0
180   SD.SysFA.PLCTask[1].Config.CycData[I%].Index1 = 0
190   SD.SysFA.PLCTask[1].Config.CycData[I%].Index2 = 0
200   SD.SysFA.PLCTask[1].Config.CycData[I%].Index3 = ""
210   SD.SysFA.PLCTask[1].Config.CycData[I%].DataSize = 0
220 NEXT I%
230 SD.SysFA.PLCTask[1].Config.Validate = 1
240 SD.SysFA.PLCTask[1].Control.Reset = 1
250 WAIT
; -----
; Set FA-Task
; -----
; Config Base Data
260 TASKTIME% = INT(SD.SysFA.NCTask.IPOCycleTime / 10) * 8
270 SD.SysFA.NCTask.MaxFARunTime = TASKTIME%
280 SD.SysFA.PLCTask[1].Config.Base.IECTaskName = "Task_FastAction"
290 SD.SysFA.PLCTask[1].Config.Base.Cycle = 1
300 SD.SysFA.PLCTask[1].Config.Base.MaxPLCTaskTime = TASKTIME%
; Confic Cycle Data: Write FA Path Override
310 SD.SysFA.PLCTask[1].Config.CycData[1].Cmd = "FAPathOvr"
320 SD.SysFA.PLCTask[1].Config.CycData[1].Access = 1
330 SD.SysFA.PLCTask[1].Config.CycData[1].Index1 = 1
340 SD.SysFA.PLCTask[1].Config.CycData[1].Index2 = 0
350 SD.SysFA.PLCTask[1].Config.CycData[1].Index3 = ""
; Confic Cycle Data: Read Path Override
360 SD.SysFA.PLCTask[1].Config.CycData[1].Cmd = "PathOvr"
370 SD.SysFA.PLCTask[1].Config.CycData[1].Access = 0
380 SD.SysFA.PLCTask[1].Config.CycData[1].Index1 = 1
390 SD.SysFA.PLCTask[1].Config.CycData[1].Index2 = 0
400 SD.SysFA.PLCTask[1].Config.CycData[1].Index3 = ""
; -----
; Validate
; -----
410 SD.SysFA.PLCTask[1].Status.CheckFin = 0
420 SD.SysFA.PLCTask[1].Config.Validate = 1
430 TIMEDELAY%=0
440 WHILE(SD.SysFA.PLCTask[1].Status.CheckFin = 0) DO
450   TIMEDELAY% = TIMEDELAY% + 1
460   IF TIMEDELAY% > 20 THEN SETERR("Error Timeout") : ENDIF
470   WAIT(,100)
480 END
; -----
; Start Fast PLC Task
; -----
490 IF ( (SD.SysFA.PLCTask[1].Status.NCError = 0) AND (SD.SysFA.PLCTask[1].Status.DataOk = 1) ) THEN
500   SD.SysFA.PLCTask[1].Control.Enable = 1
510 ELSE
520   SETERR("Error Enable")
530 ENDIF
540 WAIT
; -----
; Start Application
; -----
```

Configuring via system data file

If the fast actions should take place with a specified configuration, it is recommended to save the configurations to an SDDatFA.xmlfile after the commissioning phase. The fast actions are then immediately applied after control startup. Prerequisite: The configuration is valid and the system data structure is created as volatile system date.

14.1.5 Data exchange between fast PLC tasks, NC and drives

Data exchange between the fast PLC tasks and the NC, the interpolator and the Sercos drives is carried out via the fast NC task. From the left to the right, the following figure shows the time sequence of data transfer between a fast PLC task and the NC, the interpolator and the Sercos drives. From top to bottom, all relevant subsystems/tasks and the data to be exchanged are displayed with the corresponding fast actions.

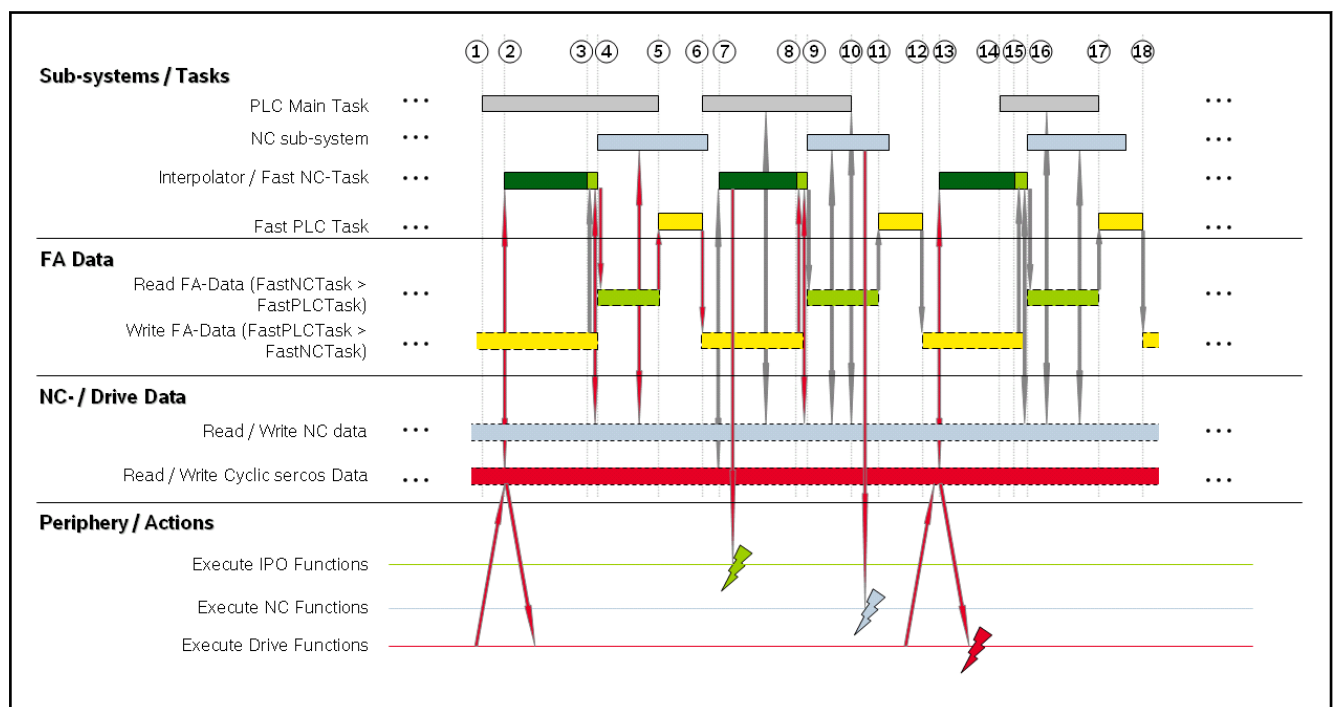


Fig. 14-5: Data exchange between the fast PLC tasks and the NC/drives (L85 – multi-core)

Explanation of the data exchange between the fast PLC tasks and the NC/drives

No.	Sub-step	Individual steps
1, 6, 14, ...		The default PLC program is executed
2, 7, 13, ...	Interpolator is started	- The drive data is read - New commands/data are forwarded to the drive
3, 8, 15, ...	The fast NC task is started	- The data provided by the fast PLC task is read and forwarded to the NC if required - The NC and drive data to be transmitted to the fast PLC task is read and forwarded
4, 9, 16, ...	The NC subsystems are running	The NC data is read and written
5, ...	PLC main task is interrupted	If the PLC main task is running, it is interrupted by the fast PLC task with a higher priority

Fast actions MTX fa

No.	Sub-step	Individual steps
5, 11, 17, ...	The fast PLC task is started	The data provided by the fast NC task is read
5-6, 11-12, 17-18, ...	The fast PLC task is processed	The corresponding programs are processed
6, 12, 18, ...	The fast PLC task is completed	Data to be forwarded to the NC and the drives are written

Tab. 14-1: Data exchange between fast PLC tasks, NC and drives

Consideration of a fast PLC task cycle (see red arrows in the figure)

Time	Process	Description
2	Read	The interpolator reads the drive data (e.g. actual position of axis X) and provides them to the fast NC task
4		The fast NC task reads the NC data (e.g. system data) and forwards them to the fast PLC task together with the drive data
5		The fast PLC task reads the NC data and drive data
5 - 6	Logic	The logic stored in the fast PLC task is executed
6	Write	At the end of the fast PLC task, the specified FA data (e.g. for the interpolator, the NC subsystem or a drive) is written
7-8		The interpolator starts the respective FAs if required
9		The fast NC task writes the NC data (e.g. system data)
9		The NC (subsystems) starts the respective FAs in the following if required
13		The interpolator writes drive data
from 13		The drive starts the respective FA if required

Tab. 14-2: Reading and writing drive data and NC data

14.2 Configuration of the cyclic data exchange

14.2.1 Cyclic data exchange

Basic information

The configuration of the cyclic data exchange between the fast NC task and the fast PLC tasks uses the FA system data. There, the fast PLC tasks can be enabled and disabled, the required transmission channels between the interpolator and the fast PLC tasks can be set up, and the data to be exchanged between the fast NC task and the fast PLC tasks can be defined.

In addition, data transmission, synchronism between the fast PLC tasks and the interpolator and the runtime behavior can be monitored.

Overview on FA data

The following table shows the FA data that can be exchanged cyclically between the fast PLC task[i] and the fast NC task if required. The corresponding command (or the corresponding ID) are listed in the respective columns. Any additional information that may be required to select the date to be exchanged cyclically can be specified using index 1, 2, 3 and 4. Also note the "Access" date. "0" (Read) means: the Fast PLC task[i] reads the respective

date (provided by the fast NC task). Correspondingly, "1" (Write) means: The fast PLC task[i] writes the respective date (forwarded to the fast NC task for distribution).

FA data	Com- mand	ID	Index 1 [int]	Index 2 [int]	Index 3 [string]	Index 4 [int]	Access	DataSize [BYTE]	NC data type	PLC data type
Interpolator data/functions										
Read the axis position	AxisPos	100000	Axis number	0: Command 1: Actual	-	-	0: Read	8	DOUBLE	LREAL
Read path override	PathOvr	100400	Channel number	-	-	-	0: Read	4	REAL	REAL
Write FA path override	FAPa-thOvr	100401	Channel number	-	-	-	1: Write	4	REAL	REAL
Read path velocity (command velocity)	PathVelo	100411	Channel number	-	-	-	0: Read	4	INT	DWORD
Read distance to go	PathDist-ToGo	100410	Channel number	-	-	-	0: Read	8	DOUBLE	LREAL
NC data/functions										
Read FA feed hold	FAFeed-Hold	100402	0.1: Channels 1..32 2: Channels 33..64	-	-	-	0: Read	4	INT	UDINT
Write FA feed hold	FAFeed-Hold	100402	0.1: Channels 1..32 2: Channels 33..64	-	-	-	1: Write	4	INT	UDINT
Read system data	System-Data	100006	-	-	XPath	-	0: Read	1	UN-SIGNED-BYTE	BYTE
								1	BYTE	SINT
								1	BOOLEAN	BOOL
								2	SHORT	INT/UINT
								4	INT	DINT/UDINT
								4	FLOAT	REAL

Fast actions MTX fa

FA data	Com- mand	ID	Index 1 [int]	Index 2 [int]	Index 3 [string]	Index 4 [int]	Access	DataSize [BYTE]	NC data type	PLC data type
Write system data	System- Data	100006	-	-	XPath	-	1: Write	1	UN- SIGNED- BYTE	BYTE
								1	BYTE	SINT
								1	BOOLEAN	BOOL
								2	SHORT	INT/UINT
								4	INT	DINT/ UDINT
								4	FLOAT	REAL
Read perma- nent CPL variables	CPLVar	100002	-	-	Variable name	-	0: Read	1	BOOLEAN	BOOL
								4	INT	DINT/ UDINT
								4	REAL	REAL
Write perma- nent CPL variables	CPLVar	100002	-	-	Variable name	-	1: Write	1	BOOLEAN	BOOL
								4	INT	DINT/ UDINT
								4	REAL	REAL
Read event	Event	100500	Event no. (1 ... 96)	-	-	-	0: Read	1	INT	SINT
Write event	Event	100500	Event no. (1 ... 96)	-	-	-	1: Write	1	INT	SINT
Write event cy- clically	Event	100500	Event no. (1 ... 96)	-	cyclic	-	1: Write	1	INT	SINT
Drive data/functions										
Read Sercos data	Sercos- Data	100005	Axis number	Sercos ID	P / S	0: AT	0: Read	2	INT	INT/UINT
						1: MDT		4	INT	DINT/ UDINT
Write Sercos data bit- by-bit	Sercos- DataBit- Mode	100004	Axis number	Sercos ID	P / S	-	1: Write	8	LONGINT	LINT/ ULINT
Write Sercos data	Sercos- Data	100005	Axis number	Sercos ID	P / S	-	1: Write	2	INT	INT/UINT
								4	INT	DINT/ UDINT
High-speed I/Os										

FA data	Com- mand	ID	Index 1 [int]	Index 2 [int]	Index 3 [string]	Index 4 [int]	Access	DataSize [BYTE]	NC data type	PLC data type
Read high-speed I/Os	High-SpeedIO	101000	- (all inputs are transmitted)	-	-	-	0: Read	1 (during reading, all high-speed I/Os are transmitted)	SINT	BYTE
Write high-speed I/Os	High-SpeedIO	101000	- (all outputs are transmitted)	-	-	-	1: Write	2 Byte0: Value Byte1: Mask	INT	WORD

Tab. 14-3: Overview on FA data: Fast data that can be exchanged between the interpolator and the fast PLC tasks



- **Characters**

- : No input required.

- **Access**

Access always has to be specified.

0: Read: Transport direction fast NC task > fast PLC task

1: Write: Transport direction fast PLC task > fast NC task

If a data element (e.g. a CPL variable) is to be both read and written by the fast PLC task, both transport directions have to be configured via the corresponding system data structure.

- **Axis and channel numbers**

Axis numbers and channel numbers start with 1.

- **Axis positions**

Axis positions can be read as command values and actual values in axis coordinates. They do not need to be created as cyclic Sercos data in the telegram.

- **NC data type**

This entry provides additional information on how the NC expects or provides the value. Type incompatibility can cause a system crash. Thus, the user has to compatibly create the data types in the PLC project. This information is stored under "PLC data type".

- **Sercos drive data**

The data transmitted in the cyclic response telegram (AT) can be provided to the fast PLC tasks for write access if required. This means that all Sercos drive data to be accessed by the fast PLC tasks have to be included in the response telegram (RT).

Use index 4 = 1 to read Sercos drive data from the master data telegram (MDT).

The data transmitted in the cyclic master data telegram (MDT) can be provided to the fast PLC tasks for write access, if required. If the fast PLC tasks are to have write access to drive data that is not in the MDT yet, they need to be included.

- **Writing Sercos drive data bit-by-bit**

The data transmitted in the cyclic master data telegram (MDT) can be overwritten bit-by-bit by the fast PLC tasks (FA tasks).

In each cycle, the FA task transfers the value to be written (as bit mask) as well as the bits to be written (four bytes each). Only the bits specified in the mask are subsequently transferred to the drive.

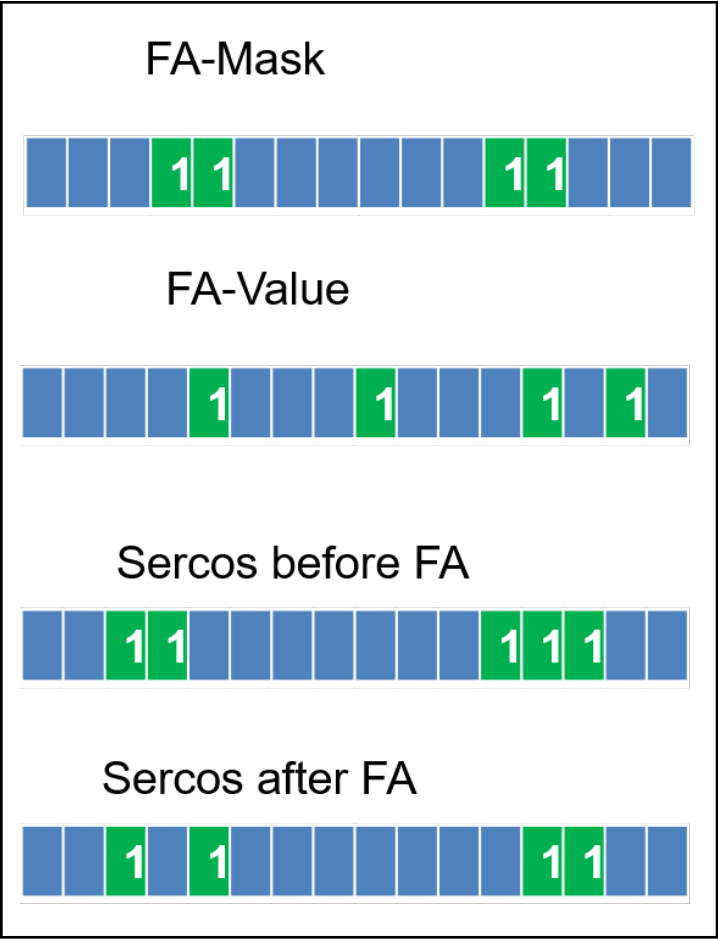


Fig. 14-6: Example on the design and effect of the masks
Value and mask are four bytes for 2-byte and for 4-byte Sercos ident numbers. For 2-byte Sercos ident numbers, only the two least significant bytes of mask and value are evaluated.

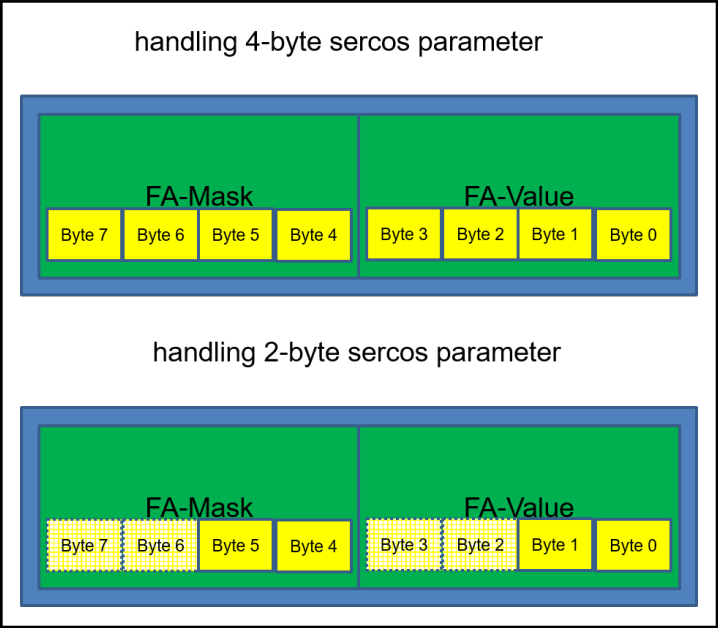


Fig. 14-7: Design and data structure of SercosDataBitMode

This can also be used to affect control data of the drive. Example: Setting the axes to standstill the fastest possible.

NOTICE

Directly controlling the drive can damage the machine. The values generated by the NC are overwritten.

This function may only be used by trained staff and commissioned in uncritical areas.

- **Path override**

The path override is provided as factor. The value 1.0 means 100%

- **Write FA path override**

The FA path override has to be specified as factor. The value 1.0 means 100%. Values < 0.001 means 0%. The permissible value range is between 0.0 and 1.0.

The resulting path override is the product of the path override and the FA path override.

- **Path velocity**

The path velocity (command velocity) is output in 1/100 mm/s.

- **Path distance to go**

By using the PathDistToGo command, the distance to go for the active NC block can be read. The block reference has to be ensured by the user (e.g. M-function). In case of split blocks, distances to go of the split partial blocks (not of the original block!) are returned. The distance to go is provided in the unit [10e-4 mm]. The command exclusively refers to the active path axes.

- **FA feed hold**

Der FA feed hold acts like a PLC feed stop: Only in automatic mode and only on synchronous axes. Both reading and writing the FA feed hold always requires a 4-byte value. The value contains bit-coded specifications for up to 32 channels.

Use Index1 to specify which channel group can be addressed:

Index1=0/1 channels 1..32

Index1=2 channels 33..60

Example: Read feed hold

_____Byte1_____Byte0_____

0000 0011 0001 0001 ; channels 1, 5, 9 and 10 FA feed hold

- **Read or write system data**

STRING type system data elements are not supported.

Avoid reading/writing system data > 4 bytes, as the consistency of the data element to be transmitted cannot be guaranteed.

- **Permanent CPL variables**

STRING type permanent CPL variables are not supported.

Avoid reading/writing system data > 4 bytes, as the consistency of the data element to be transmitted cannot be guaranteed.

- **Events**

When an NC event is written, the waiting NC function WEV/WREV is only notified if the value of the event changes (0 > 1).

If the waiting NC function WEV/WREV is last be to informed when the event is set but the edge is not changed, the keyword "cyclic" has to be entered into index3 during the configuration.

- **High-speed inputs**

During reading, the complete area (1 byte) is read and stored in byte 0 in a bit-coded way.

- **High-speed outputs**

- Only one action can be configured to write high-speed outputs.
- When writing the high-speed outputs, the complete area (1 byte) is written, with byte 0 always containing the bit-coded value and byte 1 containing the bit-coded mask.
- While writing, only the bits set in the mask are modified. All bits marked with "0" during the write operation in the mask are not written to high-speed outputs.
- When writing HS outputs, the assignment to functional units as specified in the machine data has been taken into consideration. Only bits unassigned or reserved for fast actions "PLC/DigIO/FuncAllocOut/DigIOAllocOut[.] = 7" according to machine data are written. Avoid to use unassigned bits, as they can be written by the PLC program.
- Writing by MTX fa is only possible if the interpolator exchanged the high-speed outputs (see machine data documentation).

Example: Writing high-speed outputs

_____Byte1_____Byte0_____

1010 1010 1111 0000 ; state of the high-speed outputs after writing: **1x1x 0x0x** (x: unchanged)

14.2.2 General notes

Alignment

The sequence of the data to be transmitted cyclically that was selected in the system data also has to be taken into account in the fast PLC tasks. Furthermore, the respective data types have to be used in the Fast PLC tasks (see [tab. 14-3 "Overview on FA data: Fast data that can be exchanged between the interpolator and the fast PLC tasks" on page 141](#), column "PLC Data Type"). This way, it is ensured that gaps are avoided.



Regarding read and write access to data structures, such as system data, make sure that the respective data elements that are read or written are located at an even-numbered address. Add filler bytes, if necessary.

For example, if there is a data element in a structure that is one byte in length, add a filler byte if the next following element is to be accessed.

Type definition for data exchange

Start every type definition, which is to be used to transmit FA data in a fast PLC task, with the instruction "{attribute 'pack_mode' := '1'}" (see [fig. 14-2 "Type definition for the FA input data that are to be transmitted cyclically" on page 136](#)). The example shows the type definition for reading the path override. The instruction "{attribute 'pack_mode' := '1'}" comes before the actual type definition.

14.2.3 Structure of the FA system data structure

The fast PLC task[i] is controlled and monitored using a system data structure "SDDeffa.xml" (called SysFA of type FA_t).

It can be subdivided into the following sections:

- PLCTask[i]
 - Control (> controls the fast PLC tasks)
 - Config (> configures the fast PLC tasks and the respective data transfer mechanisms)
 - Status (> monitors the fast PLC tasks and the respective data transfer mechanisms)
 - Info (> monitors the fast PLC tasks)
- NCTask (> monitors the fast NC tasks and the cyclic data transfer)



Currently, the system data structure supports two fast PLC tasks.

14.2.4 Controlling fast PLC tasks

Overview on the FA subsystem data structure Control

The FA subsystem data structure SysFA.PLCTask[i].Control controls the fast PLC task[i].

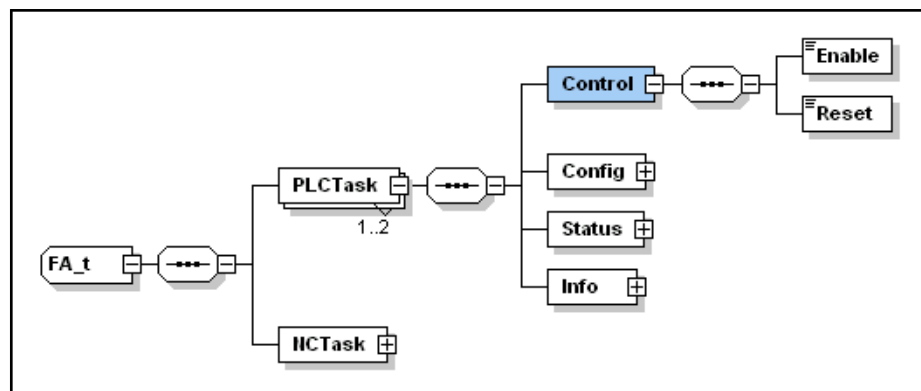


Fig. 14-8: FA subsystem data structure Control

Enable: Enable/disable the fast PLC Task[i]

ID	Enable
Type	xs:byte
Value range	0..1
Unit	-
Explanation	By setting the "Enable" element, the respective fast PLC task[i] is started after every nth interpolator cycle (= cycle). Correspondingly, by resetting the "Enable" element, the respective fast PLC task[i] is not started again
Note	-

Reset: Resetting the fast PLC Task[i]

ID	Reset
Type	xs:byte
Value range	0..1
Unit	-
Explanation	By setting the Reset bit, any reported FA error messages in the FA system data, the Wera and in the internal structures are deleted. Furthermore, the statistical data (below the substructures "SysFA.PLCTask.Info" and "SysFA.NCTask") in the FA system data is deleted
Note	A reset has no impact on the cyclic data transfer. It continues without interruption, unless it was stopped by a fatal error. If - due to an error - data exchange was not started (incorrect configuration) or interrupted (fatal error), this state persists. The only way to change this is to activate a correct configuration or to remove the cause of the fatal error

14.2.5 Setting up a transfer channel**Overview on the FA subsystem data structure Config**

The FA subsystem data structure SysFA.PLCTask[i].Config configures the fast PLC task[i] and the cyclic data transfer between the fast PLC task[i] and the interpolator. Furthermore, the substructure contains the respective error description for every configuration.

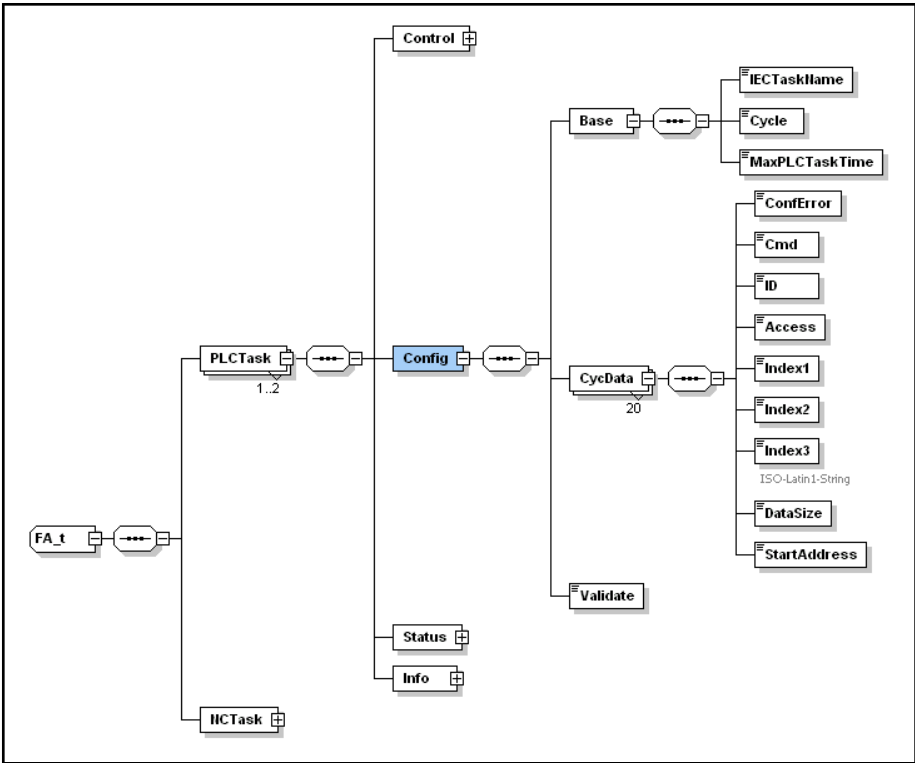


Fig. 14-9: FA subsystem data structure Config

IECTaskName: Name of the IEC task

ID	IECTaskName
Type	isoStr99_t
Value range	max. 19 characters
Unit	-
Explanation	By specifying the "IEC task name", it is assigned to the fast PLC task[i] on the PLC.
Note	The respective task is only started cyclically using the "Cycle" variable if there is a valid IEC task name. If there is an empty string, the NC concludes that there is no IEC task. During startup with Validate and Validate = 1, the existence of IEC tasks is checked on the basis of the names

Cycle: Start the fast PLC Task[i] after every nth cycle

ID	Cycle
Type	xs:unsignedInt
Value range	-
Unit	-
Explanation	Using the "Cycle" variable, the fast PLC task[i] can be started after every nth interpolator run. If the value 1 is specified, the fast PLC task[i] is started after every interpolator run
Note	If the value 0 is specified, the task is not started

MaxPLCTaskTime: Maximum permissible execution time of the fast PLC Task[i]

ID	MaxPLCTaskTime
Type	xs:unsignedInt
Value range	-
Unit	us
Explanation	With this data element, the NC implements a watchdog timer that monitors the execution time of the corresponding fast PLC task[i]. If the execution time of the fast PLC task exceeds MaxPLCTaskTime, a watchdog error is generated
Note	The maximum permissible execution time of the fast PLC task[i] is not to exceed 1/10 of the interpolation cycle time: MaxPLCTaskTime < 0.8 * IPOCycleTime Deactivation: MaxPLCTaskTime >= IPOCycleTime

14.2.6 Specifying data to be transmitted**ConfError: Configuration error (read-only)**

ID	ConfError
Type	xs:int
Value range	-
Unit	-

Fast actions MTX fa

Explanation	0: No error Individual access errors 1: Unknown ID 2: Index 1 is incorrect 3: Index 2 is incorrect 4: Index 3 is incorrect 5: Date cannot be written 6: Date cannot be read 7: Undefined access type 8: Access to double type system date (inconsistent) 9: Alignment of the system date is critical (inconsistent) 10: Invalid type 11: No address determined (variable not found) 12: No symbol access possible (IndraLogic symbol file) 13: No symbol information (system data) 14: The specified data length is invalid (less than required) 15: Redefine. The data element was configured several times 16: Unknown command syntax (system date Cmd) General configuration errors -1: No memory (internal error) -2: Task is not in the project -3: No trigger is configured -4: The task is configured several times
Note	• ConfError = 8 "Access to double type system date (inconsistent)" Recommendation: If possible, do not use this data type. • ConfError = 9 "alignment of the system date is critical (inconsistent)" Recommendation: Insert a filler byte in front of the respective date

Cmd: Command

ID	Cmd
Type	isoStr99_t
Value range	See tab. 14-3 "Overview on FA data: Fast data that can be exchanged between the interpolator and the fast PLC tasks" on page 141
Unit	-
Explanation	With this date, the user specifies the desired command using high-level language syntax
Note	The command syntax is case sensitive. If the data to be transmitted are selected using Cmd, the NC automatically establishes the corresponding ID

ID: Ident number (optional)

ID	ID
Type	xs:int

Value range	See tab. 14-3 "Overview on FA data: Fast data that can be exchanged between the interpolator and the fast PLC tasks" on page 141
Unit	-
Explanation	Optionally, the date to be transmitted can also be selected via the ID (instead of Cmd)
Note	-

Access: Read or write access

ID	Access
Type	xs:int
Value range	0,1
Unit	-
Explanation	With this date, the transport direction (between the fast PLC task[i] and the interpolator) is specified: 0: Read data (interpolator > Fast PLC task[i]) 1: Write data (fast PLC task[i] > interpolator)
Note	-

Index1: Index 1

ID	Index1
Type	xs:int
Value range	See tab. 14-3 "Overview on FA data: Fast data that can be exchanged between the interpolator and the fast PLC tasks" on page 141
Unit	-
Explanation	Use index 1 to describe the desired data in detail according to the selected ID
Note	-

Index2: Index 2

ID	Index2
Type	xs:int
	See tab. 14-3 "Overview on FA data: Fast data that can be exchanged between the interpolator and the fast PLC tasks" on page 141
Unit	-
Explanation	Use index 2 to describe the desired data in detail according to the selected ID
Note	-

Index3: Index 3

ID	Index3
Type	isoStr99_t
Value range	See tab. 14-3 "Overview on FA data: Fast data that can be exchanged between the interpolator and the fast PLC tasks" on page 141
Unit	-

Fast actions MTX fa

Explanation	Use index 3 to describe the desired data in detail according to the selected ID
Note	-

Index4: Index 4

ID	Index4
Type	xs:int
	See tab. 14-3 "Overview on FA data: Fast data that can be exchanged between the interpolator and the fast PLC tasks" on page 141
Unit	-
Explanation	Use index 4 to describe the desired data in detail according to the selected ID
Note	-

DataSize: Length of the date to be transmitted cyclically

ID	DataSize
Type	xs:int
Value range	See tab. 14-3 "Overview on FA data: Fast data that can be exchanged between the interpolator and the fast PLC tasks" on page 141
Unit	-
Explanation	Length of the respective date. The length is determined by the system
Note	Using this date, the PLC can be provided with other data while keeping the PLC data structure. In such cases, make sure that the number and length of the data packages to be transmitted remain constant. The length of the date to be transmitted cyclically always has to be $\geq$ of the length internally required (see overview table FA Data)

StartAddress: Start address of the date to be transmitted cyclically (read-only)

ID	StartAddress
Type	xs:unsignedInt
Value range	-
Unit	-
Explanation	Start address of the corresponding date in the respective transfer channel
Note	This data element is updated by the system. It can be used to check the consistency of the PLC data structure.

14.2.7 Starting the data transfer**Validate: Checking the new configuration**

ID	Validate
Type	xs:byte
Value range	0. 1
Unit	-

Explanation	0: No check required 1: Start the check With this date, the application informs whether a new configuration for cyclic data transfer has to be checked. If this bit is set, the specified data is checked. If no errors are found in the check and if the Enable bit is set, the cyclic data transfer for the corresponding fast PLC task[i] is started
Note	If this bit is set, the cyclic data transfer between the fast PLC task[i] and the interpolator is interrupted (during the check). As a result, the fast PLC task[i] is not started. When the specified data was applied successfully (correct configuration), this date is set to "0". If the Enable bit is set at that time, cyclic data transfer is continued. The "Validate" date is also analyzed during control startup

14.2.8 Monitoring the data transfer

Overview on the FA subsystem data structure Status

The FA subsystem data structure "Status" displays the current status.

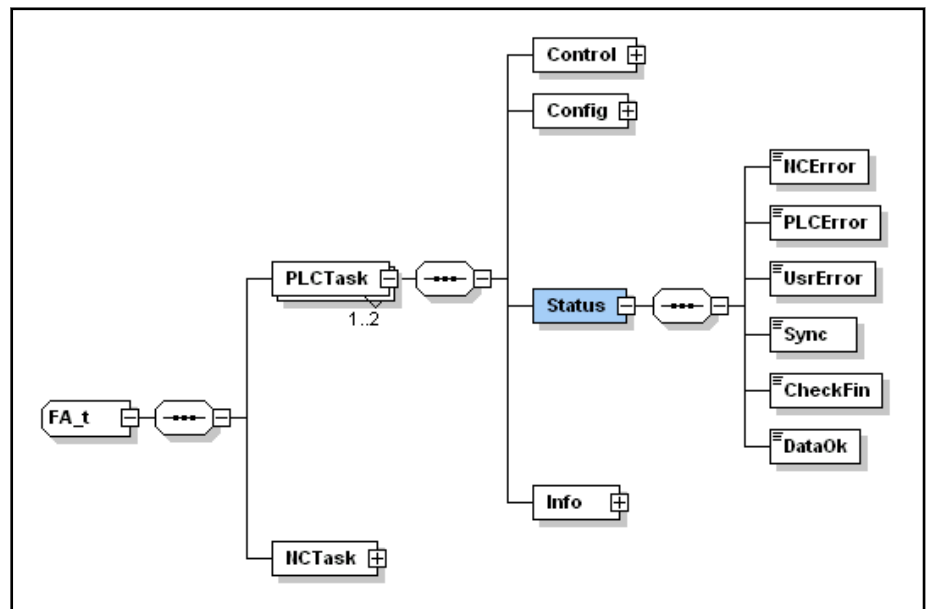


Fig. 14-10: FA subsystem data structure Status

NCErrors: Error in the NC

ID	NCErrors
Type	xs:int
Value range	-
Unit	-

Fast actions MTX fa

Explanation	If the NC detects an error, it is stored in NCErr in the FA system data using a respective error number. At the same time, an error message is created: 0: No error present 2: Synchronous run is interrupted 3: Incorrect data access was programmed (find additional information in the system data in FA_Diag and in the PLC) 4: Cyclic data transfer is not possible 5: Incorrect configuration (at least one of the existing data selections is incorrect)
Note	If there are subsequent errors, only the causative error is reported and stored in the FA system data. If an error disappears (e.g. if the synchronism between the fast PLC task[i] and the interpolator was re-established after a temporary error), this error remains stored until it is deleted using the Reset bit. FA errors can only be reset using the "Reset" bit. If the variables NCErr = 5 and CheckFin = 1, any errors in all data selections can be viewed in the FA_Diag system data structure below the "Config" substructure

PLCError: Error in the PLC

ID	PLCError
Type	xs:int
Value range	-
Unit	-
Explanation	If the PLC detects an error, the respective error number is stored in PLCError in the FA system data and an error message is reported: 0: No error -1: - -2: The forwarded task (task description) does not exist in the PLC -3: There was insufficient free memory for the FA tasks when the internal management structure was created -4: The number of the forwarded FA tasks exceeds the maximum number of FA tasks (2 tasks maximum) -5: - -6: The size of the forwarded data does not match the configured size -7: The version of the FA configuration in the request data in the NC does not match the current version in the PLC. -8: No FAs are configured in the PLC -9: When the PLC user program called MT_FaDataWrite(), unconfigured output data were transmitted -10: Null pointer when the user program calls MT_FaDataWrite() -11: - -100: Watchdog error, exceeding the monitoring time of MaxPLCTaskTime
Note	If the PLC detects an error while creating a fast PLC task with the corresponding communication channel, while configuring the data transfer, or during runtime, the respective error number is stored in the FA system data in PLCError and the NC reports the error

UsrError: User error message in the fast PLC task

ID	UsrError
Type	xs:int
Value range	-
Unit	-
Explanation	If an unexpected event/error occurs during fast PLC task execution, this can be reported in the fast PLC task using the function MT_FAWriteData (result input). The fast NC task then transmits the Result input to the system data element SD.SysFA.PLCTask[i].Status.UsrError
Note	If a user error is reported, the NC does not create an error message

Sync: The fast PLC Task[i] is synchronous to the interpolator

ID	Sync
Type	xs:byte
Value range	0. 1
Unit	-
Explanation	If the corresponding fast PLC task[i] is not synchronous with the interpolator, this is immediately displayed via this date and an error message is created at the same time: 0: The fast PLC task[i] is not synchronous to the interpolator 1: The fast PLC Task[i] is synchronous to the interpolator
Note	The Sync bit remains reset even if synchronism is re-established in the meantime. The respective error message also remains set. Both the Sync bit and the respective error message are only reset when the FA reset is carried out. Any error messages that are displayed, can be deleted using the Reset bit

CheckFin: The configuration was checked

ID	CheckFin
Type	xs:byte
Value range	0. 1
Unit	-
Explanation	This date indicates whether the configuration has already been checked. 0: The configuration has not been checked yet 1: The configuration was checked
Note	After control startup, bit CheckFin = 1 (the control has checked the current configuration). DataOK indicates if the check reported any errors

DataOk: The configuration is free of errors

ID	DataOk
Type	xs:byte
Value range	0. 1
Unit	-

Value range	-
Unit	Byte
Explanation	InSize displays the data volume transmitted from the interpolator to the fast PLC task[i]
Note	-

OutSize: Size of the data package sent from the fast PLC Task[i] to the interpolator

ID	OutSize
Type	xs: signedInt
Value range	-
Unit	Byte
Explanation	OutSize displays the data volume transmitted from the fast PLC task[i] to the interpolator
Note	-

14.2.10 Processing times and error counter of the fast NC task

Overview on the FA subsystem data structure NC task

The FA subsystem data structure NC Task monitors the fast NC tasks and the cyclic data transfer.

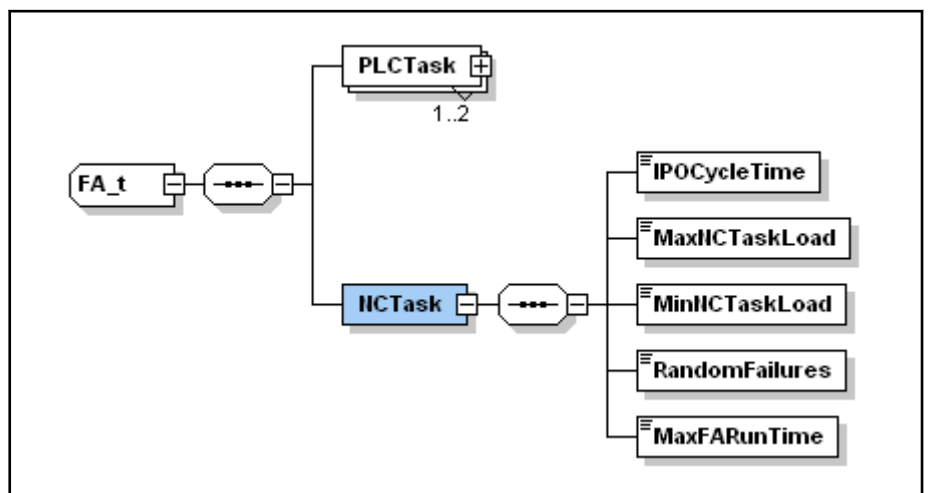


Fig. 14-12: FA subsystem data structure NCTask

IPOCycleTime: Specified interpolator time

ID	IPOCycleTime
Type	xs:unsignedInt
Value range	-
Unit	µs
Explanation	During startup, the NC reads the corresponding machine parameter (in µs) and transmits the respective value to the data element IPOCycleTime
Note	-

MaxNCTaskLoad: Maximum processing time of the fast NC task

ID	MaxNCTaskLoad
Type	xs:unsignedInt
Value range	-
Unit	µs
Explanation	The system date displays the maximum fast NC task execution time
Note	-

MinNCTaskLoad: Minimum processing time of the fast NC task

ID	MinNCTaskLoad
Type	xs:unsignedInt
Value range	-
Unit	µs
Explanation	The system date displays the minimum fast NC task execution time
Note	-

RandomFailures: Random errors

ID	RandomFailures
Type	xs:unsignedInt
Value range	-
Unit	-
Explanation	The NC counts all errors, such as losing synchronism with the interpolator. If several errors occur in a row or if an error occurs several times before the next reset, the counter RandomFailures is incremented by "1"
Note	The error counter is reset during the next FA reset

MaxFARunTime: Maximum permissible execution time

ID	MaxFARunTime
Type	xs:unsignedInt
Value range	-
Unit	µs
Explanation	Using this data element, the NC implements a watchdog timer to monitor the data exchange between the fast PLC task and the fast NC task. If it is exceeded, a watchdog error is created
Note	We recommend setting the maximum permissible execution time to approx. 3/10 of the interpolator cycle time: MaxFARunTime < 0.3 * IPOCycleTime If no value is entered, "10 microseconds" is used internally Deactivation: MaxFARunTime >= IPOCycleTime

14.3 Commissioning

14.3.1 Creating fast PLC tasks and programs

Creating FA programs

IndraWorks Engineering is open and the logic is not logged in

1. Right-click on the node "Applications" (below IndraMotion/Logic).

The following window opens:

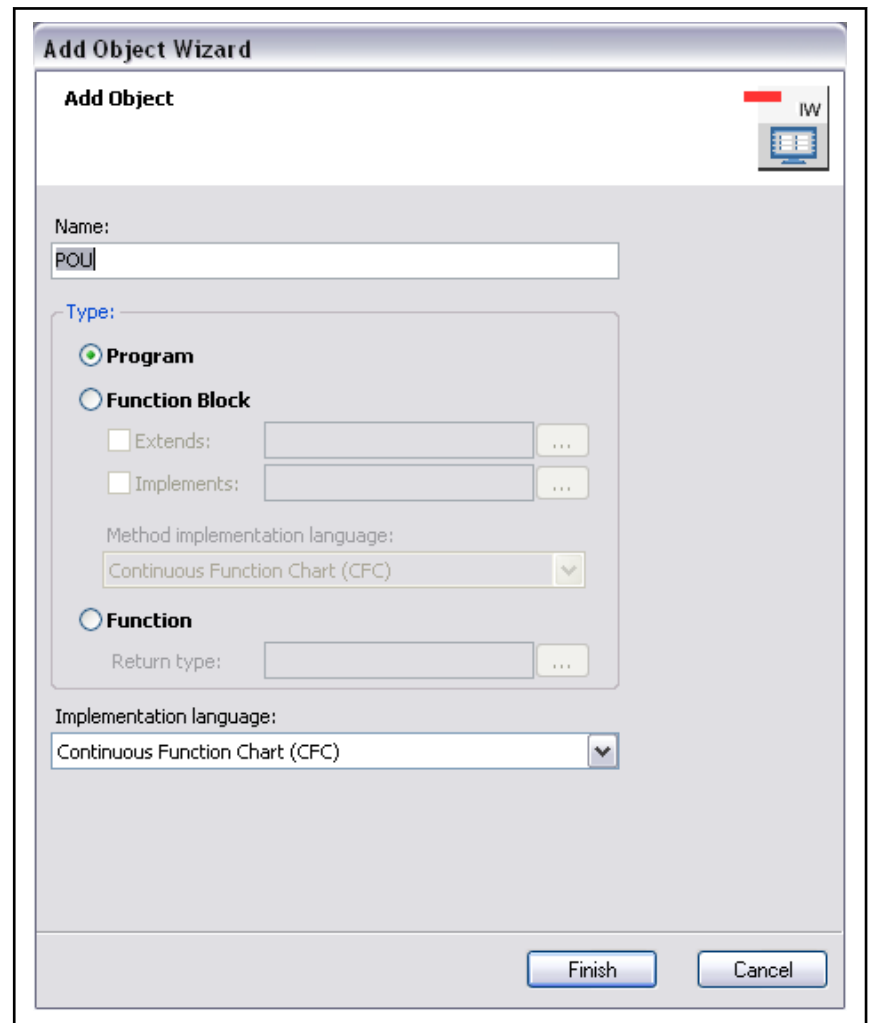


Fig. 14-13: Dialog to create a new program

2. In the open dialog, enter the program name into the first line. Then, select "Program", choose an implementation language and confirm the entries using the respective button.

Creating fast PLC tasks

IndraWorks Engineering is open and the logic is not logged in

1. Right-click on the node "Task Configuration" (below IndraMotion/Logic/Application).

First, select "Add" in the context menu, then select "Task".

A dialog opens to enter the task name.

2. Enter the name of the task in the open dialog and confirm it with the respective button.

Configuring the fast PLC task

IndraWorks Engineering is open and the logic is not logged in

1. Double-click on the newly created task.

The following dialog opens

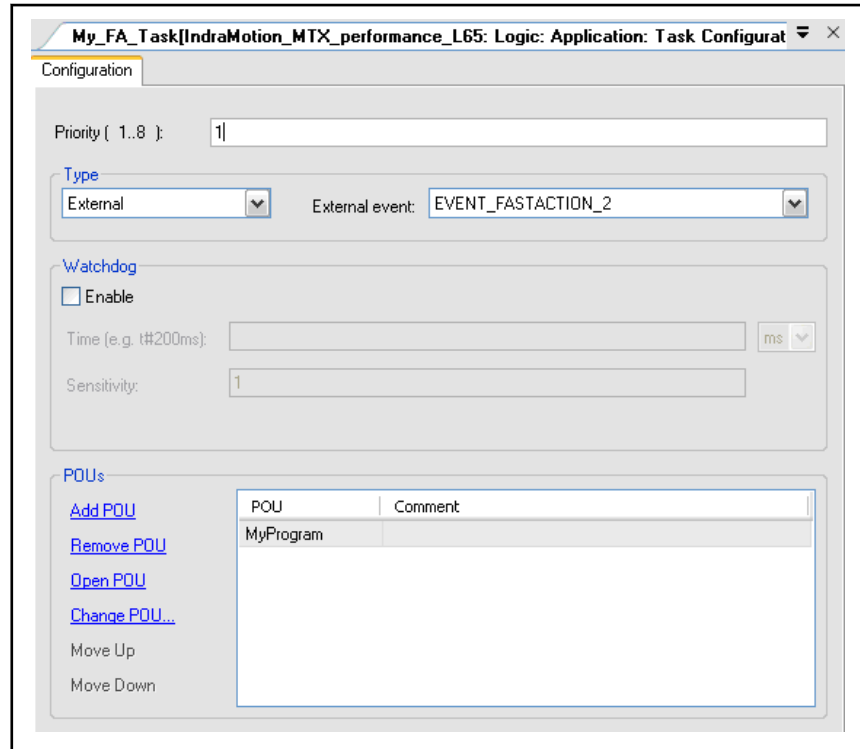


Fig. 14-14: Dialog for task configuration

2. Enter the required settings in this dialog:

- Priority = 1
- Type = External
- External event = select EVENT_FASTACTION_i (i = 1, 2, 3 or 4)
- Watchdog: Do not tick Enable!

(Using the system date "MaxFARunTime", the NC already implements a watchdog timer to monitor the data exchange between the fast PLC task and the fast NC task.)

- Assign the required programs using AddPOU.

Click on "AddPOU" to open a dialog to select the desired program.

14.3.2 Creating fast PLC programs

Type definition for the FA input data that are to be transmitted cyclically

IndraWorks Engineering is open and the logic is not logged in

1. Right-click on the node "Application".
First select "Add" in the context menu, then select "Data types ...".
A dialog opens to enter the data type name.
2. Enter the desired data type name of the FA input data structure (e.g. "FAIn_t(STRUCT)") and select "Structure" as the type.
Close the dialog.

The corresponding data type was inserted.

3. Double-click on the respective node.

Enter the data to be transmitted cyclically and their data types into the required structure.



Start every type definition, which is to be used to transmit FA data in a fast PLC task, with the instruction "{attribute 'pack_mode' := '1'}".

Type definition for the FA output data that are to be transmitted cyclically

To define the types of the FA output data, please proceed as for the FA input data (see [chapter "Type definition for the FA input data that are to be transmitted cyclically" on page 135](#)).

Reading FA input data (MT_FaDataRead)

The function MT_FaDataRead is used to read the cyclically transmitted data (from the fast NC task to the fast PLC task). At the beginning of the fast PLC task, the input data has to be read using this function. If the data transfer for the respective cycle was completed without errors, program processing can start. If the data transfer was not successful, troubleshooting has to take place correspondingly.



- The structure for the input data has to be exactly the same size as the input data configured for the respective fast PLC task on the NC.
- This function block may only be called from a fast PLC task.
- The task has to be of the type "externally event-controlled" and the external event has to be "EVENT_FASTACTION_x", where x is index 1 to 4.

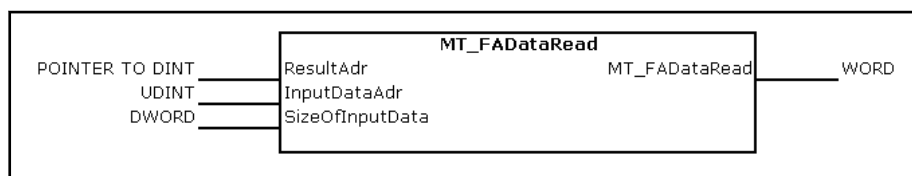


Fig. 14-15: MT_FaDataRead function

Function block parameters of the MT_FaDataRead function

I/O type	Name	Data type	Description
FUNCTION	MT_FaDataRead	WORD	The function value contains the result of the function. It is the result of a library function, not of a fast PLC task
VAR_INPUT	ResultAdr	POINTER TO DINT	Reserved for future extensions (Return value of the fast NC task). It is currently not used
	InputDataAdr	UDINT	Pointer to data transmitted from the fast NC task to the fast PLC task
	SizeOfInputData	DWORD	Total size of the data range in bytes

Tab. 14-4: I/O interface of the MT_FaDataWrite function

Status messages of the MT_FADaRead function

0x0000	Data transfer OK
0x000A	The function was not executed, since it was not called from a fast PLC task
0x000B	Configuration error: No input data is configured for the fast PLC task or the size transmitted in SizeOfOutputData is 0
0x000C	The size of the configured input data does not match the size transmitted in SizeOfInputData
0x000D	The target address in OutputDataAdr is a null pointer
0x7FFE	Internal error

Tab. 14-5: Parameters of the MT_FaDataWrite function

Writing FA output data (MT_FaDataWrite)

The function MT_FaDataWrite is used to write the cyclically transmitted data (from the fast PLC task to the fast NC task). At the end of the fast PLC task, the output data have to be transmitted to the fast NC task using this function. If the data transfer was not completed without errors for the respective cycle, troubleshooting has to take place correspondingly.



- The structure for the output data has to be exactly the same size as the output data configured for the respective fast PLC task on the NC.
- This function block may only be called from a fast PLC task.
- The task has to be of the type "externally event-controlled" and the external event has to be "EVENT_FASTACTION_x", where x is index 1 to 4.

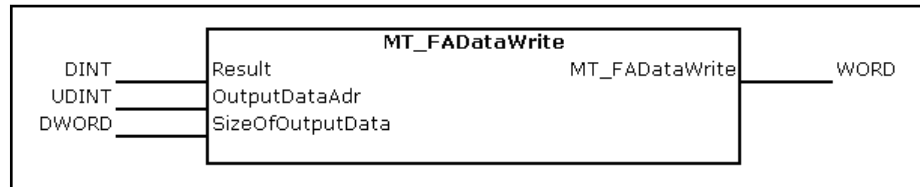


Fig. 14-16: MT_FADaWrite function

Function block parameters of the MT_FADaWrite function

I/O type	Name	Data type	Description
FUNCTION	MT_FADaWrite	WORD	The function value includes the result of the library function
VAR_INPUT	Result	DINT	Return value of the fast PLC task to the fast NC task
	OutputDataAdr	UDINT	Pointer to the data transmitted from the fast PLC task to the fast NC task
	SizeOfOutputData	DWORD	Total size of the data range in bytes

Tab. 14-6: Parameters of the MT_FaDataWrite function



If unexpected events/errors occur during fast PLC task execution, the fast PLC task can notify the NC of these events/errors using the Result input. The value applied to the Result input is transmitted to the system data element SD.SysFA.PLCtask[i].Status.UserError.

Status messages of the MT_FADaWrite function

0x0000	Data transfer OK
0x000A	The function was not executed, since it was not called from a fast PLC task
0x000B	Configuration error: No output data are configured for the fast PLC task or the size transmitted in SizeOfOutputData is 0
0x000C	The size of the configured output data does not match the size transmitted in SizeOfOutputData
0x000D	The target address in OutputDataAdr is a null pointer
0x7FFE	Internal error

Tab. 14-7: Parameters of the MT_FaDataWrite function

14.3.3 Configuring the NC FA functionality

Creating FA system data

A system data structure is used to configure the fast actions MTX fa functionality. For basic information on the system data, refer to the manual "MTX Functional Description".

Start IndraWorks Engineering, use the Project Explorer to go online and open the NC file system of IndraWorks Operation.

1. Open the "feprom" directory.
2. Copy the file "SDDeffa.xml_" to "usrfep" and remove the underscore.
The "SDDeffa.xml_" file is now available in "usrfep".
3. Open the "feprom/schemes" directory.
4. Copy the file "sdfa.xsd_" to "usrfep/schemas" and remove the underscore.



If it does not exist yet, create the "schemas" folder below "usrfep".

The "sdfa.xsd" file is now available in "usrfep/schemas".

5. Restart the NC.



- The system data creation requires RAM on the control kernel.
- By default, FA system data are created as volatile data, i.e. the desired configuration has to be written to the FA system data structure after every control startup.
- The FA system data can be read and written via the user interface, by the PLC program or a NC program.
- With the CPL command "SDSAVE", a configuration can be saved and restored using the command "SDLOAD". This way, a NC program can be used to easily save and restore a configuration.

Example: Backup the current FA configuration in a file "/usrfep/SDDatFA.xml" using the CPL command:

```
01 SDSAVE("/usrfep/SDDatFA.xml","SysFA")
```

Explicit loading of this configuration of this file:

```
01 SDLOAD("/usrfep/SDDatFA.xml","SysFA")
```

Note: The configuration is automatically enabled using this file during startup.

- The default assignment can be modified using the file SDDatFA.xml.

Recommendation: Memory attribute "volatile" and configuration via SDDatFA.xml.

- Currently, the system data structure SD.SysFA supports only two fast PLC tasks.

Introducing fast PLC tasks to the NC

The fast PLC tasks are assigned to the NC data transfer mechanisms by entering the PLC task name into the data element "SysFA.PLCTask[i].Control.IECTaskName".

Configuring the cyclic data transfer

Specify the data to be transferred cyclically below the substructure "SysFA.PLCTask[i].Config.CycData" using the data elements Cmd, Access, Index1, Index2, Index3, Index4. The overview table FA Data lists which data/functions can be entered.

Checking and applying the configuration

Before starting the cyclic transfer, check and confirm the data specified for cyclic data transfer by setting the bit "SysFA.PLCTask[i].Config.Validate".



Also check and confirm the configuration during control startup, unless it is saved permanently.

Starting fast PLC tasks

If the check is completed (SysFA.PLCTask[i].Status.CheckFin = 1) and if no errors are reported (SysFA.PLCTask[i].Status.NCError = 0) and if the data is ok (SysFA.PLCTask[i].Status.DataOk = 1), the cyclic data transfer and the cyclic start of the corresponding task can be initiated by setting the bit SysFA.PLCTask[i].Control.Enable = 1.

14.3.4 Impact of the PLC on the FA functionality



- When the IL download is started, cyclic communication is suspended.
 - At the end of the PLC download, the configuration is checked (similar to the system date "Validate").
 - Starting or stopping the PLC has no impact on the fast PLC tasks.
-

15 Real-time data

15.1 Real-time coordinates

15.1.1 Description

- Function** For interpolation real-time actions, MTX provides interpolated coordinates in the system data. Archiving of coordinates can be enabled or disabled in each channel. The coordinates are updated in the interpolation cycle of the control. Archiving of real-time coordinates is supported for the following coordinate systems:
- Workpiece coordinate system (WCS)
 - Basic workpiece coordinate system (BCS)
 - Machine coordinate system (MCS)
 - Axis coordinate system (ACS)
- Restrictions and special features**
- The coordinates can sporadically jump during correction or transformer switching.
 - If an application requires all coordinates of the respective cycle, this has to be ensured by interpolation-synchronous fast actions (see [chapter 14 "Fast actions MTX fa" on page 133](#)).
 - The coordinates are archived in the system data in ascending channel axis order. The channel axis order can change in case the axis is replaced!
 - Archiving the real-time coordinates generates extensive load on the system resources. Activate archiving only if required.
- Example** Depending on the workpiece coordinate X, the spindle speed is to be controlled dynamically during face grinding:

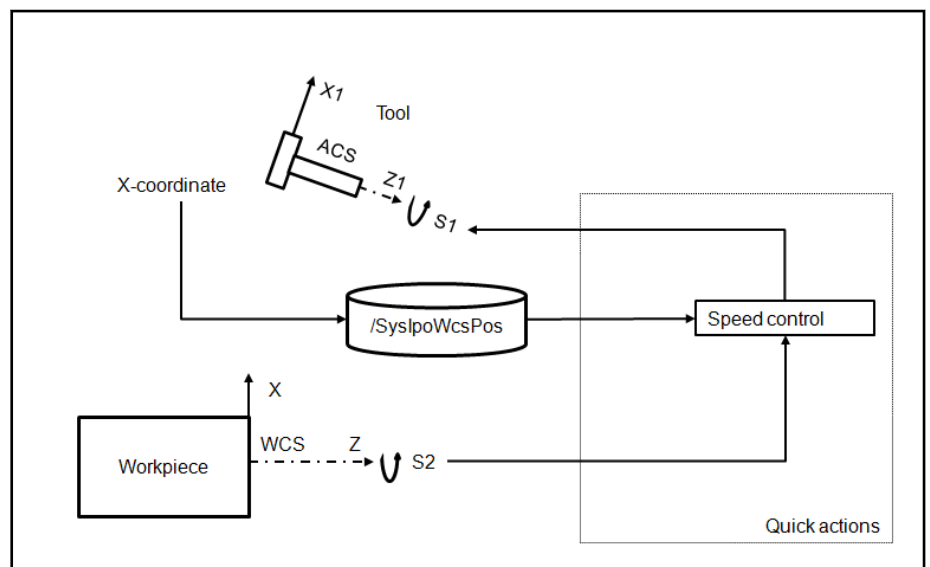


Fig. 15-1: Workpiece coordinate-dependent speed control

The workpiece coordinate X is filed with `/SyslpoWcsPos[Channel]/Active = 1` in the system data `SyslpoWcsPos[Channel]/Coord[i]` in each interpolation cycle after the activation.

The PLC program "Speed control" continuously calculates the command speed of the tool spindle (S_1) from the current workpiece speed (S_2) and the

distance from the tool to the workpiece turning center (SyslpoWcsPos[Channel]/Coord[i]).

System data overview

System element	date/	Dimension	Type	Unit	Description
SyslpoWcsPos		Channel	struct	-	Workpiece coordinates
SyslpoBcsPos		Channel	struct	-	Basic workpiece coordinates
SyslpoMcsPos		Channel	struct	-	Machine coordinates
SyslpoAcsPos		Channel	struct	-	Axis coordinates
	.Active	-	Int	-	0 - Disable archive 1 - Enable archive
	.Coord	1..8	Int	0.1 µm	Channel coordinates

Tab. 15-1: System data of the coordinate systems

Relevant system data

System date	Definition template	Scheme template
SyslpoWcsPos	/feprom/SDDefwcscrd.xml_	/feprom/schemas/sdrtpos.xsd_
SyslpoBcsPos	/feprom/SDDefbcscrd.xml_	
SyslpoMcsPos	/feprom/SDDefmcscrd.xml_	
SyslpoAcsPos	/feprom/SDDefacscrd.xml_	

Tab. 15-2: System data templates

15.1.2 Handling instruction: Applies workpiece coordinates in the interpolation cycle

Applying The definition and scheme of the system date SyslpoWcsPos has to be created in the root directory or in the usrfeep:

- Copy template /feprom/SDDefwcscrd.xml_ to /usrfeep/SDDefwcscrd.xml
- Copy template /feprom/schemas/sdrtpos.xsd_ to /usrfeep/schemas/sdrtpos.xsd
- Execute an NC restart to activate the system date

Activating /SyslpoWcsPos[Channel]/Active=1

Deactivating /SyslpoWcsPos[Channel]/Active=0

15.1.3 Handling instruction: Applies basic workpiece coordinates in the interpolation cycle

Applying The definition and scheme of the system date SyslpoBcsPos has to be created in the root directory or in the usrfeep:

- Copy template /feprom/SDDefbcscrd.xml_ to /usrfeep/SDDefbcscrd.xml
- Copy template /feprom/schemas/sdrtpos.xsd_ to /usrfeep/schemas/sdrtpos.xsd
- Execute an NC restart to activate the system date

Activate /SysIpoBcsPos[Channel]/Active=1
Deactivate /SysIpoBcsPos[Channel]/Active=0

15.1.4 Handling instruction: Applies machine coordinates in the interpolation cycle

Applying The definition and scheme of the system date SysIpoMcsPos has to be created in the root directory or in the usrfeep:

- Copy template /feprom/SDDefmcscrd.xml_ to /usrfeep/SDDefmcscrd.xml
- Copy template /feprom/schemas/sdrtpos.xsd_ to /usrfeep/schemas/sdrtpos.xsd
- Execute an NC restart to activate the system date

Activating /SysIpoMcsPos[Channel]/Active=1
Deactivating /SysIpoMcsPos[Channel]/Active=0

15.1.5 Handling instruction: Applies axis coordinates in the interpolation cycle

Applying The definition and scheme of the system date SysIpoAcsPos has to be created in the root directory or in the usrfeep:

- Copy template /feprom/SDDefacscrd.xml_ to /usrfeep/SDDefacscrd.xml
- Copy template /feprom/schemas/sdrtpos.xsd_ to /usrfeep/schemas/sdrtpos.xsd
- Execute an NC restart to activate the system date

Activating /SysIpoAcsPos[Channel]/Active=1
Deactivating /SysIpoAcsPos[Channel]/Active=0

15.2 Programmed path velocity

15.2.1 Description

Function For interpolation real-time actions, MTX provides path velocities for each channel in the system data. Archiving of velocity can be enabled or disabled in each channel. The path velocities are updated in the interpolation cycle of the control.

Restrictions and special features

- The current path feed is stored in the unit 0.1µm/interpolation cycle
- The programmed path feed is stored in the unit 0.1µm/interpolation cycle in G94 and G93.
The programmed F-value is stored for G95 - Feed/Rotation
- The real-time coordinate archive generates extensive load on the system resources. Activate archiving only if required

Example Starting from 10% of the programmed feed, a distance control is to be activated in a laser cutting application.

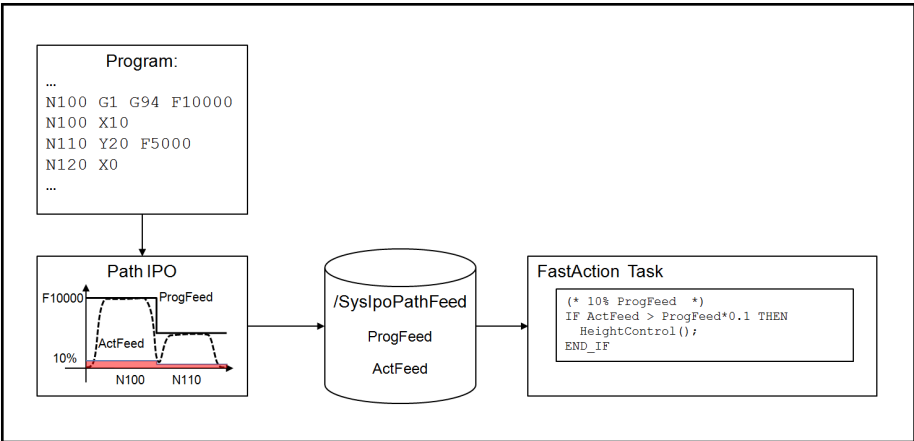


Fig. 15-2: Feed-dependent distance control

The reached feed is checked and in case of exceedance, a distance control is activated in an interpolation-synchronous FastAction task. No distance control is activated in case of range underflow.

System data overview

System element		date/	Dimension	Type	Unit	Description
SyslpoPathFeed			Channel	struct	-	Path velocities
	.Active		-	Int	-	0 - Disable archive 1 - Enable archive
	.ProgFeed		-	Float	0.1 µm/lpo	Programmed feed
	ActFeed		-	Float	0.1 µm/lpo	Current feed

Tab. 15-3: System data of the coordinate systems

Relevant system data

System date	Definition template	Scheme template
SyslpoPathFeed	/feprom/ SDDefpathfeed.xml_	/feprom/schemas/sdrtpos.xsd_

Tab. 15-4: System data template

15.2.2 Handling instruction: Applies path velocities in an interpolation cycle

Applying The definition and scheme of the system date SyslpoAcsPos has to be created in the root directory or in the usrfepeg:

- Copy template /feprom/SDDefpathfeed.xml_ to /usrfepeg/SDDefpathfeed.xml
- Copy template /feprom/schemas/sdrtpos.xsd_ to /usrfepeg/schemas/sdrtpos.xsd
- Execute an NC restart to activate the system date

Activating /SyslpoPathFeed[Channel]/Active=1

Deactivating /SyslpoPathFeed[Channel]/Active=0

16 Program-controlled events

16.1 Position-dependent high-speed output

The function "Position-dependent high-speed output" is described in detail in the programming manual.

The syntax of the function is "PosDepHSOut" or "PHS".

16.2 Programmable distance-dependent signal

16.2.1 Description

Function

The function **Advance Distance Signal (ADS)** facilitates triggering of a signal before reaching a programmed mark.

- Eight ADS signals are supported per channel.
- Each ADS signal is output via an interface signal (iCh_Customx or iCh_Cplx) or a system date
- The signal duration can be selected
- An interface signal can be assigned to several ADS signals

The ADS signal properties are specified by the /SysAdvDistSig/Sig[] system date.

After the function has been activated, a programmed mark is searched in the next 20 NC blocks. Once the mark has been found, the point on the path that has the programmed distance to the mark is marked. The selected interface signal is triggered, set or reset when passing the path point.

In the following example, a switching point of 3mm before the mark is assumed for mark 2.

```

N010 ADSENA(2)
N020 G90 G1 F1000 X100 Y100
N030 X110
N040 Y110
N050 X100
N060 Y80
N070 Y78 ADSLAB(2)
...
N230 ...
N240 X10 Y10
N250 X20 ADSLAB(2)
N260 X30
N270 ADSDIS(2)
...

```

- The search for mark ADSLAB(2) is enabled in block N010.
The mark ADSLAB(2) is found in block N070.
The switching point can be found 3 mm before the end of block N070. Thus, a task to trigger the signal is entered in block N060 at position Y81.
- The mark ADSLAB(2) is found again in block N250.
The switching point can be found 3 mm before the end of block N250. Thus, a task to trigger the signal is entered in block N250 at position X17.
- The search for the mark is enabled in block N270.

Restrictions and special features

Circumstances that can result in a delayed triggering of the signal:

- **Block pre-run**
Marks are not searched for during a block pre-run
After approaching the contour, the search for marks in the part program starts. If a mark is found whose switching point is before the approaching point, MTX triggers the signal directly at the approaching point.
- **Asynchronous subroutines**
After approaching the contour, the search for marks in the part program starts. If a mark is found whose switching point is before the approaching point, MTX triggers the signal directly at the approaching point.

Circumstances that can result in a premature triggering of the signal:

- **Rounding corners (RND, RNE)**
If the end of a block with a mark is changed by inserting a rounding, the mark is set to the start of the rounding.
- **Chamfering (CHL, CHS)**
If the end of a block with a mark is changed by inserting a chamfer, the mark is set to the start of the chamfer.

NC functions resulting in special behavior:

- In case of an active 2.5 tool radius correction "G41, G42", the path of the tool line of contact is taken into consideration
- In case of an active 3D tool radius correction "G141, G142", the path of the tool line of contact is taken into consideration
- In the operating mode "Program block", the function AdvDistSig can only take the active block into consideration
- Control reset und M30 disable the function

Relevant NC functions

Activate search for a mark

ADSENA (<Signal> {,<Signal> ...})	
with	
<Signal>	1 .. 8: Number of a signal

Set mark

ADSLAB (<Signal> {,<Signal> ...})	
with	
<Signal>	1 .. 8: The signal "DistOn" + "TimeOn" is set in front of the mark -1 .. -8: The signal DistOn + TimeOff is reset in front of the mark

Deactivates search for a mark

ADSDIS (<Signal> {,<Signal> ...})	
with	
<Signal>	1 .. 8: Number of a signal

ADSDIS ()	Deactivates the search for all marks

Reset switching signal

ADSCLR (<Signal> {,<Signal> ...})	
with	
<Signal>	1 .. 8: Number of a signal
ADSCLR ()	Resets all switching signals

Relevant IF signals

The used IF signal is selected via the system dates "/SysAdvDistSig/Sig[1..8]/Type" and "/SysAdvDistSig/Sig[1..8]/Index".

iCh_Custom1 .. 8	Client output 1 ... 8
iCh_Cpl01 .. 16	CPL customer output 1 ... 16

Tab. 16-1: Relevant IF signals

Relevant machine parameters

Max. number of blocks for other functions "MaxBlkMiscFunc"

The "ADSENA" function searches for the programmed marks "ADSLAB" in the following NC blocks. Limits the number of blocks in the block look-ahead.

Relevant system data

The behavior of the 8 signals is determined by "/SysAdvDistSig[*]/Sig[n]"

/SysAdvDistSig[*]/Sig[n]/DistOn	
	Distance to Set mark ADSLAB(<Signal>) The sum of DistOn and the distance (TimeDistOn) calculated from TimeOn results in the effective distance (EffDistOn). The signal sets EffDistOn to 1 using the programmed mark before reaching the end of the block.
/SysAdvDistSig[*]/Sig[n]/DistOff	
	Distance to Reset mark ADSLAB(-<Signal>) The sum of DistOff and the distance calculated from TimeOff results in the effective distance (EffDistOff). The signal EffDistOff is set to 0 using the mark before reaching the end of the block.
/SysAdvDistSig[*]/Sig[n]/TimeOn	
	Time interval to set mark ADSLAB(<Signal>) A distance (TimeDistOn) is calculated from the time TimeOn by using the last programmed feed(*). The sum of this value and DistOn returns the effective distance (EffDistOn). (*) Note possibly set bit in /SysAdvDistSigE[*]/OveForTime. The signal EffDistOn is set to 1 using the mark before reaching the end of the block.

Program-controlled events

/SysAdvDistSig[*]/Sig[n]/TimeOff	
	Time interval to reset mark ADSLAB(--<Signal>) A distance (TimeDistOff) is calculated from the time TimeOff by using the last programmed feed(*). The sum of this value and DistOff returns the effective distance (EffDistOn). (*) Note possibly set bit in /SysAdvDistSigE[*]/OveForTime. The signal EffDistOff is set to 0 using the mark before reaching the end of the block.
/SysAdvDistSig[*]/Sig[n]/DistCalc	
	Type of distance calculation 0: Coordinates of the active plane 1: Coordinates of the active space 2: All coordinates of the channel
/SysAdvDistSig[*]/Sig[n]/Duration	
	0.5 - 10000.0: Signal duration -1: The signal level is only switched on by ADSLAB()
/SysAdvDistSig[*]/Sig[n]/Type	
	Signal type selection 0: iCh_Customx 1: iCh_Cplx 10: iCh_Customx + The HS output of the PosDepHSOut function 11: iCh_Cplx + The HS output of the PosDepHSOut function
/SysAdvDistSig[*]/Sig[n]/Index	
	Index of the PLC signal 0 (not used) No IF signal is used 1 .. 8: Index for type = 0 and 10 (iCh_Custom1 .. iCh_Custom8) 1..16: Index for type = 1 and 11 (iCh_Cpl01 .. iCh_Cpl16)

Tab. 16-2: Relevant system data

The DistOn and DistOff unit is determined by "/SysAdvDistSig[*]/Unit"

/SysAdvDistSig[*]/Unit	
	Specifies the unit of the distance 0: mm 1: Inch

Tab. 16-3: Relevant system data

Active signals are additionally displayed in "/SysAdvDistSig[*]/SignalStatus"

/SysAdvDistSig[*]/SignalStatus	
	The current status of all signals (bit-coded) Signal 1 has the valence 1 Signal 2 has the valence 2 Signal 3 has the valence 4 ... Signal 8 has the valence 128 The valences of the single bit add up. Example: Signal 3 and 5 are active. Signal status has the value 20 (4 + 16)

Tab. 16-4: Relevant system data

Optional extensions (from 14V22 Patch6)

/SysAdvDistSigE[*]/OveForTime	
	Note feed potentiometer for TimeOn and TimeOff (bit-coded) The position of the feed potentiometer during block preparation is considered when converting from TimeOn and TimeOff into one distance.
/SysAdvDistSigE[*]/ReqLookAhead	
	Refer to the block preview for "Programmable distance-dependent signal" The block preview range to search for programmed marks is generally 20 programmed NC blocks. If this number of blocks is not sufficient in some cases, the block preview can be extended from 21 to 100 NC blocks using this parameter.

Tab. 16-5: Relevant system data

Example 1: The signals are set for a certain duration

In channel 1, a plane workpiece is machined.

Machining is executed in two steps. The machining operation requires a slightly earlier disconnection of the functional units A and B in the 2nd section.

In the first machining step, the signals 1 and 2 are used. In the second machining step, the signals 3 and 4 are used.

The signals 1 and 3 trigger the PLC to disconnect the functional unit A.

The signals 2 and 4 trigger the PLC to disconnect the functional unit B.

The signal configuration:

The signals 1 and 3 and the signals 2 and 4 trigger the same action. Thus, they are assigned to the same interface signal.

The signal duration has to be 40 ms and the distance is calculated in this plane.

SignalConfig

```

00 SD.SysAdvDistSig[1].Unit = 0                : REM mm
11 SD.SysAdvDistSig[1].Sig[1].DistOn = 1.0      : REM 1.0 mm
12 SD.SysAdvDistSig[1].Sig[1].DistOff = 0
13 SD.SysAdvDistSig[1].Sig[1].TimeOn = 0

```

Program-controlled events

```

14 SD.SysAdvDistSig[1].Sig[1].TimeOff = 0
15 SD.SysAdvDistSig[1].Sig[1].Duration = 40.0      : REM 40.0 msec
16 SD.SysAdvDistSig[1].Sig[1].DistCalc = 0         : REM active plane
17 SD.SysAdvDistSig[1].Sig[1].Type = 0             : REM iCh_Custom1
18 SD.SysAdvDistSig[1].Sig[1].Index = 1            : REM iCh_Custom1
21 SD.SysAdvDistSig[1].Sig[2].DistOn = 0.4
22 SD.SysAdvDistSig[1].Sig[1].DistOff = 0
23 SD.SysAdvDistSig[1].Sig[1].TimeOn = 0
24 SD.SysAdvDistSig[1].Sig[1].TimeOff = 0
25 SD.SysAdvDistSig[1].Sig[2].Duration = 40.0
26 SD.SysAdvDistSig[1].Sig[2].DistCalc = 0
27 SD.SysAdvDistSig[1].Sig[2].Type = 0             : REM iCh_Custom2
28 SD.SysAdvDistSig[1].Sig[2].Index = 2            : REM iCh_Custom2

31 SD.SysAdvDistSig[1].Sig[3].DistOn = 1.2
32 SD.SysAdvDistSig[1].Sig[1].DistOff = 0
33 SD.SysAdvDistSig[1].Sig[1].TimeOn = 0
34 SD.SysAdvDistSig[1].Sig[1].TimeOff = 0
35 SD.SysAdvDistSig[1].Sig[3].Duration = 40.0
36 SD.SysAdvDistSig[1].Sig[3].DistCalc = 0
37 SD.SysAdvDistSig[1].Sig[3].Type = 0             : REM iCh_Custom1
38 SD.SysAdvDistSig[1].Sig[3].Index = 1            : REM iCh_Custom1
41 SD.SysAdvDistSig[1].Sig[4].DistOn = 0.5
42 SD.SysAdvDistSig[1].Sig[1].DistOff = 0
43 SD.SysAdvDistSig[1].Sig[1].TimeOn = 0
44 SD.SysAdvDistSig[1].Sig[1].TimeOff = 0
45 SD.SysAdvDistSig[1].Sig[4].Duration = 40.0
46 SD.SysAdvDistSig[1].Sig[4].DistCalc = 0
47 SD.SysAdvDistSig[1].Sig[4].Type = 0             : REM iCh_Custom2
48 SD.SysAdvDistSig[1].Sig[4].Index = 2            : REM iCh_Custom2

```

PartProgram

```

; Stage 1
ADSENA(1,2)
G1 X... Y... F...
G2 X... Y...
G1 X... Y...
...
...
G1 X... Y...
ADSLAB(1,2) G1 X... Y... ; set iCh_Custom1 1.0 mm before end of block
                        ; set iCh_Custom2 0.4 mm before end of block
G1 X... Y...
...
ADSDIS(1,2)
...
; Stage 2
ADSENA(3,4)
G1 X... Y... F...
G2 X... Y...
G1 X... Y...
...
...
G1 X... Y...
ADSLAB(3,4) G1 X... Y... ; set iCh_Custom1 1.2 mm before end of block
                        ; set iCh_Custom2 0.5 mm before end of block
G1 X... Y...
...
ADSDIS(3,4)
...
M30

```

Example 2: The signal is set and deleted by the parts program

In channel 1, a sheet is machined with a laser.

The laser is activated 50 ms before reaching a point on the path and is deactivated 20 ms before reaching a 2nd point on the path.

An HS output of the MTX activates/deactivates the laser.

Additionally, the signal iCh_Custom1 is switched for the PLC.

SignalConfig

```

11 SD.SysAdvDistSig[1].Sig[1].DistOn = 0          : REM 0 mm
12 SD.SysAdvDistSig[1].Sig[1].DistOff = 0
13 SD.SysAdvDistSig[1].Sig[1].TimeOn = 50         : REM 50 msec

```

```

14 SD.SysAdvDistSig[1].Sig[1].TimeOff = 20      : REM 20 msec
15 SD.SysAdvDistSig[1].Sig[1].Duration = -1     : REM switching signal
16 SD.SysAdvDistSig[1].Sig[1].DistCalc = 0      : REM active plane
17 SD.SysAdvDistSig[1].Sig[1].Type = 10        : REM iCh_Custom1
; Type10 = iCh_Custom1 + the high speed output signal which is
; assigned to the function PosDepHSOut
18 SD.SysAdvDistSig[1].Sig[1].Index = 1        : REM iCh_Custom1

```

PartProgram

```

ADSENA(1)
G1 X... Y... F500
G2 X... Y...
G1 X... Y... ADSLAB(1) ;set signal 50msec before reaching the end of this
                        ;block. Distance will be calculated with F500
...
G1 X... Y... F300
G1 X... Y...
G1 X... Y... ADSLAB(-1) ;reset signal 20msec before end of block
                        ;this block. Distance will be calculated with F300
G1 X... Y... F500
G2 X... Y...
G1 X... Y... ADSLAB(1) ;set signal 50msec before reaching the end of this
                        ;block. Distance will be calculated with F500
...
G1 X... Y... F300
G1 X... Y...
G1 X... Y... ADSLAB(-1) ;reset signal 20msec before end of block
                        ;this block. Distance will be calculated with F300
...
M30 ;internal deactivation of ADS

```

16.2.2 Applying

Create system data

The signal properties are determined by the system date. This system date has to be created.

1. Copy the /feprom/schemas/sdadvdistsig.xsd_ file to /usrfep/schemas/sdadvdistsig.xsd.
2. Copy the /feprom/SDDefAdvDistSig.xml_ file to /usrfep/SDDefAdvDistSig.xml.
3. Restart the NC.

Adjust the machine parameters block look ahead

The "ADSENA" function searches for the programmed marks "ADSLAB" in the following NC blocks. Limits the number of blocks in the block look-ahead.

1. Increase parameter "Max. number of blocks for other functions NCO/LookAh/Ch[k]/MaxBlkMiscFunc" to 45 in the relevant channels.
If the block preview is set to more than 20 NC blocks when using / SysAdvDistSigE[*]/ReqLookAhead, the parameter "Max. number of blocks for other functions NCO/LookAh/Ch[k]/MaxBlkMiscFunc" has to be set to at least (2x ReqLookAhead + 5) NC blocks.
2. If required, adjust the total number of blocks and the percentage of the total number.
 1. Increase parameter "Total number of blocks for block preparation and interpolation NCO/LookAh/Ch[k]/NofBlkPrep" in the relevant channels if required.
 2. Adjust the parameter for percentage distribution of the total number of blocks "NCO/LookAh/Ch[1]/PercSplit/..." in the relevant channels.

Adjust the machine parameters digital inputs/outputs (DigIO)

Program-controlled events

If a fast MTX output is to be used to signalize an event, a digital MTX output of the "Position-dependent HighSpeed output (PosDepHSOut, PHS)" function has to be assigned.

1. Configure the digital outputs of the MTX according to the "MTX Machine Parameters", chapter "Digital inputs/outputs (DigIO)".
2. Assign a digital output via parameter "Assignment of the digital outputs[1..8] PLC/DigIO/FuncAllocOut/DigIOAllocOut[1..8]" to the function "Position-dependent HS output for channel x".

Adjust the channel initialization program

When using switching signals (...Sig[<Signal>].Duration = -1) it is recommended to enter the "ADSCLR()" command in the channel initialization program. Thus, the interface signals are reset after the home position and M30.

Adjust the PLC program

Link the selected signals to the corresponding function.

16.2.3 Activating/deactivating

The function can be used at any time within a part program.

ADSENA() activates search for a mark. ADSLAB() represents a searched mark. ADSDIS() deactivates search for a mark.

SignalConfig

```

001 SD.SysAdvDistSig.Unit = 0 : REM mm
002 REM Configure Signal 3
003 SD.SysAdvDistSig.Sig[3].DistOn = 1.2      : REM 1.2 mm
004 SD.SysAdvDistSig[1].Sig[1].DistOff = 0    : REM distance only
005 SD.SysAdvDistSig[1].Sig[1].TimeOn = 0
006 SD.SysAdvDistSig[1].Sig[1].TimeOff = 0
007 SD.SysAdvDistSig.Sig[3].Duration = 40.0   : REM 40 msec
008 SD.SysAdvDistSig.Sig[3].DistCalc = 0      : REM active plane
009 SD.SysAdvDistSig.Sig[3].Type = 0          : REM iCh_Custom2
010 SD.SysAdvDistSig.Sig[3].Index = 2         : REM iCh_Custom2

```

PartProg

```

...
N20 P SignalConfig
N21 ADSENA(3)           ; Activate look-ahead for label ADSLAB(3)
N22 G1 X... Y... F...
N23 G2 X... Y...
N24 G1 X... Y...
...
N50 G1 X... Y... ADSLAB(3) ; Set iCh_Custom2 1.2 mm before end of block
N51 G1 X... Y...
...
N80 ADSDIS(3)          ; Deactivate look-ahead for label ADSLAB(3)
...

```

17 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)

Glossary

ACS

Axis coordinate system

BCS

Basic coordinate system

MCS

Machine coordinate system

PCS

Program coordinate system

TCP

Tool center point

TCS

Tool coordinate system

WCS

Workpiece coordinate system

Index

A

Abbreviations.....	7
About this documentation.....	1
Information representation.....	6
Validity of the documentation.....	1
Angle error compensation.....	39
Apply macros.....	126
Apply, XML-based macros.....	130
Area monitoring with system data.....	111
Changing the operating data.....	114
Comparison with the NC command Area.....	118
Creating monitoring areas in the system data.....	111
Deactivating all channel areas.....	114
Example.....	111
General area properties.....	112
Geometric area properties.....	113
Handling instruction example.....	118
Introduction.....	111
Invalid area data.....	116
Optional system data for monitoring areas.....	117
Synchronizing the block preparation with the interpolator.....	115
Time of area violation detection.....	116
Asynchronous axes (auxiliary axes).....	15
AT1CoordNo.....	43
Automatic parameter download	
Activate.....	81
Apply.....	79
Deactivate.....	81
Description.....	79
Auxiliary axes.....	15
Axis reset	
Activate.....	16

B

Backlash compensation	
Activate.....	49
Apply.....	48
Deactivate.....	49
Description.....	48
basic_ty.xsd.....	87

C

CCOMP.....	30
Chan.....	43
Channel axes.....	15
Channel control.....	57
Circle compensation at the quadrant transition	
Description.....	50
Compensate mechanical inaccuracies.....	25
Backlash compensation.....	48
Circle compensation at quadrant transition....	50
Cross compensation (CCOMP).....	30

Lead screw error compensation (LSEC).....	25
Overview.....	25
Straightness and angle error compensation.....	39
Switch compensation tables (LSEC and CCOMP).....	36
Compensating mechanical inaccuracies	
Volumetric compensation (VCP).....	54
Complaints.....	7
Create macros.....	126
Criticism.....	7
Cross compensation (CCOMP)	
Content of the cnc03r02.....	34
Description.....	30
Customer Feedback.....	7

D

dbg command.....	82
Definition files	
Description.....	88
Structure.....	88
Delayed NC error reaction for non-fatal drive errors	
Activate.....	13
Apply.....	13
Deactivate.....	13
Description.....	11
Deviations	
Geometric.....	30
Documentation	
Change record.....	1
Download Sercos files in phase 3 only if required	
Description.....	83
Drive parameterization.....	77
Automatic parameter download.....	79
Download Sercos files in phase 3 only if required.....	83
Log transmitted Sercos parameters.....	82
Sercos initialization.....	77
Sercos multiplex channel.....	83
Drives (axes, spindle)	
Asynchronous axes (auxiliary axes).....	15
Delayed NC error reaction in case of non-fatal drive errors.....	11
Sercos gateway.....	14
Synchronous axes (channel axes).....	15
Drives (axes, spindles).....	11

E

Execution tools - Macro programming.....	125
--	-----

F

Feedback.....	7
Functions for drive parameterization.....	77
Download Sercos files in phase 3 only if required.....	83

Index

Drive parameterization.....	79	Special characters.....	125
Log transmitted Sercos parameters.....	82	XML.....	125
Sercos Initialization.....	77	Macro programming description	
Sercos multiplex channel.....	83	Function.....	121
Functions for Sercos handling		Macro type.....	124
Overview.....	77	CPL macro.....	125
Functions on drive parameterization		DIN macro.....	125
Function for drive parameterization.....	77	Macros.....	121
G		Manage schema files	
Geometric deviations.....	30	Description.....	101
H		Monitoring areas with system data.....	111
Handling instruction: Circle compensation at		N	
quadrant transition		NC macros.....	126
Description.....	52	Nest macros.....	124
Helpdesk.....	181	O	
Hotline.....	181	Online correction with handwheel.....	17, 22
I		Apply.....	20
Information representation		Deactivate.....	22
Names and abbreviations.....	7	Online correction with handwheel (in work-	
Safety instructions.....	6	piece coordinates)	
Symbols used.....	7	Description.....	17
Initialization files.....	92	Operating functions	
Intended use		Online correction with handwheel.....	17
Areas of application.....	9	Operating functions (manual operation).....	17
Introduction.....	9	Operating ranges with system data.....	111
Use cases.....	9	Override NC functions.....	123
L		P	
Lead screw error compensation (LSEC)		Parameterizable macros.....	122
Description.....	25	Path motion.....	71
Log transmitted Sercos parameters		PTP traversing motion.....	72
Activate.....	82	TCS travel motion.....	71
Deactivate.....	82	Position-dependent high-speed output.....	173
Description.....	82	Processing the backward and forward pro-	
LSEC.....	25	gram.....	59
M		Program-controlled events.....	173
Macro definition file		Position-dependent high-speed output.....	173
Structure.....	126	Programmable distance-dependent sig-	
Macro definition files.....	126	nal.....	173
Description.....	126	Programmable distance-dependent signal.....	173
Names and back up of definition files.....	127	Activating.....	180
Macro programming.....	121	Applying.....	179
Apply macros.....	126	Description.....	173
Create macros.....	126	Protected ranges with system data.....	111
Description.....	121	PTP.....	73
Example on XML-based macros.....	127	PTP traversing motion.....	72
Execution tools.....	125	Description.....	72
Macro definition files.....	126	R	
Macro types.....	124	Real-time coordinates.....	169
Nest macros.....	124	Axis coordinates.....	171
Override NC functions.....	123	Basic workpiece coordinates.....	170
Parameterizable macros.....	122	Machine coordinates.....	171
Reserved characters.....	125	Workpiece coordinates.....	170
		Real-time data.....	169

Real-time coordinates.....	169	Apply.....	15
Real-time path velocities.....	171, 172	Deactivate.....	15
Programmed path velocity.....	171	Description.....	14
RectStart / RectSpac.....	43	Sercos initialization	
Restrictions area monitoring with system data		Apply.....	78
Restrictions.....	118	Description.....	77
Retrace.....	57	Sercos multiplex channel	
Retrace on contour.....	57	Description.....	83
Return to machining.....	60	Parameterize.....	84
		Transfer list elements.....	84
S		Service hotline.....	181
SaciCorr / SaciBal.....	43	Special characters in the replacement text or	
Safety/monitoring.....	65	parameters.....	125
Sag.....	30	startup time.....	83
saveph3.scs.....	83	Straightness and angle error compensation.....	39
schema files		Activate.....	47
facets.....	104	Apply.....	45
Schema files.....	101	Control reset.....	45
Apply.....	109	Deactivate.....	47
Apply schemas.....	102	Description.....	39
Create new schema file.....	109	gct file - Structure.....	42
Declare attributes.....	107	gct files.....	42
Declare elements.....	106	Measure angle error.....	41
Define anonymous types.....	108	Measure straightness.....	41
Define complex types.....	105	Measuring method.....	41
Define simple types.....	104	Switch off behavior.....	45
General information.....	101	Switch on behavior.....	45
Manage schema files.....	101	Straightness compensation.....	39
Memory management for schemas.....	102	Structure file types (XML schema files)	
MTX kernel: Data transfer.....	110	Name and storage location of a structure	
PC / schema editor: Creating new sche-		file type.....	92
ma.....	109	Structure type files (XML schema files).....	91
PC / schema editor: Save schema files.....	110	Name and storage location.....	92
Regular expressions.....	109	Structure.....	91
Schema directories.....	101	Structure types.....	87
Schema structure.....	103	Characteristics.....	87
Schema syntax.....	102	Suggestions.....	7
Simple predefined types.....	103	Support.....	181
Types.....	101	Switch compensation tables (LSEC and CCOMP)	
SD definition file		Activate.....	39
Name and storage location.....	91	Apply.....	38
Name and storage location of initializa-		Deactivate.....	39
tion file.....	93	Description.....	36
SD initialization file.....	93	Extend table format.....	36
Name and storage location.....	93	Switch table.....	36
SD Initialization file		Switching from normal to retrace mode.....	59
Structure.....	92	Synchronous axes (channel axes).....	15
SDDat*.xml.....	92	System data.....	85
SDDef*.xml.....	88	Area monitoring with system data.....	111
Sercos.....	77	Definition files.....	88
Log ID numbers.....	82	Execution tools.....	86
Parameter download.....	79	Re-initialization of selected SD elements	
Parameter download in phase 3 if required...	83	after NC restart.....	93
Sercos initialization.....	77	SD initialization file.....	92
Sercos multiplex channel.....	83	SD.ColpDeact.....	117
Sercos gateway.....	14	SD.ColpLimit.....	117
Activate.....	15	SD.ColpQuad.....	111
		SD.ColpValid.....	111

Index

Structure type files (XML schema files).....	91
Structure types.....	87
Structure types, basic type.....	87
Structure types, MTX type.....	87
Structure types, self-defined type.....	87
System data (SD)	
Access.....	86
Save.....	86
System Data (SD).....	85
T	
Target groups.....	1
TCM.....	71
TCS travel motion.....	71
Description.....	71
Linear orientation.....	71
No orientation.....	71
Tensor orientation.....	71
Vector orientation.....	71
U	
Unintended use.....	10
Consequences, non-liability.....	9
use command.....	80
Use macros.....	121
V	
VCP.....	54
Volumetric compensation (VCP)	
Description.....	54
X	
XML - macro programming.....	125
XML schema files.....	91
XML-based definition files.....	127
XML-based macros.....	127
Access to SD.SysCyc.SL as R11.....	128
Apply.....	130
Changing the permitted axis acceleration at G0/G1.....	127
Handling instruction: Example.....	130
Hiding the underlying SD structure when checking a number.....	129
Moving to a safe position with G0_Safe.....	128
Shortened programming for the substructure of SD.SysG81.Rvs by means of RVS.....	129
Specify.....	127

Notes

Bosch Rexroth AG

P.O. Box 13 57

97803 Lohr a.Main, Germany

Bgm.-Dr.-Nebel-Str. 2

97816 Lohr a.Main, Germany

Phone +49 9352 18 0

Fax +49 9352 18 8400

www.boschrexroth.com/electrics



R911342624