rexroth
A Bosch Company

MTX 15VRS

Functional Description - Extension

Reference Book
R911393316

Edition 05



Title MTX 15VRS
Functional Description - Extension

Type of Documentation Reference Book

Document Typecode DOK-MTX***-NC*F*EX*V15-RE05-EN-P

Internal File Reference RS-7fcaab0b87a3d485c0a802862461ad27-5-en-US-3

Change record Edition 05, 2021-09
Refer to tab. 1-1 "Change record" on page 1

Copyright © Bosch Rexroth AG 2021
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, GeKo (TaDo/PiaSt)

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 ... 4
1.3.3 Commissioning ... 4
1.3.4 Operation ... 6
1.3.5 OEM Engineering ... 6
1.3.6 AddOns ... 7
1.4 Information representation ... 7
1.4.1 Using safety instructions ... 7
1.4.2 Names and abbreviations ... 9
1.5 Customer feedback ... 9

2 Important instructions on use ... 11
2.1 Intended use ... 11
2.1.1 Introduction ... 11
2.1.2 Areas of use and application ... 11
2.2 Unintended use ... 12

3 Drives (axes, spindles) ... 13
3.1 Linear Modulo Axis ... 13
3.1.1 Description ... 13
3.1.2 Handling instruction: Applies linear axis as modulo axis ... 14
3.2 Hirth Axis ... 15
3.2.1 Description ... 15
3.2.2 Handling instruction: Applies a Hirth axis ... 16

4 Operating functions (manual mode) ... 19
4.1 Jogging in workpiece coordinates ... 19
4.1.1 Description ... 19
4.1.2 Handling instruction: Jogging in workpiece coordinates ... 20
4.2 Handwheel in workpiece coordinates ... 23
4.2.1 General information ... 23
4.2.2 Handling instruction: Handwheel in workpiece coordinates ... 27

5 Channel control ... 31
5.1 Leaving and returning to contour ... 31
5.1.1 Description ... 31
5.1.2 Returning to contour options ... 34
5.1.3 Handling instruction: Leaving and Returning To Contour ... 37
5.2 Block pre-run ... 40

Page

5.2.1 General information about the block pre-run ... 40
5.2.2 Description ... 42
5.2.3 Block pre-run approach ... 43
5.2.4 Configuring the block pre-run ... 47
5.2.5 Switching modal states in the adjustment program or Action Block ... 58
5.2.6 Block pre-run applications ... 59
5.2.7 Spindles ... 62
5.2.8 System data ... 63
5.2.9 Interface signals ... 65
5.2.10 Look ahead ... 65
5.2.11 General information ... 66
5.2.12 Examples ... 68
5.3 Block switching via high-speed signal ... 74
5.3.1 Description ... 74
5.3.2 Handling instruction: Block Switching via High-Speed Signal with Cancellation of the Distance to Go ... 77
5.3.3 Handling instruction: Block Switching via High-Speed Signal with On-the-Fly Block Switching ... 79
5.4 Continuing Program with Block Pre-Run after Program Abort ... 80
5.4.1 Description ... 80
5.4.2 Activating ... 81
5.4.3 Deactivating ... 81
5.4.4 Handling instructions ... 81

6 Dynamics and Velocity Control ... 83
6.1 Scaling Rotary Axis Velocities ... 83
6.1.1 Configuring Scaling Factors for Rotary Axis Velocities ... 83
6.1.2 Programming Scaling Factors for Rotary Axis Velocities ... 86
6.2 Path Lengths and Curve Parameter Interpolation ... 88
6.2.1 Description ... 88

7 Safety/monitoring ... 91
7.1 Torque reduction ... 91
7.1.1 Description ... 91
7.1.2 Handling instruction: Activating/deactivating torque reduction ... 92
7.2 Reduced rapid traverse ... 92
7.2.1 Description ... 92
7.2.2 Handling instruction: Reduced rapid traverse ... 93
7.3 Test feed, test rapid traverse ... 94
7.3.1 Description ... 94
7.3.2 Handling instruction: Test feed (DryRun) ... 96
7.3.3 Handling instruction: Test rapid traverse (DryRun) ... 97
7.4 One-dimensional axis collision monitoring ... 98
7.4.1 Description ... 98
7.4.2 Handling instruction: One-dimensional axis collision monitoring ... 100
7.5 Axis position monitoring ... 103

Page

7.5.1 Description ... 103
7.5.2 Handling instruction: Axis position monitoring ... 104
7.6 Distance to end point ... 105
7.6.1 Description ... 105
7.6.2 Handling instruction: Distance To End Point ... 105
7.7 Retract vector as system date ... 106
7.7.1 Description ... 106
7.7.2 Handling instruction: Retract Vector as System Date ... 109

8 Accuracy ... 111
8.1 Accuracy programming ... 111
8.1.1 Description ... 111
8.1.2 Handling instruction: Accuracy programming ... 113

9 Path motion ... 115
9.1 Spline ... 115
9.1.1 Overview ... 115
9.1.2 Spline with coefficient programming ... 119
9.1.3 Spline with data point programming and tangential transitions (C1-continuous cubic splines) ... 122
9.1.4 Cubic hermite splines ... 128
9.1.5 Spline with data point programming and curvature-continuous transitions (C2-continuous cubic splines) ... 130
9.1.6 Spline with checkpoint programming (B-spline) ... 135
9.1.7 Approximate B-splines with tolerance specification ... 141
9.1.8 Spline programming ... 143
9.1.9 Handling instruction: Applying spline ... 148
9.1.10 Handling instruction: Activating spline ... 149
9.1.11 Handling instruction: Deactivating spline ... 149
9.2 Additional spline functions ... 149
9.2.1 Automatic detection of corners and lines ... 149
9.3 Automatic linear circular spline conversion ... 152
9.3.1 Description ... 152
9.3.2 Applying ... 154
9.3.3 Activating ... 154
9.3.4 Deactivating ... 154

10 Tool orientation und movement of orientation ... 157
10.1 General information ... 157
10.1.1 Description ... 157
10.2 Orientation definitions ... 157
10.2.1 Tool coordinate system TCS ... 157
10.2.2 Euler Angles ... 158
10.2.3 Orientation vector ... 162
10.2.4 Orientation tensor ... 164
10.3 Orientation motion ... 166

Page

10.3.1 Description ... 166
10.3.2 Vector orientation motion ... 166
10.3.3 Linear orientation motion ... 174
10.3.4 Tensor orientation motion ... 176
10.4 Tangential tool guidance ... 180
10.4.1 Description ... 180

11 Axis transformation ... 183
11.1 Introduction ... 183
11.1.1 Explanations ... 183
11.1.2 Axis transformation principle ... 183
11.1.3 Definitions ... 184
11.1.4 Points of action ... 186
11.1.5 Configuring axis transformations ... 187
11.1.6 Activating and deactivating axis transformations ... 188
11.1.7 Displaying active transformation data ... 190
11.1.8 Overview on axis transformations(in a Table) ... 192
11.2 Bipod axis transformations ... 196
11.2.1 Description ... 196
11.2.2 Handling Instruction: 3/4-Axis Transformation Bipod ... 198
11.3 Scara-axis transformation ... 205
11.3.1 Description ... 205
11.4 Handling Instruction: 3-Axis/4-Axis Transformation SCARA ... 208
11.5 Non-orthogonal 2- and 3-axis transformation ... 210
11.5.1 Variant 1 ... 210
11.5.2 Variant 2 ... 212
11.5.3 Variant 3 ... 213
11.5.4 Handling Instruction: Non-Orthogonal 2/3-Axis Transformation ... 214
11.6 4-axis transformation - Type 1: SCARA on linear axis ... 219
11.6.1 Description ... 219
11.6.2 Handling instruction ... 221
11.7 2-dimensional delta kinematics with additional rotary axis in the point (RRR) ... 227
11.7.1 Description ... 227
11.7.2 Handling instruction ... 239
11.8 4-axis transformation - Type 2: LLLR ... 242
11.8.1 Description ... 242
11.8.2 Handling instruction ... 244
11.9 4-axis transformation - Type 3: RLLL ... 248
11.9.1 Description ... 248
11.9.2 Handling instruction ... 253
11.10 5-axis transformation - Type 1: LLLR ... 256
11.10.1 General information on variants 1-3 ... 256
11.10.2 Variant 1 (with vector orientation) ... 257
11.10.3 Variant 2 (with linear orientation) ... 258
11.10.4 Variant 3 (with linear motion of the rotary axes) ... 258
11.10.5 Common characteristics of the variants 1-3 ... 259

Page

11.10.6 Handling instruction: 5-axis transformation LLLRR ... 263
11.10.7 Variant 4: 5-axis transformation LLLAB ... 267
11.11 5-axis transformation - Type 2: RLLLR (with rotary table) ... 273
11.11.1 Description ... 273
11.11.2 Handling Instruction: 5-Axis Transformation RLLLR ... 280
11.12 5-axis transformation - Type 3: RRLLL(with swivel/rotary table) ... 283
11.12.1 Description ... 283
11.12.2 Handling instruction: 5-Axis Transformation RRLLL ... 289
11.13 5-axis transformation LLLCB - Type 4: Cardanic head with vector orientation motion ... 293
11.13.1 Description ... 293
11.13.2 Handling instruction: 5-axius transformation cardanic LLLRR ... 299
11.14 5-axis transformation LLLCB - Type 4.1: Cardanic head with linear orientation motion ... 303
11.14.1 Description ... 303
11.14.2 Handling instruction: Cardanic head with linear orientation motion ... 305
11.15 5-axis transformation LLLCA - Type 4/4.1: Cardanic head with vector or linear orientation motion ... 305
11.15.1 Description ... 305
11.16 5-axis transformation - Type 5: Less-equipped RRLLL (with swivel/rotary table) ... 308
11.16.1 Description ... 308
11.16.2 Applying ... 311
11.16.3 Activating ... 313
11.17 5-axis transformation - Type 6: Cardanic RLLLR ... 317
11.17.1 Description ... 317
11.17.2 Handling Instruction: 5-Axis Transformation Cardanic RLLLR ... 323
11.18 5-axis transformation - Type 7: Tripod transformation with wobble plate ... 326
11.18.1 Description ... 326
11.18.2 Handling instruction: Tripod transformation with wobble plate ... 329
11.19 5-axis transformation - Type 8: Tripod transformation with wobble plate and additional c-axis for the tangential tool guidance ... 332
11.19.1 Description ... 332
11.19.2 Handling instruction: Tripod transformation with wobble plate and additional c-axis for a tangential tool guidance ... 333
11.20 3-axis transformation LLR ... 337
11.20.1 Description ... 337
11.20.2 Applying ... 340
11.20.3 Activation ... 343
11.21 6-axis transformation ... 343
11.21.1 Description ... 343
11.21.2 Variant 1 (with tensor orientation) ... 344
11.21.3 Variant 2 (with linear motion of the rotary axes) ... 345
11.21.4 Common characteristics of the variants 1, 2 ... 346
11.21.5 Handling instruction: 6-axis transformation LLLRRR ... 354
11.22 3-axis cylinder jacket transformation ... 359
11.22.1 Description ... 359
11.22.2 Variant 1 (unwinding) ... 359
11.22.3 Variant 2 (plane projection) ... 361
11.22.4 Common characteristics of the variants 1, 2 ... 362

Page

11.22.5 Handling instruction: 3-Axis Cylinder Jacket Transformation ... 364
11.23 4-Axis Cylinder Jacket Transformation ... 367
11.23.1 Description ... 367
11.23.2 Handling instruction: 4-axis cylinder jacket transformation ... 372
11.24 5-axis cylinder jacket transformation ... 375
11.24.1 Description ... 375
11.24.2 Handling instruction: 5-axis cylinder jacket transformation ... 380
11.25 Face transformation ... 383
11.25.1 Description ... 383
11.25.2 Variant 1 (2-axis transformation) ... 384
11.25.3 Variant 2 (3-axis transformation) ... 384
11.25.4 Common characteristics of the variants 1, 2 ... 384
11.25.5 Handling Instruction: 2/3-Axis Transformation Face ... 390
11.26 3-axis transformation LLR - Cartesian ... 393
11.26.1 Description ... 393
11.26.2 Handling instruction ... 397
11.27 Polar kinematics ... 401
11.27.1 Description ... 401
11.27.2 Handling instruction: Polar kinematics ... 404
11.28 Axis kinematic calibration ... 407
11.28.1 Description ... 407
11.28.2 Measuring methods ... 408
11.28.3 Handling instruction ... 425

12 Technologies ... 427
12.1 Tapping ... 427
12.1.1 Description ... 427
12.1.2 Handling instruction: Applying tapping without compensating chuck ... 432
12.2 Thread cutting ... 435
12.2.1 Description ... 435
12.2.2 Relevant system data ... 440
12.2.3 Applying ... 440
12.2.4 Activating ... 440
12.2.5 Deactivating ... 441
12.3 Circular rounding of corners ... 441
12.3.1 Description ... 441
12.3.2 Handling instruction: Applying and Activating/Deactivating Rounding of Corners Specifying the Deviation ... 442
12.3.3 Handling instruction: Applying and Activating/Deactivating Rounding of Corners Specifying the Radius ... 444
12.4 Rounding corners with splines ... 445
12.4.1 Description ... 445
12.4.2 Handling instruction: Rounding corners with splines ... 449
12.5 Punching and nibbling ... 450
12.5.1 Description ... 450
12.5.2 Handling instruction: Activate punching and nibbling ... 453

Page

12.6 Tangential tool orientation ... 455
12.6.1 Description ... 455
12.6.2 Activation ... 456
12.6.3 Deactivating ... 457
12.7 Laser machining ... 457
12.7.1 Path velocity-dependent laser power control ... 457
12.7.2 Handling instruction: Path Velocity-Dependent Laser Power Control ... 460

13 Axis configuration in the channel ... 465
13.1 Axis transfer ... 465
13.1.1 Description ... 465
13.1.2 Handling instruction: Axis transfer ... 471

14 Couplings ... 485
14.1 Gantry axes ... 485
14.1.1 Description ... 485
14.1.2 Handling instruction: Applying gantry axis group ... 498
14.1.3 Handling instruction: Activating Gantry Group ... 510
14.1.4 Handling instruction: Establishing reference at absolute measuring system ... 513
14.1.5 Handling Instruction: Applying Gantry Axis Group E-Stop ... 515
14.2 Selective additive coordinate coupling ... 516
14.2.1 Description ... 516
14.2.2 Handling instruction: Selective additive coordinate coupling ... 518
14.3 Selective additive coordinate coupling with tables ... 520
14.3.1 Description ... 520
14.3.2 Activation ... 524
14.3.3 Deactivation ... 524
14.4 Coupling tables ... 524
14.4.1 Description ... 524
14.5 System axis coupling ... 528
14.5.1 Description ... 528
14.5.2 Applying the system axis coupling ... 532
14.5.3 Activation/deactivation ... 560
14.5.4 Synchronization ... 563
14.5.5 System axis coupling with spindles ... 564
14.5.6 Exception Handling ... 567
14.5.7 Application cycles ... 576
14.5.8 System data overview ... 580
14.6 Channel axis coupling ... 588
14.6.1 Description ... 588
14.6.2 Applying ... 591
14.6.3 Activating ... 591
14.6.4 Deactivating ... 591

Page

15 Measuring functions ... 593
15.1 Traveling against touch probe (G75) ... 593
15.1.1 Description ... 593
15.1.2 Commissioning ... 595
15.1.3 Handling instruction: Measuring with the touch probe functions G75.1 and G75.2 ... 600
15.2 Flying measurement (FME) ... 603
15.2.1 Description ... 603
15.2.2 Commissioning ... 605
15.2.3 Handling instruction: Flying Measurement ... 608
15.3 Measuring on a contour (MOC) ... 610
15.3.1 Description ... 610
15.3.2 Applying ... 615
15.3.3 Activation ... 615
15.3.4 Deactivation ... 616
15.3.5 Handling instruction: Measuring on a contour ... 616
15.3.6 MOC with "Quick Stop via Probe Input" ... 618
15.4 Digitizing (recording measured values) ... 619
15.4.1 Description ... 619
15.4.2 Handling instruction: Digitizing (recording measured values) ... 628
15.5 The Log File ... 632
15.5.1 Description ... 632
15.6 Axis distance control for digitizing ... 635
15.6.1 Description ... 635
15.6.2 Monitoring deviations ... 641
15.6.3 Handling instruction: Axis Distance Control for Digitizing ... 643
15.6.4 Handling instruction: Adjust the Axis Distance Control for Digitizing Parameter via CPL ... 646
15.7 Measurement at fixed stop ... 648
15.7.1 Description ... 648
15.7.2 Handling instruction: Measurement at fixed stop ... 648
15.8 Measuring in space ... 649
15.8.1 Description ... 649
15.8.2 Commissioning ... 652
15.8.3 Troubleshooting ... 655

16 Synchronizing functions ... 657
16.1 Introduction ... 657
16.2 Synchronizing with NC preparation ... 659
16.2.1 NC Preparation and NC Execution (WAIT) ... 659
16.2.2 NC Preparation and Interface/PLC (WAIT(BITIF(...))) ... 660
16.2.3 Interface/PLC and NC Preparation (BITIF, PLC, SD) ... 661
16.2.4 Interface (OPC) and NC Preparation (SD) ... 665
16.3 Synchronizing with NC execution ... 667
16.3.1 NC Execution and NC preparation ... 667
16.3.2 NC execution between channels (WEV/WREV/SEV, WPV/SPV) ... 668
16.3.3 NC Execution and Interface/PLC (WAITA/WAITO) ... 671
16.3.4 NC execution and positions (*STOPA/*STOPO) ... 673

Page

16.3.5 PLC and NC Execution (Auxiliary Function SSD)..... 675
16.3.6 Interface (OPC) and NC Execution (SSD/SSDE/SSDQ)..... 676

17 Service and support..... 683

Glossary..... 685

Index..... 687

1 About this documentation

Editions of this documentation

Edition	Release date	Note
01	2019-03	First edition
02	2020-11	Changes in the chapter "Channel control" → "Block pre-run": "Configuring the block pre-run" supplemented Chapter "Path motion" → "Spline" → "Programming a spline" supplemented by discontinuation of the spline types 4201, 4202, 4203, 4204 and 4205 Chapter "Axis transformation" supplemented Chapter "Couplings" supplemented Chapter "Axis transformation" supplemented Additions in chapter "Measuring functions"
03	2021-03	Changes and additions in chapter "Measuring functions" Changes in chapter "Couplings"
04	2021-04	Changes and additions in chapter "Measuring functions"
05	2021-09	New: chapter 11.14 "5-axis transformation LLLCB - Type 4.1: Cardanic head with linear orientation motion" on page 303 (PBI 322340) Changes and additions in "Synchronous run monitoring of the slave axis:" on page 558 (PBI 312413)

Tab. 1-1: Change record

1.1 Validity of the documentation

Purpose

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

Overview on target groups and product phases

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

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

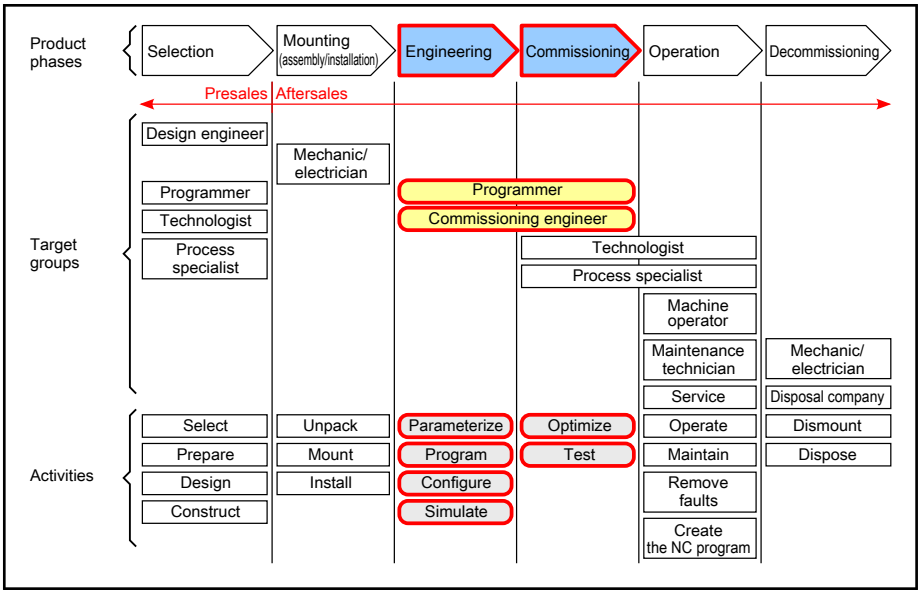


Fig. 1-1: Assigning this documentation to the target groups, product phases and target group activities

1.2 General information

This manual provides support 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 15VRS System Description DOK-MTX***-SYS*DES*V15-PRRS-EN-P, R911397742 This documentation describes the MTX control. It includes the designs, technical data, interfaces as well as the configuration of the control components.
MTX 15VRS SafeLogic System Overview DOK-MTX***-SL**SYS*V15-PRRS-EN-P, R911398637 This documentation describes the use of the safety control SafeLogic in the MTX.
IndraControl XM42 Controls DOK-CONTRL-XM4X*CTRL**-ITRS-EN-P, R911345566 This documentation describes the IndraControl XM42 controls.

RS Corresponding edition

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

1.3.2 Configuration

Documentation titles with type codes and part numbers

MTX 15VRS Machine Parameters DOK-MTX***-MA*PAR**V15-RERS-EN-P, R911400176 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 15VRS PLC Interface DOK-MTX***-PLC*INT*V15-PRRS-EN-P, R911400172 This documentation describes the interface signals and the program function blocks for the integrated PLC.
MTX 15VRS Functional Description - Basics DOK-MTX***-NC*F*BA*V15-RERS-EN-P, R911401401 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 15VRS Functional Description - Extension DOK-MTX***-NC*F*EX*V15-RERS-EN-P, R911393316 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 15VRS Functional Description - Special Functions DOK-MTX***-NC*F*SP*V15-RERS-EN-P, R911393309 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 wit 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.

RS Corresponding edition

Tab. 1-3: MTX documentation overview - Configuring

1.3.3 Commissioning

Documentation titles with type codes and part numbers

MTX 15VRS Commissioning DOK-MTX***-STARTUP*V15-CORS-EN-P, R911393281 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 15VRS Basic Libraries, IndraLogic 2G DOK-IL*2G*-BASLIB**V15-LIRS-EN-P, R911398633 This documentation describes the system-comprehensive PLC libraries.

IndraWorks 15VRS Field Buses DOK-IWORKS-FB******V15-APRS-EN-P, R911393284 This documentation describes the field bus and local periphery connections supported by the 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 15VRS Software Installation DOK-IWORKS-SOFTINS*V15-CORS-EN-P, R911393450 This documentation describes the IndraWorks installation.
IndraWorks 15VRS Engineering DOK-IWORKS-ENGINEE*V15-APRS-EN-P, R911393303 This documentation describes the use of IndraWorks in which the Rexroth Engineering tools are integrated. It includes instructions on how to work with IndraWorks and how to operate the oscilloscope function.
IndraWorks 15VRS PLC Programming System IndraLogic 2G DOK-IWORKS-IL2GPRO*V15-APRS-EN-P, R911396137 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 15VRS HMI DOK-MLC***-HMI*****V15-APRS-EN-P, R911399270 This documentation describes the functions, configuration and operation of the user interfaces IndraWorks HMI Engineering and IndraWorks HMI Operation.
MTX 15VRS Shape Cutting Technology DOK-MTX***-TECHCUT*V15-CORS-EN-P, R911399147 This documentation describes the shape cutting technology used by the machines to cut materials using an NC-controlled tool head.
MTX-integrated CAD/CAM System for Shape Cutting DOK-MTX***-CAD*TECHCUT-CO01-EN-P, R911393320 This documentation describes the mode of operation and the area of application of the CAD/CAM nesting software Lantek Expert Inside.

RS Corresponding edition

Tab. 1-4: MTX documentation overview - Commissioning

1.3.4 Operation

Documentation titles with type codes and part numbers

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 15VRS Programming Manual
DOK-MTX***-NC**PRO*V15-RERS-EN-P, R911393318
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 15VRS Standard NC Operation
DOK-MTX***-NC*OP***V15-APRS-EN-P, R911393314
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 15VRS Multitouch
DOK-MTX***-MULTI***V15-APRS-EN-P, R911393311
The Multitouch user interface of the NC control MTX is described in this documentation.
MTX 15VRS Standard NC Cycles
DOK-MTX***-NC*CYC**V15-PRRS-EN-P, R911394940
This documentation describes the application of the standard cycles of the different technologies for the MTX control.
MTX 15VRS NC Simulation
DOK-MTX***-NC*SIM**V15-APRS-EN-P, R911393273
This documentation describes the NC simulation for the MTX control.
MTX 15VRS Measuring Functions
DOK-MTX***-MES*FUN*V15-APRS-EN-P, R91194938
This documentation describes the measuring cycles of the MTX control.

RS 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 15VRS OPC Communication
DOK-MTX***-OPC*COM*V15-PRRS-EN-P, R911399272
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.

RS Corresponding edition

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 15VRS Efficiency Workbench MTX cta, MTX ega DOK-MTX***-EWB*****V15-APRS-EN-P, R911400178 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 / RS Corresponding edition

Tab. 1-7: MTX documentation overview - AddOns

1.4 Information representation

1.4.1 Using safety instructions

Structure of the safety instructions

The safety instructions are structured as follows:

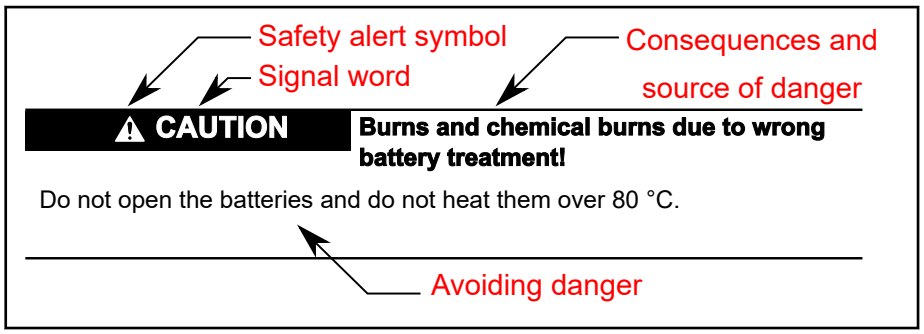


Fig. 1-2: Structure of the safety instructions

Explaining signal words and safety alert symbol

The safety instructions in this documentation contain specific signal words (danger, warning, caution, notice) and, if necessary, a safety alert symbol (according to ANSI Z535.6-2006).

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

The safety alert symbol (triangular safety reflector with exclamation marks), preceding the signal words Danger, Warning, Caution indicates hazards for persons.

⚠ DANGER

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

⚠ WARNING

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

Symbols used

Pointers are displayed as follows:



This is a note.

Tips are displayed as follows:



This is a tip.

Explaining the signal alert symbol on the device



If this symbol is on your device, you have to observe the documentation on the device. The respective documentation informs on the type of hazard as well as the steps required to avoid this hazard.

1.4.2 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
MP	Machine Parameters

Tab. 1-8: Terms and abbreviations

1.5 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 user 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:

- Anyone 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 its 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 documentations 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 previously specified 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 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 Linear Modulo Axis

3.1.1 Description

Function

Applies a linear axis as endless axis. When the linear axis reaches the modulo value, its command value is automatically set to "0". This prevents the computational overflow of axis values and allows a programmable reset to zero of the program coordinate system to the current axis position by LinModZp (LMZ).

Thus, the axis can theoretically be traversed endlessly in one direction. The function is recommended for applications of conveyor belt type, such as conveying a wire of a drum or winding of springs.

Velocity specification for asynchronous linear endless axes

For asynchronous endless axes, velocity programming is possible in position control mode. This can especially be used for master axes in electronic gears (system axis coupling).

The velocity is specified with the S-attribute (<Axis>=S<Velocity>) in the unit modulo intervals/minute. If the velocity command is reached, the axis interface signal "iAx_ProgSpReach" is set. The velocity specification can be freely modified and become a positioning motion on-the-fly.

Restrictions

- The relevant drive has to be able to perform a modulo computation independently. On the drive side, the modulo value can typically be parameterized via S-0-0103.
- In a traversing block, only position values less than or equal to the modulo value can be programmed.
- Negative position values may only be programmed if they are smaller than the modulo value.
- The control reset returns the programming zero point to the axis zero point.
- Measuring (InitMeas(IME) / FlyMeas(FME)) together with linear modulo axes is only allowed for traversing motions in positive direction.
- The limit velocity for modulo axes cannot exceed half the modulo range (S-0-103) per interpolation cycle.

  Example:

  S--0-103 = 20 mm; Sercos cycle 4 ms corresponds to the limit velocity 150,000 mm/min or to an S-value of 7,500 MOD/MIN.

Relevant NC functions

LinModZp(...)	Setting modulo axis of zero
<Axis>=S(...)	Velocity programming in modulo intervals/min

Tab. 3-1: Relevant NC functions

Relevant Sercos parameters

S-0-1050.1.6	Configuration list AT Depending on the encoder used in each case, the Sercos parameter S-0-0051 (actual position value 1 of motor encoder) or S-0-0053 (actual position value 2 of external encoder) has to be entered respectively.
S-0-1050.0.6	Configuration list MDT The Sercos parameter S-0-0047 (position command value) has to be entered. Together with the IndraDrive drives, P-0-100 (position command extension) can be entered.
S-0-0032	Primary mode Has to be set to "position control" mode.
S-0-0033	Secondary mode 1 Has to be set to the "position control without following errors" mode.
S-x-0103	Modulo value Enter modulo value

Tab. 3-2: Relevant Sercos parameters

Relevant IF signals

qAx_SwLimOff	Hiding software limit switches
iAx_ProgSpReach	Specified velocity of the endless axis reached.

Tab. 3-3: Relevant IF signals

Relevant machine parameters (MP)

1003 00004	Axis motion type

Tab. 3-4: Relevant machine parameters

3.1.2 Handling instruction: Applies linear axis as modulo axis

Applies linear axis as endless axis. When the linear axis reaches the modulo value, its command value is automatically set to "0". This prevents a numerical overflow.

NC: Creating linear axis

Before enabling the modulo computation, create a linear axis.

		Instruction
Instruction:		Creating linear axis

IW Engineering/configuration: Editing parameters

- **SysDrName**
  "Physical axis name" (1003 00001): Enter the system axis name.
- **EnablDr**
  "Enable drive" (1001 00001): Enable relevant drive with the value "Yes".
- **EnablModCalc**
  "Enable modulo calculation" (1003 00004): Set to "Yes".

- **ModePosLog**

  "Positioning logic" (1003 00005): Select desired positioning logic.

- **EnablChgPosLog**

  "Positioning logic can be switched via NC prog" (1003 00050) determines, whether the positioning logic in the NC program can be switched using PosMode(..).

☞ The relevant drive has to be able to perform a modulo computation independently. On the drive side, the modulo value can typically be parameterized via the Sercos parameter **S-0-0103** "modulo value".

In a traversing block, only position values between -modulo value and +modulo value may be programmed.

Measuring **(InitMeas(...)** "Initialization of the touch probe logic"/ **FlyMeas(...)** "Flying measurement" together with linear modulo axes is only allowed for traversing motions in positive direction.

As long as **G91** is active (relative dimension programming), the modulo computation is suppressed NC-internally for synchronous modulo axes in automatic manual data input modes except for the output command value and its NC-internal display.

Adapt unit if necessary

	Documentation	Instruction
Instruction:		Editing machine parameters
Documentation:	MTX Machine Parameters	Axis Parameters

IW Operation/NC programming: Setting modulo axis of zero

LinModZp(...) allows a programmable zero setting of the program coordinate system to the current axis position

	Documentation	Instruction
Instruction:		Editing PLC signals
Documentation:	MTX Machine Parameters	Linear axis signals

3.2 Hirth Axis

3.2.1 Description

Function

Applies a rotary or endless axis as Hirth axis. Hirth axes only approach positions on a specified grid (Hirth grid; ⇒ MP 1003 00055).

Thus, this function is suitable for the axes of tool changers or axes with toothing.

Optionally, the programming can be set in degrees or in places (⇒ MP 1003 00056).

For the **place programming**, enter the target position as integer place number without decimal positions.

Place 1 always corresponds to the position 0 degree. All places are numbered depending on MP 1003 00055 and are displayed as place number in

the position display. Negative place numbers, place number 0, or place numbers greater than those specified in MP 1003 00055 can only be programmed when G91 is active (relative dimension programming).

If the Hirth axis is to be **programmed in degrees**, an automatic end point correction is possible (⇒ MP 1003 00057). Then, the NC corrects a programmed position to the next position of the Hirth grid. Without end point correction, the control issues an error message.

Restrictions

- A Hirth axis cannot be used as slave axis of a coupling group. It can be used as master axis.
- Handwheel mode not permitted.
- A synchronous Hirth axis cannot be part of the axes selected with the "inclined plane" function.
- Reference point, contour and zero point offsets of the Hirth axis have to be on the Hirth grid
- At "incremental jogging" (incremental step is active), after "Feed hold", "Feed inhibit", "E-Stop" or "Control reset", the Hirth axis can also stop between the valid positions.

Relevant IF signals

qAx_NextNotch	"Next notch position" (axis interface) The high signal starts the traversing motion if the "Jog" mode is active, but the Hirth axis is not jogged manually at the moment. The NC reports the current traversing direction via the axis interface iAx_TrvDirNeg "negative traversing direction".
iAx_NotchPos	"Axis on grid position" (axis interface) The signal for a non-referenced axis is always "0". For a referenced axis, it is only set when the axis is in the "in-pos window".

Tab. 3-5: Relevant IF signals

Relevant machine parameters (MP)

1003 00055	Places per circle for Hirth axis
1003 00056	Place programming for Hirth axis
1003 00057	End point correction for Hirth axes

Tab. 3-6: Relevant machine parameters

3.2.2 Handling instruction: Applies a Hirth axis

Applies a rotary or endless axis as Hirth axis. Hirth axes only approach positions located on a specified grid. Thus, this function is suitable for the axes of tool changers or axes with toothing.

NC: Apply rotary or endless axis

The axis already exists in the system as rotary or endless axis and - in respect of velocity, dynamics, and positioning logic if applicable - has been correctly parameterized according to the requirements.

		Instruction
Instruction:		Appling rotary and endless axis

IW Engineering/configuration: Editing parameters

- **ModeLocProg** "Type of place programming" (1003 00056):
  Defines the desired programming.
- **NofLoc** "Number of places" (1003 00055):
  Defines the necessary Hirth grid.
- **EnablAutoCorr** "Enable automatic place correction" (1003 00057):
  If necessary, the end point correction can be enabled.

	Documentation	Instruction
Instruction:		Editing machine parameters
Documentation:	MTX Machine Parameters	Place programming

- The necessary axis-based interface signals are analyzed or set in the PLC program.
- Adapt unit if necessary.

IW Engineering/IndraLogic: Relevant interface signals (PLC → NC)

- **qAx_NextNotch:**
  "Next grid position" (axis interface). The high signal starts the traversing motion if the "Jog" mode is active, but the Hirth axis is not jogged manually at the moment. The NC reports the current traversing direction via the axis interface **iAx_TrvDirNeg** "negative traversing direction".
- **qAx_JogPlus** "Manual +" / **qAx_JogMinus** "Manual -":
  Is effective for channel axes (synchronous axes) and auxiliary axes (asynchronous axes) in the operation modes "manual setup" and "setup approach reference point". In addition, the signals are used to jog synchronous axes manually when leaving the contour. The active part program is stopped with feed hold.

	Documentation	Instruction
Instruction:		Editing PLC signals
Documentation:	MTX PLC Interface	Hirth axis signals

IW Engineering/IndraLogic: Interface Signals for control (NC → PLC)

- **iAx_NotchPos:**
  "Axis on grid position" (axis interface). The signal for a non-referenced axis is always 0. For a referenced axis, it is only set when the axis is in the "inpos window".
- **iAx_TrvDirNeg:**
  "Negative traversing direction". Set means that the corresponding axis performs or has already performed a negative traversing direction.

	Documentation	Instruction
Instruction:		Editing PLC signals
Documentation:	MTX PLC Interface	Hirth axis signals

Troubleshooting: Known status messages

State: Hirth axis is not positioned on the Hirth grid...

...following "Feed hold" or "Feed inhibit":

1. Remove "Feed hold" or "Feed inhibit".

   ⇒ The previously programmed end position is immediately approached on the Hirth grid.

...following "E-Stop" or "System reset":

1. Set manual feed to "Rapid traverse", "Fast", "Medium" or "Slow".
2. Activate jog mode.
3. Briefly jog Hirth axis (via the axis interface signals qAx_JogPlus "Manual +" / qAx_JogMinus "Manual-").

   ⇒ Depending on the jog direction (+/-), the NC travels to the nearest position on the Hirth grid.

- or -

1. Activate jog mode
2. Set the axis interface signal "travel to next grid position".

   ⇒ The NC travels to the next possible position of the Hirth grid. The traversing direction is automatically defined by the current state of the axis interface signal iAx_TrvDirNeg "negative traversing direction".

...after "incremental jogging" (incremental step is active):

1. Set manual feed to "Rapid traverse", "Fast", "Medium" or "Slow".
2. Briefly jog Hirth axis (via the axis interface signal qAx_JogPlus "Manual +" or qAx_JogMinus "Manual-").

   ⇒ Depending on the jog direction (+/-), the NC travels to the nearest position on the Hirth grid.

   - or -

3. Set the axis interface signal "travel to next grid position".

   ⇒ The NC travels to the next possible position of the Hirth grid. The traversing direction is automatically defined by the current state of the axis interface signal iAx_TrvDirNeg "negative traversing direction".

	Documentation	
Documentation:	MTX Functional Description	Hirth axis

4 Operating functions (manual mode)

4.1 Jogging in workpiece coordinates

4.1.1 Description

Function

Use this function to jog the workpiece coordinates (WCS) and - if an orientable axis transformation is active - the z-direction of the tool (TCS). For this purpose, the operation mode **"Setup mode, jogging in workpiece coordinates"** is provided.

The coordinate to be jogged is selected via a special coordinates selection program (whose name is stored in MP 705001110). Alternatively, it can also be selected from the PLC via the NC block specification or via any part program. In all cases, the NC function "JogWCSSelect" is used for the selection.

☞ Only one coordinate can be moved at a time.

Before jogging, the "Setup mode, jogging in workpiece coordinates" operation mode must be selected (via channel interface signal qCh_OpModeSel_00 "Operating mode selection bit 0" to qCh_OpModeSel_03 "Operating mode selection bit 3").

Jog variants:

- **Continuous jogging**:

  At jogging, a positive or negative end position is specified for the selected coordinate. The motion stops at an axis limit switch or a workspace limit. The motion stops when releasing the jog key.

- **Incremental jogging**:

  Press the jog key to specify the selected increment.

☞ If the selected path increment cannot be traversed, the NC moves to the maximum possible end point!

Use the **initialization program** to create a special environment for jogging (axis transformation, placements, etc). The name of the program is entered into MP 70500 1100. This program is executed together with the first jog motion.

Relevant NC functions

JogWCSSelect	For information on the syntax, refer to the "Programming Manual"

Tab. 4-1: Relevant NC functions

Relevant IF signals

qCh_JogPlusWcs	WCS Manual+
qCh_JogMinusWcs	WCS Minus-
qCh_OpModeSel_00 to qCh_OpModeSel_03	Operating mode selection bit 0 to 3

Tab. 4-2: Relevant IF signals

Relevant machine parameters

1010 00002	Jog acceleration
7050 01000	Jog WCS: Manual feeds
7050 01010	Jog WCS: Feed and incremental step, system axis number
7050 01020	Jog WCS: Resolution of an increment
7050 01030	Jog WCS: Variable increment step
7050 01100	Jog WCS: Program name for default configuration
7050 01110	Jog WCS: Program name for coordinate selection

Tab. 4-3: Relevant machine parameters

4.1.2 Handling instruction: Jogging in workpiece coordinates

Applying

IW Operation/NC programming: Selecting the coordinates to be moved

- with **coordinate selection program:**

  A coordinate is determined using a coordinate selection program. The program name (coordinate selection program) is entered into the parameter "**CoordProgName** - Program name for the coordinate selection (7050 01110)". This program is executed each time the jog key is pressed.

  Enter the initialization program into the parameter **TrafoProgName** "Program name for the standard configuration (7050 01110)". This program is executed together with the first jog motion.

Initialization program "InitstringForJogWCS"

Program:

```
N10 Coord (@AXTRAFO)
```

Coordinate selection program "CoordSelectForJogWCS"

Program:

```
10 COORDNO% = PLC(<Typ>,<DB-Nummer>,<Adresse>,<Gr÷¯e>)
  N10 JogWCSSelect JWSCOORD[COORDNO%]
```

- with **any part program**

  - or -

- via the **PLC with NC block specification** using the PLC function block **MT_NcBlk**

  - or -

  MT_NcBlkExt.

☞
- **JogWCSSelect**, refer to "MTX Programming Manual".
- Store the created programs in the search path.

	Documentation	Instruction
Documentation:	MTX Programming Manual	Coordinate selection program

IW Engineering/configuration: Editing parameters

- **AccAxJog**

"Jog acceleration" (1010 00002)

- **WcsJogIf**
  "System axis number of the interface to jog in WCS" (7050 01010)
- **IncrWcsJog**
  "Scaling a jog increment" (7050 01020)
- **DistWcsJog**
  "Variable of jog distance" (7050 01030)
- **SlowVelWcs**
  "Jog velocity slow" (7050 01000 [1])
- **MedVelWcs**
  "Jog velocity medium" (7050 01000 [2])
- **FastVelWcs**
  "Jog velocity fast" (7050 01000 [3])
- **RapidVelWcs**
  "Jog velocity rapid traverse" (7050 01000 [4])
- **DefStepVelWcs**
  "Jog velocity defined steps" (7050 01000[5])
- **VarStepVelWcs**
  "Jog velocity variable steps" (7050 01000[6])
- **TrafoProgName**
  "Program name for default configuration"(7050 01100)
- **CoordProgName**
  "Program name for coordinate selection"(7050 01110)

☞ Store the created programs in the search path.

The resulting jog variants are:

Jog mode(1)	Length of the distance to go(2)	Effective velocity	Effective acceleration
Rapid traverse	Depending on the period of the set channel interface signal "Manual -" or "Manual +"	RapidVelWcs	AccAxJog (if assigned with "0", the value from FeedAxAcc applies)
Fast		FastVelWcs	
Medium		MedVelWcs	
Slow		SlowVelWcs	
x increments	per L/H edge to "Manual +" or "Manual -": DistWcsJog * IncrWcsJog	VarStepVelWcs	
1000 increments	per L/H edge to "Manual +" or "Manual -": 1000 * IncrWcsJog	DefStepVelWcs	
100 increments	per L/H edge to "Manual +" or "Manual -": 100 * IncrWcsJog		
10 increments	per L/H edge to "Manual +" or "Manual -": 10 * IncrWcsJog		
1 increments	per L/H edge to "Manual +" or "Manual -": 1 * IncrWcsJog		

(1) Selection by setting the axis interface signals from "qAx_ManFeed_00" to "qAx_ManFeed_03 "Manual feed from Bit0 to Bit3" by WcsJogIf "System axis number of the interface to jog in WCS".

(2) The functions G70/G71 and DIA/RAD are considered at incremental jogging. If a linear coordinate is moved and G70 becomes active, the NC multiplies the value entered in "IncrWcsJog" with the factor 2.54. If the "DIA" function is active for the selected coordinate, the NC divides the value entered into "IncrWcsJog" by 2.

Tab. 4-4: Overview on parameters "jogging in WCS"

	Documentation	Instruction
Instruction:		Editing machine parameters
Documentation:	MTX Machine Parameters	Jogging in WCS

Activating **IW Operation: Select operation mode**

Selection of the channel mode "Manual setup workpiece coordinates" via the interface signals **qCh_OpModeSel_00 .. qCh_OpModeSel_03**

Operation mode	Operation mode selection			
	Bit 3 (qCh_OpModeSel_03	Bit 2 (qCh_OpModeSel_02	Bit 1 (qCh_OpModeSel_01	Bit 0 (qCh_OpModeSel_00
Manual setup: workpiece coordinates	1	1	1	0

Example	Documentation	Instruction
Example:		Channel modes in WCS
Instruction:		Editing PLC Signals
Documentation:	MTX PLC Interface	Selecting operation mode

IW Operation/IndraLogic: Select velocity range or step width

- Select the "velocity level" or "step width" at the axis interface determined by **WcsJogIf** "System axis number of the interface to jog in WCS" by setting the signals from "qAx_ManFeed_00" to "qAx_ManFeed_03" "Manual feed Bit0 to Bit3".

	Documentation	Instruction
Instruction:		Editing PLC Signals
Documentation:	MTX PLC Interface	Velocity range/increment

IW Operation/IndraLogic: Start motion

Start the motion by setting **qCh_JogPlusWcs** "WCS Manual +" or **qCh_JogMinusWcs** "WCS Manual -".

	Documentation	Instruction
Instruction:		Editing PLC Signals
Documentation:	MTX PLC Interface	Start motion

Deactivating

IW Operation/IndraLogic: Abort motion

A continuous motion is stopped when the respective jog key is released while the channel interface signal **qChJogPlusWcs** "WCS Manual +" or **qCh_JogMinusWcs** "WCS Manual - " is reset.

	Documentation	Instruction
Instruction:		Editing PLC Signals
Documentation:	MTX PLC Interface	Abort motion

IW Operation/IndraLogic: Exit jogging

- Exiting the channel mode "Manual setup: workpiece coordinates" completes each jog motion.
- Channel reset and system reset abort an active jog motion.

	Documentation	Instruction
Instruction:		Editing PLC Signals
Documentation:	MTX PLC Interface	Exit jogging

4.2 Handwheel in workpiece coordinates

4.2.1 General information

Function

With the "Handwheel in workpiece coordinates" function, the handwheel moves a workpiece coordinate of a channel. A motion in the active plane can be optionally executed by setting the system date "SysHandwWCS". Straight line and circle are available as contour elements.

Example:

Handwheel in workpiece coordinates

If a corresponding axis transformation is active, a workpiece can be traversed via handwheel in TCS-z-direction by a combined motion of the linear spatial coordinates (x, y, z).

Coordinate selection in the WCS:

The coordinate or the contour element to be jogged is selected via a special coordinate selection program whose name is stored in CoordProgName (MP 705010110). This program is called at each handwheel selection.

The coordinate can also be selected by the PLC or via any part program.

The NC function "JogWCSSelect" and the system date "SysHandwWCS" is always used for the selection. The selection via "SysHandwWCS" has a higher priority than the selection by "JogWCSSelect".

☞ At each handwheel selection, the current preselection of the coordinate or the contour element is applied.

Handwheel selection

For the handwheel selection, the coordinate to be moved is subject to the following boundary conditions:

- The "setup mode, jogging in workpiece coordinates" mode (relevant operation mode for "handwheel in WCS") has to be selected.
- Only one workpiece coordinate can be moved at a time.
- The coordinate can only be moved in the determined position limits (maximum positive and negative end position).
- The state of the qCh_HandwPosMode interface signal is applied at the handwheel selection.
- If the maximum possible path velocity for the selected coordinate is exceeded by the handwheel specification, the response is based on the state of the interface signal qCh_HandwPosMode at the time of handwheel selection:
  - qCh_HandwPosMode not set:

    The velocity is limited and the summed distance is reduced accordingly.
  - qCh_HandwPosMode set:

    The velocity is limited.
- Active end position coupling (EPC) of the selected coordinate is considered.
- G70 (Inch) active: For linear coordinates, the handwheel increments are multiplied by the factor 2.54 (0.001 mm → 0.0001 inch).
- Active diameter programming (DIA) of the selected coordinate is considered.
- Handwheel motions are ignored in case of an active ...
  - feed inhibit or feed hold (for path motion)
  - Feed inhibit, drive off or safety off (for axis motion of an involved axis)

Use the initialization program to create a special environment for the handwheel mode (axis transformation, placements, etc). The name of the program is entered into MP 70500 1100. This program is executed in the first handwheel selection.

Restriction

- Only the coordinates enabled for online correction can be moved via handwheel.

Refer to the documentation "MTX Functional Description Special Functions", chapter "Online Correction with Handwheel (in workpiece coordinates)".

- The coordinates of the following axis transformations cannot be moved via the handwheel:

  Transformation type 3232201: Phi and theta

  Transformation type 3232202: Phi and theta

  Transformation type 3232203: Phi and theta

  Transformation type 3232211: Phi and theta

  Transformation type 3021002: x and y

  Transformation type 2011001: x and y

  Transformation type 2103011: x and y

Relevant NC functions

JogWCSSelect	Coordinate selection and other parameters For information on the syntax, refer to the "MTX Programming Manual").

Tab. 4-5: Relevant NC functions

Relevant IF signals

qCh_OpModeSel_00 to qCh_OpModeSel_03	"Operating mode selection bit 0 ... 3"
qCh_OpModePLC	"PLC operating mode"
qCh_HandwSelWcs00	"Handwheel selection, bit 0"
qCh_HandwSelWcs01	"Handwheel selection, bit 0"
qCh_HandwDirWcs	"Handwheel direction of rotation"
qChHandwPosMode	"Position handwheel"
iCh_OpMode_00 to iCh_OpMode_03	"Active operation mode bit 0 ... 3"
qAx_ManFeed_00 to qAx_ManFeed_03	"Manual feed bit 0 to 3" Axis number, see MP 7050 01010

Tab. 4-6: Relevant IF signals

Relevant machine parameters

1010 00002	Jog WCS: Jog acceleration of the coordinate relevant for the channel.
7050 01010	Jog WCS: Feed and incremental step; system axis number.
7050 01020	Jog WCS: Resolution of an increment.
7050 01030	Jog WCS: Variable increment step.
7050 01100	Jog WCS: Program name for the default configuration (initialization program).

7050 01110	Jog WCS: Program name for the coordinate selection.
9060 00001	Number of increments per revolution (handwheel).

Tab. 4-7: Relevant machine parameters

Relevant system data

Element name	Description	Type	Size in bytes	Write access
PathMode	Contour handwheel? 0: (No) 1: (Yes)	UnsignedByte_t	1	Everyone
ContourType	Type of contour? 0: Straight line 1: Arc	UnsignedByte_t	1	Everyone
Angle	Direction of the straight line Angle between the positive main axis direction and the straight line in degrees. For ContourType = 0.	Float_t	4	Everyone
MainCP	Position of the circle center in the main axis in mm. For ContourType = 1.	Float_t	4	Everyone
ScndCP	Position of the center point in the secondary axis in mm. For ContourType = 1.	Float_t	4	Everyone

Tab. 4-8: Structure elements "SysHandwWCS"

Setting the system date

SysHandwWCS[<ChannelNo.>].PathMode

determines the coordinates to be moved from the main and secondary axis of the active plane. The motion type results from the content of the system date

SysHandwWCS[<ChannelNo.>].ContourType.

ContourType = 0

results in a straight line. The positive direction of the straight line in the current plane is determined by the angle parameter SysHandwWCS[<ChannelNo.>].Angle .

ContourType = 1

results in an arc. The position of the center point of the arc is determined via the system data SysHandwWCS[<ChannelNo.>].MainCP and .ScndCP.

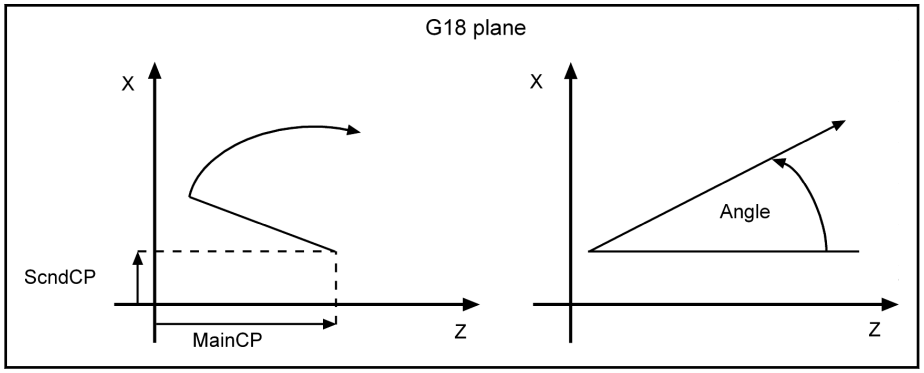


Fig. 4-1: G18 plane

4.2.2 Handling instruction: Handwheel in workpiece coordinates

Applying

IW Operation/NC programming: Selecting the coordinate to be moved

- with a **coordinate selection program**:

  A coordinate is determined using a coordinate selection program. The program name (coordinate selection program) is entered into the parameter "**CoordProgName** - Program name for the coordinate selection (7050 01110)". This program is executed in each handwheel selection.

  Enter the initialization program name into the parameter **TrafoProgName** "Program name for standard configuration" (7050 01110). This program is executed together with the first handwheel selection.

Initialization program "InitstringForJogWCS"

Program:

```
N10 Coord (@AXTRAFO)
```

Coordinate selection program "CoordSelectForJogWCS"

Program:

```
10 COORDNO% = PLC(<Typ>,<DB-Nummer>,<Adresse>,<Gr÷¯e>)
  N10 JogWCSSelect JWSCOORD[COORDNO%]
```

- with **any part program**

  - or -

- via the **PLC with NC block specification** using the PLC function block **MT_NcBlk**

  - or -

  MT_NcBlkExt.

☞
- **JogWCSSelect**, refer to "MTX Programming Manual".
- Store the created programs in the search path.

	Documentation	Instruction
Documentation:	MTX Programming Manual	Coordinate selection program

IW Engineering/configuration: Editing parameters

- **AccAxJog**

  "Jog acceleration" (1010 00002)

- **IncrPerRev**
  "Number of increments/revolutions" (9060 00001)
- **WcsJogIf**
  "System axis number of the interface to jog in WCS" (7050 01010)
- **IncrWcsJog**
  "Scaling a jog increment" (7050 01020)
- **DistWcsJog**
  "Variable jog distance" (7050 01030)
- **TrafoProgName**
  "Program name for default configuration" (7050 01100)
- **CoordProgName**
  "Program name for coordinate selection" (7050 01110)

☞ Store the created programs in the search path.

Incremental step(1)	Length of the distance to go(2)	Effective velocity
x increments	Per scale division at the handwheel: DistWcsJog * IncrWcsJog	depending on the rotation velocity of the handwheel.
1000 increments	Per scale division at the handwheel: 1000 * IncrWcsJog	
100 increments	Per scale division at the handwheel: 100 * IncrWcsJog	
10 increments	Per scale division at the handwheel: 10 * IncrWcsJog	
1 increments	Per scale division at the handwheel: 1 * IncrWcsJog	

(1) Selection by setting the axis interface signals from "qAx_ManFeed_00" to "qAx_ManFeed_03 "Manual feed from Bit0 to Bit3" by WcsJogIf "System axis number of the interface to jog in WCS".

(2) The functions G70/G71 and DIA/RAD are considered when calculating the length of the traversing distance. If a linear coordinate is moved and G70 becomes active, the NC multiplies the value entered in "IncrWcsJog" with the factor 2.54. If the "DIA" function is active for the selected coordinate, the NC divides the value entered into "IncrWcsJog" by 2.

Tab. 4-9: Overview on parameters in WCS

	Documentation	Instruction
Instruction:		Editing machine parameters
Documentation:	MTX Machine Parameters	Handwheel in WCS

Activating

IW Operation/IndraLogic: Selecting operation mode

Selection of the channel mode "Manual setup workpiece coordinates" via the interface signals **qCh_OpModeSel_00...qCh_OpModeSel_03**.

- **qCh_HandwPosMode**, Select between the handwheel modes "Position-exact" or "Motion-exact" via the interface signal.
- The channel interface signals **qCh_JogPlus** "Manual+" or **qCh_JogMinus** "Manual-" are not active.

Operation mode	Operation mode selection			
	Bit 3 (qCh_OpMode-Sel_03	Bit 2 (qCh_OpMode-Sel_02	Bit 1 (qCh_OpMode-Sel_01	Bit 0 (qCh_OpMode-Sel_00
Manual setup: workpiece coordinates	1	1	1	0

	Documentation	Instruction
Instruction:		Editing PLC Signals
Documentation:	MTX PLC Interface	Selecting handwheel

IW Engineering/IndraLogic: Selecting handwheel

Selection via axis interface signals **qCh_HandwSelWcs_00** "Handwheel selection bit 0" or **qCh_HandwSelWcs_01** "Handwheel selection bit 1".

- **qCh_HandwPosMode**, Select between the handwheel modes "Position-exact" or "Motion-exact" via the interface signal.
- The interface signals **qCh_JogPlusWcs** "Manual+" or **qCh_JogMinusWcs** "Manual-" are not active.

	Documentation	Instruction
Instruction:		Editing PLC Signals
Documentation:	MTX PLC Interface	Selecting handwheel

IW Engineering/IndraLogic: Specifying direction of rotation

Define how the control has to interpret the direction of rotation of the handwheel via the interface signal **qCh_HandwDirWcs** "Handwheel direction of rotation".

	Documentation	Instruction
Instruction:		Editing PLC Signals
Documentation:	MTX PLC Interface	Specifying direction of rotation

Handwheel: Triggering traversing motion

Triggering the traversing motion by rotating the handwheel.

Deactivating

Deactivating IW Operation/IndraLogic:

Reset the respective axis interface signal **qAx_HandwSelWcs_00** "Handwheel selection bit 0" or **qAx_HandwSelWcs_01** "Handwheel selection bit 1".

☞ After deselecting the "Jogging in workpiece coordinates" mode or after channel or system reset, the handwheel has to be reselected as the relation between coordinate and handwheel does not exist anymore.

	Documentation	Instruction
Instruction:		Editing PLC Signals
Documentation:	MTX PLC Interface	Deactivating handwheel

5 Channel control

5.1 Leaving and returning to contour

5.1.1 Description

Function

Use this function to leave and return to contour after pausing a part program or a block sequence in the operation mode "MDI block input" with "feed hold". This can be required, for example, after a tool breaks, to manually measuring a workpiece or tool or for visual inspection.

The following can be freely selected for the return to contour strategy:

- Approach mode

  (**How** the NC is to return to the contour: manual, automatic, single block)
- Approach point

  (**The point at which** the NC is to return to contour: point of interruption, start of block, end of block)
- Approach strategy

  Defines whether the selected point is approached linearly or tangentially in an arc from the so-called placement point.



Use the machine parameter OPF/ReturnPath/Ch[k]/ReturnMode (7060 00310) and MP OPF/ReturnPath/Ch[k]/ReturnPoint (7060 00320) to preset the approach mode and approach point.

For detailed information on the mode of actions of the individual options, refer to chapter 5.1.2 "Returning to contour options" on page 34.

If a part program (or a block sequence in the "MDI block input" mode) is paused with "feed hold", the "leaving and returning" to contour function can be used. Enable the "Leaving the contour" mode first.

Leaving the contour always internally triggers a delete distance to go and thus an axis standstill. Subsequently, all blocks, prepared before leaving the contour are discarded.

All discarded blocks have to be prepared again after returning to the contour.



If correction values (e.g. zero points or tool data) have been changed within the range of already prepared blocks, the changed values now apply to all blocks to be prepared again.

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 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).

From this point in time, the control "stores" all traversing motions triggered via the axis-related interface signals qAx_JogPlus "Manual+" or qAx_JogMinus "Manual-". At axis standstill, the control outputs the channel interface signal iCh_ReadyReEnter "Ready to return to contour".

☞ Use the machine parameter OPF/ReturnPath/Ch[k]/NofJogSteps (7060 00330) to configure the maximum number of the traversing motions to be stored.

Enable the "Return to contour" mode if no more traversing motions are to be stored. The NC then sets the channel interface signal iCh_RemoveFinish "Leaving completed". It is still possible to jog the axes in this mode.

The return to contour can be started via "NC start" depending on the selected approach strategy (seechapter 5.1.2 "Returning to contour options" on page 34). After the return to contour motion, it is switched back to the previously active operation mode and the machining is continued.

When returning to contour, the "placement point" is significant, since the distance between the placement point and the approach point (re-entry point on the contour) is used to calculate the required correction values.

The placement point corresponds to

- "Manual" approach mode of the current position
- "Automatic" or "single block" approach mode of the position at the end of the first recorded traversing motion.

Select a favorable placement point for the distance "placement point - approach point" to achieve the "smoothest" contour entry possible.

☞ In the "Leaving the contour" mode, machining can be continued at any time at the set approach point via "NC start" provided the input signal "Return to contour" (channel interface "qCh_RetCont") is not set simultaneously.

In this case, the control runs directly from the current position to the approach point, records any necessary correction values in the control and continues machining.

Select the best placement point to achieve the smoothest contour entry possible.

After NC start to machine in the placement point, an asynchronous subroutine (ASUP) is automatically called if logged into the system data. The name of the asynchronous subroutine is stored in the **SysReentry** system date in the **NameOfAsup** component.

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

10 SD.SysReentry.NameOfAsup="/usrfep/ReentryAsup.cnc"

At the end of the asynchronous subroutine, the start, interruption or end point of the abort block is approached according to the setting. The contour can be approached linearly or tangentially. Afterwards, normal machining is continued.

While the asynchronous subroutine is running, the "SD.SysReentry.AsupActive" system date is set to 1.

For the approach strategy "tangential in the arc", the contour is entered into an arc with a radius to be specified by a system date. The arc is in the active working plane (G17, G18, G19). With regard to this working plane, the contour is entered tangentially.

The previous motion from the placement point to the approach circle is along a straight line. Thus, with regard to the active working plane, there is also a tangential transition from this straight line to the arc.

Defines if a feed stop is enabled after a return to contour using the **FeedHold** component.

Restrictions

- When leaving the contour, the jog signals (qAx_JogPlus "Manual+" or qAx_JogMinus "Manual-") directly affect the respective machine axes. Neither an axis transformation nor a rotation of the program coordinate system (PCS) with regard to the basic coordinate system (BCS) is considered!
- Returning to contour is not always possible in several channels at the same time.
- Returning is only possible after the end of a possibly active CPL waiting state (e.g. WAIT). The control outputs a warning if a CPL waiting state is active at the time of interruption.
- G63 (tapping) must not be active.
- Any running spindles have to be stopped after an interruption either manually (via switching function) or via the PLC and restarted on time before the machining is restarted.
- Leaving and returning to contour cancels the retrace mode.
- When returning to contour, it is not possible to leave and return to the contour again until the approach point was reached. Switching to retrace mode is not possible either.

Relevant IF signals

at the **axis interface**	
qAx_JogPlus	Manual+
qAx_JogMinus	Manual-
at the **channel interface**	
qCh_NCStart	NC start
qCh_OpModeSel_00 to qCh_OpModeSel_03	Operation mode selection bit 0 to 3 Required operation mode: "Return to contour"
qCh_OpModePlc	Operation mode specified by PLC
qCh_RetCont	Return (to contour)
qCh_FeedHold	Feed hold
iCh_OpMode_00 to iCh_OpMode_03	Active operation mode bit 0 to 3 (Feedback for operation mode selection bit 0 to 3)
iCh_ReadyReEnter	Ready to return to contour
iCh_ReEnterAct	Re-entry active
iCh_RemoveFinish	Leaving completed

Tab. 5-1: Relevant IF signals

Relevant machine parameters (MP)

OPF/ReturnPath/ Ch[k]/ReturnMode (7060 00310)	Return to contour operation mode Presetting approach mode (manual, automatic, single block)
OPF/ReturnPath/ Ch[k]/ReturnPoint (7060 00320)	Approach point: Return to contour Presetting point of return to contour (start of block, interruption point or end of block)
OPF/ReturnPath/ Ch[k]/NofJogSteps (7060 00330)	Maximum record to return to contour in blocks. Specifies the maximum number of traversing motions that can be recorded after the "Leaving the contour" mode was activated.

Tab. 5-2: Relevant machine parameters

Relevant system data

Structure element	Description
FeedVal	Individual return to path velocity
OptFeedVal	Optional velocity if no feed was programmed
TangCircleRadius	Return to path radius
NameOfAsup	Adjustment program name of the return to path logic (ReentryAsup)
Approach	Return to contour
FeedMode	Return to path velocity selection
FeedHold	Activate feed stop after a return to contour
AsupActive	Adjustment program of the return to path logic (ReentryAsup) active

Tab. 5-3: Overview on structure elements "SysReentry"

5.1.2 Returning to contour options

Selecting approach point

Available options:

- Start of block
- Point of interruption
- End of block

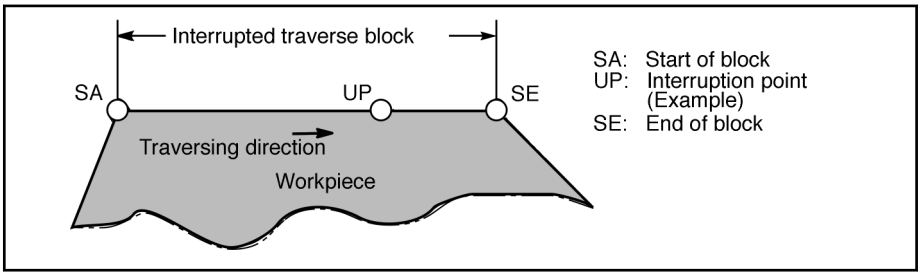


Fig. 5-1: Selecting approach point

NOTICE **Workpiece or machine can be damaged!**

Note that the contour has not yet been processed from the point of interruption when using the "end of block" option!

Selecting approach mode

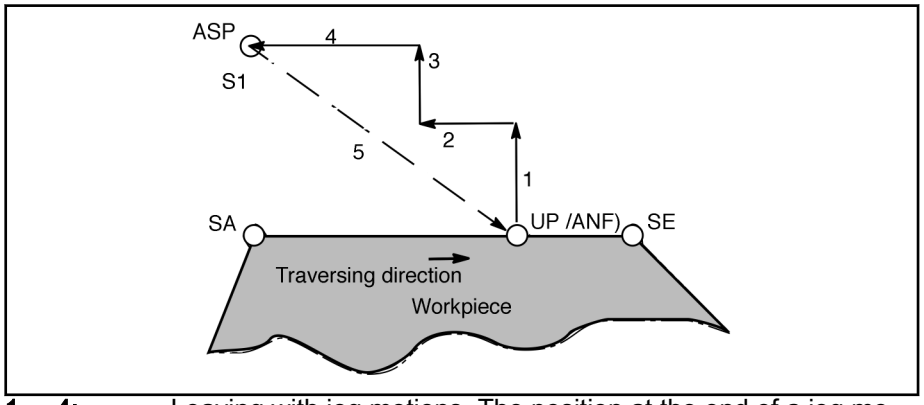
Available options:

- Manual
- Automatic
- Single block

☞ The following applies to the example figures shown in the following:

The selection of the approach point is set to "point of interruption".

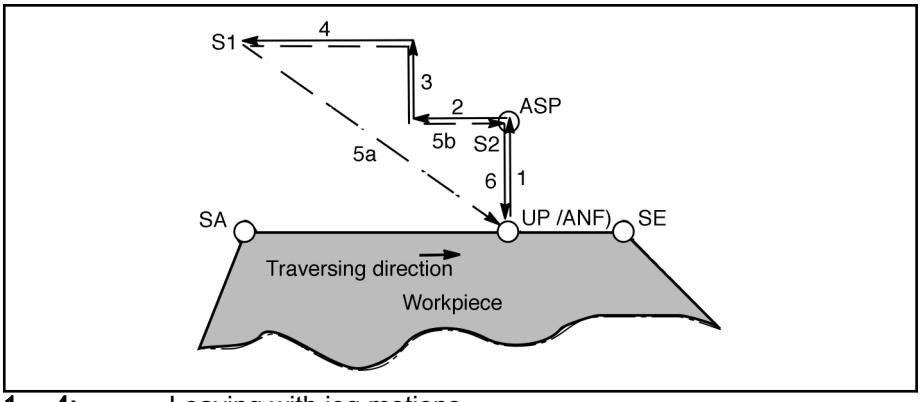
Manual:



1 ... 4: Leaving with jog motions. The position at the end of a jog motion always becomes the (new) placement point.
S1 : Triggering of NC start
Asup: If an asynchronous subroutine is logged into the system data, it is called here.
5: Approaching motion to the approaching point (here: point of interruption) and immediate continued machining.
ASP: Placement point (example)
SA: Start of block
UP: Point of interruption (example)
ANF: Approach point
SE: End of block

Fig. 5-2: Approach mode - Manual

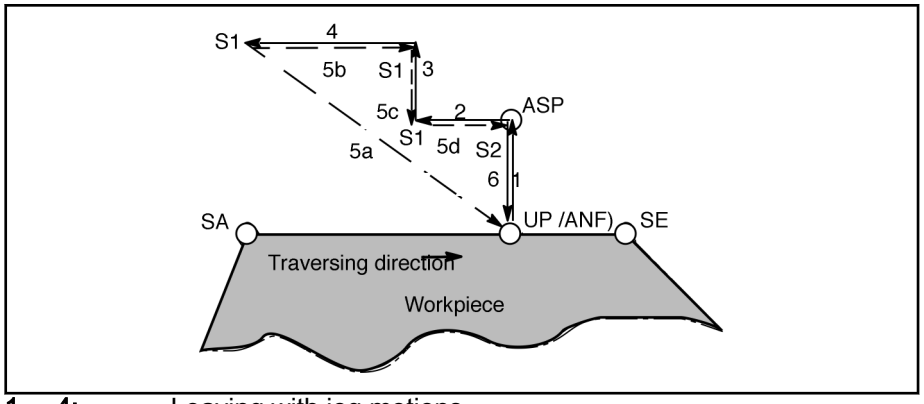
Automatic:



1 ... 4: Leaving with jog motions
S1, S2: Triggering of NC start
5a: Approach motion with immediate continued machining at the approach point (here: point of interruption) if the channel input signal qCh_RetCont "Return to contour" is not set when NC start (S1) is triggered. If an asynchronous subroutine is logged into the system data, it is called before the approach motion takes place.
5b: Approach motion to the placement point if the channel input signal qCh_RetCont "Return to contour" is set when NC start (S1) is triggered. The placement point can still be changed afterwards via jog motion. Subsequently, the NC travels to the approaching point (here: point of interruption) and continues machining via NC start (S2). If an asynchronous subroutine is logged into the system data, it is called after NC start (S2) before the approach motion takes place.
ASP: Placement point (example)
SA: Start of block
UP: Point of interruption (example)
ANF: Approach point
SE: End of block

Fig. 5-3: Approach mode - Automatic

Single block:



1 ... 4: Leaving with jog motions
S1, S2: Triggering of NC start
5a: Approach motion with immediate continued machining at the approach point (here: point of interruption) if the channel input signal qCh_RetCont "Return to contour" is not set when NC start (S1) is triggered. If an asynchronous subroutine is logged into the system data, it is called before the approach motion takes place.
5b ... 5d: Separate approach motion to the placement point if the channel input signal qCh_RetCont "Return to contour" is set when NC start (S1) is triggered. If the input signal "Return to contour" is not set when the NC start (S1) is triggered, the NC directly moves towards the approach point and continues machining. The placement point (ASP) can still be changed with jog motions. Subsequently, the NC travels to the approaching point (here: point of interruption) and continues machining via NC start (S2). If an asynchronous subroutine is logged into the system data, it is called before the approaching motion after: - NC start (S1), if the channel input signal qCh_RetCont "Return to contour" is not set at the contour. - NC start (S2)
ASP: Placement point (example)
SA: Start of block
UP: Point of interruption (example)
ANF: Approach point
SE: End of block

Fig. 5-4: Approach mode - Single block

5.1.3 Handling instruction: Leaving and Returning To Contour

Applying:

NOTICE **Workpiece or machine can be damaged!**

Do not reset the system during machining.

1. Adjust MP 7060 00310, 7060 00320 and 7060 00330 according to the desired standard return to contour strategy.
2. Trigger system reset.
3. If applicable, adapt the system date /SysReentry according to the desired return to contour strategy.

Activating:

☞ A channel reset aborts an active mode!
Any leaving motions already stored in the memory are deleted.

Activating the "Leaving the contour via PLC" **mode**:

Prerequisites:

- The active part program is interrupted using "feed hold".
- The channel interface signal iCh_RemoveFinish "Leaving completed" is not set.
- The channel interface signal qCh_RetCont "Return to contour" is not set.
- The channel interface signal qCh_OpModePlc "PLC mode" is set.

Activate the channel mode "Return to contour" via the channel interface signals qCh_OpModeSel_00 to qCh_OpMode Sel_03.

From now on, the control "stores" all traversing motions triggered via the axis-related interface input signals qAx_JogPlus "Manual+" or qAx_JogMinus "Manual-". At axis standstill, the control outputs the channel interface signal iCh_ReadyReEnter "Ready to return to contour".

Activating "Return to contour via PLC" **mode:**

Prerequisites:

- The "leaving the contour via PLC" mode (see above) was activated.
- Set the positive edge at the interface input qCh_RetCont "Return to contour".

  The NC then sets the interface output signal iCh_RemoveFinish "Leaving completed". More traversing motions triggered by jogging can be performed, but not recorded.

Depending on the selected approach strategy (see chapter 5.1 "Leaving and returning to contour" on page 31), returning to contour can be started via "NC start".

Use this function to leave and return to contour after a part program is interrupted using "feed hold" (NC stop). This can be required, for example, after a tool breaks, to manually measuring a workpiece or tool or for visual inspection.

IW Engineering/configuration: Configuring the returning to path mode:

- Determine the approach strategy in the parameters **ReturnMode** "approach strategy" (7060 00310).
- In the parameter **ReturnPoint** "approach point" (7060 00320), select the approach point.
- In the parameters **NofJogStep** "max. number of jog motions to be recorded" (7060 00330), determine the number of maximum possible jog motions to be recorded while leaving the contour.



Depending on the selected approach strategy, there are several variants to return to contour. For more details, refer to chapter 5.1 "Leaving and returning to contour" on page 31.

	Documentation	Instruction
Instruction:		Editing machine parameters
Documentation:	MTX Machine Parameters	Defining point of return to contour

IW Engineering/IndraLogic: Leaving the contour:

- Trigger "Feed hold" (NC stop) by setting the interface signal **qCh_FeedHold = 1** "Feed hold".

- The interface signal **qCh_RetCont** "Return to contour" has to be set to logic "0".
- The interface signal **qCh_OpModePlc** "PLC mode" has to be set to logic "1".

  qCh_OpModePlc = 1:

  The operation mode is specified by the PLC via **qCh_OpModeSel_00...03**. It is not possible to preselect an operation mode via the NC user interface.
- Activate the channel mode "Return to contour" via the interface signals **qCh_OpModeSel_00, to qCh_OpModeSel_03** with the coding **0b1011**.

☞ The channel interface signal **iCh_RemoveFinish** ,"Leaving completed" is not set.

Operation mode	Operation mode selection			
	Bit 3 (qCh_OpModeSel_03	Bit 2 (qCh_OpModeSel_02	Bit 1 (qCh_OpModeSel_01	Bit 0 (qCh_OpModeSel_00
Return to contour	**1**	0	1	1

	Documentation	Instruction
Instruction:		Editing PLC signal
Documentation:	MTX PLC Interface	Leaving the contour

IW Engineering/IndraLogic: Check or complete leaving the contour

Leaving the contour and thus recording the points is completed if the interface signal **qCh_RetCont** "Return to contour" is set.

- or -

if a level change from 0 to 1 at the interface signal **qCh_RetCont** "Return to contour" completes the recording of the leaving motion.

☞
- If leaving the contour is completed, the NC sets the interface signal iCh_RemoveFinish "Leaving completed".
- More traversing motions triggered by jogging can be performed, but not recorded.

	Documentation	Instruction
Instruction:		Editing PLC signal
Documentation:	MTX PLC Interface	Check Leaving the Contour

IW Engineering/IndraLogic: Returning to contour

- The "Leaving the contour via PLC" mode was performed.
- The setting of the interface signal **qCh_RetCont** "Return to contour" is interpreted as follows:

  qCh_RetCont = 1:

  Following the "NC start", the tool returns to the contour on the saved leaving motion.

  qCh_RetCont = 0:

  Machining is continued immediately after the "NC start".

- Depending on the selected approach strategy, there are several variants to return to contour. For more details, refer to chapter 5.1 "Leaving and returning to contour" on page 31.

☞ The interface signal **iCh_ReadyReEnter** "Ready to return to contour" is set to logic "1" if the system is ready to return to contour.

	Documentation	Instruction
Instruction:		Editing PLC signal
Documentation:	MTX PLC Interface	Leaving the contour

5.2 Block pre-run

5.2.1 General information about the block pre-run

Term definition

Block pre-run

When machining a workpiece, situations can occur (tool breakage, periphery error, etc.) that result in a program interruption and an NC program restart. In most cases, the workpiece is not severely damaged by interrupting the machining. Machining can be continued after the cause for interruption has been found and fixed.

Machining would have to be restarted without the "Block pre-run" function. The already processed program part would unnecessarily be traversed again. To reduce the machining time, the machine operator selects an NC block as target block close to the abort point (can also be in a subroutine), to resume machining at this point.

The MTX recalculates the entire NC program from program start up to this target block (computation), without generating axis motions or auxiliary functions. However, the logic status of the control is modified to how it would have been like in case of a regular program run at the target block. For example, the modal NC functions, the programmed velocities and the speed are set correctly so that machining at the selected target block can be resumed.

Computation

Processing the NC program from start block to the selected target block without generating axis motions or auxiliary functions is called computation. It can be queried in the NC program if a program is processed or if a computation is currently executed. The computation ends with the selected target block. Thus, the preconditions to start the interrupted program with the target block are created.

Adjustment

After reaching the target block, it has to be ensured that the physical machine status corresponds to the logic status of the control. The axes have to traversed from their current actual position to the command position of the NC program calculated during the block pre-run. Simultaneously, the spindles have to reach the last commanded speeds, directions of rotation and states again. All modal NC functions have to be restored. The states are automatically adjusted by the NC as well as by executing special adjustment programs. Optionally, individual adjustment functions can be triggered by manual input. Information for the user which adjustment actions have to be used are provided on special adjustment screens. Subsequently, the NC program is continued at the target block as if it has been started without a block pre-run.

Operation

The block pre-run is enabled via an M-key in the standard application.

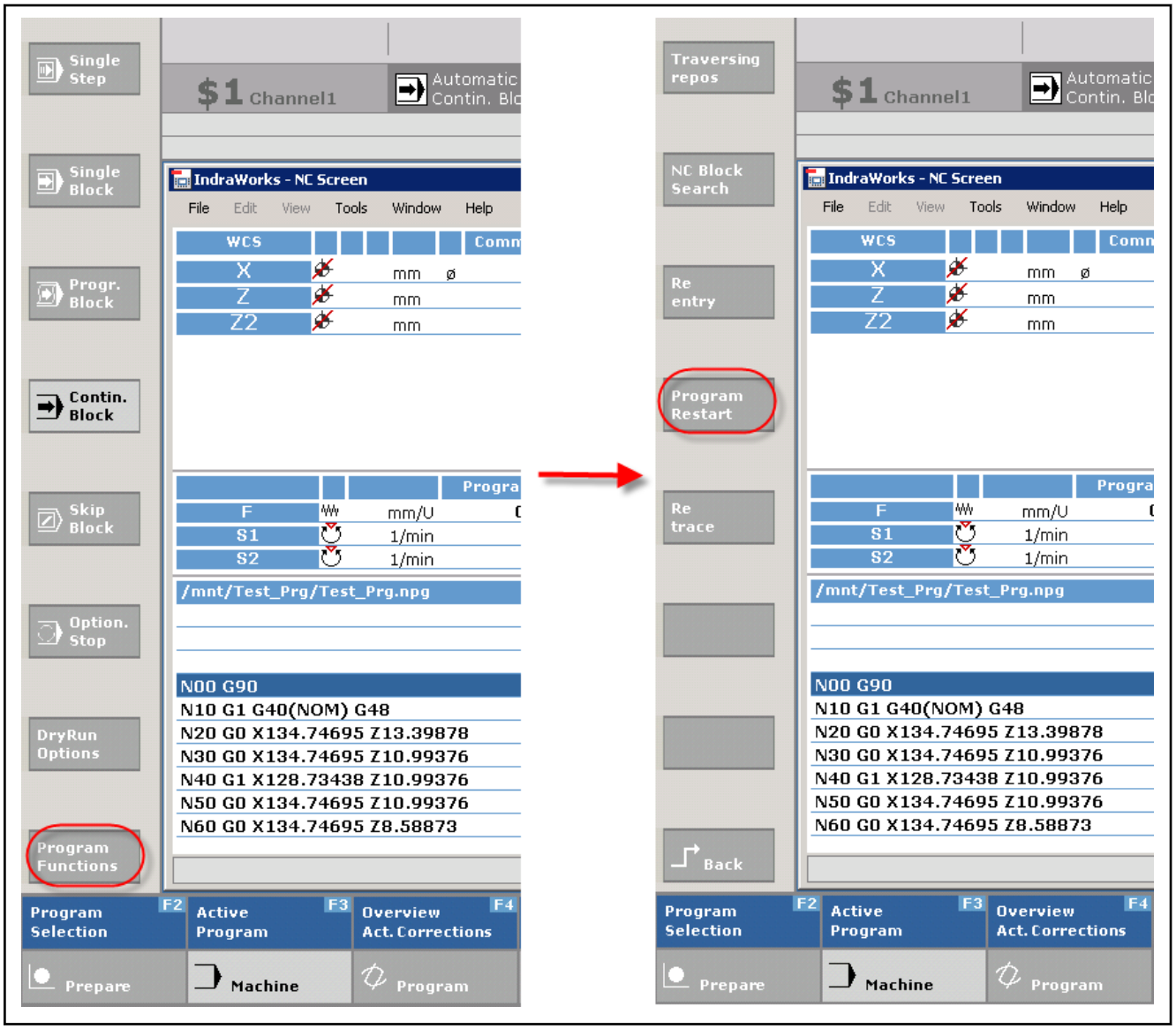


Fig. 5-5: Enable block pre-run

For further information on this topic, refer to the documentation "MTX xxVRS Standard NC Operation" (DOK-MTX***-NC*OP***Vxx-APRS-EN-P), chapter "Block search/block pre-run".

Defining the scope of the block search

Block search

By means of the block search, the user can start the selected part program from a defined program block. The selected program has to be set to program start so that the user can select the target block. This function acts like an absolute branch from the start of the program to the target block. This means that all other statements are ignored.

After an "NC start" has been triggered, the program processing is started at this block, based on the current machine and control states. All preceding blocks are not executed. They are not taken into consideration with regard to their content.

The settings can be selected under "MDI operation" if conditions have to be available at the machine.

Differentiating between block search and block pre-run

The "block pre-run" functionality runs through the entire part program and its subroutines, from start to target block and considers all programmed modal conditions, corrections and system settings that have been programmed up to this point. As in the block search, no axes are moved and no auxiliary functions are output.

5.2.2 Description

Function

After a part program selection, it is possible to perform a block pre-run via the "Adjustment functions" M-key.

On this topic, also refer to the documentation "MTX Standard NC Operation", Chapter "Block search/block pre-run".

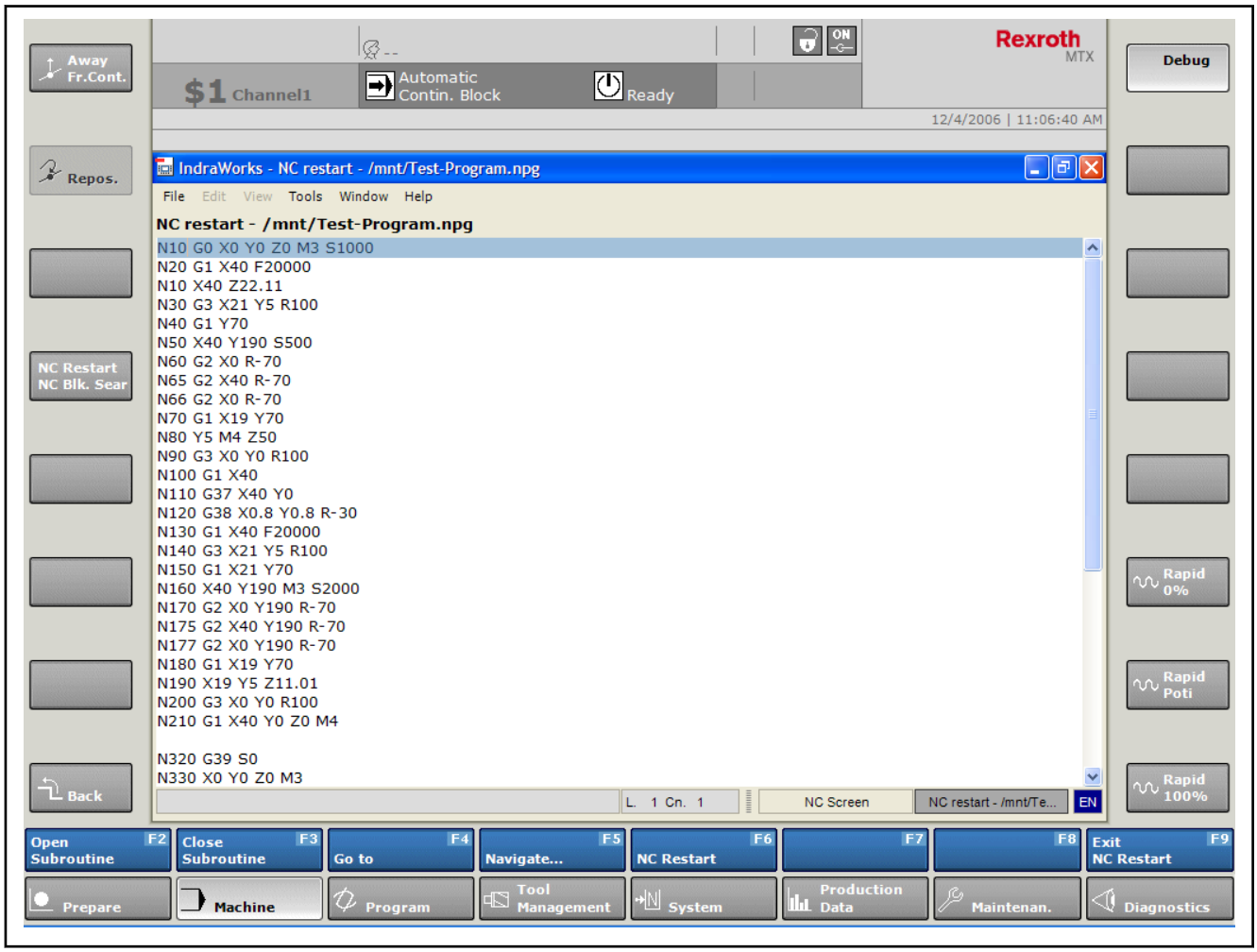


Fig. 5-6: Screen layout of block search/block pre-run

At block pre-run, a target block is specified first. The target block can be located in the main program or in subroutines. After the "Block pre-run" function is triggered, the NC blocks of the part program are calculated up to the "target block-1". However, **no** traversing motions are performed. The target block-1 is also called "abort block". There is no interpolation during block pre-run. Only one computation takes place. Dwell times, WAIT functions with parameters, NC synchronization functions and auxiliary functions are skipped during

block pre-run, but Sercos communication (writing of Sercos ID numbers) is executed.

Via the user interface, any NC block or CPL block can be selected as target block.

After a computation, block pre-run can be continued. The target block does not have to be changed. Thus, it is possible to select a defined cycle within program passages (loops) that has to be run through multiple times.

After block pre-run, the channel state is on "Ready". As a result of the NC start, the axes of the channel travel in a linear approach motion from their current position to the computed starting position or target position of the target block. Then, the part program is executed starting with the target block. The selection of the starting or end point is determined by a system date on which the part program can write. The selection - that is the system date - has to be specified before the adjustment.

Optionally, an "initialization adjustment program" (InitAsup) can be executed after NC start. InitAsup is logged into the system data. The path can be entered into SysSRun.NameOfInitAdjPrg.

InitAsup starts at the state which was active before the program was selected. In the InitAsup, it can be switched step-by-step to the state it was before the target block. This applies mainly to axis configurations, spindle configurations, coordinate transformations or modal states in the channel. At the end of the InitAsup, one or several MDI blocks can be additionally processed. Additionally, the respective menu item in the user interface has to be selected. At the end of the adjustment, it is switched back to the state it was before the target block.

Optionally, an "action block" and/or an (asynchronous) subroutine can be executed as "adjustment program" before the approach motion.

5.2.3 Block pre-run approach

Enabling the Block Pre-Run

The block pre-run can be called via M-keys after a program has been selected. The target block is selected using the "Apply Block" function key that simultaneously starts the computation...

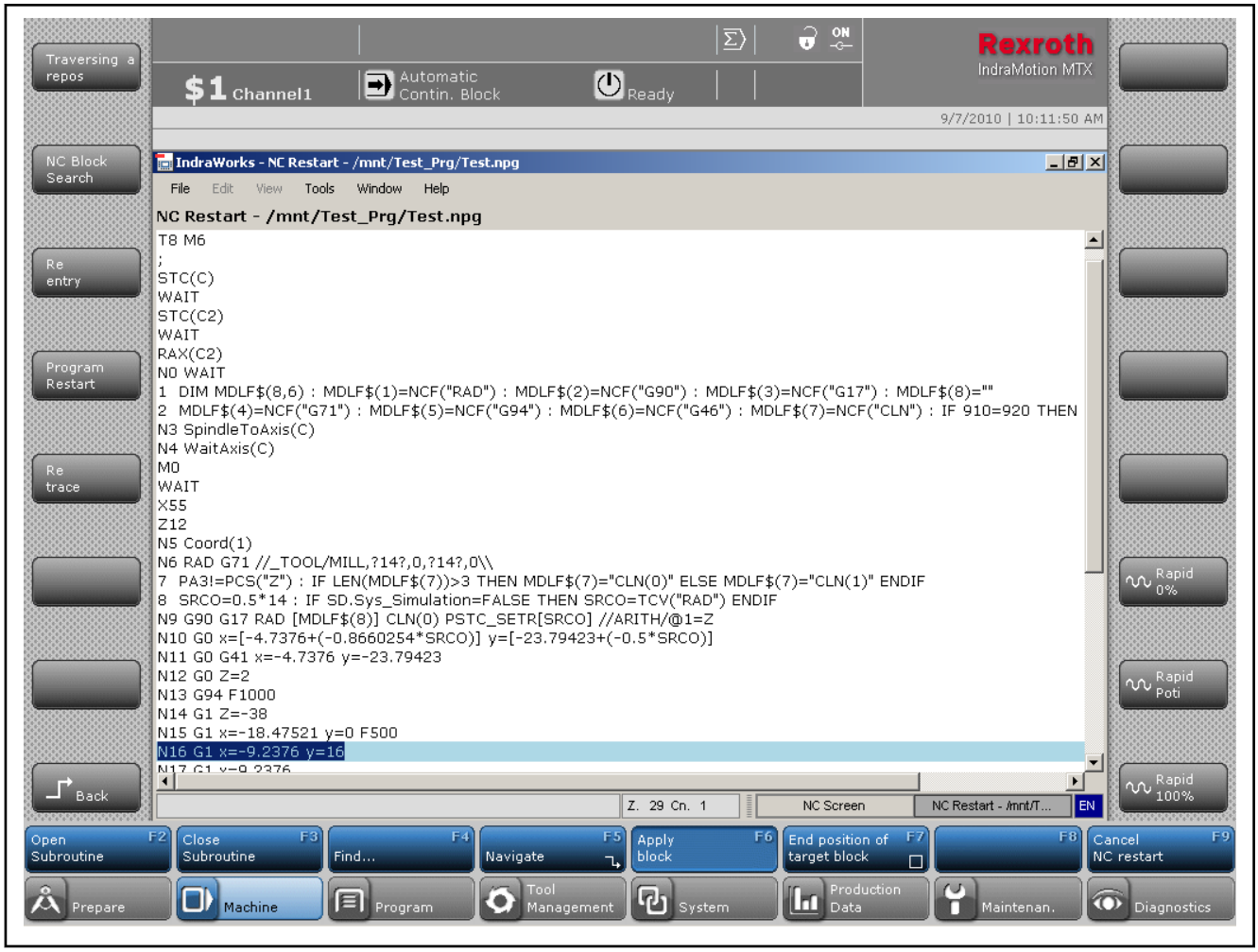


Fig. 5-7: Apply block

…and changes to the screen "Machine" after the computation has been completed successfully. Due to individual configurations, an additional adjustment is required for the individual areas before the actual approaching motion. This adjustment is represented by an additional symbol.

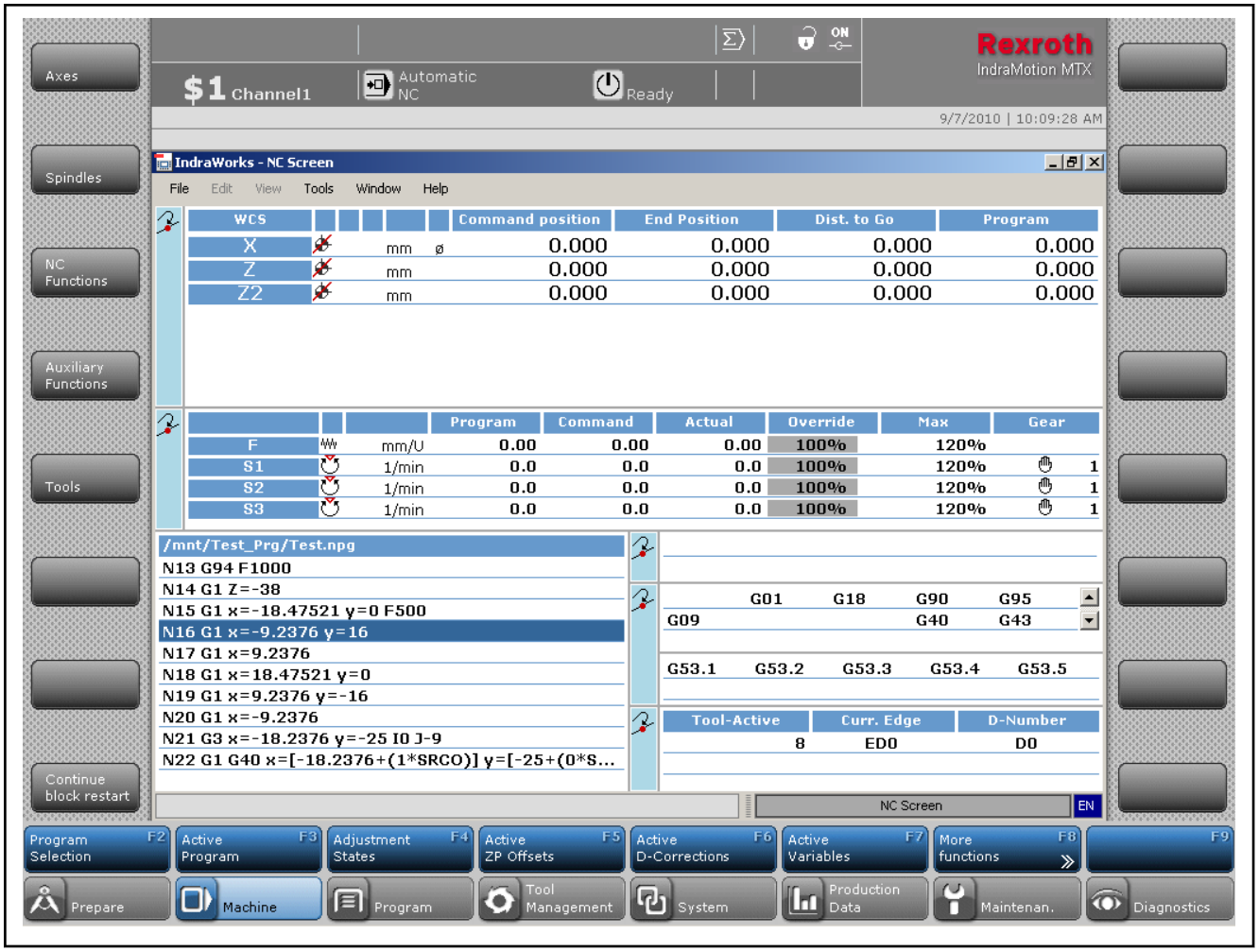


Fig. 5-8: Adjustment screen

More information about the individual adjustment groups (axes, spindles, etc.) can be obtained by using the left M-keys.

In this state, an "NC start" triggers a block pre-run that optionally starts with the adjustment before the approaching motion.

Operations during the computation

Schematic representation

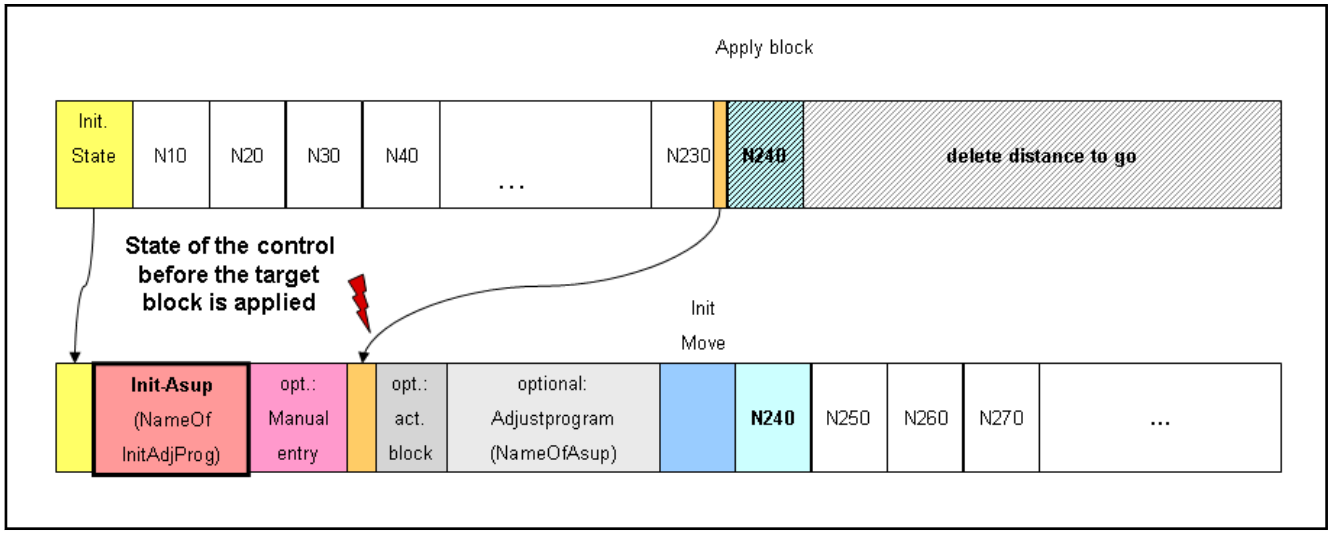


Fig. 5-9: Block pre-run adjustment

Procedure

The computation is defined by its specifications in the system data in the block pre-run operating mode (see chapter 5.2.4 Configuring the Block Pre-Run, page 47). The control-internal block preparation recalculates itself and executed a "Delete Distance to Go". Subsequently, the adjustment conditions are executed according to the configured system data before executing an approaching motion to the target block.

The adjustment conditions are implemented in two steps:

- 1. Generate program conditions (states) before or in the target block
- 2. Optionally add supplementing conditions

The basic conditions of all program blocks up to the target block are fulfilled, starting with the "initialization status" and the "InitAdjustPrg" adjustment program. Important for the basic conditions are the axis, coordinate and spindle configuration. Furthermore, the NC functions, tools and if required the auxiliary functions have to be adjusted. After having executed the InitAdjustPrg, special functions and states can be manually adjusted via an optional system date "manual input". If this option is used, an "NC start" has to be set after the InitAdjustPrg to continue the adjustment in the block pre-run.

Once the first step of the adjustment is completed, it is verified if the axis configuration has been adjusted to the target block status.

The now saved status at the start of the target block is used again in the control-internal block preparation. If individual states cannot be adjusted, error messages are reported or - worst case scenario - the machine fails to operate correctly.

In the second step, more actions such as e.g. auxiliary function can be optionally enabled in an action block and/or an additional adjustment program "TargetAdjustPrg". To complete the adjustment in the block pre-run, an approaching motion occurs, i.e. the axes are automatically positioned to their target position (position adjustment). Some modal NC function groups have been adjusted automatically if they have been changed in the action block or under "TargetAdjustPrg".

⚠ CAUTION **All axes to be adjusted are traversed simultaneously**

If this is not possible, the axes have to be adjusted manually via G77 in TargetAdjustPrg, see example.

5.2.4 Configuring the block pre-run

System Data

Description

The block pre-run is parameterized by means of the system data. All settings and status queries are saved in the system data structures.

Only the most important system data relevant to the block pre-run are described in detail in this documentation.

Overview on the system data for the block pre-run

For more information, see the documentation "MTX Machine Parameter", chapter "System Data".

Block pre-run data for axes "/SysAxSRun"

Structural element	Description
NotAdjusted	Axis not adjusted
PosNotAdjusted	Axis position not adjusted
ConfNotAdjusted	Channel assignment not adjusted
ReqPosAdjust	Position adjustment required
TargChan	Target channel
ReleaseChan	Release channel

Tab. 5-4: Overview on the "SysAxSRun" structural elements

Spindle block pre-run data "/SysChSpSRun"

The system data structure "SysChSpSRun" is created for each channel in the volatile memory.

The system data structure consists of three parts:

General data (SysChSpSRun):

Structure element	Description
MaxSpNumber	Highest spindle number available in the channel
PrgSpNumber	Highest programmed spindle number in pre-run
PrgCpNumber	Highest programmed coupling number in pre-run
ConstCutStatus	Adjust G-function group G96/97
SpeedStr	Speed syntax of a spindle
MoveStr	Movement syntax of a spindle
GearStr	Gear syntax of a spindle
SpConfNotAdjusted	Spindle configuration adjustment required

Tab. 5-5: Overview on structure elements "SysChSpSRun" (general data)

Data channel spindles (SysChSpSRun.Sp[<ChannelSpindleNumber>]):

Structure element	Description
SysSpNo	System spindle number
DriveNo	Drive number
IsProg	Spindle programmed in pre-run
NotAdjusted	Adjustment required
MoveNotAdj	Adjustment status of the movement function
GearNotAdj	Adjustment status of the gear
SpeedNotAdj	Adjustment status of the speed
LimitNotAdj	Adjustment status of the speed limitation
CcvNotAdj	Adjustment status of the G96 state
CAxisNotAdj	Adjustment status of c-axis required
OpModeNotAdj	Adjustment status of the drive operation mode
CpDataNotAdj	Adjustment status of the coupling data
CoupleNotAdj	Adjustment status of the coupling state
Move	Movement function
Gear	Gear level
AutoGearShift	Automatic gear shifting
CAxisOn	Switch on c-axis
CAxisType	C axis type
CcvOn	Switch on constant cutting velocity
CcvAxis	Channel reference axis for constant cutting velocity
CcvCoordSystem	Coordinate system for constant cutting velocity
PosCtrlOn	activates positioning interface
OriPos	Orientation position
CcvVelocity	Cutting velocity
ProgSpeed	Programmed speed
ProgMinSpeed	Minimum speed
ProgMaxSpeed	Maximum speed
CoupleDistance	Coupling distance
SyncWindow	Synchronization window for coupling
SyncErrorWindow	Synchronization window error for coupling
CouplePosOffset	Angular offset in case of active coupling
CouplePosOffsetSpeed	Speed to approach the angular offset

Tab. 5-6: Overview on structure elements "SysChSpSRun" (data for channel spindles)

Data for spindle couplings (SysChSpSRun.Cp[<CouplingNumber>]):

Structure element	Description
IsProg	Couple link programmed in pre-run
NotAdjusted	Adjustment status of spindle coupling

Structure element	Description
CoupleOn	Switch on couple link
Master	Channel spindle number of the master spindle
Slave[1..8]	Channel spindle number of the slave spindle

Tab. 5-7: Overview on structure elements "SysChSpSRun" (data for spindle couplings)

Channel-specific block pre-run data "/SysChSRun"

Structural element	Description
Prep/ChAxNames[1..8]	Channel axis names in the block preparation
Act/ChAxNames[1..8]	Active channel axis names in adjustment
Targ/ChAxNames[1..8]	Channel axis names in the target block
PosAdjust[1..8]	Axis names for position adjustment
TargPos[1..8]	Target position of the approaching block
AcsTargPos[1..8]	Target position in the axis coordinate system
ActAxConf[1..8]	Current axis configuration in adjustment
PrepAxConf[1..8]	Axis configuration in the block preparation
TargAxConf[1..8]	Axis configuration in the target block
TechAxAdjust	Technological axis adjustment
Active	Computation active
ActMode	Block pre-run mode
TargAdjPrgActive	Target block adjustment program (TargAdjPrg) active
InitAdjPrgActive	Init adjustment program (InitAdjPrg) active
LookAheadPreRun	Block pre-run active in look-ahead range after target block
ChanNotAdjusted	Channel is not adjusted
NcfNotAdjusted	Modal NC functions are not adjusted
AuxNotAdjusted	Modal auxiliary functions are not adjusted
AxConfNotAdjusted	Axis configuration is not adjusted
CoordNotAdjusted	Axis transformation is not adjusted
ActStatus	Active adjustment state
TargBlockSrc	Target block code
ReentryInfoDir	File path on re-entry data
TargetBlkAvailable	Backed up target block available
LockTargBlock	Target blocks locked

Tab. 5-8: Overview on structure elements "SysChSRun"

Data of the repositioning logic "/SysReentry"

Structure element	Description
FeedVal	Individual return to path velocity
OptFeedVal	Optional velocity if no feed was programmed
TangCircleRadius	Return to path radius
NameOfAsup	Adjustment program name of the return to path logic (ReentryAsup)
Approach	Return to contour
FeedMode	Return to path velocity selection
AsupActive	Adjustment program of the return to path logic (ReentryAsup) active

Tab. 5-9: Overview on structure elements "SysReentry"

General block pre-run data "/SysSRun"

Structural element	Description
Active	See: SysChSRun.Active
AsupActive	Adjustment program active
LookAheadPreRun	See: SysChSRun.LookAheadPreRun
Reserved01	Reserved
TargPos	See: SysChSRun.TargPos
TargBlockSrc	See: SysChSRun.TargBlockSrc
ReEntryIndex	Latest used storage index of abort information
LockTargBlock	See: SysChSRun.LockTargBlock
LockWarning	Note text for blocked target block
NameOfAsup	Name of target block adjustment program (Asup)
NameOfInitAdjPrg	Name of the Init adjustment program
ActionBlock	Action block
Timeout	Runtime monitoring for block pre-run
RetPaStrategy	Approaching strategy after block pre-run
TargetBlkCheck	Backed up point of interruption available
ManWaitForStart	Manual data input after block pre-run
SuppressMsg	Message suppression of messages in block pre-run
ReEntryStore	Backing up aborted information for a later re-entry

Tab. 5-10: Overview on the "SysSRun" structural elements

Active and preselected tool for block pre-run "/SysToolSRun"

Structure element	Description
ActTool.K1	Tool data element K1 of the active tool
ActTool.K2	Tool data element K2 of the active tool
ActTool.SKQ	Tool data element SKQ of the active tool
ActTool.IKQ1	Tool data element IKQ1 of the active tool

Structure element	Description
ActTool.IKQ2	Tool data element IKQ2 of the active tool
ActTool.IKQ3	Tool data element IKQ3 of the active tool
ActTool.IQ1	Tool data element IQ1 of the active tool
ActTool.IQ2	Tool data element IQ2 of the active tool
ActTool.IQ3	Tool data element IQ3 of the active tool
ActTool.BQ1	Tool data element BQ1 of the active tool
ActTool.BQ2	Tool data element BQ2 of the active tool
ActTool.BQ3	Tool data element BQ3 of the active tool
PreTool.K1	Tool data element K1 of the prepared tool
PreTool.K2	Tool data element K2 of the prepared tool
PreTool.SKQ	Tool data element SKQ of the prepared tool
PreTool.IKQ1	Tool data element IKQ1 of the prepared tool
PreTool.IKQ2	Tool data element IKQ2 of the prepared tool
PreTool.IKQ3	Tool data element IKQ3 of the prepared tool
PreTool.IQ1	Tool data element IQ1 of the prepared tool
PreTool.IQ2	Tool data element IQ2 of the prepared tool
PreTool.IQ3	Tool data element IQ3 of the prepared tool
PreTool.BQ1	Tool data element BQ1 of the prepared tool
PreTool.BQ2	Tool data element BQ2 of the prepared tool
PreTool.BQ3	Tool data element BQ3 of the prepared tool
TargEd	Number of the active edge in the target block
TargDNr	Number of the active D correction in the target block
TCorrNotAdj	Tool correction not adjusted

Tab. 5-11: Overview on the structure elements "SysToolSRun"

Important System Data for Configuring the Block Pre-Run

For more information, see the documentation "MTX Machine Parameter", chapter "System Data".

Computation active	Active

Provides the channel-specific information that the computation of block pre-run is active in the channel.

By querying in the program, NC or CPL parts (such as a tool change or modifications in the database) can be hidden during the computation.

Path /SysChSRun[<Channel>]/Active

Description of parameter values

0 Block pre-run is not active.

1 Block pre-run is active.

Lock target blocks	LockTargBlock

Allows in the respective channel to lock areas for the target block selection in the part program.

In the part program, ranges in which no target blocks are permissible can thus be locked. When the lock is activated, the target block is applied, but an NC start is not assumed. In this case, either a general warning or – if present – an individual note text (see **/SysSRun/LockWarning** system date) is output. The block pre-run cannot be continued on another target block.

The date is generally written in the NC program.

Path /SysChSRun[<Channel>]/LockTargBlock

Description of parameter values

0 Target blocks are not locked.

1 Target blocks are disabled.

Individual return to path velocity	FeedVal

Indicates an individual return to path velocity. It becomes active if the value 2 is entered in "/SysReentry/FeedMode".

The value can be modified in the part program.

Path /SysReentry/FeedVal

Description of the variable values Approaching velocity

Optional return to path velocity	OptFeedVal

Indicates an optional return to path velocity. It is applied if a value of 1 is entered in "/SysReentry/FeedMode" and no valid F-value is applied in the freeze block or if time programming is active (G93).

If an invalid value is parameterized for the optional return to path velocity, the emergency strategy to use the **low jog velocity** of the respective first axis in the channel as optional return to path velocity.

It is recommended to enter a reasonable velocity value. The value can also be entered in the part program.

Path /SysReentry/OptFeedVal

Description of the variable values Approaching velocity

Return to path velocity selection	FeedMode

Defines how the approaching velocity is to be determined.

The value can be modified in the part program.

Path /SysReentry/FeedMode

Description of the variable values

0 Approaching velocity = jog velocity

1 Approaching velocity = active feed in the freeze block (programmed feed)

2 Approaching velocity = value from "/SysReentry/FeedVal"

Note text for blocked target block	LockWarning

For each channel, defines an individual note text if the target block is locked.

If nothing is specified here, the general warning in the respective language "This target block is locked." is output. The ID "%s" can be used in the text. It is replaced by the general warning.

Path /SysSRun[<Channel>]/LockWarning

Description of the variable values

<EmptyString> If the target block is locked, a general warning is output.

<Note text> If the target block is locked, this note text is output.

Name of the target block adjustment program	NameOfAsup

Defines the name including the path specification of the program to be called a target block adjustment program.

If no program is specified, no target block adjustment program is called before the approaching movement.

Path /SysSRun[<Channel>]/NameOfAsup

Description of the variable values

<EmptyString>	No target block adjustment program is called before the approaching movement.
<invalid program name>	The indicated target block adjustment program is called before the approaching movement.

Name of the Init adjustment program	NameOfInitAdjPrg

For each channel, defines the path and name of the subroutine that is to be called as the initialization adjustment program.

If no subroutine is specified, no Init adjustment program is called before the approaching movement.

Path /SysSRun[<Channel>]/NameOfInitAdjPrg

Description of the variable values

<EmptyString>	No Init adjustment program is called before the approaching movement.
<invalid program name>	The indicated Init adjustment program is called before the approaching movement.

Action block	ActionBlock

Defines an action block for each channel.

If an NC block is specified here, it is processed at the start of machining after a block pre-run and before the start of the adjustment program which is also defined.

Path /SysSRun[<Channel>]/ActionBlock

Description of the variable values

<EmptyString>	There is no action block.
<Action block>	When machining is started after block pre-run, the defined action block is processed first.

Runtime monitoring for block pre-run	Timeout

It allows canceling the block pre-run in the respective channel after monitoring time has elapsed.

The runtime monitoring can be used if a target block was skipped in a program due to an unfulfilled condition and if the block pre-run is not automatically closed due to an endless loop in the program. Programs containing potential endless loops should set this system date to a sufficient high value at the beginning and subsequently restore it.



If the control load is high, the block pre-run might be aborted due to runtime exceedance in the case that the timeout value is only set a bit over the block pre-run runtime of the unloaded control.

Set a sufficiently high value or completely disable monitoring.

Path /SysSRun[<Channel>]/Timeout

Description of the variable values

0	The runtime monitoring is deactivated.
>0	The runtime monitoring is active (e.g. 10000 ms).

ActTool.K1	ActTool.K1

Tool data element K1 of active tool

Path /SysTool[<Channel>]/ActTool/K1



The date can be freely assigned. Usually it is used for the sector selection and is e.g. used to differentiate between spindle, gripper and tool list.

ActTool.K2	ActTool.K2

Tool data element K2 of active tool

Path /SysTool[<Channel>]/ActTool/K2



The date can be freely assigned. Usually, the date is used to select the position. It is used to divide the sectors into positions.

Number of the active edge in the target block	TargEd

Number of the active edge in the target block.

Path /SysToolSRun[<Channel>]/TargEd

Description of the variable values

0	No edge active
1..16	Number of the active edge in the target block

Number of the active D-correction in the target block	TargDNr

Number of the active D-correction in the target block.

Path /SysToolSRun[<Channel>]/TargDNr

Description of the variable values

0	No correction active
1..99	Number of the active D correction in the target block

Adjusting Modal NC Functions "Adjust, ADJ"

Description: Using the adjustment function "Adjust", modal NC functions within the adjustment programs can be comfortably adjusted after block pre-run. The syntax of a modal NC function is provided as parameter to the adjustment function. The control enables the NC function in the respective modal group in front of the target block of the block pre-run.

Syntax:

ADJ(<NCF Name>)	
<NCF name>	Syntax of a modal NC function of the group to be adjusted.

Tab. 5-12: ADJ syntax

Example: ADJ(G0)

The modally effective geometry function (G0, G1, G2, G3...) before the target block becomes active and thus the NC function group is adjusted.

The function-specific parameters for the NC functions listed in the following are adjusted simultaneously:

Modal group	NC functions	Supported parameters
Interpolation functions	G0	Polar programming and exact stop window (NIPS, IPS<n>)
	G1	Polar programming and exact stop window (IPS<n>)
	G2, G3	Polar programming
Path slope	G8	Path shape
	G9	Path shape, axis shape
Level	G17, G18, G19, G20	Axis assignment
Tool length correction	G47	Length assignment
Zero point offset 1	G52.1	Programmable ZO 1*)
	G53.1	ZO 1 off *)
	G54.1,...,G59.1	Active ZO table bank 1*)
...		
Zero point offset 5	G52.5	Programmable ZO 5*)
	G53.5	ZO 5 off *)
	G54.5,...,G59.5	Active ZO table bank 5*)
Placements: "Inclined plane" 1	G151.1	Tool orientation**)
	G152.1	Programmable inclined plane 1**)
	G153.1	Inclined plane 1 off**)
	G154.1,...,G159.1	Active inclined plane; table bank 1**)
...		
Placements: "Inclined plane" 5	G151.5	Tool orientation 5**)
	G152.5	Programmable inclined plane 5**)
	G153.5	Inclined plane 5 off**)
	G154.5,...,G159.5	Active inclined plane; table bank 5**)
Exact stop	G61	Exact stop window (IPS<n>)
Tool length assignment	G78	ActPlane or length assignment
Path feed	G93	F-value
	G94	F-value, F2-value
	G95	F-value, reference axis (FRA)
Spindle programming	G96	Reference axis, spindle assignment
	G97	Spindle assignment
D-correction	D0	D-correction off***)
	D1,...,D99	Active D-correction***)

Modal group	NC functions	Supported parameters
Axis acceleration	AAC	Programmed axis accelerations
Axis jerk	AJK	Programmed axis jerks
Additive program coordinate offset	ATR	Program zero point coordinates
Maximum axis velocity	AVE	List of axis velocities
Workpiece position	BCR	Workpiece zero point coordinates, angle
Chamfer/roundings	CHL CHS RND RNE	Chamfer length, chamfer segment Radius, epsilon
Automatic corner detection	CLD	Angle, angle factor, maximum point distance, distance factor, accuracy limit
Milling path collision detection	CLN	Monitoring type, preview area
Axis transformation	COORD	Axis transformations 1 and 2 with transformation options
Radius/diameter programming	DIA RAD	Diameter coordinates
Channel spindles and spindle groups	DSP, GSP, RSP SPGn, SPGALL	Channel spindles and channel group assignment (corresponds to SPADJSP)
Block transition without reduction of velocity	DTT	Transition angle
Feed masking axes	FAD	List of masking axes
Feed coordinate group	FDG FDGT	List of path coordinates, feed group type
Straightness and angle error compensation	GCT	-
Active axis jerk	JAL	List of path-relevant axis jerk axes
Jerk limitation	JKC	-
Input tool "Mirroring"	MIR	Mirroring axes
Main spindle	MSP	Spindle
Path acceleration	PAC	Path acceleration, path deceleration
Pole definition	PLS	Mirroring/turning point
Rotary axis positioning mode	PMD	Positioning mode
Pole definition	POP	Pole coordinates
Accuracy programming	PRP	Radius contour error (EPS), lagging (DIST)
Maximum radial acceleration	RAC	Acceleration
Input help: Turning	ROT	Angle

Modal group	NC functions	Supported parameters
Input help: Scaling	SCL	Axis factors
Input help: Offset	SHT	Contour zero point coordinates
Spline definition	SPD	Spline type, spline coordinates, starting conditions (SBC), end conditions (EBC), spline parameter length (PL)
Geometric splitting	SPT	Mode, length
Program coordinate offset	TRS	Program zero point coordinates
Tapping spindle	TSP	Spindle

Tab. 5-13: Adjustable modal parameterizable NC functions

*) Implicit ZO table selection (ZOS)

**) Implicit placement table selection (PMS)

***) Implicit D-correction table selection (DCS)

Special features and restrictions:

- If necessary, the adjustment of modal parameterizable NC functions not mentioned above has to be performed using pre-run system data in the adjustment program (InitAdjPrg) or using the action block.
- Each bank of a zero point offset or an inclined plane has to be adjusted separately: ADJ(G54.1), ADJ(G54.2), etc.
- The tool radius correction G41/G42 becomes only active in the adjusted state with the approaching block after the adjustment. During the adjustment, the internal effect is suppressed. The correction is not calculated.

Modal NC function groups that are restored during the approaching motion

The following modal functions and states are restored automatically in the approaching block in case they have been switched over or changed in the action block or the TargetAdjustPrg.

Modal group	NC functions	Supported parameters
Milling cutter path correction	G40	Milling cutter path correction OFF
	G41	Milling cutter path correction to the left of the workpiece
	G42	Milling cutter path correction to the right of the workpiece
Tool length correction	G47	Tool length correction ON
	G48	Tool length correction OFF
Exact stop	G61	Exact stop ON
	G62	Exact stop OFF
Abs/Rel	G90	Absolute dimension programming
	G91	Relative dimension programming
Feed programming	G93	Time programming
	G94	Feed programming in mm
	G95	Feed programming in revolutions

Tab. 5-14: Modal NC functions during the approaching motion

5.2.5 Switching modal states in the adjustment program or Action Block

In the action block or the first block of the adjustment program, some NC functions are reset (switched off) implicitly if they were activated in the block pre-run:

- G41/G42 → G40
- G47 → G48

Analogously, these functions are automatically activated again in the approach block. In addition, the following modal functions and states are restored (in case they were switched or changed in the action block or adjustment program:

- G90/G91
- G93/G94/G95
- F-value

If the states of all other modal functions are changed in the adjustment program, they have to be managed manually in this program. The CPL command "NCF" is recommended for example. It can be used to read the active functions of the individual modal groups and the configured auxiliary function groups to store them in variables.

Thus, all auxiliary functions to be activated after the block pre-run have to be assigned to auxiliary function groups.

The modified groups must be restored by the end of the adjustment program at the latest. If this is not the case, the modified modal state is also applied to the selected part program.

Also note that the NC function "AUXFUNC" which outputs the current auxiliary functions of all auxiliary function groups.

Apart from special system data, signals are also available at the bit interface for third-party communication. While the system data is used to control the block pre-run and the adjustment and it is set at the time of block preparation (prep), the interface signals are output at the active point in time (act).

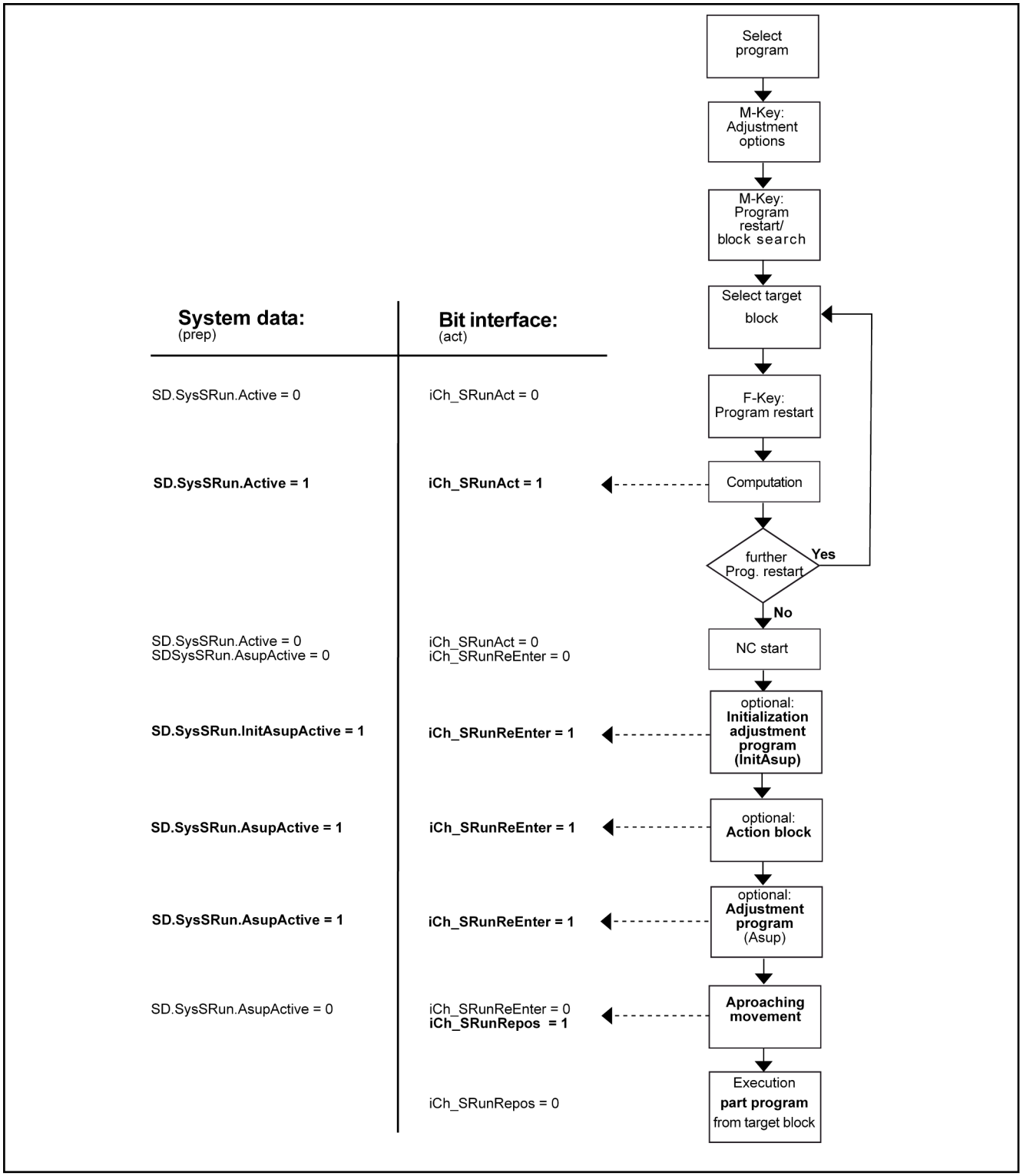


Fig. 5-10: Switching modal states in the adjustment program or action block

5.2.6 Block pre-run applications

Manual input

One or more NC blocks can be specified and processed via the manual input after the block pre-run. The NC states can thus be manually adjusted.

The manual input is optional and can be called after InitAdjustPrg.
A preceding InitAdjustPrg is mandatory.

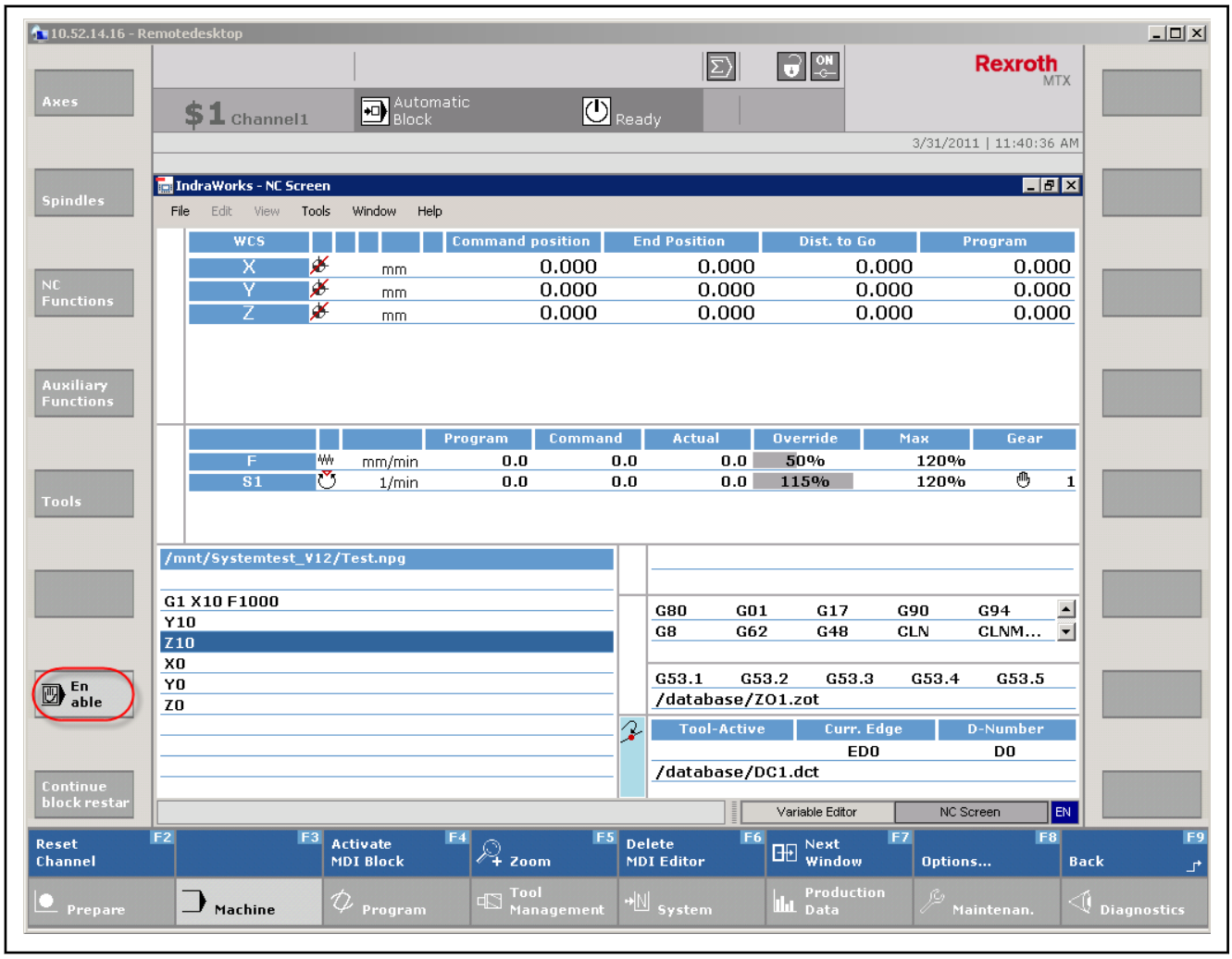


Fig. 5-11: Releasing the manual input

If the manual input is released, the block progress display is automatically replaced by the manual entry editor after exiting the adjustment program (InitAdjustPrg) in the overview adjustment screen. NC blocks can be entered for the MDI operating mode. They can be transferred to the NC and they can be processed with the NC start key (control panel).

Without having entered an NC block, the next NC start calls an action block and the adjustment program (TargetAdjustPrg), if it is parameterized. The manual input is completed. The manual data input editor in the adjustment screen is replaced by the block progress bar.

Restrictions

The manual input is only available if an InitAdjustPrg is used.

No CPL statements and no subroutine calls are possible.

Only individual blocks and no buffered NC block specification is possible.

Only basic NC function can be programmed.

For further information on this topic, refer to the documentation "MTX xxVRS Standard NC Operation" (DOK-MTX***-NC*OP***Vxx-APRS-EN-P), chapter "Block search/block pre-run".

Re-entry marks “RCB marks”

Description Marks a position in the program from which the program can be continued after having been canceled. In contrast to the normal block pre-run with manual specification of the target block, it is possible to skip successfully processed program passages or to allow finding the target block under certain prerequisites.

Application If an NC program is terminated during processing due to different reasons, processing shall begin at a predefined position immediately. Reasons for an abort can be tool rupture, power failure, NC control error, etc. In these cases, the NC programmer can determine where (NC block) a repositioning in the NC program is reasonable.

Please refer to the documentation "MTX Programming Manual", chapter "Program Continuation with Block Pre-Run After Program Cancelation RCB".

Init adjustment program "InitAdjPrg"

In the "InitAdjPrg" adjustment program, the target block state (-1) has to be reached.

The following has to be adjusted:

- Axis configuration
- Coordinate configuration
- Spindles
- Couplings
- Tools
- NC functions
- Auxiliary functions
- Axis and axis name configuration (unless the NC command "TAX" is used!)
- Other



The "TAX" NC command automatically creates the axis and axis name configuration!

The control uses the TAX command to activate the axis configuration prior to the target block of the block pre-run. Superfluous axes are removed. Required axes are included into the channel. Axis names are also applied.

Axes assigned otherwise are not taken into account:

- An axis that is assigned to another channel cannot be included.
- A C-axis that is in spindle mode cannot be included.
- An axis that is part of a transformation cannot be removed.

Please refer to the documentation "MTX Programming Manual", chapter "Adjust axis configuration, TAX".

Adjustment program sequence Possible functional sequence for adjustment in the InitAdjPrg.

- Adjust channel spindle configuration
- Switch off active transformations
- Convert C-axis to spindles
- Remove channel spindle
- Remove redundant axes (RAX)
- Get axis (WAX)

- Adjust spindle configuration
- Get C-axes
- Enable transformation
- Tool change
- Adjust spindle
- If required, adjust more NC functions
- Adjust auxiliary functions

⚠ CAUTION **The adjustment sequence depends on the application**

Thoroughly check the application and customize the adjustment. Thoroughly check the machine state after the adjustment before triggering a traversing motion.

5.2.7 Spindles

Description

Function

Spindles are asynchronous drives. During a block pre-run, the spindles are not controlled. The programmed data is logged internally in the NC and stored in the system data

SD.SysChSpSRun[<Channelno.>]

.

In this system date, the NC also stores the information necessary for the spindle adjustment. This adjustment information is updated while processing the adjustment program stored under SysSRun[<ChannelNo.>].NameOfInitAdjPrg and SysSRun[<ChannelNo.>].NameOfAsup in each spindle programming.

Restrictions

The following spindle functions are not supported:

- SpAdmin / SPA
- SSpAdm
- SSpAdmOff

Relevant NC functions

Adjust(..) / ADJ(..)

ADJ(G96) adjusts the NC function group for the constant cutting velocity (G96) and the direct spindle speed programming (G97).

SpAdjSp

The function "SpAdjSp" adjusts the logic spindles of the channel (automatic sequence of "RemSpindle" and "GetSpindle").

SpAdjStr <Spindleno.>

The function "SpAdjStr" <SpindleNo.> fills the following components of the SD.SysChSpSRun system data when being called in an adjustment program:

- SpeedStr
- MoveStr
- GearStr

Example

Refer to chapter "Adjusting the spindle" on page 68.

This is an example program to adjust simple spindle functions.

This program can be used as subroutine of an adjustment program.

5.2.8 System data

The block pre-run data is summarized in the system date "SysSRun". The block pre-run data can be read or modified from the user interface, the PLC or the part program. Thus, it is possible to react flexibly during block pre-run and in the adjustment program.

Element name	Description	Type	Size in bytes	Write access
Active	Assumes the value 1 during block pre-run, otherwise 0. To query in the part program. Thus, it is possible to skip program passages during block pre-run, for example a tool change or table modifications.	Byte_t	1	NC kernel
AsupActive	Assumes the value 1 when executing the adjustment program (Asup) after block pre-run, otherwise 0. Used for queries in the adjustment program. Thus, general subroutines can be used as adjustment program which respond differently in the Asup of the block pre-run for example.	Byte_t	1	NC kernel
InitAdjPrgActive	Assumes the value 1 if InitAsup is active after SV, otherwise 0. To query in the program. Thus, general subroutines can be used which can respond differently in the Asup of the SV for example.	Byte_t	1	NC kernel
LookAheadPreRun	Assumes the value 1 if the block pre-run prepares blocks beyond the target block within the set LookAhead. To query in the part program. With this information, it is possible to prevent CPL components following behind the target block from being run through twice (during block pre-run and during the actual preparation after block pre-run).	Byte_t	1	NC kernel
TargPos	After the pre-run, the control writes into data array the target position of the approach block with regard to the current workpiece coordinate system of the (target block-1).	Double_t	8x8	NC kernel
TargBlockSrc	Target block After reaching the target block, the target block string is entered here. CPL parts are already integrated. If the target block is a CPL block, the following DIN block is entered.	IsoLatin1String	512	NC kernel

Element name	Description	Type	Size in bytes	Write access
LockTargBlock	This date is used to block areas in the part program for the selection as target block. When the lock is set (value = 1), the target block is accepted, but no NC start is assumed. A general warning is output. If there is an information text in the "LockWarning" date, it is output. It is possible to continue the block pre-run on another target block.	Int_t	4	Everyone
LockWarning	Note text for blocked target block If nothing is specified, the general warning "This target block is locked." is output in the language selected. The ID "%s" can be used in the text. It is replaced by the general warning.	IsoLatin1String	80	Everyone
NameOfAsup	Name of the subroutine (including path) to be called as adjustment program. If no name is specified, no adjustment program is called before the approach motion.	IsoLatin1String	100	Everyone
NameOfInitAdjPrg	Name of the Init adjustment program. If the path name of an Init adjustment program is specified, it is activated automatically.	isoLatin1String	100	Everyone
ActionBlock	Action block If an action block is specified, it is executed upon the machining start after block pre-run and before an adjustment program if defined.	IsoLatin1String	512	Everyone
Timeout	Period after which the block pre-run is aborted. The value is given in ms. The value "0" means that the runtime monitoring is deactivated. The runtime monitoring can be used if a target block was skipped in a program due to an unfulfilled condition and if the block pre-run is not automatically closed due to an endless loop in the program. Programs containing potential endless loops should set this system date to a sufficiently high value at the beginning and subsequently restore it. **Attention:** If the control load is very high, the block pre-run can be aborted due to exceeded runtime if the timeout value is only set a bit higher than the block pre-run runtime of the unloaded control. Set a sufficiently high value or completely disable monitoring.	Int_t	4	Everyone

Element name	Description	Type	Size in bytes	Write access
RetPaStrategy	0 - Approach the starting point (standard) 1 - Approach to end point 2 - Approaching the point of interruption The approaching motion is executed with regard to the system date on the starting point or end point of the target block or the point of interruption. The system date has to be specified before starting the adjustment cycle. It can be set in the block pre-run, but not anymore in the init adjustment program or in the adjustment program.	Int_t	4	Everyone
ManWaitForStart	Manual data input after block pre-run 0 - Deactivate manual data input 1 - Activate manual data input After the init adjustment program has been processed, NC blocks can be specified for manual adjustment after the manual data input is activated.	Int_t	4	Everyone

Tab. 5-15: System data

5.2.9 Interface signals

The following interface signals inform the PLC on the block pre-run state at the channel interface:

SrunAct	Block pre-run active This signal is output as long as the program is processed during "computation" up to the abort block (target block-1).
SrunReEnter	Re-entry active The signal is output as long as the action block or the adjustment program is processed.
SrunRepos	Return to contour active The signal is output as long as the approach block is active towards the target block.

Tab. 5-16: Interface signals

5.2.10 Look ahead

During block pre-run, NC functions can be active before the target block which requires a block preview (Look Ahead), e.g. milling cutter path correction G41/G42. The NC blocks after the target block are thus prepared until the last block before the target block (the interruption block) became active.

Within the Look Ahead function, CPL expressions are executed as well. Some values, e.g. of variables, can change.

When machining begins, these program passages are re-run. Before, the contents of the **CPL variables** are reset to their original state.

However, the contents of **permanent variables** and **system data** remain unchanged!

To exclude a passage from being run through in the block preview behind a target block, the part program has to be modified and the passages in ques-

tion have to be locked from a programming point of view by querying the system date "SD.SysSRun.LookAheadPreRun".

5.2.11 General information

Functional restrictions

Currently, the following functional restrictions apply:

- Only the Init adjustment program starts with the state before the program selection. The action block or the adjustment program jump to the state resulting from the target block. In the Init adjustment program, ensure that both states are adjusted (the most possible).
- No C-axis switching
- No block pre-run in tapping (G63): Lock in the TP
- No block pre-run into a spline sequence. However, the target block can be a switching block to G6
- No block pre-run into the center of a ramp interpolator sequence, but at the beginning of such a sequence
- No block pre-run on thread cutting G33 (also see G63).
- No 3D tool correction (G140 - G142)
- No fixed stop
- No axis change
- Axis coupling AXC

Entering subroutines

In subroutines called with the CPL variable, it is not possible to use the MTX block pre-run dialog.

Feed programming (per revolution) "G95"

G95 requires a rotating main spindle.

If a G95 has been enabled as last cycle in the computation but no spindle speed has been programmed, the feed of the channel axes is 0. A feed is required for the approaching motion of the axes at the end of a block pre-run. This feed cannot be generated via the "FeedVal" or "OptFeedVal" system data in this case.

The axes are not adjusted.



Select the target block to enable a spindle speed in the computation.

Manually adjust the axes in the manual input.

Adjust the axes using G77 in the TargetAsupPrg.

Relevant interface signals

iCh_SrunAct	Block pre-run active This signal is output as long as the program is processed during "computation" up to the abort block (target block-1).
iCh_SrunReEnter	Re-entry active The signal is output as long as the action block or the adjustment program is processed.
ICh_SrunRepos	Return to contour active The signal is output as long as the approach block is active towards the target block.

Tab. 5-17: Relevant interface signals

Relevant system data

SysSRun.Active	To query in the part program. Thus, it is possible to skip program passages during block pre-run, for example a tool change or table modifications.
SysSRun.AsupActive	Used for queries in the adjustment program. Thus, general subroutines can be used as adjustment program which respond differently in the Asup of the block pre-run for example.
SysSRun.InitAdjPrgActive	Used for queries in the adjustment program. Thus, general subroutines can be used as InitAsup which respond differently in the InitAsup of the block pre-run for example.
SysS-Run.LookAheadPreRun	To query in the part program. With this information, it is possible to prevent CPL components following the target block from being run through twice (during block pre-run and during the actual preparation after block pre-run).
SysSRun.TargPos	After the block pre-run, the control writes the target position of the approach block with regard to the current workpiece coordinate system of (target block -1) into this data array.
SysSRun.LockTargBlock	This date is used to block areas in the part program for the selection as target block.
SysSRun.LockWarning	Note text for blocked target block
SysSRun.NameOfAsup	Name of the subroutine (including path) to be called as adjustment program.
SysSRun.NameOfInitAdjPrg	Name of the subroutine (including path) to be called as init adjustment program.
SysSRun.ActionBlock	Action block

Tab. 5-18: Relevant system data

5.2.12 Examples

Axis configuration and coordinate configuration

The axes can be configured in the InitAdjPrg using the NC command TAX. An axis transformation is previously disabled and enabled again after the axes have been adjusted. The coordinates are configured using the adjustment command "COORD".

Axis and coordinate adjustment

Program:

```
; switch off transformers
G40 G48 COORD(0)
; generate axis configuration
TAX
; adjust transformer
ADJ(COORD)
ADJ(G47)
ADJ(G40)
```



The tool change and the spindle adjustment have to be taken into consideration, depending on the application.

Adjusting the spindle

The spindle adjustment has to be separated into different sections, depending on the application. The example gives a rough overview and suggests a possible adjustment sequence.

Adjust channel spindles and spindle groups

```
0000 DIM SADR$(8)  :REM For spindle syntax     (e.g., S1=)
0001 DIM GADR$(8)  :REM for gear          (e.g., M142)
;
N100  SPADJSP
```

Disable all non-adjusted C-axes

```
0200  FOR SP% = 1 TO SD.SysChSpSRun.MaxSpNumber
0201       IF SD.SysChSpSRun.Sp[SP%].CAxisNotAdj <> 0 THEN
N202      RAX([SD.SysChSpSRun.Sp[SP%].DriveNo])
N203      ATSW([SD.SysChSpSRun.Sp[SP%].DriveNo])
0204    ENDIF
0290  NEXT
```

Adjust spindles

```
0300  FOR SP% = 1 TO SD.SysChSpSRun.MaxSpNumber
0301    IF SD.SysChSpSRun.Sp[SP%].NotAdjusted <> 0 THEN
N302      SpAdjStr[SP%]
0303      SADR$ = SD.SysChSpSRun.SpeedStr
0305      GADR$ = SD.SysChSpSRun.GearStr
0306      MOVE% = SD.SysChSpSRun.Sp[SP%].Move
;
;         Gear
0310      IF SD.SysChSpSRun.Sp[SP%].GearNotAdj <> 0 THEN
0311        IF SD.SysChSpSRun.Sp[SP%].AutoGearShift THEN
0312          SYSSPNO% = SD.SysChSpSRun.Sp[SP%].SysSpNo
N313          SSPGEAR([SYSSPNO%],[SD.SysChSpSRun.Sp[SP%].Gear])
0314        ELSE
N315          [GADR$]
0316        ENDIF
0317      ENDIF
;
;         Speed limit
0320      IF SD.SysChSpSRun.Sp[SP%].LimitNotAdj <> 0 THEN
N321        SMX([SADR$][SD.SysChSpSRun.Sp[SP%].ProgMaxSpeed])
N322        SMN([SADR$][SD.SysChSpSRun.Sp[SP%].ProgMinSpeed])
0323      ENDIF
;
```

```
;        SERCOS operation mode
0330      IF SD.SysChSpSRun.Sp[SP%].OpModeNotAdj <> 0 THEN
0331        IF SD.SysChSpSRun.Sp[SP%].PosCtrlOn THEN
N332          SPM([SADR$]1)
0333        ELSE
N334          SPM([SADR$]0)
0335        ENDIF
0336      ENDIF
0337    ENDIF
0390  NEXT SP%
N400  ADJ(G96)
```

Specify S-values and if required AutoGear

```
0500  FOR SP% = 1 TO SD.SysChSpSRun.MaxSpNumber
0501    IF SD.SysChSpSRun.Sp[SP%].NotAdjusted <> 0 THEN
N502      SpAdjStr[SP%]
;
;        Speed or cutting velocity
0510      IF SD.SysChSpSRun.Sp[SP%].SpeedNotAdj <> 0 THEN
0511        SADR$ = SD.SysChSpSRun.SpeedStr
N512        [SADR$][SD.SysChSpSRun.Sp[SP%].ProgSpeed]
0513      ENDIF
;
;        AutoGear
0520      IF SD.SysChSpSRun.Sp[SP%].AutoGearShift THEN
0521        GADR$ = SD.SysChSpSRun.GearStr
N522        [GADR$]
0523      ENDIF
0524    ENDIF
0590  NEXT SP%
```

☞ If a spindle coupling is available, it has to be adjusted, too. A possible sequence is given in the following example.

Switch on all required C-axes asynchronously

```
0700  FOR SP% = 1 TO SD.SysChSpSRun.MaxSpNumber
0701    IF SD.SysChSpSRun.Sp[SP%].CAxisNotAdj AND 2 = 2 THEN
0702      IF SD.SysChSpSRun.Sp[SP%].CAxisType AND 32 = 32 THEN
0703        SETERR("Block pre-run cannot be executed in a G63 sequence!")
0704      ENDIF
0705      IF SD.SysChSpSRun.Sp[SP%].CAxisType AND 16 = 16 THEN
0706        SETWARN("Block pre-run in turret operation! Intentionally?")
N707        M0   ; Turret operation
N708        STAW([SD.SysChSpSRun.Sp[SP%].DriveNo])
0709      ELSE
N710        STAW([SD.SysChSpSRun.Sp[SP%].DriveNo])
0711      ENDIF
0712    ENDIF
0760  NEXT SP%
```

Adjustment successful

```
0800  WAIT
0801  IF SD.SysChSpSRun.SpConfNotAdjusted <> 0 THEN
0802    SETERR("Incorrect channel spindles.")
0803  ENDIF
0811  FOR SP% = 1 TO SD.SysChSpSRun.MaxSpNumber
0812    IF SD.SysChSpSRun.Sp[SP%].NotAdjusted <> 0 THEN
0813      IF SD.SysChSpSRun.Sp[SP%].GearNotAdj <> 0 THEN
0814        SETERR("Incorrect gear.")
0815      ENDIF
0816      IF SD.SysChSpSRun.Sp[SP%].LimitNotAdj <> 0 THEN
0817        SETERR("Incorrect speed limit.")
0815      ENDIF
0819      IF SD.SysChSpSRun.Sp[SP%].CAxisNotAdj <> 0 THEN
0820        SETERR("Incorrect C-axes.")
0821      ENDIF
0822      IF SD.SysChSpSRun.Sp[SP%].OpModeNotAdj <> 0 THEN
0823        SETERR("Incorrect drive operating mode.")
0824      ENDIF
0825      IF SD.SysChSpSRun.Sp[SP%].CpDataNotAdj <> 0 THEN
0826        SETERR("Incorrect coupling data.")
0827      ENDIF
0828      IF SD.SysChSpSRun.Sp[SP%].CoupleNotAdj <> 0 THEN
0829        SETERR("Incorrect coupling state.")
0830      ENDIF
0831      IF SD.SysChSpSRun.Sp[SP%].SpeedNotAdj <> 0 THEN
```

```
0832        SETERR("Incorrect S-value.")
0833      ENDIF
0834    ENDIF
0835  NEXT SP%
```

The spindles are only started at the end of the adjustment

Adjust speed and motion function

```
0000 DIM SADR$(8)  :REM for spindle syntax     (z.B. S1=)
0001 DIM MADR$(8)  :REM for motion function (e.g. M103)
;
;
;   Specify speed and direction of travel
0100  FOR SP% = 1 TO SD.SysChSpSRun.MaxSpNumber
0101    IF SD.SysChSpSRun.Sp[SP%].NotAdjusted <> 0 THEN
N102      SpAdjStr[SP%] ; load syntax elements of current spindle
0103      SADR$ = SD.SysChSpSRun.SpeedStr
0104      MADR$ = SD.SysChSpSRun.MoveStr
0105      MOVE% = SD.SysChSpSRun.Sp[SP%].Move
;
;        Speed or cutting velocity
0110      IF SD.SysChSpSRun.Sp[SP%].SpeedNotAdj <> 0 THEN
N111        [SADR$][SD.SysChSpSRun.Sp[SP%].ProgSpeed]
0112      ENDIF
;
;        Motion function
0120      IF SD.SysChSpSRun.Sp[SP%].MoveNotAdj <> 0 THEN
0121        IF MOVE% = 19 THEN
N122          [MADR$][SADR$][SD.SysChSpSRun.Sp[SP%].OriPos]
0123        ELSE
N124          [MADR$]
0125        ENDIF
0126      ENDIF
0180    ENDIF
0190  NEXT SP%
;
;  Adjustment successful?
0200 WAIT
0201  FOR SP% = 1 TO SD.SysChSpSRun.MaxSpNumber
0202    IF SD.SysChSpSRun.Sp[SP%].NotAdjusted <> 0 THEN
0203      SETERR("Spindle adjustment failed.")
0204    ENDIF
0205  NEXT SP%
```



If a TargetAdjustPrg becomes active, the speed and the motion function should be adjusted in this adjustment program.

Spindle coupling

Different steps have to be executed during the adjustment in a spindle coupling. A rough classification with an example explains the connection.

- Adjust coupling data of spindles
- Adjust coupling state
- Evaluating master and slave
- Enable new coupling or disable old coupling
- Position phase offset

⚠ CAUTION **The spindles are moving.**

Ensure a basic level of safety.

Adjust spindle coupling

```
0000 DIM SADR$(8)  :REM For spindle syntax     (e.g. S1=)
;
; Adjust coupling data of spindles
0100  FOR SP% = 1 TO SD.SysChSpSRun.MaxSpNumber
0101    IF SD.SysChSpSRun.Sp[SP%].CpDataNotAdj <> 0 THEN
N102      SpAdjStr[SP%]
```

```
0103      SADR$ = SD.SysChSpSRun.SpeedStr
N104      SPCS([SADR$][SD.SysChSpSRun.Sp[SP%].SyncWindow])
N105      SPCE([SADR$][SD.SysChSpSRun.Sp[SP%].SyncErrorWindow])
N106      SPCD([SADR$][SD.SysChSpSRun.Sp[SP%].CoupleDistance])
0107    ENDIF
0108  NEXT
;
; Adjust coupling state
0200  FOR CP% = 1 TO SD.SysChSpSRun.PrgCpNumber
0201    IF SD.SysChSpSRun.Cp[CP%].NotAdjusted <> 0 THEN
0202      NADJ% = SD.SysChSpSRun.Cp[CP%].NotAdjusted
0203      MAST% = SD.SysChSpSRun.Cp[CP%].Master
0300      IF NADJ% = 1 THEN
;       Incorrect master, delete coupling and generate new coupling
0301        IF SD.SysChSpSRun.Cp[CP%].CoupleOn THEN
N302          SPCC(CP=[CP%],MA=0)
N303          SPC_WAIT(CP=[CP%])
0304          INDEX% = 1
0305          SP% = SD.SysChSpSRun.Cp[CP%].Slave[INDEX%]
0310          WHILE SP% <> 0 DO
N311            SpAdjStr[SP%]
0312            SADR$ = SD.SysChSpSRun.SpeedStr
N313            SPCC(CP=[CP%],MA=[MAST%],[SADR$]1)
0314            INDEX% = INDEX% + 1
0315            IF INDEX% <= 8 THEN
0316              SP% = SD.SysChSpSRun.Cp[CP%].Slave[INDEX%]
0317            ELSE
0318              SP% = 0
0319            ENDIF
0320          END
N330          SPC_WAIT(CP=[CP%])
0331        ELSE
0332          SETERR("MTX error: Cp[].NotAdjusted=1 und CoupleOn=FALSE")
0333        ENDIF
0334      ENDIF
0400      IF NADJ% = 2 THEN
;       Incorrect slaves
0401        IF SD.SysChSpSRun.Cp[CP%].CoupleOn THEN
0410          FOR SP% = 1 TO SD.SysChSpSRun.MaxSpNumber
0411            CPNOTADJ% = SD.SysChSpSRun.Sp[SP%].CoupleNotAdj
0412            IF (CPNOTADJ% = 1) OR (CPNOTADJ% = 4) THEN
N413              SpAdjStr[SP%] ;Load syntax elements of the current spindle
0414              SADR$ = SD.SysChSpSRun.SpeedStr
N415              SPCC(CP=[CP%],MA=[MAST%],[SADR$]0)
0416            ENDIF
0417          NEXT
N418          SPC_WAIT(CP=[CP%])
0420          FOR SP% = 1 TO SD.SysChSpSRun.MaxSpNumber
0421            CPNOTADJ% = SD.SysChSpSRun.Sp[SP%].CoupleNotAdj
0422            IF CPNOTADJ% = 2 THEN
N423              SpAdjStr[SP%] ;Load syntax elements of the current spindle
0424              SADR$ = SD.SysChSpSRun.SpeedStr
N425              SPCC(CP=[CP%],MA=[MAST%],[SADR$]1)
0426            ENDIF
0427          NEXT
N428          SPC_WAIT(CP=[CP%])
0430        ELSE
0431          SETERR("MTX error: Cp[].NotAdjusted=2 und CoupleOn=FALSE")
0432        ENDIF
0433      ENDIF
0500      IF NADJ% = 3 THEN
;         New coupling
0501        IF SD.SysChSpSRun.Cp[CP%].CoupleOn THEN
0502          INDEX% = 1
0503          SP% = SD.SysChSpSRun.Cp[CP%].Slave[INDEX%]
0504          WHILE SP% <> 0 DO
N505            SpAdjStr[SP%]
0506            SADR$ = SD.SysChSpSRun.SpeedStr
N507            SPCC(CP=[CP%],MA=[MAST%],[SADR$]1)
0508            INDEX% = INDEX% + 1
0509            IF INDEX% <= 8 THEN
0510              SP% = SD.SysChSpSRun.Cp[CP%].Slave[INDEX%]
0511            ELSE
0512              SP% = 0
0513            ENDIF
0514          END
0515        ELSE
;           Resolve coupling
N516          SPCC(CP=[CP%],MA=0)
0517        ENDIF
N518        SPC_WAIT(CP=[CP%])
0519      ENDIF
;
```

```
; If required, position the phase offset of coupling
; Do not confuse with coupling distance
0700      IF SD.SysChSpSRun.Cp[CP%].CoupleOn THEN
0701        FOR SP% = 1 TO SD.SysChSpSRun.MaxSpNumber
;            Is the phase offset correct?
0702          IF SD.SysChSpSRun.Sp[SP%].CoupleNotAdj AND 16 = 16 THEN
;               Adjust phase offset and speed
N703            SpAdjStr[SP%]  ;Load syntax elements of the current spindle
0704            SADR$ = SD.SysChSpSRun.SpeedStr
0705            SPEED = SD.SysChSpSRun.Sp[SP%].CouplePosOffsetSpeed
N706            SPCP([SADR$][SD.SysChSpSRun.Sp[SP%].CouplePosOffset],POSVEL[SPEED])
0707          ELSE
0708            IF SD.SysChSpSRun.Sp[SP%].CoupleNotAdj = 8 THEN
;                 Adjust phase offset
N709              SpAdjStr[SP%] ;Load syntax elements of the current spindle
0710              SADR$ = SD.SysChSpSRun.SpeedStr
N711              SPCP([SADR$][SD.SysChSpSRun.Sp[SP%].CouplePosOffset])
0712            ENDIF
0713          ENDIF
0714        NEXT
N715        SPCP_WAIT(CP=[CP%])
0716      ENDIF
0800    ENDIF
0990  NEXT
```

Tools

Depending on the application, processing the tool change in the block pre-run can be structured very differently. Via the system data, it can be differentiated between block pre-run and program processing in the tool change program.

Query in the tool change program

```
100 IF SD.SysChSRun[1].Active=1 THEN
110   ...
120   ...
130 ENDIF
```

The tool change is not actually executed during the block pre-run. Some data information has to be provided for the computation in the block pre-run. Amongst others, the DCT corrections are affected for the path correction. The last active tool can be saved in a system date. The E- and D-corrections are automatically archived in the system data.

Tool data in block pre-run

```
300 FOR I% = 1 TO EDGE%
350   SD.EdGeo  = SD.ToolPre.UD.Ed[I%].Geo
360   SD.EdLife = SD.ToolPre.UD.Ed[I%].Life
370   SD.EdWear = SD.ToolPre.UD.Ed[I%].Wear
380   SD.EdOffset = SD.ToolPre.UD.Ed[I%].Offset
410   DCT(1,I%,0)=SD.EdGeo.L1 + SD.EdWear.L1 + SD.EdOffset.L1
420   DCT(2,I%,0)=SD.EdGeo.L2 + SD.EdWear.L2 + SD.EdOffset.L2
430   DCT(3,I%,0)=SD.EdGeo.L3 + SD.EdWear.L3 + SD.EdOffset.L3
440   DCT(4,I%,0)=SD.EdGeo.Rad + SD.EdWear.Rad + SD.EdOffset.Rad
450   DCT(5,I%,0)=SD.EdGeo.Ori
460 NEXT I%
810 SD.SysToolSRun[1].ActTool.K1=1
810 SD.SysToolSRun[1].ActTool.K2=SD.ToolPre.Hd.IKQ3
```

The tool change can be executed in the InitAdjPrg.

Tool change

```
N10 M6 T[SD.SysToolSRun[1].ActTool.K2]
100 IF SD.SysToolSRun[1].TargDNr = 0 THEN
N50   D0
110 ELSE
N60   D[SD.SysToolSRun[1].TargDNr]
120 ENDIF
150 IF SD.SysToolSRun[1].TargEd = 0 THEN
N80   ED0
160 ELSE
N90   ED[SD.SysToolSRun[1].TargEd]
170 ENDIF
```

NC functions

A part of the modal NC functions can be adjusted via the adjust command. Some modal NC function are also considered in the approaching block, see above.

Adjusting NC functions

```
...
ADJ(G47)
ADJ(G1)
ADJ(G61)
ADJ(G8)
ADJ(RAD)
ADJ(G53.1)
ADJ(G94)
ADJ(G96)
ADJ(G16)
ADJ(G40)
...
```

Auxiliary functions

Auxiliary functions are suppressed in the computation. Auxiliary functions such as e.g. coolant can be executed in the TargetAdjustPrg.

The "/SysChSRun/AuxNotAdjusted" system date specifies that not all modal auxiliary functions have been adjusted.

Individual axis adjustment with G77

All axes to be adjusted are traversed simultaneously during the approaching motion. However, this is not required for some machine types. The axes have to be approached depending on the type of machining.

At a lathe, the z-axis can only be traversed for e.g. machining of internal diameters if the x-axis has already been adjusted.

In TargetAdjustPrg, individual axes can be adjusted using the NC command G77. In the following example, a differentiation of the adjustment sequence is shown. Depending on a PLC variable that can e.g. be set via M-keys, the adjustment sequence is influenced in the TargetAdjustPrg.

Axis adjustment with G77 in the TargetAdjustPrg

```
WAIT
1     CASE @MODEG77% OF
1     LABEL 1
1         PRN#(0,"Mode 1: first X then Z")
          G77 X1 F2000
          G77 Z1 F2000
1     LABEL 2
1         PRN#(0,"Mode 2: first Z then X")
          G77 Z1 F2000
          G77 X1 F2000
1     LABEL 3
1         PRN#(0,"Mode 3: X and Z together")
          G77 X1 Z1 F2000
1     OTHERWISE
1         SETERR("Invalid startup mode G77")
1     ENDCASE
WAIT
```

5.3 Block switching via high-speed signal

5.3.1 Description

Function

The block switching via "high-speed signal" (HsBlkSwitch/HSB) function enables a "premature" block switching in the NC via the high-speed signal input or a customer input of the channel interface to be triggered.

Thus, external control or monitoring devices can abort an active linear traversing motion. Afterwards, the control continues machining with the next programmed block.

The function permits:

- Block switching "on-the-fly":

  Programmed end points of axes are not deleted.

  If these axes are not programmed with new end points in the subsequent block, the control uses their end points from the prematurely completed block and includes these axes in the interpolation.

  The current velocity in the prematurely terminated block is not influenced at block switching (see chapter "Restrictions" on page 74 for exceptions). If a change in velocity is required, the NC initiates it only in the subsequent block, but without checking the axis jump capacity.

- Block switching with "Cancel distance to go":

  Programmed end points of axes are deleted.

  If these axes are not specifically programmed again in the subsequent block, they are not traversed further.

  The current path velocity of the prematurely completed block is always reduced to v=0 at block switching. Use the parameter HSSTOP to select either a down slope with the current acceleration or a velocity jump to v=0.

Block-switching "on-the-fly" can, for example, be used if it is to be traversed with different feeds along a straight line depending on an external event.

Block switching with "Cancel distance to go" is suitable for example, for applications in which a pressure or torque-controlled "move to fixed stop" is required.

See section HsBlkSwitch (HSB) of the "MTX Programming Manual" for examples.

NOTICE	**This function changes the programmed contour! Risk of damage to the workpiece and/or the tool!**

Only use the function when the tool is no longer active or when damage to the workpiece and/or the tool can be excluded!

Restrictions

- A linear traversing motion has to be programmed in the prematurely completed block and in the subsequent block (while G0 or G1 is active).
- Interface for high-speed signal coupling required ("High-speed I/O").
- If the block switching is activated via a high-speed input, it has to be applied in **PLC/DigIO/FuncAllocIn/DigIOAllocIn[1..8]** (MP 4075 00101) with the "HsBlkSwitch/HSB" functionality.

- *Block switching "on-the-fly" in the "continuous block" mode takes place at axis standstill if*
  - there is a > 90 degree contour bend between the block to be aborted and the subsequent block
  - the HsBlkSwitch block always ends with v=0 (with G0 active) due to the currently effective type of interpolation
  - the "Exact stop" function is active (G61)
  - the subsequent block starts with v=0 by reason of additional programming (e.g. KvProg/KVP, FeedForward/FFW or WriteId/WID)
  - the "SHAPE" function is active (see under G8 or G9).
- Velocity in the "single block" and "single step" modes is decelerated to v=0 despite the programmed "on-the-fly" block switching, as the control processes only one single program block per NC start.
- In the operation modes, "program block" and "manual data input" is decelerated to v=0 despite the programmed "on-the fly" block switching **and the distance to go** is deleted! Thus, a subsequent NC block with incremental programming does not result to the previously programmed end point!

Relevant NC functions

HsBlkSwitch/HSB	"Activate block switching via high-speed signal" for the duration of the HsBlkSwitch block. The triggering conditions and the selection of the digital input can be set via the **parameter HS**:	
	HSx=0:	Low level (x: input number)
	HSx=1:	High level
	-or-	
	Can be set via CI parameter:	
	CIx=0:	Low level (x: input number)
	CIx=1:	High level
	Parameter HSSTOP affects the mode of action:	
	Without HSSTOP:	Block switching "on-the-fly". No change in velocity during block switching.
	HSSTOP=0:	Block switching with "Cancel distance to go" and down slope to v=0.
	HSSTOP=-1:	Block switching with "Cancel distance to go" and velocity jump to v=0.
	If the level of the used high-speed input does not change during the processing time for the "HsBlkSwitch" block, the block is traversed to the end position according to its programming (with or without the "HSSTOP" parameter). Subsequently, the NC processes the next program block.	

Tab. 5-19: Relevant INC functions

Relevant IF signals

HSI-0	Fast digital input, Can be set via MP DigIOAllocIn[1..8] (4075 00101). Trigger block switching, e.g. at HSB(HS1=1). The input is queried in the interpolator cycle.
qCh_Customx	Customer input x Trigger block switching, e.g. at HSB(CI1=1)

Tab. 5-20: Relevant IF signals

Relevant machine parameters (MP)

PLC/DigIO/ FuncAllocIn/ DigIOAllocIn[1..8] (4075 00101)	Assignment of digital inputs Specifies which high-speed digital input is assigned to the "HsBlkSwitch/HSB" function.

Tab. 5-21: Relevant machine parameters

Example:

Block switching "on-the-fly"

The resulting traversing motion to the end point (X50/Y20) can change depending on the signal changes at the high-speed entry, but the entry remains unchanged.

During block switching, the programmed velocity remains.

N05G1F1000 X0 Y5	End point: X0 Y5
N10HSB(HS1=1) **G90**X50	End point: X50 Y5
N20HSB(HS1=0) **G91**Y10	End point: X50 Y15 (abs. Y5+incr. Y10)
N30 **G91**Y5	End point: X50 Y20 (abs. Y15+incr. Y5)

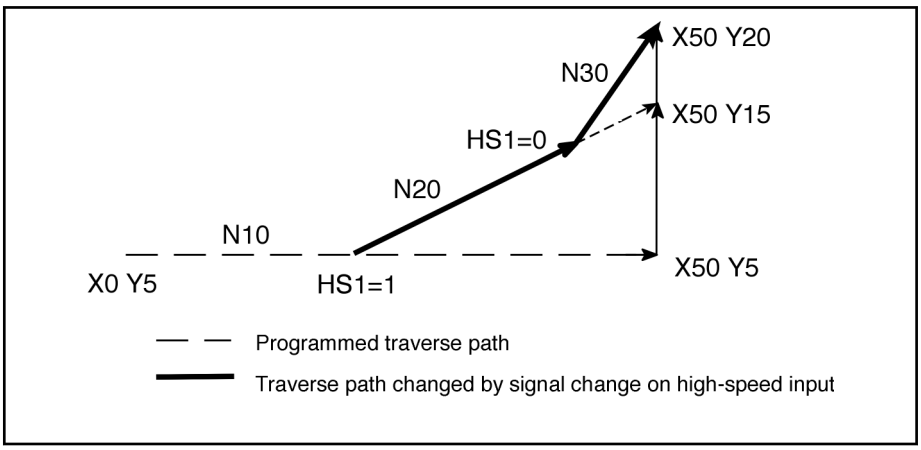


Fig. 5-12: Example: Block switching "on-the-fly"

Example:

Block switching with "cancel distance to go"

The programmed end point (X50/Y20) only remains active if the signal at the high-speed input does not change. In all other cases, the end point depends on the signal change at the high-speed input.

During block switching, velocity is decelerated to v=0 using a down slope.

N05G1F1000 X0 Y5	End point: X0 Y5
N10HSB(HS1=1) HSSTOP=0 **G90**X50	End point: External event for Xp50
N20HSB(HS1=0)HSSTOP=0 **G91**Y10	End point: External event for Yp10
N30 **G91**Y5	End point: Incrementally "removed" by Y5 from the last external event.

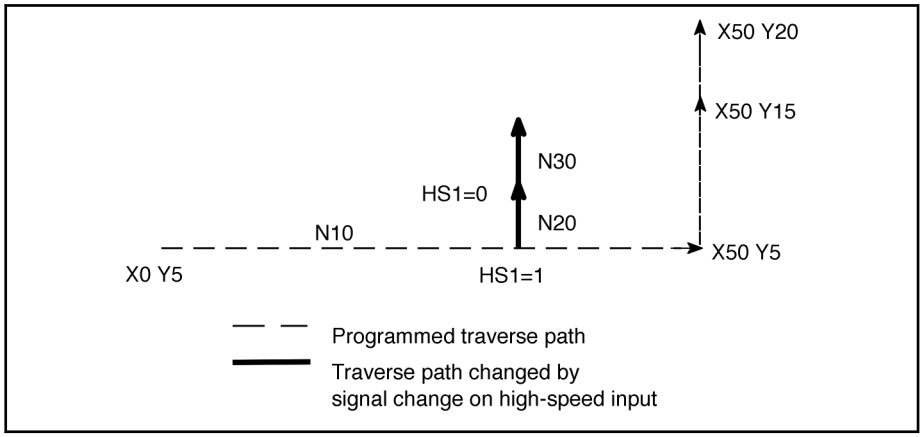


Fig. 5-13: Example - Block switching with "Cancel distance to go"

5.3.2 Handling instruction: Block Switching via High-Speed Signal with Cancellation of the Distance to Go

Applying

NOTICE **Workpiece or machine can be damaged!**

Do not reset the control during machining.

1. Ensure that all high-speed inputs to be used in connection with the "Block switching via high-speed signal" function are assigned with the value "0003" in the machine parameter 4075 00101.
2. If the configuration parameter was changed, trigger control reset.

Activating

Prerequisite:

A linear type of interpolation is active (G0 or G1).

Program "HsBlkSwitch/HSB" in connection with the parameters HS or CL and - if applicable - HSSTOP (see "Relevant NC functions").

The function is active until the block switching is triggered, but not longer than for the duration of the "HsBlkSwitch" block. Afterwards, it is deactivated automatically unless a new "HsBlkSwitch block" follows immediately.

Use the "Block switching via high-speed signal" function (HsBlkSwitch/HSB) to trigger a "premature" block switching in the NC via the high-speed signal input or a customer input of the channel interface.

IW Engineering/configuration: Parameterizing Digital I/O

To use the "Block switching via high-speed signal" function, assign the value "**Block switching via HS signal**" to the digital input using the parameter **DigIOAllocIn** "Assignment of the digital inputs [1..8]" (4075 00101).

- Set the hardware used for digital I/O via the parameter **DigIOHw** "Hardware selection" (4075 00200)
- Enter the starting address for digital I/Os into the parameter **DigIOStartAddrIn** "Starting address of the digital inputs" (4075 00105).

	Documentation	Instruction
Instruction:		Editing machine parameters
Documentation:	MTX Machine Parameters	Parameterizing digital I/Os

IW Operation/NC programming:

Program NC block (cancel distance to go)

Program the NC function **HsBlkSwitch(HS<x>=<y>,HSSTOP=<z>)** or **HSB(HS<x>=<y>,HSSTOP=<z>)** "Block switching with abort".

<x>: Number of the high-speed signal (1..8)

<y>: Logical signal state required for block switching (0 or 1)

<z>: Deceleration type if the event occurs

(**"0"** = Ramp down to v=0 with max. deceleration and

"1" = Velocity jump to V=0)

- The function requires a linear traversing motion (G0 or G1) in the "HsBlkSwitch" block and in its subsequent block.
- The function has to be written with path information. It can be written together with other path conditions.
- Exact stop is active, i.e. G0(IPS...) or G1(IPS...).
- Interface for high-speed signal coupling required ("High-speed I/O").

	Documentation	Example
Example:		Example of block switching on-the-fly
Documentation:	MTX Programming Manual	Programming block switching on-the-fly with cancel distance to go

Digital interface: Triggering block switching via digital input

Set the corresponding logic signal (0 or 1) at the digital input for the type of block switching.

It is possible to assign a digital input to an interface signal **qAx_Custom1 ... 8** "Customer input 1...8".

	Documentation	Instruction
Instruction:		Editing PLC Signals
Documentation:	MTX PLC Interface	Triggering Block Switching

5.3.3 Handling instruction: Block Switching via High-Speed Signal with On-the-Fly Block Switching

Use the "Block switching via high-speed signal" function (HsBlkSwitch/HSB) to trigger a "premature" block switching in the NC via the high-speed signal input or a customer input of the channel interface.

IW Engineering/configuration: Parameterizing Digital I/O

To use the "Block switching via high-speed signal" function, assign the value **"Block switching via HS signal"** to the digital input using the parameter **DigIOAllocIn** "Assignment of the digital inputs [1..8]" (4075 00101).



- Set the hardware used for digital I/O via the parameter DigIOHw "Hardware selection" (4075 00200)
- Enter the starting address for digital I/O into the parameter DigIOStartAddrIn "Starting address of digital inputs" (4075 00105).

	Documentation	Instruction
Instruction:		Editing machine parameters
Documentation:	MTX Parameter Description	Parameterizing digital I/Os

IW Operation/NC programming: Programming NC Block (on-the-fly block switching)

Program the NC function **HsBlkSwitch(HS<x>=<y>)** or **HSB(HS<x>=<y>)** "Block switching on-the-fly".

<x>: Number of the high-speed signal (1..8)

<y>: Logical signal state required for block switching (0 or 1)



- The function requires a linear traversing motion (G0 or G1) in the "HsBlkSwitch" block and in its subsequent block.
- The function has to be written with path information. It can be written together with other path conditions.
- Exact stop is active, i.e. G0(IPS...) or G1(IPS...).
- Interface for high-speed signal coupling required ("High-speed I/O").

	Documentation	Example
Example:		Example: Block switching on-the-fly
Documentation:	MTX Programming Manual	Programming block switching on-the-fly

Digital interface: Triggering block switching via digital input

Set the corresponding logic signal (0 or 1) at the digital input for the type of block switching.



It is possible to assign a digital input to an interface signal **qAx_Custom1 ... 8** "Customer input 1...8".

	Documentation	Instruction
Instruction:		Editing PLC Signals
Documentation:	MTX PLC Interface	Triggering Block Switching

5.4 Continuing Program with Block Pre-Run after Program Abort

5.4.1 Description

Function

If an NC program is completed during processing, the program can be continued at the last predefined point. This point is specified in the NC program by programming the NC function "RCB". After the interpolator processed this block, this point can be used as target block in the block pre-run.

If several points to return to contour with RCB are programmed in the NC program, the latest successfully processed point is used as specification for the block pre-run. It is possible to delete the latest stored point using a certain parameter value. Thus, the block pre-run to the previously stored point is not possible after processing this block.

Optionally, up to three CPL conditions can be transferred to the RCB function. These conditions are analyzed at the point in time the block is prepared and stored internally at the point in time the block is processed. The block pre-run uses the same conditions to decide whether the target block was found. In contrast to the normal block pre-run with manual specification of the target block, it is possible to skip successfully processed program passages or to allow finding the target block under certain prerequisites.

Restrictions

The conditions are analyzed when the block is prepared or at block pre-run. The block preparation has to be synchronized with the IPO using the WAIT function before programming the RCB. Thus, the state of the CPL variables is consistent. This also applies for the block pre-run. Possible external signals are already to be applied properly before starting the block pre-run, as the time of analysis of RCB only depends on the processing velocity of the control system.

The directly activating programming of RCB is not possible in manual input mode. Nevertheless, an activating RCB can be used in a program called in manual input mode.

Only one point per channel can be active at a time that can be used as target block in the block pre-run.

Calling activating RCB functions in ASUPs is permitted, but does not have any effects.

When using conditions, the program has to have been linked.

Relevant NC functions

RCB(1, {Cond. 1}, {Cond. 2}, {Cond. 3}) or **ReentContBlk(1, {Cond. 1}, {Cond. 2}, {Cond. 3})**	Highlights a point the block search is able to find. Conditions 1 to 3 are optional. The validity of a point possibly stored before is overwritten.
RCB(0) or **ReentContBlk(0)**	Deletes the point highlighted last. A block pre-run to the last activating point is no longer possible.

Tab. 5-22: Relevant NC functions

5.4.2 Activating

Program **RCB(1)** (without condition) or **RCB(1, Cond. 1{, Cond. 2{, Cond. 3}}})** (with up to three optional conditions)

5.4.3 Deactivating

Programming **RCB(0)**.

5.4.4 Handling instructions

If the program was completed by a deactivating RCB programming after processing an activating RCB programming, IndraWorks Operation can trigger the block pre-run using the latest point stored.

Thus, select the canceled main program and start the block pre-run using the internally stored point. If the target block is found, the program can be continued at this point via NC start. A possible error message indicates that either the target block was not found or the target block was found, but one of the conditions specified was not met.

Remark 1: Target block not found:

Only possible if the latest processed RCB was programmed in a subroutine not called in the block pre-run.

Remark 2: Target block found, but one condition was not met:

All specified conditions are checked in the block pre-run and whether the state in the block pre-run is identical to the state which previously existed during normal processing. If this is the case due to modified input signals or variable assignments, the initial state of the variables has to be restored. This allows the block pre-run to find and stop the target block.

6 Dynamics and Velocity Control

6.1 Scaling Rotary Axis Velocities

6.1.1 Configuring Scaling Factors for Rotary Axis Velocities

Description

Function

This function ensures that the control specifies the velocities of synchronous rotary and endless axes correctly with metric as well as with inch-based programs.

Accordingly, any existing "country-specific" part programs can be processed without any modification.

By scaling the rotary axes velocity, it is determined how the path specification of a rotary/endless axis in degree is to be scaled with regard to a linear axis with a path specification in mm or inch in terms of velocity.

Depending on G70/G71, set via machine parameters whether one degree angle specification is to be handled as path specification of 1 mm or 1 inch. Depending on the setting, a different motion period results even though the programmed path velocity is equal.

Individual settings are required for G70 (programming in inch) and G71 (programming in metric units) to specify for the velocity computation whether one degree is to be handled as 1 mm or like 1 inch.

1. European setting: [1]: 1

[2]: 1

- Both with metric and with inch programming, one degree always corresponds to 1 mm. If the rotary axis is alone programmed for G71, the feed is interpreted in degree/min.
- For G70 and exclusive rotary axis programming, the rotary axis traversed to rapidly by a factor or 25.4 (feed value interpreted in degree/min.

  Example:

  F100 = 100 inch/min = 2540 mm/min=2540 degree/min).
- If the rotary axis and linear axis move together, both are scaled equally at G71.

  Example:

  The linear axis moves by 100 mm, the rotary axis by 100 degrees. Both axes move at a velocity of 0.7071* programmed feed in [mm/min] or [degree/min], since both distances to be traversed are assumed to be of equal length.
- If the linear axis travels by 100 inch and the rotary axis by 100 degree for G70, the linear axis travels at almost 100% of the programmed velocity, since the distance traveled by the linear axis is assumed to be 25.4 times longer than the distance to be traveled by the rotary axis.

2. American setting: [1]: 2

[2]: 2

- For metric as well as inch programming, one degree always corresponds to one inch. If only the rotary axis is programmed in G70, the feed is interpreted in degree/min.

- With G71 and exclusive rotary axis programming, the rotary axis traverses too slowly by a factor of 25.4 (feed value not interpreted in degree/ min.

  Example:

  F100 = 100 mm/min = 3.937 inch/min = 3.397 degree/min).
- If the rotary axis and linear axis move together, both are scaled equally at G70.

  Example:

  The linear axis moves by 100 inch, the rotary axis by 100 degree. Both these axes move with 0.7071* programmed feed in [inch/min] or [degree/min], since both distances to be traveled are assumed to be of equal length.
- If the linear axis travels by 100 mm and the rotary axis by 100 degree in G71, the rotary axis travels at almost 100% of the programmed velocity, since the distance traveled by the rotary axis is assumed to be 25.4 times longer than the distance to be traveled by the linear axis.

3. General setting

[1]: 2

[2]: 1

- For G70 and for G71, the feed is interpreted in degree/min if only the rotary axis is programmed.
- If the rotary axis and the linear axis move together, both are scaled equally at G70 and G71.

Example:

G70

The linear axis travels by 100 inch, the rotary axis travels by 100 degree. Both these axis traverse at 0.7071* programmed feed in [inch/min] or [degree/min].

Example:

G71

The linear axis travels by 100 mm, the rotary axis travels by 100 degree. Both these axes traverse at 0.7071*programmed feed in [mm/min] or [degree/min] respectively.

Significance of parameter values:

1	1 degree is scaled like 1 mm
2	1 degree is scaled like 1 inch

Significance of the individual parameters:

[1]	Scaling for G70
[2]	Scaling for G71

Restrictions

- The FeedAd (FAD) function must not be active (⇒ all synchronous axes of a channel are considered to compute the path feed).

  To ensure that FeedAd is deselected after startup and after reset, include "FAD()" in MP 7060 00020 (reset).
- This function only applies for synchronous axes of "rotary" or "endless" type of axis motion (⇒ MP 1003 00004).
- The function only refers to velocities, not to positions.

Relevant NC Functions

FeedAd(...)	Considers all channel axes for feed calculation

Tab. 6-1: Relevant NC Functions

Relevant Machine Parameters (MP)

7040 00110	Scaling the rotary axis velocity at G70/G71.

Tab. 6-2: Relevant Machine Parameters

Handling Instruction: Scaling of Rotary Axis Velocities

Applying

NOTICE **Danger of damages to the workpiece or the machine!**

Do not reset the system during machining.

1. Adjust MP 7040 00110

   The following applies:

   - The control is operated at a site where only metric part programs are used (G71 is active; e.g. in continental Europe):

     7040 00110 [1]: 1

     7040 00110 [2]: 1

   - The control is operated at a site where only inch-based part programs are used (G70 is active; e.g. in the USA):

     7040 00110 [1]: 2

     7040 00110 [2]: 2

   - The control is operated at a site where metric as well as inch-based part programs are used:

     7040 00110 [1]: 2

     7040 00110 [2]: 1

2. Trigger system reset

Enables the use of metric and non-metric part programs for rotary axes.

IW Engineering/Configuration: Adapt rotary axis velocity

- Parameter **FactRotInch** "Scaling factor for non-metric rotary axis velocity (7040 00110[1])" and **FactRotMetr** "Scaling factor for metric rotary axis velocity (7040 00110[2])"
  - G70 active
    - **RotInch** (7040 00110[1]): 1 degree = 1 inch
    - **RotMetr** (7040 00110[2]): 1 degree = 1 inch
  - G71 active
    - **RotInch** (7040 00110[1]): 1 degree = 1 mm
    - **RotMetr** (7040 00110[2]): 1 degree = 1mm
  - For use of metric and inch-related part programs
    - **RotInch** (7040 00110[1]): 1 degree = 1 inch
    - **RotMetr** (7040 00110[2]): 1 degree = 1 mm

☞ The NC function **FeedAd (..)** must not be active.

6.1.2 Programming Scaling Factors for Rotary Axis Velocities

Description

Function

The feed is computed based on the calculated (virtual) path length **L** of the programmed blocks. The traveled velocity of an axis **va** results as a part of the path length **La** divided by the total path length **L** multiplied by the programmed feed **v**.

va = La / L * v

If the path length is calculated for a motion, the MTX uses the geometrical distance (root of (A1² + A2² + A3² …)). This is based on the implicit assumption that linear axes and rotary axes span a rectangular coordinate system.

Example:

Calculating feed of linear axes

Axis X travels [300] mm, axis Y travels 400 [mm], programmed feed 2000 [mm/min]

Path length L = root of (300²+400²) = 500 [mm]

Axis X: va = 300/500*2000 = 1200[mm/min]

Axis Y: va = 400/500*2000 = 1600[mm/min]

Rotary axes, rotary coordinates and orientation are programmed in degree. 1 degree = 1 mm is set to calculate the path length. Geometrically speaking, this corresponds to an effective radius of 57.3 mm. Hence, if a workpiece is mounted at a distance of 57.3 mm from the rotary axis of a rotary table, it moves exactly on the circular path with the programmed path feed while the rotary axis is rotating.

Example:

Calculating feed of linear and rotary axes

Axis X travels [100] mm, axis C travels 50 [degree], programmed feed 500 [mm/min]

Path length L = root of (100²+50²) ≈ 111.8 [mm]

Axis X: va = 100/111.8*500 ≈ 447.2[mm/min]

Axis C: va = 50/111.8*500 ≈ 223.6[degree/min]

In certain cases, the scaling of the part of the rotary axis motion in the feed is to be scaled higher (if the axes move too fast) or lower (if the exact orientation is irrelevant).

Therefore, an additional factor can be defined for rotary axes/coordinates to modify the effect of the axis on the feed (root of (L1² + L2² + L3² + (R1*F1)² + (R2*F2)²…)). If this factor is equal to 1, 1° = 1 mm applies. This corresponds to an effective radius of 57.3 mm.

This factor can be directly programmed in the part program if the "ActRadFact" NC function is used. In addition, the effective radius factor to inch/metric can be adapted in the configuration parameter (chapter 6.1.1 "Configuring Scaling Factors for Rotary Axis Velocities" on page 83).

Example:

Calculating feed of linear and rotary axes with ARF (reduced)

Axis X travels [100] mm, axis C travels 50 [degree], programmed feed 500 [mm/min], ARF(C0.5)

Path length L = root(100²+(50*0,5)²) ≈ 103.1[mm]

Axis X: va = 100/103.1*500 ≈ 485.9[mm/min]

Axis C: va = 50/103.1*500 ≈ 242.5[degree/min]

Example:

Calculating feed of linear and rotary axes with ARF (increased)

Axis X travels [100] mm, axis C travels 50 [degree], programmed feed 500 [mm/min], ARF(C3)

Path length L = root of (100²+(50*3)²) ≈ 180.3 [mm]

Axis X: va = 100/180.3*500 ≈ 277.3[mm/min]

Axis C: va = 50/180.3*500 ≈ 138.7[degree/min]

Restrictions

- Effective radius factors may not be negative.
- After system start, all effective radius factors of ActRadFact are set to 1. The factors configured in the machine parameters are applied again.
- If an axis is removed from the channel, the effective radius factor is also lost.
- If the effective radius of an axis/coordinate is equal to 0, motions only containing parts of this axis/coordinate may not be programmed.
- If the active transformation is changed, the effective radius factors of equal coordinates are applied. All other effective radius factors are reset to 1.
- This function is only effective for synchronous axes of "rotary" or "endless" type of axis motion (MP 1003 00004) or for rotary coordinates and the orientation.
- The function only refers to velocities, not to positions.
- The function has no effect in rapid traverse.

Relevant NC Functions

ActRadFact, ARF	Programs an effective radius factor

Tab. 6-3: Relevant NC Functions

Relevant Machine Parameters (MP)

7040 00110	Scaling the rotary axis velocity at G70/G71.

Tab. 6-4: Relevant Machine Parameters

Handling Instruction

Case 1

Problem The orientation in the part program changes abruptly -> rotary axes travel jerkily and too fast

Solution Increase the scaling of the rotary axes in critical areas.

Program:

```
N0010 G1 F800
…
N1010 X563.3 Y325.2 A34.1 C270.0
N1020 X254.1 Y325.2 A34.1 C270.0
N1030 X253.3 Y326.1 A34.1 C225.0 ARF(C2.5)    ; Scaling of c-factor 2.5
N1040 X252.6 Y327.2 A34.1 C180.0
```

```
N1050 X252.6 Y414.4 A34.1 C180.0 ARF()        ; Scaling of C again on 1.0
…
```

Result The MTX travels along the curve more slowly; rotary axes move smoother.

Case 2

Problem The orientation in the part program is irrelevant during the positioning motion -> rotary axes decelerate the motion

Solution Reduce the scaling of rotary axes during the positioning motion

Program:

```
…
N1010 G1 F1000 X550 Y220 A0 C0 ARF(A0.1, C0.1)      ; Scaling reduced
N1020 F355 X554.1 Y225.2 A34.1 C270.0
N1030 X556.1 Y227.2 A34.1 C270.0 ARF()              ; Scaling again on 1.0
…
```

Result The MTX moves faster to the target position.

6.2 Path Lengths and Curve Parameter Interpolation

6.2.1 Description

Function

For contour elements with curve parameters - the spline types 0 to 4 and the rounding splines (syntax SCO) – curve parameter interpolation can be selected instead of path length interpolation. The channel coordinates do not follow the path synchronously to the distance "s", but synchronously to the curve parameter "w". With path types without curve parameters (straight line, helix), an activated curve parameter interpolation has no effect.

The following figure shows one example. Curve parameter interpolation results in an xy-velocity proportionate to the drawn curve point distances.

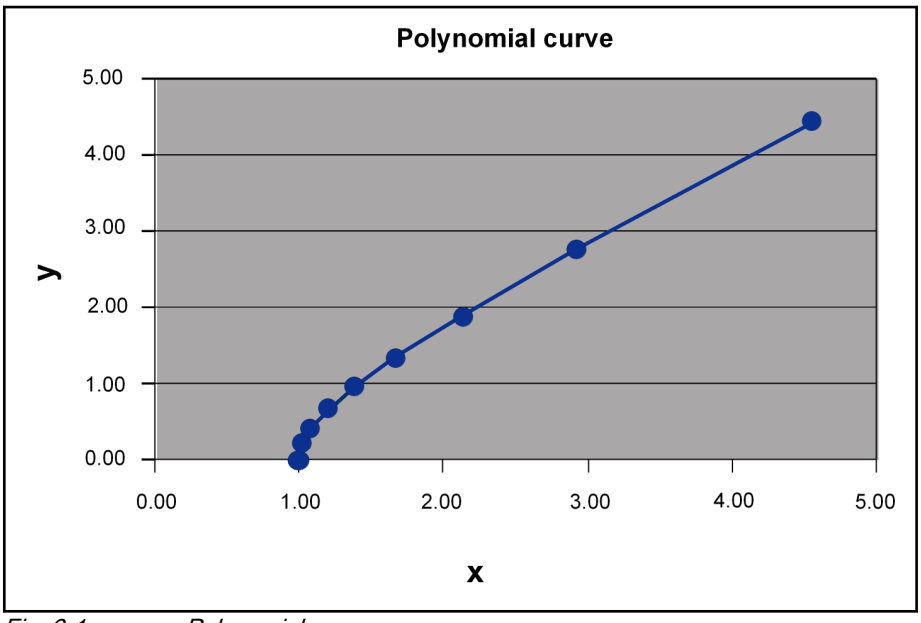


Fig. 6-1: Polynomial curve

Polynomial curve described by

$$x(w) = \frac{1+w^2}{1-w^2} \quad y(w) = \frac{2w}{1-w^2} \quad w \in [0,0.8]$$

The curve points for equidistant are drawn. $\Delta w = 0.1$.
The curve corresponds to the following NC program section:

N10 SplineDef(0002)
N20 G1 x1 y0
N20 G06 x(1,0,1) y(0,2) DN(1,0,-1) PL0.8

Relevant NC Functions

CPI(1)	Activating curve parameter interpolation
CPI(0)	Switch back to path length interpolation

Tab. 6-5: Relevant NC Functions

7 Safety/monitoring

7.1 Torque reduction

7.1.1 Description

Function

Torque reduction allows to temporarily influence the set torque limit value of an axis (S-0-0092: Torque limit value bipolar).

Features:

- By means of the axis-based interface signal qAx_TrqLim "Torque reduction", this function can be switched on and off.
- Feedback of the function status is given via the axis-related interface signal iAx_TrqLim "Torque reduction active".
- The reduced torque transferred to the drive using the setting of the interface signal "Torque reduction" is set via MP 1003 00010.

  The machine parameter value can be changed program-controlled via the "RedTorque" NC function. A torque limit value changed via RedTorque is applied at the next positive edge of the axis-related interface signal qAx_TrqLim.

  Once the interface signal is reset, the torque limit value (S-0-0092) current at the time of control startup is transferred again to the drive.

Restriction

- Switching on and off the torque reduction is only allowed at axis standstill.
- In the drive, S-0-0092 has to be interpreted as % of the nominal torque (select "percentage scaling" as scaling type in S-0-0086).

Relevant NC functions

RedTorque	Torque reduction

Tab. 7-1: Relevant NC functions

Relevant IF signals

qAx_TrqLim	"Torque reduction" (axis interface) L/H edge: Switch on torque reduction. H/L edge: Switch off torque reduction. When switching off, the torque limit value set in S-0-0092 after the last Sercos phase startup is set again in the drive.
iqAx_TrqLim	"Reduced torque" (axis interface) is active.

Tab. 7-2: Relevant IF signals

Relevant machine parameters

1003 00010	Presetting for reduced maximum axis torque.

Tab. 7-3: Relevant machine parameters

7.1.2 Handling instruction: Activating/deactivating torque reduction

Applying: **Engineering/configuration: Reduced torque at active torque reduction**

Correct the machine parameter

- **RedMaxTorq** "Reduced torque at active torque reduction" (1003 00010)

	Documentation	Instruction
Instruction:		Editing machine parameters
Documentation:	MTX Machine Parameters	Torque reduction

☞ In the drive, **S-0-0092** "Bipolar torque/force limit value" has to be interpreted as **%** of the nominal torque (select "percentage scaling" as scaling type in **S-0-0086**).

Activating/deactivating: **IW Operation/NC programming: Reduced torque at active torque reduction**

Using the NC function, **RedTorque(<AxisName><Value>)**, the value of the reduced torque can be changed.

	Documentation	Instruction
Documentation:	MTX Programming Manual	Torque reduction

IW Engineering/IndraLogic: Activating/deactivating torque reduction

- **qAx_TrqLim = 1**:

  Overrides the Sercos parameter S-0-0092 "Bipolar torque/force limit value" using the value from the parameter **RedMaxTorq** "Reduced torque at active torque reduction" (1003 00010) or the NC function **RedTorque**.

- **qAx_TrqLim = 0**:

  The original torque limit value from the Sercos parameter **S-0-0092** "Bipolar torque/force limit value" is transmitted to the drive.

	Documentation	Instruction
Instruction:		Editing PLC Signals
Documentation:	MTX PLC Interface	Torque reduction

7.2 Reduced rapid traverse

7.2.1 Description

Function

When entering part programs, it is frequently not desired to execute rapid traverses at maximum velocity. In case of faulty programming, a reaction by the operator by means of "Override" may be too late to prevent potential damage.

Thus, the control - depending on the channel-based interface signal qCh_RedRap "Reduced rapid traverse" - reduces all rapid traverse motions of synchronous axes in the corresponding channel to

- a parameterizable value (if MP 7030 00110 > 0)

  - or -

- to the feed rate last programmed in the channel (provided MP 7030 00110 = 0).



Provided no reference points have been approached yet, this function can also be used to reduce the rapid traverse.

If the "Override" function is active in the relevant channel, it also affects the reduced rapid traverse. In this case, the reduced rapid traverse velocity equals to an override of 100%.

Restrictions

- Applies to the "Manual Data Input of NC Block", "Continuous Block", "Program Block", "Single Block" and "Single Step" modes.
- A modification of the qCh_RedRap interface signal has no effect on program blocks already processed by block preparation at the time of modification.

  If this behavior is not desired, stop machining with "Feed hold", cancel the already prepared blocks by "Cancel distance to go" and continue machining via NC start.

Relevant IF signals

qCh_RedRap	"Reduced rapid traverse" (channel interface)

Tab. 7-4: Relevant IF signals

Relevant machine parameters

7030 00110	Value for reduced rapid traverse velocity

Tab. 7-5: Relevant machine parameters

7.2.2 Handling instruction: Reduced rapid traverse

The function reduces the maximum velocity of the rapid traverse motion (used for test purposes for example)

Applying:

IW Engineering/configuration: Configuring reduced rapid traverse

Assigning desired reduced rapid traverse velocity to the parameter **RedChVel** "Reduced rapid traverse velocity" (7030 00110).

	Documentation	Instruction
Instruction:		Editing machine parameters
Documentation:	MTX Machine Parameters	Reduced rapid traverse

IW Operation/machine: Triggering system reset

To apply the changed machine parameters, reset the system.

Activating:

NOTICE **Workpiece or machine can be damaged!**

Do not reset the system during machining.

IW Engineering/IndraLogic: Activating reduced rapid traverse

Set the axis interface signal **qCh_RedRap Reduced rapid traverse** as follows:

- **qCh_RedRap = 1**:

  to activate the "reduced rapid traverse" function.

Is not effective on spindle speeds or the active path feed.

A modification of the **qCh_RedRap** interface signal has no effect on program blocks already processed by block preparation at the time of modification.

If the "Override" function is active in the relevant channel, it also affects the reduced rapid traverse. In this case, the reduced rapid traverse velocity equals to an override of 100%.

	Documentation	Instruction
Instruction:		Editing PLC Signals
Documentation:	MTX Parameter Description	Reduced rapid traverse

Disabling: **IW Engineering/IndraLogic: Activating/deactivating reduced rapid traverse**

Deleting the axis interface signal **qCh_RedRap Reduced rapid traverse** as follows:

- **qCh_RedRap = 0**:
  to deactivate the "reduced rapid traverse" function.

A modification of the **qCh_RedRap** interface signal has no effect on program blocks already processed by block preparation at the time of modification.

	Documentation	Instruction
Instruction:		Editing PLC Signals
Documentation:	MTX Parameter Description	Reduced rapid traverse

7.3 Test feed, test rapid traverse

7.3.1 Description

Function

For the duration of the currently active function, it overrides the feed or rapid traverse velocity previously applied to the channel provided G8 or G9 is active.

The following applies:

- Via the "test feed" function, the effective feed can temporarily be increased or decreased to a defined value.
- Via the "test rapid traverse" function, the effective rapid traverse velocity can temporarily be reduced to a defined value.

NOTICE

During tool contact, invalid high feed possible!

When the test feed is activated during machining, the technologically maximum permissible feed may be exceeded.

This can result in damages to the workpiece, the tool or the machine.

Use the function test feed only for test purposes, not for machine parts!

Features

- The functions can either be influenced via channel-specific interface signals or the user interface.

  The control returns the currently active state at the channel-specific interface.
- The velocity values effective during test feed or test rapid traverse are stored for each channel in system date (in mm/min).
- The current feed override in the channel is retained even in case of a test feed or test rapid traverse.
- This function is effective under G93 (time programming), G94 (direct feed programming) and G95 (feed/revolution).
- Both functions affect the target velocity immediately after activation. The control includes the currently set acceleration to determine the required velocity changes.

Restrictions

- G8 or G9 has to be active.
- Is not effective with the "velocity profiles" functionality (ramp functions).
- During a running part program, any modifications of velocity values in the system date are only applied in the next NC block to be prepared.
- If the test rapid traverse set by the system date is greater than the rapid traverse value calculated by the control for the path, the latter value is applied.

Relevant system data

SysTestMode[<Channel no.>].Enable	
	A velocity value higher than the programmed feed and saved in the SysTestMode[<channel no.>]. TestFeedrate system date can only be reached if SysTestMode[<channel no.>].Enable has the value 1. During the processing, SysTestMode[<channel no.>].Enable should be set to value 0; otherwise, an undesired reduction of the path acceleration can occur during the processing operation.
SysTestMode[<Channel no.>].TestFeedrate	
	Velocity in mm/min for test feed in channel <ChannelNo.>
SysTestMode[<Channel no.>].TestRapid	
	Velocity in mm/min for test rapid traverse in channel <ChannelNo.>

Tab. 7-6: Relevant system data

Relevant IF signals

qCh_TestFeed	"Test feed" (channel interface) Switches on or off the "test feed" function.
qCh_TestRap	"Test rapid traverse" (channel interface) Switches on or off the "test rapid traverse" function.

iCh_TestFeed	"Test feed active" (channel interface) Mirrors the status of the "test feed" function.
iCh_TestRap	"Test rapid traverse active" (channel interface) Mirrors the status of the "test rapid traverse" function.

Tab. 7-7: Relevant IF signals

7.3.2 Handling instruction: Test feed (DryRun)

Activate/deactivate the test feed function (DryRun). The test feed function overrides the feedrate previously effective in the channel.

Activating: **IW Operation/NC programming: Assigning a value to a system date**

SD.SysTestMode[<Channel no.>].TestFeedrate = <Velocity in mm/min>

Save test feed velocity in system date.

	Documentation	
Documentation:	MTX Programming Manual	Test feed

IW Operation / HMI / M-panels: Assigning the M-key

		Instruction
Instruction:		Assigning the M-key

IW Operation/program: Activating test feed

Activating via GUI:

- Select the left M-key **"DryRun options"** (L7)



- Activate **"test feed"** (L4) using the left M-key



- Test feed is active if the "$" sign is displayed in the status field



or via PLC:

qCh_TestFeed = 1:

Activate the test feed with the channel-specific velocity value from the CPL variable **TEST_FEEDRATE**.

☞ G8 or G9 has to be active.

The interface signal **iCh_TestFeed** "Test feed active" (channel interface) reflects the status of the "test feed" function.

	Documentation	
Documentation:	IndraWorks HMI	Test feed

Disabling: **IW Operation/program: Deactivating test feed**

Deactivating via GUI:

- Select the left M-key **"DryRun options"** (L7)



- Deselect **"test feed"** (L4) using the left M-key



or via PLC:

qCh_TestFeed = 0:

Deactivate test feed

	Documentation	
Documentation:	IndraWorks HMI	Test feed

7.3.3 Handling instruction: Test rapid traverse (DryRun)

The "test rapid traverse" function overrides the rapid traverse velocity configured in the channel at G0 for running in an NC program with the (lower) test rapid traverse velocity.

Activating: **IW Operation/NC programming: Assigning a value to a system date**

SD.SysTetsSD.SysTestMode[<Channel no.>].TestRapid = <Velocity in mm/min>

Save test feed velocity in system date.

	Documentation	
Documentation:	MTX Programming Manual	Test feed

IW Operation / HMI / M-panels: Assigning the M-key

		Instruction
Instruction:		Assigning the M-key

IW Operation/program: Activating test rapid traverse

Activating via GUI:

- Select the left M-key **"DryRun options"** (L7)



- Activate **"test rapid traverse"** (L4) using the left M-key



- Test feed is active if the "$" sign is displayed in the status field



or via PLC:

qCh_TestRap = 1:

Activating test rapid traverse with the channel-specific velocity value from the CPL variable **TEST_RAPID**.

☞ G8 or G9 has to be active.

The interface signal **iCh_TestRap** "Test feed active" (channel interface) reflects the status of the "test feed" function.

	Documentation	
Documentation:	IndraWorks HMI	Test feed

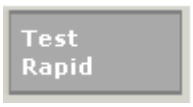
Disabling: **IW Operation/program: Deactivating test rapid traverse**

Deactivating via GUI:

- Select the left M-key **"DryRun options"** (L7)



- Deactivate **"test rapid traverse"** (L4) using the left M-key



or via PLC:

qCh_TestRap = 0:

Deactivating test rapid traverse

	Documentation	
Documentation:	IndraWorks HMI	Test feed

7.4 One-dimensional axis collision monitoring

7.4.1 Description

Function

Monitors the difference or total of the two axis positions Pos(1) and Pos(2):

Lower limit value ≤ Pos(1) ± Pos(2) ≤ upper limit value

The attempt to leave the monitored area initiates an error.

Thus, two parallel linear system axes can be monitored for

- Collision

  - or -

- Excessive divergence

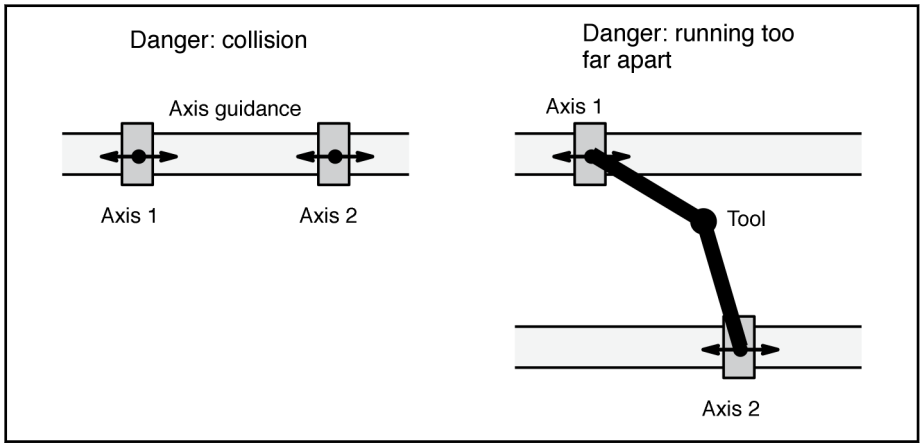


Fig. 7-1: Use cases

Features

- Ten groups of two axes each can be monitored simultaneously. One axis may belong to several groups.
- The axes of one group may be assigned to different channels.
- The monitoring function also takes the required deceleration distances into consideration. Accordingly, the preset limit values are not exceeded at standstill of the relevant axes.

Restrictions

- A maximum of ten groups is system-wide possible.
- Axis collision monitoring for a group is only effective if both axes involved are referenced.
- Modulo axes are not allowed.
- In cases where the purely position-based programmed data does not cause a collision, excessive axis velocities (v_{axis1}, v_{axis2}) and/or interpolator cycle times (tIpo) can trigger the collision monitoring function.

  The cause is that it is fallen below the internally formed safety distance:

  $d_{min} = 4 * |max(v_{axis1}, v_{axis2})| * t_{Ipo}$.

Relevant machine parameters (MP)

8008 00001	The monitoring network is activated Enables monitoring for the respective group. The monitoring function becomes active immediately after referencing the two axes involved.
8008 00002	System axis number of monitoring axis 1 Physical axis index of the first axis of the monitored group.
8008 00003	System axis number of the monitoring axis 2 Physical axis index of the second axis of the monitored group. Monitoring axes 1 and 2 may be located in different channels.

8008 00004	Specifies whether the difference between or the total of the two axis positions is monitored: 0: The difference is monitored: Lower limit value ≤ Pos(1) - Pos(2) ≤ upper limit value 1: The total is monitored, i.e.: Lower limit value ≤ Pos(1) + Pos(2) ≤ upper limit value
8008 00005	Activate monitoring of the upper limit value. 0: Upper limit value (MP 8008 00007) is not monitored, i.e.: Lower limit ≤ Pos(1) ± Pos(2) 1: Upper limit value (MP 8008 00007) is monitored, i.e.,: Lower limit value ≤ Pos(1) + Pos(2) ± upper limit value MP 8008 00006 specifies whether the lower limit value is monitored.
8008 00006	Activating monitoring the lower limit value. 0: Lower limit value (MP 8008 00008) is not monitored, i.e.: Pos(1) ± Pos(2) ≤ upper limit value 1: Lower limit value (MP 8008 00008) is monitored, i.e.: Lower limit value ≤ Pos(1) + Pos(2) ± upper limit value MP 8008 00005 specifies whether the upper limit value is monitored.
8008 00007	Upper limit value The entered numerical value is only effective if the value 1 is entered under MP 8008 00005.
8008 00008	Lower limit value The numerical value entered is only effective if the value 1 is entered under MP 8008 00006.

Tab. 7-8: Relevant machine parameters (MP)

7.4.2 Handling instruction: One-dimensional axis collision monitoring

The one-dimensional axis collision monitoring monitors any two linear system axes for collision or excessive divergence. The system axes travel on the same path behind each other or on parallel paths.

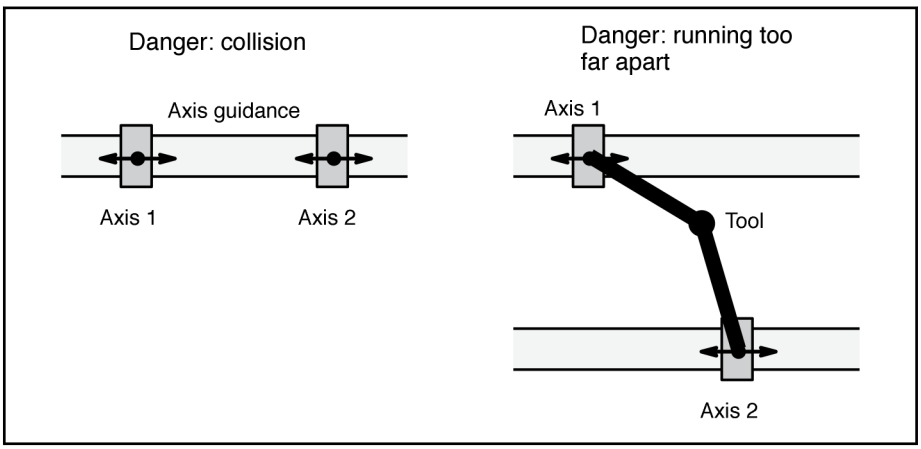


Fig. 7-2: Danger of collision or divergence

Applying:

NOTICE **Danger of collision or impermissibly wide divergence! Danger of damage to the machine.**

Ensure that the axes can be stopped without problems at any time!

☞ A maximum of ten groups is system-wide possible.

Modulo axes are not allowed.

In cases where the purely position-based programmed data does not cause a collision, excessive axis velocities [v(Axis1), v(Axis2)] and/or interpolator cycle times (tIpo) can trigger the collision monitoring function. Cause: It is fallen below an internally formed safety distance: dmin = 4 * |max[v(Axis1), v(Axis2)] | * tIpo

IW Engineering/configuration: Configuring axis collision monitoring

Parameterize as described in the following for each individual monitoring group:

- Enter the system axis number for axis 1 to be monitored in the parameter **MonAx1** "Monitoring axis 1" (8008 00002).

  Enter the system axis number for axis 2 to be monitored in the parameter **MonAx2** "Monitoring axis 2" (8008 00003).

  Enter the orientation of the coordinate systems of the two axes participating in the parameter**Dir**: "Traversing direction" (MP 8008 00004).
- Ensure that the relevant axes are referenced.
- Move the two desired axes carefully up to the desired "collision limit". The collision limit is the critical distance of both axes in case of danger.
- Note the axis position of both axes:

  Pos1: Position of the monitoring axis 1

  Pos2: Position of the monitoring axis 2

  Both values are required in the following!
- Determine monitoring type; logical "0" = No and logical "1" = Yes.

  ActUpLim "Monitor upper limit" (8008 00005)

  ActLowLim "Monitor lower limit" (8008 00006)

Monitoring type	Required machine parameter setting	
	ActUpLim	ActLowLim
No monitoring	0	0
Monitoring for collision if axis 1 runs in front of axis 2 *)	0	1
Monitoring for collision if axis 2 runs in front of axis 1 *)	1	0
Monitoring for divergence	1	1

*) With regard to the positive traversing direction of axis 2 with synchronous traversing direction of both axes.

Tab. 7-9: Monitoring type

- Determine limit values. Use the values for Pos1 and Pos2 above (point 3):

  PosUpLim "Upper limit value" (8008 00007)

PosLowLim "Lower limit value" (8008 00008)

Danger	When setting the parameters			Computation of parameter values	
	Dir	ActUpLim	ActLowLim	PosUpLim	PosLowLim
Collision	0 (equal)	0	1	-	Pos1 - Pos2
		1	0	Pos1 - Pos2	-
Collision	1 (opposed)	0	1	-	Pos1 +Pos2
		1	0	Pos1 + Pos2	-
Diverging	0 (equal)	0	1	Pos1 - Pos2	-
		1	0	-	Pos1 - Pos2
Diverging	1 (opposed)	0	1	Pos1 + Pos2	-
		1	0	-	Pos1 + Pos2

	Documentation	Instruction
Instruction:		Editing machine parameters
Documentation:	MTX Machine Parameters	Applying axis collision monitoring

Activating:

Prerequisite:

The function is applied correctly.

☞ Monitoring a group becomes active when both participating axes are referenced.

NOTICE **Workpiece or machine can be damaged!**

Do not reset the system during machining.

IW Engineering/configuration: Activating axis collision monitoring

The parameter **EnablCollMon** "Activate one-dimensional collision monitoring" (8008 00001) has to be set as follows:

- **EnablCollMon = Yes**:

  Activates collision monitoring

	Documentation	Instruction
Instruction:		Editing machine parameters
Documentation:	MTX Machine Parameters	Activating axis collision monitoring

IW Operation/machine: Triggering system reset

To apply the changed machine parameters, a system reset or a control start-up is required.

If an active monitoring group is not kept, a warning is issued directly after a system reset. Thus, motions in the direction of the permissible range are allowed. Motions away from the permissible range are prohibited and cause an error message.

☞ Modulo axes are not allowed.

In cases where the purely position-based programmed data does not cause a collision, excessive axis velocities [v(Axis1), v(Axis2)] and/or interpolator cycle times (tIpo) can trigger the collision monitoring function. Cause: It is fallen below an internally formed safety distance:

dmin = 4 * |max[v(Axis1), v(Axis2)] | * tIpo

Disabling:

NOTICE **Workpiece or machine can be damaged!**

Do not reset the system during machining.

IW Engineering/configuration: Activating/deactivating axis collision monitoring

The parameter **EnablCollMon** "Activate one-dimensional collision monitoring" (8008 00001) has to be set as follows:

- **EnablCollMon = No**:
  Deactivates collision monitoring

	Documentation	Instruction
Instruction:		Editing machine parameters
Documentation:	MTX Machine Parameters	Activating axis collision monitoring

IW Operation/machine: Triggering system reset

To apply the changed machine parameters, a system reset or a control start-up is required.

7.5 Axis position monitoring

7.5.1 Description

Function

Indicates via interface signal in which range an axis is located in relation to a parameterizable position (⇒MP 2010 00110). The position always relates to the axis coordinate system (ACS).

The control includes active LSE corrections automatically when defining the current axis position.

The following applies for signal generation:

- Current axis position ≥ Parameterized position: Signal set
- Current axis position < Parameterized position: Signal not set

This function is used to activate other functionalities by means of the PLC.

Features:

- Up to eight positions can be defined with one corresponding interface signal for each axis.
- For rotary axes, the monitoring position can refer to the complete traversing range or for the range between 0 and 360 degrees.
- Optionally, monitoring can be set only for a referenced or for a non-referenced axis.

Restriction

- The monitoring positions relate to the axis coordinate system

- A maximum of 64 monitoring positions system-wide

Relevant IF signals

iAx_PosSwitch1 to iAx_PosSwitch8	
	"Position switching point 1 ... 8" (AxisInterface). High signal: Current axis position ≥ parametr. Position Low signal: Current axis position < parametr. Position

Tab. 7-10: Relevant IF signals

Relevant machine parameters (MP)

2010 00100	Monitored axis
2010 00110	Position of the point
2010 00120	Interface signal of the point
2010 00130	Cyclic monitoring at rotary axes

Tab. 7-11: Relevant machine parameters (MP)

7.5.2 Handling instruction: Axis position monitoring

Applying: **IW Engineering/configuration: Setting axis position monitoring**

For each Point[i] (disconnection point[i]), the following parameters have to be adjusted:

- **Dr** "Axis number of the physical axis" (2010 00100)
- **Pos** "Position" (2010 00110)
- **MonType** "Type of monitoring" (2010 00100)
- **MonRotAx** "Monitoring for rotary axes" (2010 00130)
- **IfSig Interface signal of the point** (2010 00120) contains the desired interface signal **iAx_PosSwitch1** to **iAx_PosSwitch8**.

☞ The monitoring positions relate to the axis coordinate system (ACS).

Up to 64 monitoring positions are system-wide possible.

	Documentation	Instruction
Instruction:		Editing machine parameters
Documentation:	MTX Machine Parameters	Axis position monitoring

Activating: **IW Engineering/IndraLogic: Using axis position monitoring signal**

Up to eight interface signals can be used as position switching point for each axis:

iAx_PosSwitch1 to **iAx_PosSwitch8** "Position switching point 1 ...8" (axis interface)

- **High signal**: Current axis position >= Parameterized position
- **Low signal**: Current axis position < Parameterized position:

	Documentation	Instruction
Instruction:		Editing PLC Signals
Documentation:	MTX PLC Interface	Axis position monitoring

7.6 Distance to end point

7.6.1 Description

Function

An axis-related interface signal is output as soon as the current distance of the command position to the programmed end point of an NC block falls below a defined distance.

Thus, the PLC can for example already initiate or prepare customer-specific functions before the axis reaches its actual end point. This can result in a reduction of the total time required for a machining process.

Restriction

"Distance to end point" is without function for

- Slave axes at active axis coupling
- Handwheel mode
- "Approach reference point " mode

Relevant IF signals

iAx_DistCtrl	"Axis to end point" (axis interface)

Tab. 7-12: Relevant IF signals

Relevant machine parameters (MP)

1015 00010	Distance to end point.

Tab. 7-13: Relevant machine parameters (MP)

7.6.2 Handling instruction: Distance To End Point

Is used to output an axis interface signal as soon as the current distance of the command position to the programmed end point of an NC block falls below a defined distance.

Applying:

NOTICE **Workpiece or machine can be damaged!**

Do not reset the control during machining.

IW Engineering/configuration: Editing parameters

- **EnablAxDistCtrl** "Activate distance monitoring" (1015 00010)

  EnablAxDistCtrl = Yes: Enable "distance to end point"
- Enter the desired distance in the parameter **DistEndPoint** “Distance to the programmed endpoint” (1015 00010), where the axis interface signal **iAx_DistCtrl** "Axis from end point" is to be set when it is fallen below this distance.

	Documentation	Instruction
Instruction:		Editing machine parameters
Documentation:	MTX Machine Parameters	Distance To End Point

IW Operation/Diagnostics: Triggering control startup

To enable the function, a control startup is required.

IW Engineering/IndraLogic: Using signal distance to endpoint

The axis interface signal **iAx_DistCtrl** "Axis to end point" is set if it is fallen below the distance defined in the parameter **DistEndPoint** "Distance to the programmed end point"(1015 00010).

☞ "Distance to end point" is without function for:

- Slave axes at active axis coupling
- Handwheel mode
- "Approach reference point " mode

	Documentation	Instruction
Instruction:		Editing PLC Signals
Documentation:	MTX Machine Parameters	Distance To End Point

Disabling:

NOTICE **Workpiece or machine can be damaged!**

Do not reset the control during machining.

IW Engineering/configuration: Editing parameters

- **EnablAxDistCtrl** "Activate distance monitoring" (1015 00010)
  EnablAxDistCtrl = No: Deactivate "distance to end point"

	Documentation	Instruction
Instruction:		Editing machine parameters
Documentation:	MTX Machine Parameters	Distance To End Point

7.7 Retract vector as system date

7.7.1 Description

Function

If the program is aborted during machining, it is in most cases preferable to lift off the tool from the workpiece at right angles to the machined workpiece surface. This is important after power failure. After power has returned, the customer wants to lift off the tool from the workpiece surface without damaging the latter.

To facilitate this, the storage of the following data can be activated at interpolator cycle intervals by means of an NC function:

- Storage of the orientation vector to system date SD.RevVec.Orientation[1..3]

- Storage of the binormal vector to system date SD.RecVec.Binormal[1..3]

  This is the standardized radius correction vector if radius correction is activated.

  If radius correction is not activated, it is the unit vector perpendicular to the direction and the orientation vector.

  An exception are the G41/G42 correction build-up block and the G40 correction build-down block. In these cases, the binormal vector extends vertically to the center point path of the milling tool. In practice, however, this is hardly of any relevance, since there is no contact to the tool in these blocks yet.

System date **SD.RevVec** is not included in the scope of delivery by default. The functionality described can only be used if it has been created as follows:

```
<Variable Dimension="Channel" Storage="permanent">
  <Name> RevVec </Name>
  <Type> RevVec_t </Type>
</Variable>
```

If the wrong type is selected, the minor system error 4144 is output the next time the system is started up.

If the wrong dimension is selected, the minor system error 4145 is output the next time the system is started up.

If the system date is not created at all, there is no information the next time the system is started up.

If RRV is activated without there being any suitable system date, error 4147 is output.

If RRV is activated and the system date type is wrong, error 4148 is output.

If RRV is activated and the system date dimension is wrong, error 4149 is output.

Storage = "permanent" is also allowed instead of storage = "volatile". In this case, the content of the system date is lost in the event of a power failure.

When relating to the direction (forward by definition) and to the orientation vector pointing away from the workpiece, the perpendicular to these two directions points either to the left or to the right. The preferred option must be defined when storage is activated.

Storage with a binormal vector pointing to the left is activated with the following NC command (cf. Rexroth MTX Programming Manual):

RecordRevVec(1) or alternatively **RRV(1)**, **RRV**

Storage with a binormal vector pointing to the right is activated with the following NC command (cf. Rexroth MTX Programming Manual):

RecordRevVec(2) or alternatively **RRV(2)**

Once processing of the NC block in which this activation takes place is started, the system date SD.RevVec is initialized to zero. Then, the current orientation and binormal vectors are stored at interpolator cycle intervals.

If a radius correction is active, the decision whether the binormal vector should point to the left or to the right is not made by the parameter of NC function **RecordRevVec** but by the radius correction itself.

G41, G141: The tool is arranged to the left of the workpiece. The binormal vector also points to the left. These two statements also apply to a negative milling tool radius.

G42, G142: The tool is arranged to the right of the workpiece. The binormal vector also points to the right. These two statements also apply to a negative milling tool radius.

This means that, in this case, it does not matter whether you program parameter 1, 2 or no parentheses at all.

If a radius correction is deactivated while storage is active, the decision whether the binormal vector should point to the left or to the right is made by the parameter of the active NC function RecordRevVec.

The two vectors always refer to the basic coordinate system (BCS). If used to lift off the tool from the workpiece, coordinate transformations or input tools may therefore not be active.

Storage of the system date is deactivated via the following NC command:

RecordRevVec(0)

or alternatively

RecordRevVec(), **RRV(0)**, **RRV( )**

Once processing of the NC block in which this deactivation takes place is started, nothing is written to the system date SD.RevVec any longer. The values that were the last to be stored there remain.

Remarks

- The orientation vector is not imperatively perpendicular to the direction.
- For example, negative milling tool radiuses are required if the CAD system already provides the center point path of the milling tool for a specific milling tool radius and the user decides to work with a different milling tool radius. In this case, the existing path must be corrected with the following "milling tool radius":

  (actual milling tool radius) – (CAD milling tool radius)

  However, the return direction must always remain the same, no matter whether this "milling tool radius" is > 0, = 0 or < 0.
- If mirror is active with regard to one or all of the three spatial coordinates, the binormal vector is inverted.
- If G20 and RRV are active at the same time, the system date is nulled and the warning 4146 is output, as no statement can be made for the reverse factor if G20 is active.

Example: **Program abortion due to power failure**

Program:

```
001  DCT(4,1,0)=10
N10  RRV(1)
N20  G71 G17 G1 F1000 X-100 Y0 Z0
N30  X-70
N40  X-30 G41 ED1
N50 X0
N60  X40 Y30      ; Power failure during block processing
...
```

The following NC program was interrupted by a power failure at the point marked as such:

Example: **Selected retraction**

After the control has started up, for example, the following NC program or NC cycle can be used to retract by 17.0 mm to the side of the workpiece surface:

Program:

```
10   A = SD.RevVec.Binormal[1]      :REM = -0.6
20   B = SD.RevVec.Binormal[2]      :REM =  0.8
30   C = SD.RevVec.Binormal[3]      :REM =  0.0
40  W = 17.0                         :REM Retract path

N50  G91 G17 G1 F100
N60  X[W*A] Y[W*B] Z[W*C]

; It is assumed that after control startup
; no milling path correction, no coordinate transformation
; and no input help is active. The decision about the
; retract distance, that it the distance that may be retracted
; safely is the user`s responsibility.
```

Relevant NC functions

RecordRevVec, RRV, G16..G20, G40, G41, G42, G141, G142

7.7.2 Handling instruction: Retract Vector as System Date

Activating Activation of storage for a binormal vector pointing to the left:

RRV(1)

Activation of storage for a binormal vector pointing to the right:

RRV(2)

Deactivating **RRV(0)**

The user should be able to retract the tool from the workpiece after an NC program has been interrupted, e.g., due to a power failure, without damaging the workpiece.

1. Insert NC command RRV(1) or RRV(2), respectively, at the point in the NC program where continuous recording of the orientation and binormal vectors should start.
2. If you do not have to record the orientation and the binormal vectors any more at a certain point in the NC program and thereafter, insert the NC command RRV(0) (in short) at this point.
3. Program a cycle for retracting the tool in case of an interruption of the NC program.

Example: If the NC program was interrupted while a milling tool path correction was active, the user might want to retract the milling tool from the workpiece in the direction of the binormal vector. In this case, this cycle might be as follows:

Program:

```
20  A = SD.RevVec.Binormal[1]
21  B = SD.RevVec.Binormal[2]
22  C = SD.RevVec.Binormal[3]
23  W = 17.0                         :REM Retract path

G91 G17 G40 G1 F100

X[W*A] Y[W*B] Z[W*C]
```



The decision on how far the tool may be safely retracted in the direction specified (17.0 mm in the example) must be made based on the specific workpiece geometry.

The orientation and binormal vectors always refer to the basic coordinate system (BCS).

If an axis transformation was active while the two vectors were recorded, this axis transformation must therefore also be activated during retraction.

If, however, input tools, a radius correction or coordinate transformations, e.g., placements, were active during recording, they may not be activated during retraction.

8 Accuracy

8.1 Accuracy programming

8.1.1 Description

Function

In certain use cases, a **controlled** "linishing" of contour transitions can be expressly required (when machining free-form surfaces for example).

Thus, "Accuracy programming" limits

1. - at the contour transitions - the
   - contour error (e)

     - or -
   - path lag (d)

   and
2. - for circular path segments (circles, helical, helicalN) - the
   - radius error (e)

to a programmable value (in mm or inch; depending on whether G71 or G70 is active).

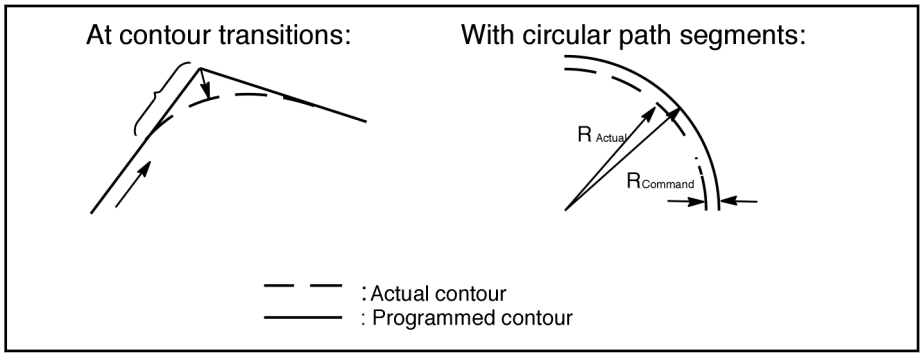


Fig. 8-1: Accuracy programming

The control calculates the maximum path velocity which must not be exceeded to maintain the boundaries of the respective limit value for each contour transition or for each circular path segment.

Restrictions

- The "Accuracy programming" function is applied to contour transitions only if G8 is active. It is always applied to circular path segments.
- The "Exact stop" function has to be switched off (G62 is active).
- Only the controller parameters of the first axis in the channel are used to calculate the maximum path velocity. All axes involved at the path have thus to be parameterized identically with regard to dynamics.
- To calculate the maximum path velocity, the control assumes that there is a linear connection between velocity and lag. The influence of accelerations on the current lag is not taken into account.
- When "Feed Forward" is active in the drive (moving without following errors, feed forward), the control considers the Sercos parameter P-0-0500 only in connection with ServoDyn drives to calculate the valid path velocity. It reads the parameter value from the drive at the Sercos phase startup and saves the value.

If P-0-0500 is assigned with values > 99%, a following error of 1% of the following error without feed forward is assumed.

For other drives, the control, when calculating the valid path velocity, proceeds on the assumption that the lag is reduced by half when the feed forward control is active in the drive.

Relevant NC functions

PrecProg **– or –**	
PrecProg(EPS <ContourError>) **– or –**	
PrecProg(DIST<path-lagging>)	
	In each case, the accuracy programming is ON. If **only PrecProg** is programmed, the control limits the contour/radius error ε at contour transitions or with circular path segments. The limit value is the value parameterized in MP 8003 00001. If **PrecProg is programmed together with EPS**, the control limits the contour/radius error ε at the contour transitions or for circular path segments to the value programmed after EPS. If **PrecProg is programmed together with DIST**, the control limits the path lagging at contour transitions δ to the value programmed after DIST. When traversing circular path segments, the control restricts the radius error ε according to MP 8003 00001. DIST and EPS must not be programmed simultaneously.
PrecProg(0)	Accuracy programming OFF.

Tab. 8-1: Relevant NC functions

Relevant machine parameters (MP)

8003 00001	Contour deviation tolerance for the accuracy programming "PrecProg".

Tab. 8-2: Relevant machine parameters

Relevant SD functions

SD(328,1)	Provides the latest programmed maximum permitted contour/radius error ε in the active channel as rounded integer. Alternative programming: SD(328).
SD(328,2)	Provides the latest valid programmed maximum path lagging δ in the active channel as rounded integer.
SDR(328,1)	identical to SD(328,1); return value in real format. Alternative programming: SDR(328).
SDR(328,2)	identical to SD(328,2); return value in real format.

Tab. 8-3: Relevant SD functions

If the "Accuracy programming" is not active, all functions provide 0.

8.1.2 Handling instruction: Accuracy programming

Activating/deactivating the accuracy programming. The accuracy programming limits the maximum deviation at contour transitions and for circular path segments (circles, helical, helicalN).

IW Engineering/configuration: Determining tolerance

The parameter **ContDevTol** "Contour deviation tolerance" (8003 00001) determines maximum path error allowed at the contour transition and the radius error at the arc.

	Documentation	Instruction
Instruction:		Editing machine parameters
Documentation:	MTX Machine Parameters	Accuracy programming

IW Operation/NC programming: Activating/modifying

- **PrecProg(1)**

  - or -

- **PrecProg(EPS<Contour error ε>)**

  - or -

- **PrecProg(DIST<Path lag δ>)**

  programming.



The "Accuracy programming" function is applied to contour transitions only if G8 is active. It is always applied to circular path segments.

The "Exact stop" function has to be switched off (G62 is active).

All axes participating in the path have to be parameterized identically with regard to the dynamics.

DIST and EPS must not be programmed simultaneously.

	Documentation	
Documentation:	MTX Machine Parameters	Accuracy programming

IW Operation/NC programming: Deactivating

Program **PrecProg(0)**

	Documentation	
Documentation:	MTX Programming Manual	Accuracy programming

9 Path motion

9.1 Spline

9.1.1 Overview

Function

Compared to linear interpolation, spline interpolation achieves the same surface and contour accuracy despite its lower number of data points, as **curves** are generated between the points.

The spline function provides:

- Different options for processing for spline data of the CAD/CAM or digitizing system
- Different spline types for practically-oriented requirements (e.g. 3- and 5-axis free-form surface machining),
- Parameterization of boundary conditions for application-oriented spline processing in the NC program

Spline data origin: Point sequences are specified as follows:

- By manual programming in the NC program
- By generating a spline using a CAD program with postprocessor function, while only the spline coefficients are transmitted in the NC program

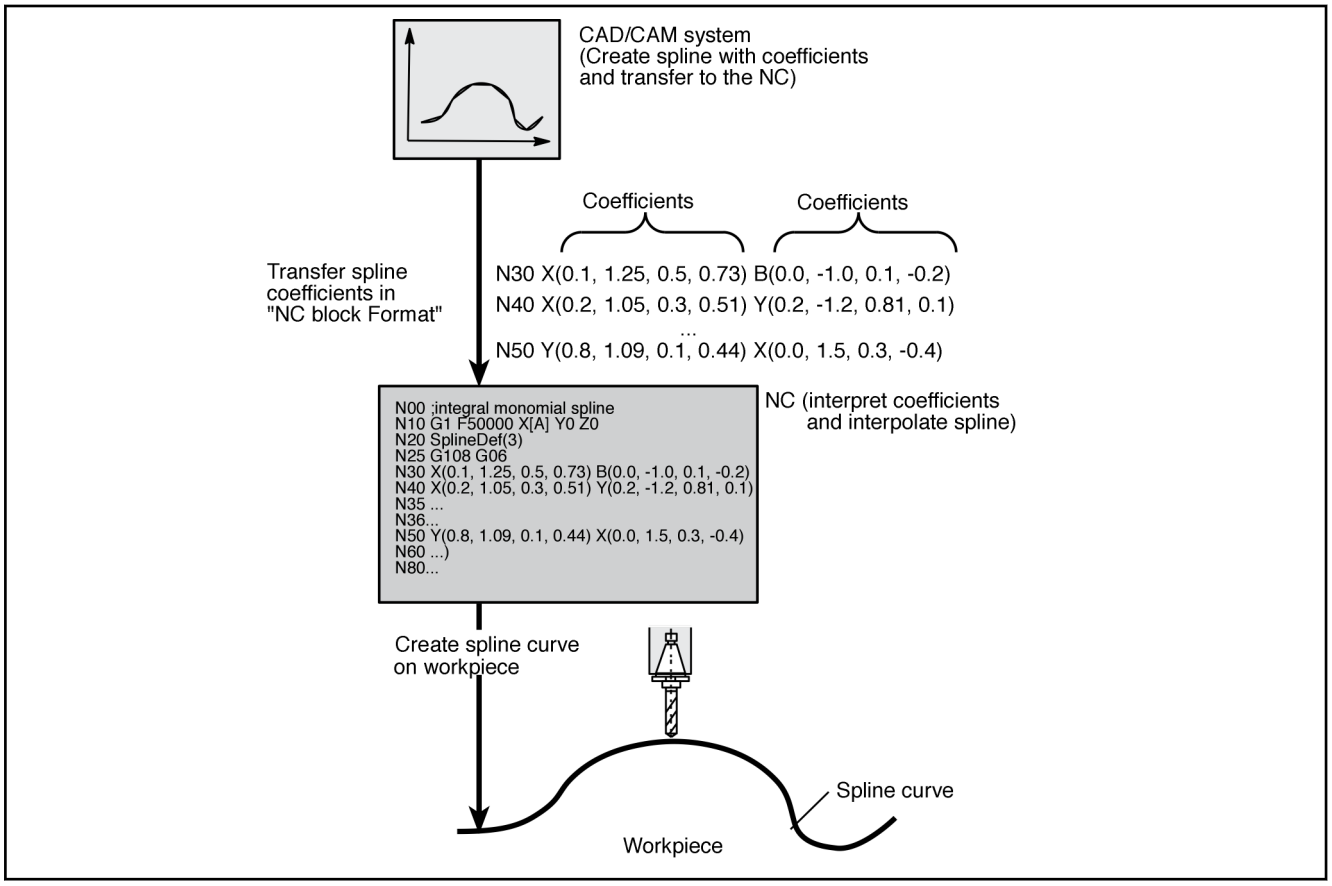


Fig. 9-1: By a CAD/CAM system (generating spline with coefficients and transmitting it to the NC)

- By a CAD/CAM system creating data points or checkpoints for the NC program used to generate NC spline curves (see the figure below).
- While digitizing: Recording surfaces as individual points (see the figure below)

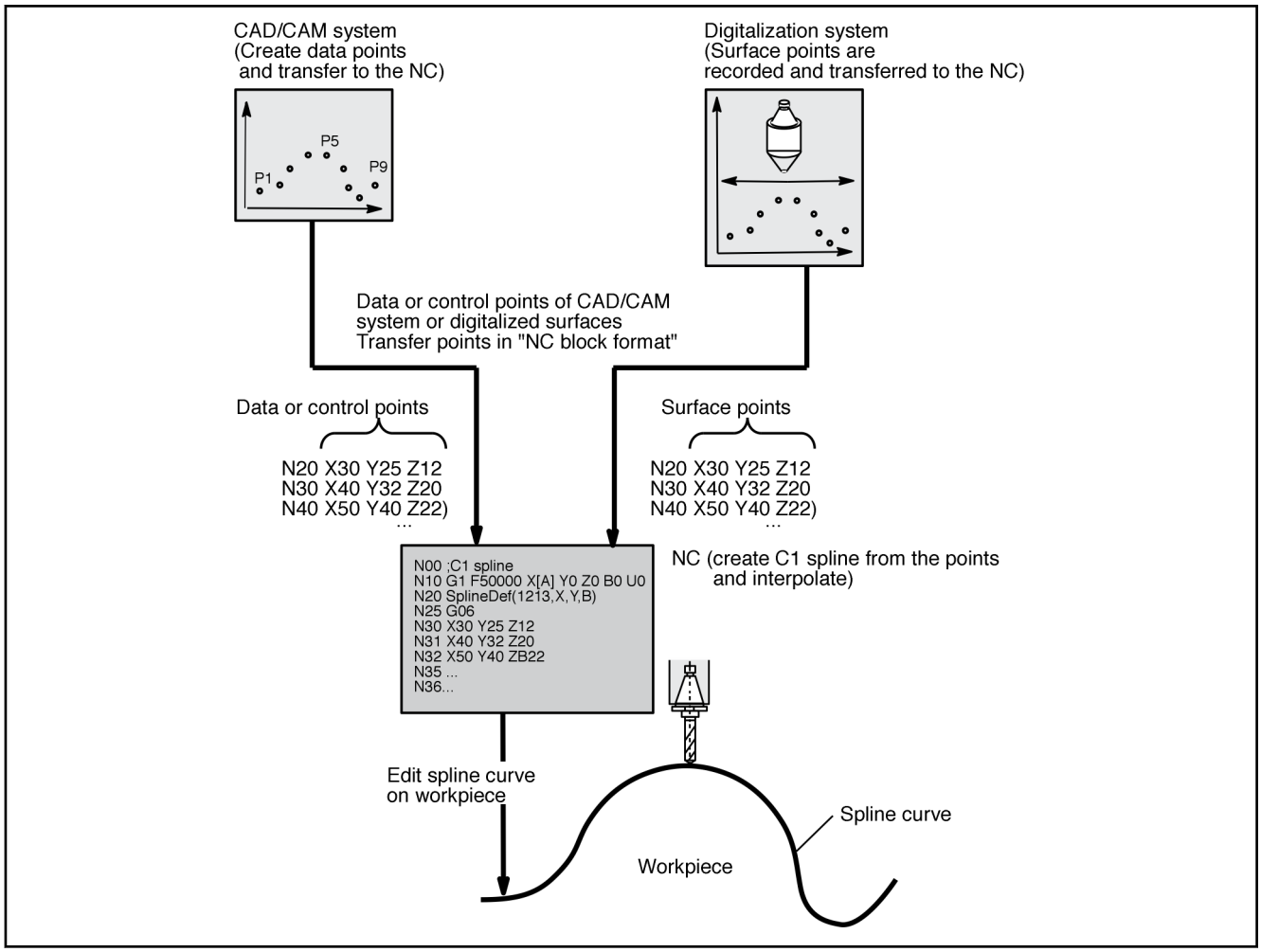


Fig. 9-2: Digitizing

Advantages:

- Softer contour alignment and smoother workpiece surfaces
- The number of points generated by the CAD/CAM program or by digitizing can possibly be reduced provided the required accuracy requirements are met
- Compared to the angular transition with linear interpolation, spline interpolation causes **less** stress for the machine and its moving parts as the direction changes of the tool are less abrupt

Spline types (overview): *The NC supports the following spline programming types:*

- **Type 0: Splines with coefficient programming**

  Each spline can be represented as a polynomial (e.g. parabola=square polynomial). Its coefficients define the visual appearance of the curve, its end and starting points. The coefficients are provided from the CAD/CAM system. There, they are individually programmed as NC blocks, transmitted to the NC and interpolated in the NC.

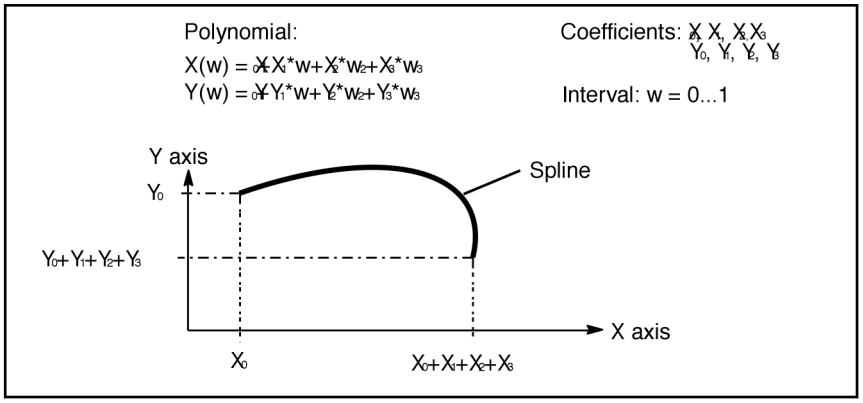


Fig. 9-3: Splines with coefficient programming

Advantages:

The spline is defined by the data provided by the CAD/CAM system.

Disadvantages:

The CAD/CAM system has to have a corresponding postprocessor interface. Parameterizing in the NC does not allow any corrections. Potential deviations becomes only visible during machining.

- **Type 1: C1-constant cubic splines with data point programming**

  The MTX distinguishes between two subtypes:

  – There are only data points in the NC program. The NC generates cubic splines tangential at the data points, i.e. c^1-continuous. The tangents at the transitions are calculated according to the Akima, Bessel or False position method and take effect on up to four data points.

  – Hermite splines: In the NC program, there are tangents (derivation of the coordinate according to the path length) next to the data points. On their basis, the NC creates cubic splines that are at least c^1-continuous.

  Advantages:

  Reduced computation time in the NC. Only one block look-ahead is required (local spline).

  Disadvantages:

  The transitions at the data points have no continuous curvature. This causes jumps in the axis acceleration.

- **Type 2: C^2-constant cubic splines with data point programming**
- There are only data points in the NC program. The NC creates cubic splines tangential at the data points (that means c^2-continuous).

  Advantages:

  No jump in the axis acceleration.

  Disadvantages:

  High computation and memory load in the NC. Large number of blocks in the look-ahead (global spline).

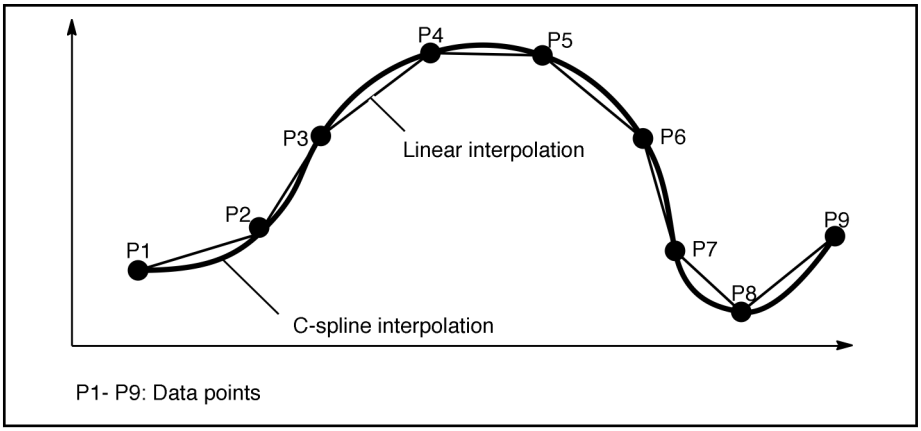


Fig. 9-4: Cubic splines

- **Type 3: B-splines with checkpoint programming**

  For the B-spline programming **(e.g. NURBS)**, the checkpoints of the B-spline curve are located in the NC program. The curve does not cross the checkpoints, but approximates the checkpoint polygon (refer to the following figure).

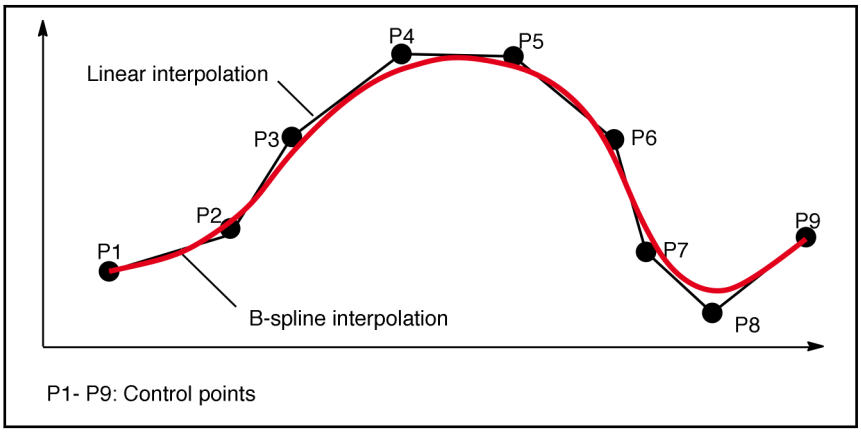


Fig. 9-5: B-splines with checkpoint programming

Advantages:

– Lower computation load in the NC than for the C^2-continuous cubic splines

– Smaller number of blocks in the look-ahead (local spline)

– All curves defined by conics (ellipses, circle, parabola, hyperbola) can be displayed by B-splines

Disadvantages:

The curve does not cross the checkpoints, only approximates them.

- **Type 4: Approximating B-splines with tolerance specification (compressor)**

  The resulting spline curve approximates the programmed points with a specified accuracy.

  This

  1. smoothes imprecisely programmed points
  2. summarizes short NC blocks (compressor)

9.1.2 Spline with coefficient programming

General information



The prerequisite for coefficient programming of splines is a corresponding CAD/CAM programming system able to generate the coefficients in an NC program.

Integral monomial splines with coefficient programming

The spline is represented by a polynomial up to the degree of n=5.

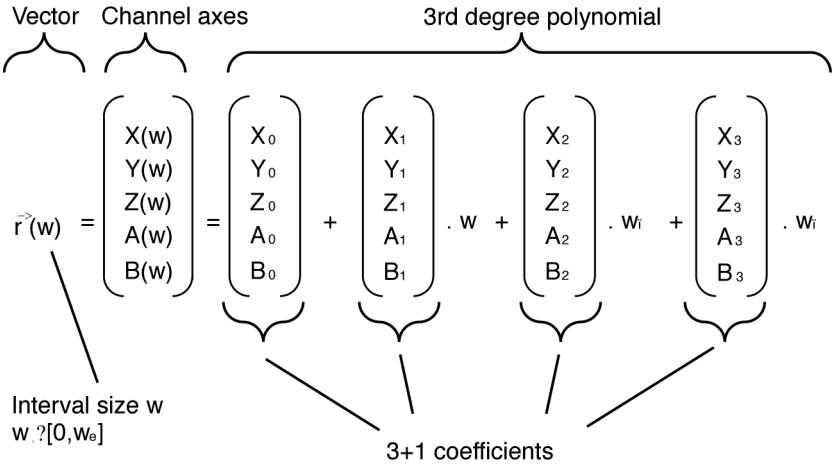
The n+1 coefficient of the polynomial are specified **directly** in the NC block.

Vector Polynomial

$$\vec{r}(w) = \sum_{j=0}^{n} \vec{r}_j w^j \qquad w \in [0, w_e]$$

Example:

Monom splines (e.g. cubic spline: degree 3) with coefficient programming. It is assumed that the channel axes X, Y, Z, A and B form the spline motion.



Example:

Specifying coefficients in the NC program for the channel axes X and B

G06 **X(0.1, 1.25, 0.5, 0.73) B(0.0, -1.0, 0.1, -0.2)**

Thus, the following results for the channel axes X and B:

$X(w) = 0.1 + 1.25w + 0.5w^2 + 0.73w^3$

$B(w) = 0.0 - 1.0w + 0.1w^2 - 0.2w^3$

In the definition interval 0 to $_e$, the vectors creating the spline curve are calculated. The vectors components are the channel coordinates. From the channel coordinates, the axis positions of the physical axes are calculated internally in the NC, taking into account an active axis transformation.

Example:

Spline curve

$\vec{r}(w)$ With the two channel coordinates x and y

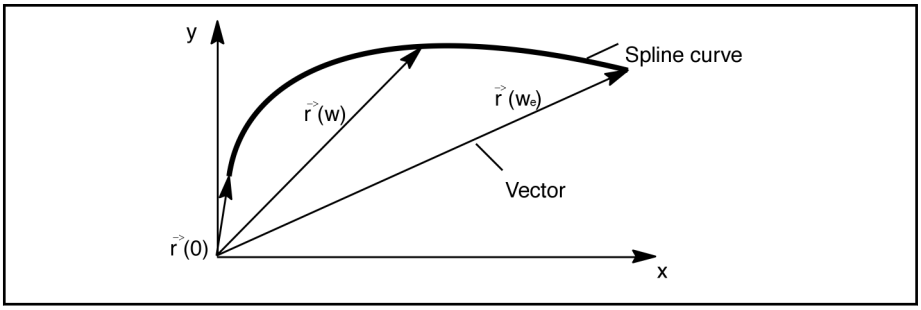


Fig. 9-6: Spline curve with the two coordinates x and y

Spline parameter length

The spline parameter length defines the length of the **definition interval** of **w**. The **definition interval** in the NC block is programmed using the rule **PL..**.

Example:

Spline_ParameterLength

G06 X(0.1, 1.25, 0.5, 0.73) B(0.0, -1.0, 0.1, -0.2) **PL0.6**

Mixed programming

Within each NC block, coordinates can either be programmed as **spline** or as **straight line**.

Example:

Mixed programming - Exemplary use case

N10 G1 X100 Y100 Z20 B10 U0 F1000

N20 SplineDef(3)

N30 G2 X200 Y100 Z30 I100 J0

N40 G06 **X(200,1,0.5,-0.1) Y(100,-1,0.2,0.2) B(10,1.0) Z60 U10** PL10

N50 **X(160,20) Y(310,4.0,0.2,-0.01) Z0**

N60 G1 X10 Y20 Z30 B40

N70 G06 Y(...) PL12

N20	A cubic spline is assigned to the syntax G06.		
N40	Switching from path type "helix" to "spline". X, Y, and B are provided with coefficient programming and are considered spline members in this block. Z and U are provided with end point programming and move linearly. The spline parameter "w" runs through the interval [0,10].		
N40	$X(w)=200+1.0w+0.5w^2-0.1w^3$	End point:	X(10)=160
	$Y(w)=100-1.0w+0.2w^2+0.2w^3$	End point:	Y(10)=310
	$B(w)=10+1.0w+0.0w^2-0.0w^3$	End point:	B(10)=20
N50	Second spline in the NC program. The spline parameter length is 10 (modal effect of PL from N40) X and Y are spline members Z moves linearly		

N60	Switching to path type "straight line"
N70	Switching to path type "spline" PL has to be programmed again

Rational monom splines

If the polynomial of the integral monomial spline is extended by a common **denominator polynomial**, the result is a rational monomial spline.

Vector Polynomial

$$\vec{r}(w) = \frac{\sum_{j=0}^{n} \vec{r}_j \cdot w^j}{\underbrace{\sum_{j=0}^{n} r_j \cdot w^j}_{\text{Denominator polynomial}}} \qquad w \in [0, w_e]$$

The denominator polynomial is programmed by coefficient programming in the NC block using the rule DN(...).

Example:

Specifying coefficients in the NC program for the channel axes X and B

G06 X(0.1, 1.25, 0.5, 0.73) B(0.0, -1.0, 0.1, -0.2) **DN(1.0,0.0,1.0)**

The following results for the channel axis:

$$X(w) = \frac{0.1 + 1.25w + 0.5w^2 + 0.73w^3}{1 + w^2}$$

and for

$$B(w) = \frac{0.0 - 1.0w + 1.0w^2 - 0.2w^3}{1 + w^2}$$

Example:

Use case

N00; ellipses as rational monomial spline
0001 A=100
0002 B=50
N10 G1 F50000 X[A] Y0 Z0
N20 SplineDef(2)
N25 G108 G06
N30 X(A,0,-A) Y(0,2*B) **DN(1,0,1)**PL1
N40 X(0,-2*A,A) Y(2*B,-2*B) **DN(2,-2,1)**
N50 X(-A,0,A) Y(0,-2*B) **DN(1,0,1**)
N60 X(0,2*A,-A) Y(-2*B,2*B) **DN(2,-2,1)**
N80 G1

All conic sections (ellipse,..) can be represented by rational monomial splines.

The spline are provided with the additional following characteristics:

- A C^0-continuity is present when the starting point of a spline is identical to the end point in the previous block. Inaccurate coefficient programming can result in differences between starting and end point. For the block transition, MP 8007 00010, a tolerance value e_0 can be set.
- The path velocity on the spline curve is approximated internally using the relation between the path length and the spline parameter w via a "path length function".

  In MP 8007 00020, a tolerance ε_v can be set as **maximum occurring relative path velocity error**. With increasing accuracy requirements, a smaller value has to be entered. This, however, also results in an increasing NC block cycle time. If the preset accuracy is not achieved, the NC issues a warning, but machining is continued.

  If MP 8007 00020 is set to 100%, the path length function is approximated by a straight line.
- Axes also involved in the spline curve can be removed from the feed computation (FeedGrp programming). In this case, however, the spline curve must not contain a path point in which all coordinates/axis computing the feed have a so-called zero tangent (i.e. no offset in axis direction). In this case, a runtime error is generated.

Example:

The space curve in the following figure is considered as curve of the axes X and Y.

The spline $\vec{r}(w)$ has two path points with a tangent along the x-axis. If X does not calculate the feed (FeedGrp(Y), this error case results.

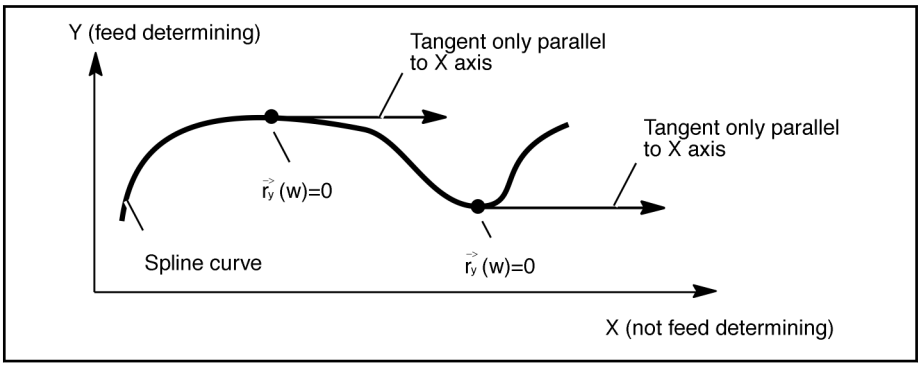


Fig. 9-7: Error case

9.1.3 Spline with data point programming and tangential transitions (C1-continuous cubic splines)

Function

The splines are calculated from the specified points (n data points) and the tangents of these points. The points are specified in the NC program.

The tangents in the points $\vec{r}i(i = 2,...n-1)$ are calculated from three subsequent points $\vec{r}i-1$, $\vec{r}i$, $\vec{r}i+1$.

The tangents at the points $\vec{r}1$, and $\vec{r}n$ result from the starting and end conditions (see chapter "Tangent calculation" on page 123).

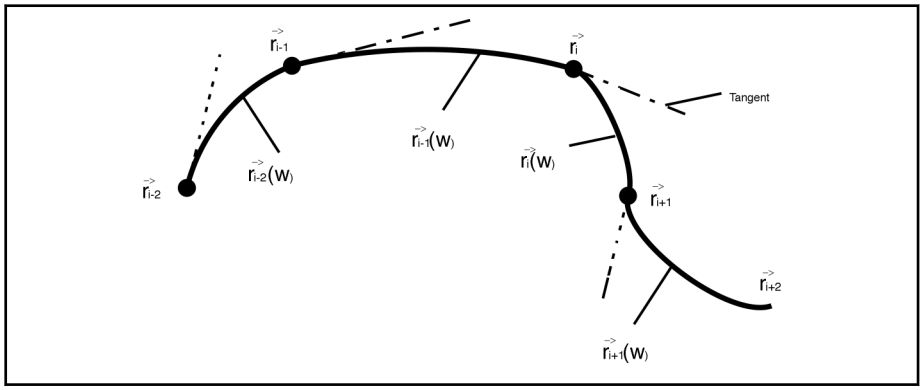


Fig. 9-8: Spline with data point programming and tangential transitions

The splines are provided with the following characteristics:

- From n points and one starting and end condition of the spline sequence, n-1 splines are created.

  A spline $\vec{r}i(w)$ connects the points $\vec{r}i$ and $\vec{r}i+1$.

- At the transition points, they are C^1-continuous (tangential) $\vec{r}i$.

- They are local, i.e. the change of a point $\vec{r}i$ only affects a limited number of splines in its neighborhood.

  These are the splines $\vec{r}i-2(w)$, $\vec{r}i-1(w)$, $\vec{r}i(w)$, $\vec{r}i+1(w)$.

  Vice versa, a spline is only defined by the points $\vec{r}i-1$, $\vec{r}i$, $\vec{r}i+1$, $\vec{r}i+2$.

Tangent calculation

The tangential transition is set via parameterization (see chapter "Function" on page 135 for spline ID). To this end, the following methods are available:

Bessel method:

By the programmed points $\vec{r}i-1$, $\vec{r}i$, $\vec{r}i+1$ a parabola (2nd degree spline) is defined. This results in the tangent in $\vec{r}i$.

Length and direction of the tangents depend on the selected spline parameterization (see chapter "Tangent calculation" on page 123)

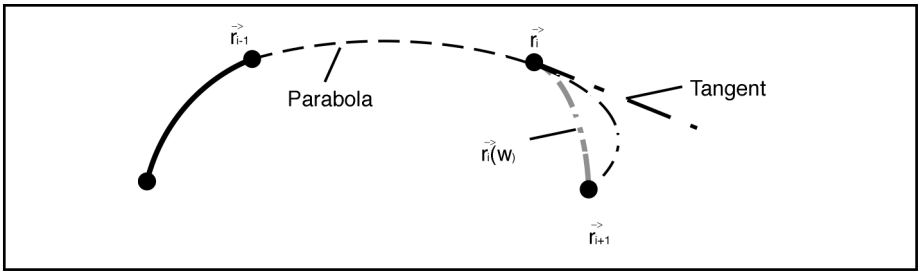


Fig. 9-9: Bessel method

Akima method:

The tangent in $\vec{r}_i$ indicates the direction of the angle bisector of both chords $\vec{r}_{i-1}\vec{r}_i$ and $\vec{r}_i\vec{r}_{i+1}$.

The tangent length depends on the selected spline parameterization (see chapter "Tangent calculation" on page 123).

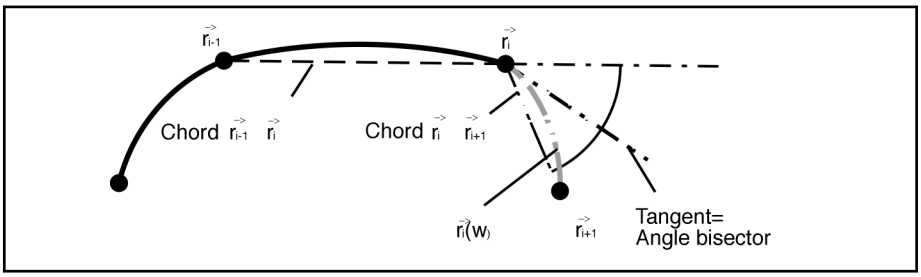


Fig. 9-10: Akima method

False position method:

The tangent in $\vec{r}_i$ points to the direction of the chord $\vec{r}_{i-1}\vec{r}_{i+1}$.

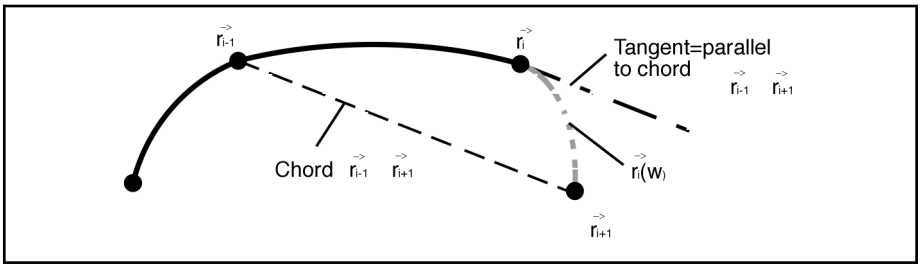


Fig. 9-11: False position method

Starting and end condition:

Boundary conditions can be programmed at any position within a spline sequence, usually at the start and the end.

It is programmed with the syntax

- "Start boundary condition" **SBC**((<Type>[,<values>]) and
- "End boundary condition" **EBC**((<Type>[,<values>]).

The following options are available for the starting and end conditions:

Type = 1: Specifying the tangential direction at the starting/end point of the spline sequence. Only the relation of the values of the spline members is significant. The tangential length is calculated internally (for chord parameterization, the tangential length is 1).

Type = 2: Specification of the second derivation r_i __ after w at the starting/end point of the spline sequence.

Type =10: The spline tangent direction runs along the connecting line "starting point - end point".

Type = 11: The spline tangent direction runs along the connecting line "starting point - endpoint of the previous block (with SBC) or the subsequent block (with EBC).

By using SBC and EBC within a spline sequence, edges can be specifically generated on the spline curve. When no starting and end condition is programmed, the so-called natural boundary condition (2,0,...,0) or EBC(2,0,...,0) is used.

When type 1 and 2 is used, enter a value into the <Values> list for each spline member. If vector orientation is active as spline, the boundary conditions also comprise the orientation vector.

Example:

SplineDef(1213,x,y,z,O)

G06 ..

... EBC(1,1.0,1.5,-0.5,0.0,-1.0.0.0)

The first value triple (1.0,1.5,-0.5) is the tangent direction of the spatial motion x, y, z.

The second triple (0.0,-1.0.0.0) is the tangent direction of the orientation vector.

Spline parameter length: The spline parameter length takes the data point width and the spline curvature into account. Depending on the application, three different parameterizations are valid:

- **Equidistant** parameterization (w=1) :

  This parameterization is suitable for identical spline data point distances. If the spline data points have different distances to each other, this results in the "bulging" of the curve, above all in case of the continuity curvature (see next figure).

- **Chord** parameterization $w = \left| \vec{n}_{i+1} - \vec{n}_i \right|$:

  If the spline data points have different distances to each other, the distance and direction of the data points is taken into account.

- **Centripetal** parameterization $w = \left| \sqrt{\vec{n}_{i+1} - \vec{n}_i} \right|$:

  This parameterization is used when the spline data points have different distances to each other and the spline curves have strong curvatures.

The better the position and the distances of the data points are taken into account, the smoother the curve shape and the lower the number of unexpected curvatures. Accordingly, it can be necessary to combine the above parameterizations depending on the specified data points.

Instead of using the parameterization calculated by the NC, the spline parameter length can also be specified manually via the PL<w> parameter.

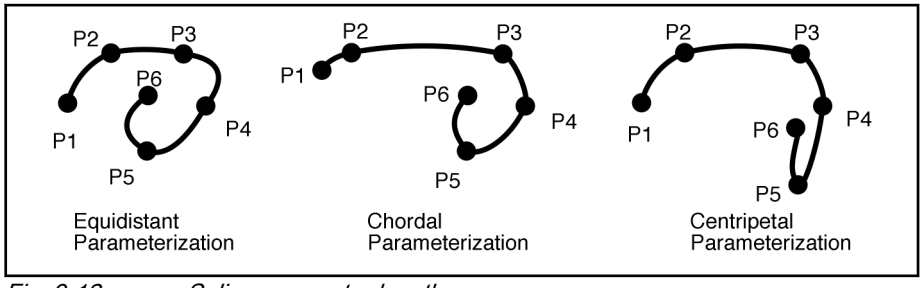


Fig. 9-12: Spline parameter length

Motion of the tool orientation

Not only the tool tip but also the **tool orientation** can traverse on a spline curve if the tool in question is rotation-symmetric and programmed together with a corresponding 5-axis transformation. A distinction is made between vector orientation and linear orientation

(see also chapter 3 "Drives (axes, spindles)" on page 13).

☞ The orientation of non-rotation-symmetric tools (tensor orientation in connection with a corresponding 6-axis transformation) cannot be programmed as spline.

- **Vector orientation as C^1-continuous cubic spline:**

  With the vector orientation as C^1-continuous cubic spline, a spline in the six coordinates x, y, z, ρ_x, ρ_y, ρ_z is generated in each NC block with a common parameterization (equidistant, chordal, centripetal). The tangential method (Bessel, Akima, and False position method) specified in the "SplineDef" function is also used for the orientation vector $\vec{\rho}(w)$. At the block transitions, the path motion (x,y, z) as well as the orientation vector C^1 are continuous.

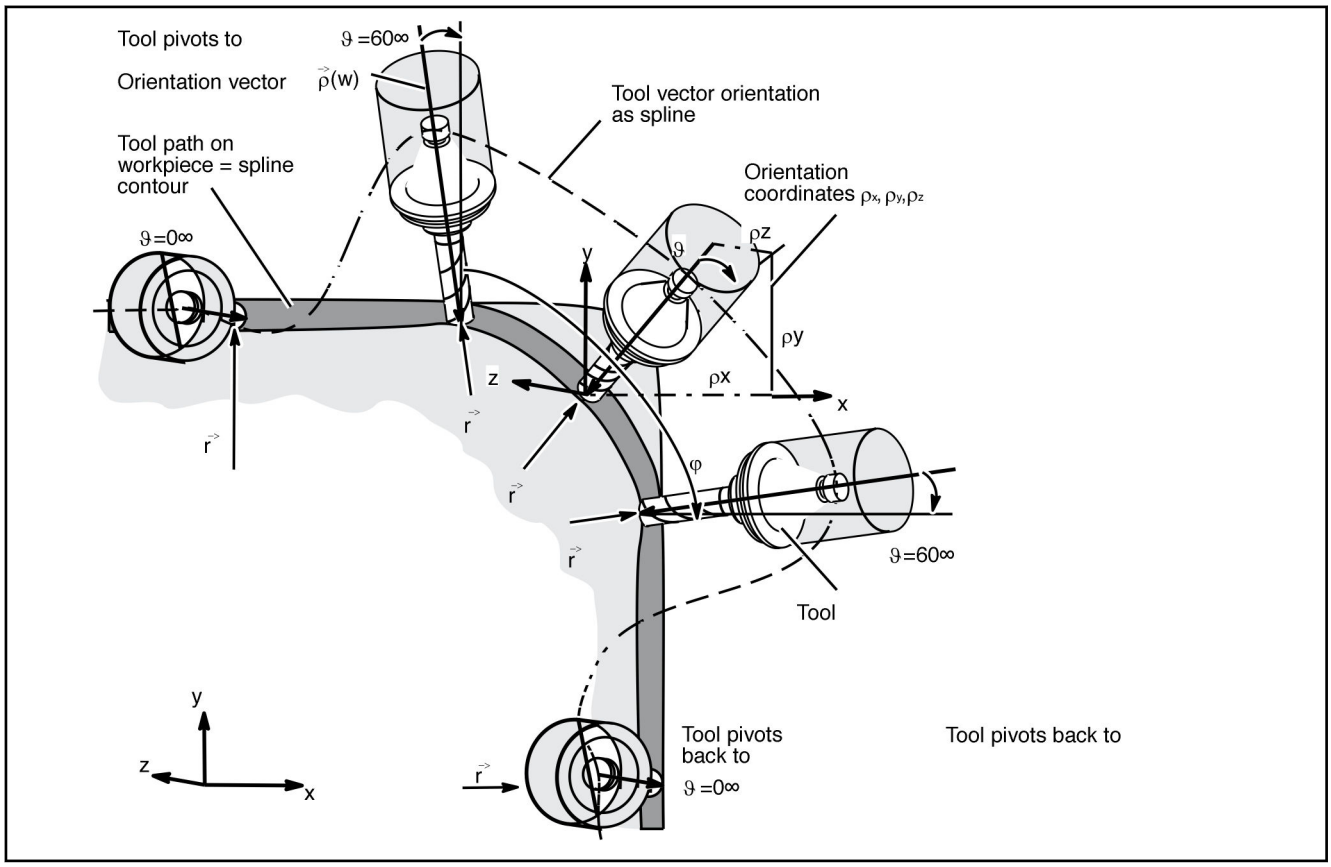


Fig. 9-13: Vector orientation

- **Linear orientation motion as C^1-continuous cubic spline**

  For the linear orientation motion, splines are created in the five coordinates x, y, z, phi and theta. The splines are mathematically equivalent to

the splines when axis moves in three linear and two rotary axes (see chapter "Motion of the tool orientation" on page 126).

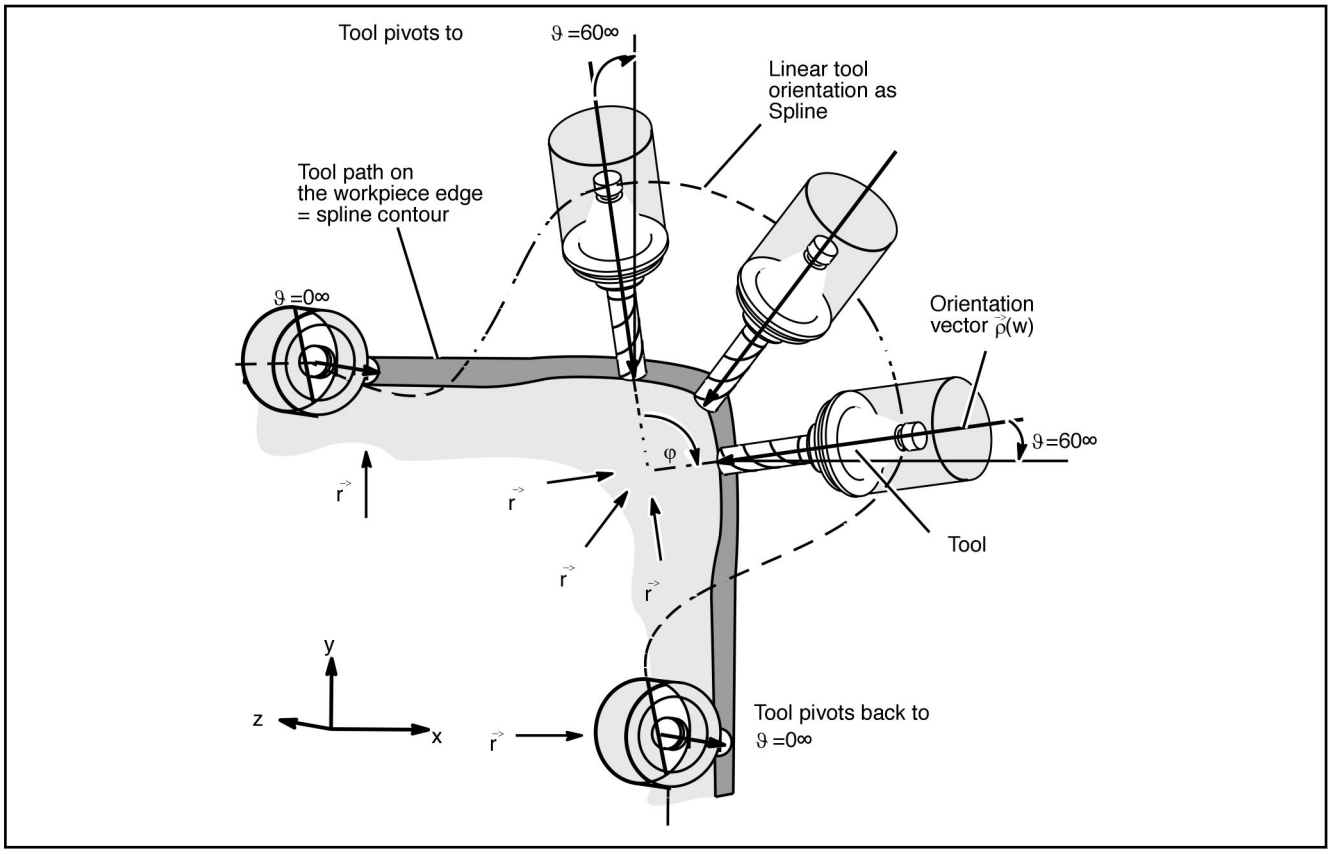


Fig. 9-14: Linear orientation motion

The spline are provided with the additional following characteristics:

- The $\vec{r}$ motion ($\vec{r} = (x, y, z)$) depends on the $\vec{\rho}$ motion and vice versa. This means that e.g. in block in which only $\vec{\rho}$ is programmed, an additional $\vec{r}$ motion can occur if not all $\vec{r}$ spline determining points $\vec{n}_{i-1}$,..., $\vec{n}_{i+2}$ are identical.
- The spline parameter $w \in [0, w_e]$ applies to $\vec{n}_i(w)$ and to $\vec{p}_i = (w)$.

Example:

Linear orientation motion

N0 SplineDef(**1213**,x,y,z,O)

;Linear coordinates and orientation are spline members **(C1)**

; 5-axis transformation with **vector orientation**

N0 **Coord(1)**

;alternative N0 **Coord(3)**; 5-axis transformation with **linear orientation**

N0 x0 y0 z0 phi0 theta0

N0 G06 G8

N1 y20

N2 y40 theta30

N3 y60 theta0

N4 y80

N5 ...

N1	Theta orientation motion of the tool starts already in this block.
N4	Theta orientation motion of the tool ends only in this block.

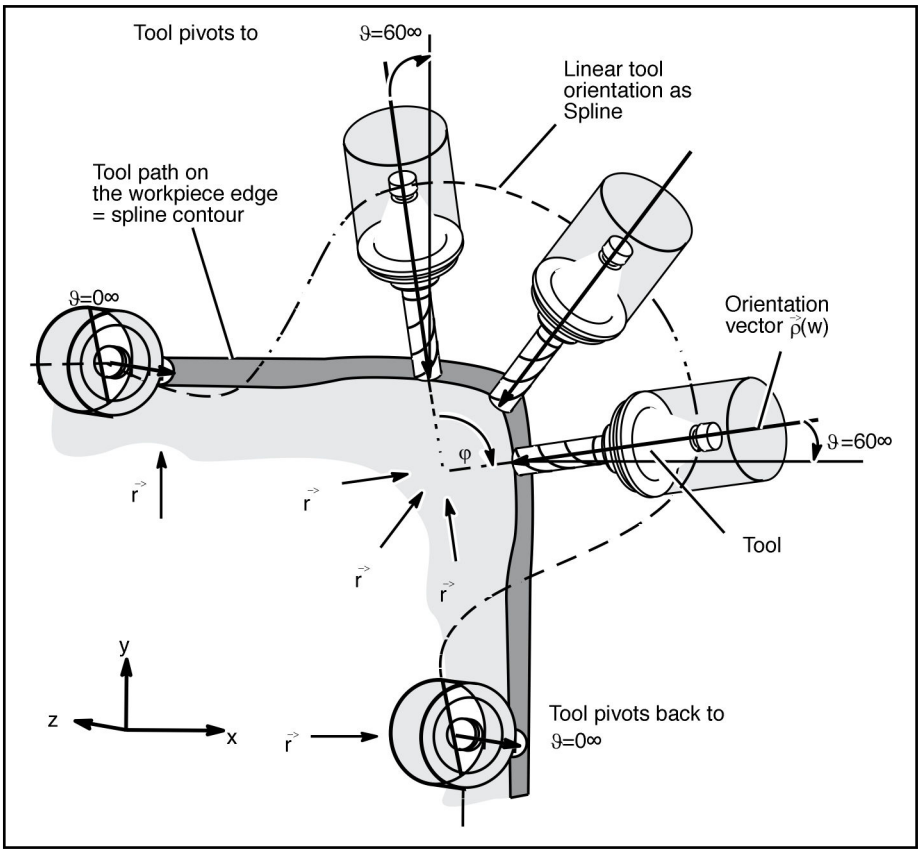


Fig. 9-15: Example

9.1.4 Cubic hermite splines

General information

☞ The prerequisite for Hermite splines is a corresponding CAD/CAM system which can generate points as well as their derivations

A cubic spline is generally presented by the following monomial polynomial

$$\vec{r}(w) = \vec{c}_0 + \vec{c}_1 \cdot w + \vec{c}_2 \cdot w^2 + \vec{c}_3 \cdot w^3 \; ; \; w \in [0, W]$$

with the curve parameter w and the four spline coefficients c_0, c_1, c_2, c_3. c_0 is the starting point and c_1 the starting tangent. C_2, c_3 cannot be directly interpreted geometrically.

Hermite splines can be presented mathematically equivalent to the monomial polynomial by means of Bézier polynomials

$$\vec{r}(w) = \sum_{i=0}^{3} \vec{P}_i B_{i,3}(u) \quad ; \quad u \in [0,1]$$

with the Bernstein basic polynomials

$$B_{i,3}(u) = \binom{3}{i} u^i (1-u)^{3-i}$$

For Hermite splines, a CAD/CAM system generally calculates the data points r_i (i=1,..,n) and the tangents at the data points (derivations with respect to the path) and provides them to the NC (cf. following figure).

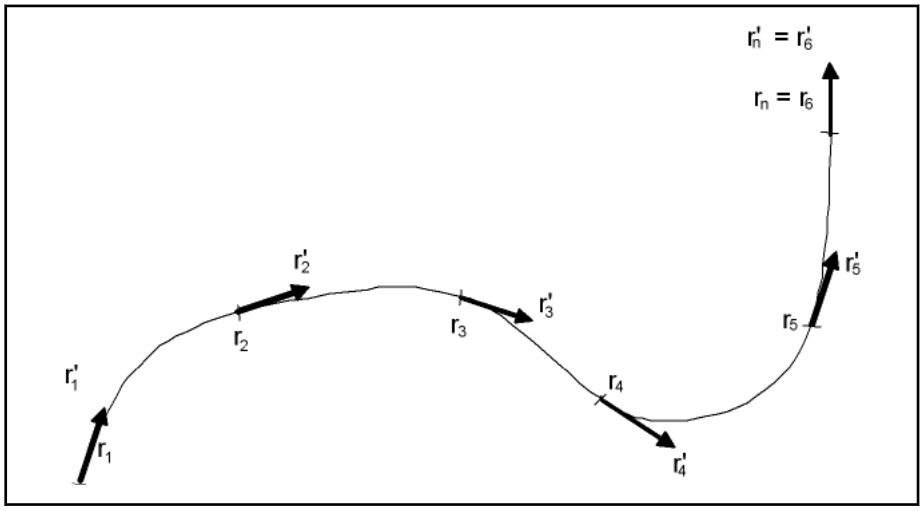


Fig. 9-16: Hermite spline

Thus, there are four conditions r_i, r'_i, r_{i+1} , r'_{i+1} control-internally for each cubic partial polynomial $r_i(w)$ used to compute the coefficients. The block transitions are at least c^1-continuous. At the starting point r_1, a starting tangent with the value 0 is assumed.

Each channel axis or coordinate involved in the spline motion is programmed with its position value and its derivation value. These values are specified in brackets following the channel axis or coordinate name. As with splines with coefficient programming, the parameter length can also be programmed

Example:

N10 SplineDef(1143)	A Hermite spline is assigned to G6
N20 G1 X100 Y100 Z0 F60	Approach initial position
N25 G6 CPI(1) CVP	Activate spline, curve parameter interpolation "CPI(1)" and jump interpolator "CVP"
N30 G6 N40 X(200.0,1.0) Y(100.0,-1.0) PL10	Parameter length = 10 (corresponds to 10 s because of F = 60)
N50 X(160.0,20.0) Y(310.0,4.0) Z(100.0,2.0)	Parameter length = 1 (since no PL programmed)

N70 F60000 X(210.0,2.0) Y(220.0,2.1) PL3	Parameter length = 3 (corresponds to 3 ms because of F=60 000)
N90 G1 X10 Y20 Z30	Complete spline, approach end position

Hermite splines are provided with the following characteristics:

- From n points and n tangents of the spline sequence, n-1 splines are created. A spline $r_i(w)$ connects the points r_i and r_{i+1}.
- At the transition points, they are c1-continuous (tangential).
- They are local, i.e. changing a point r_i only affects a limited number of splines in its neighborhood In this case, the splines $r_{i-1}(w)$, $r_i(w)$ and $r_{i+1}(w)$.
- Under two additional conditions, the parameter length PL can be interpreted in case of Hermite splines. Therefore, as specified in line N25 in the example program, the curve parameter interpolation "CPI(1)" and the jump interpolator "CVP" have to be activated. With F=60, the PL value is the time in [sec] and with F=60000, the PL value is the time in [ms].

9.1.5 Spline with data point programming and curvature-continuous transitions (C2-continuous cubic splines)

Function

The splines are calculated from the specified points $\vec{r}_i$ with i=1,..,n and one starting and end condition of the spline sequence.

The splines are provided with the following characteristics:

- From n points and one starting and end condition of the spline sequence, n-1 splines are created. A spline $\vec{r}_i(w)$, $w \in [0, w_e]$ connects the points $\vec{r}_i$ and $\vec{r}_{i+1}$.
- At the transition points, they are C^2-continuous (curvature-continuous) $\vec{r}_i$.
- They are global, i.e. the modification of one point $\vec{r}_i$ influences all splines of the spline sequence. However, the influence significantly decreases with the distance to the point $\vec{r}_i$.

Due to the global spline characteristic, calculating the spline requires a significantly higher look-ahead range than for C^1-continuous cubic splines.

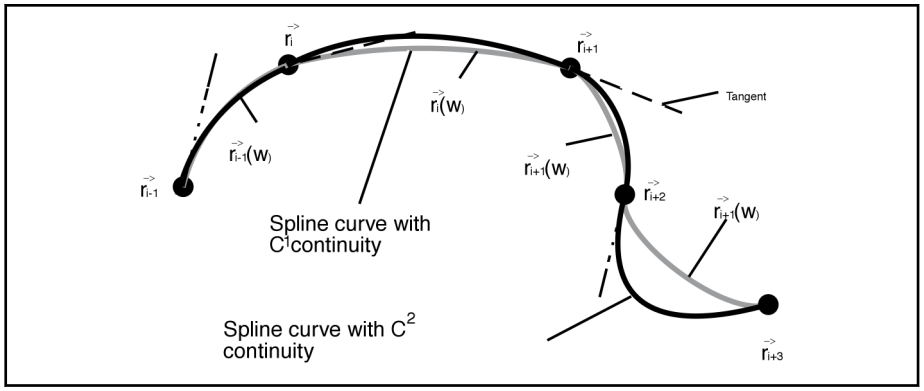


Fig. 9-17: Spline curve

Starting and end condition:

Apart from the boundary conditions type 1, 2, 10 and 11, which are also possible for C^1-continuous splines (see "Starting and end condition:" on page 124), there are two more conditions:

- Type = 3:

  The **De-Boor boundary condition** links the second derivatives at the first two data points or at the two last data points.

  SBC(3,<v>) or EBC(3,<v>)

  v is typically =1

- Type = 4:

  The **periodic boundary condition** is used for closed curves (e.g. ellipses). In this case, the last and the first point of the spline sequence have to be identical.

  The periodic boundary condition cannot be combined with another boundary condition, i.e. with SBC(4) at the beginning, an EBC(4) at the end is mandatory.

  The spline sequence of SBC(4) to EBC(4) has to be in the look-ahead area of the NC. If this is not the case, a runtime error is generated.

SBC and EBC can also be programmed within a spline sequence (G06,..,G1). Thus, the original sequence is divided into two partial sequences to specifically generate corners within the spline curve.

When no start and end condition is programmed, the natural boundary condition SBC(2,0,…0) or EBC (2,0,...,0) is used.

Spline parameter length:

See C^1-continuous cubic spline (see chapter "Motion of the tool orientation" on page 126).

Motion of the tool orientation as C2-continuous cubic Spline:

The same conditions as for the C^1-continuous cubic spline applies (see chapter "Motion of the tool orientation" on page 126).

The vector orientation can also by programmed as C^2-continuous cubic spline. Two variants, which can be specified in the machine parameter 800700099[3], are distinguished:

- **Vector orientation (variant 1) as a common C^2-cubic spline with the spatial spline of the linear coordinates x, y, z.**

  In this variant, a spline in the six coordinates x, y, z, ρ_x,ρ_y,ρ_z is used with a mutual parameterization.

In this case, the spline vector orientation is provided with the following characteristics:

– The $\vec{r}$ motion ($\vec{r} = (x, y, z)$) depends on the $\vec{\rho}$ motion and vice versa. This means that e.g. in block in which only $\vec{\rho}$ is programmed, an additional $\vec{r}$ motion occurs (see "Vector orientation (variant 2)" on page 132.

– The spline parameter $w \in [0, w_e]$ applies to $\vec{r}(w)$ and to $\vec{\rho}(w)$.

- **Vector orientation (variant 2) as an independent C²-cubic spline which is independent of the spatial spline of the linear coordinates x, y, z.**

  This variant requires a high number of blocks in the look-ahead range. The number of blocks (MP 706000120[3]) is to be approximately three times as high as the variant 1.

  Variant 2 is provided with the following specific characteristics:

  – The $\vec{r}$ motion ($\vec{r} = (x, y, z)$) is independent from the $\vec{\rho}$ motion. A $\vec{\rho}$ motion only takes place in blocks with a starting orientation unequal to the end orientation (see fig. 9-18 "Vector orientation - according to variant 1" on page 134).

  – $\vec{\rho}(w)$, $w \in [0, w_e]$ always has an individual chordal parameterization. $\overrightarrow{w_e}$ is the angle between the starting and the end orientation vector of the block.

  This $\overrightarrow{w_e}$ of the spline $\vec{p}_i = (w)$ differs from w_e of the spline $\vec{r}(w)$.

  – Special boundary conditions for the orientation vector $\vec{\rho}$ cannot be programmed.

Example:

Vector orientation (variant 2)

When the program is processed with the setting MP 8007 00099[3] = 1, the Theta-y-course results as shown in the next figure.

Vector orientation and path motion as common spline;

(Variant 1)N0 SplineDef(**2203**,x,y,z,O)

;Linear coordinates and orientation are spline members (C²)

N0**Coord(1)**; 5-axis transformation with **vector orientation**

N0 x0 y0 z0 phi0 theta0

N0 G06 G8

N1 y20

;Theta orientation motion of the tool already starts in N1

N2 y40 theta30

N3 y60 theta0N4 y80

;Theta orientation motion of the tool ends in N4

N5 ...

When the program is processed with the setting MP 8007 00099[3] = 2, the Theta-y-course results as shown in the next figure. A $\vec{\rho}$ motion also results in the blocks N1 and N4 in which the orientation vector has not been programmed.

;vector orientation as independent spline

;(Variant 2)

N0 SplineDef(**2203**,x,y,z,O)

;Linear coordinates and orientation are spline members (C^2)

N0**Coord(1)**; 5-axis transformation with **vector orientation**

N0 x0 y0 z0 phi0 theta0

N0 G06 G8

N1 y20

N2 y40 theta30

;Theta orientation motion of the tool N3 y60 theta 0 starts in N2

;Theta orientation motion of the tool N4 y80 ends in N3

N5 ...

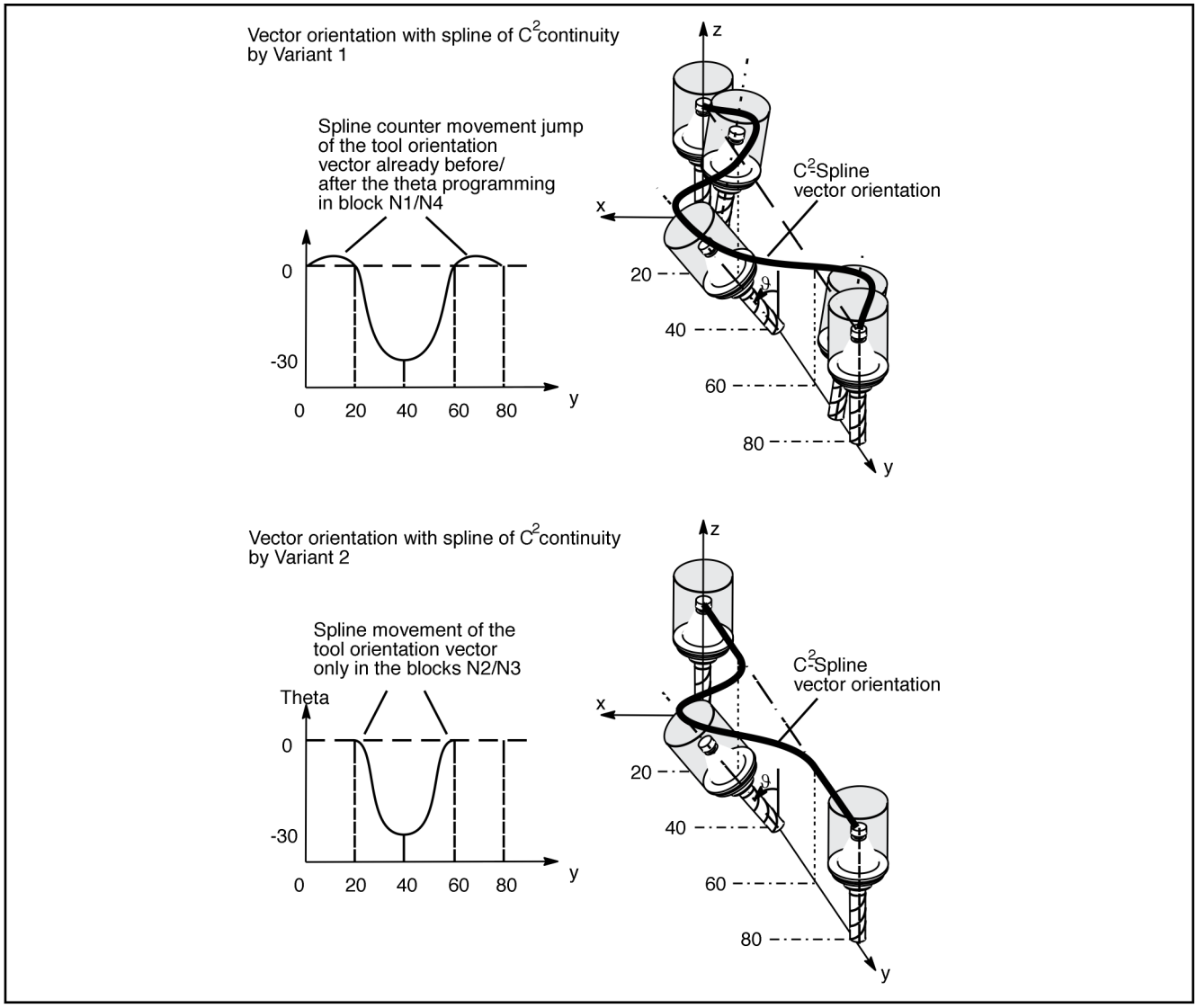


Fig. 9-18: Vector orientation - according to variant 1

- **Linear orientation motion as C^2-continuous cubic spline**
  - For the linear orientation motion as C^2-continuous cubic spline, applies the same as for the linear orientation motion as C^1-continuous cubic spline (see from chapter 9.1.3 "Spline with data point programming and tangential transitions (C1-continuous cubic splines) " on page 122).
  - With long spline sequences (number of blocks between SBC and EBC), which are not fully in the look-ahead range (see chapter "Relevant machine parameters (MP)" on page 147) of the NC, a suitable method is used to generate two or more partial spline sequences which again merge C^2-continuously. Thus, the splines can be used for contours of any length (free-from surfaces machining).

    The number of partial sequences are to be kept to a minimum, i.e. the MP 800700099[2] is to be set as high as possible.
  - With periodic boundary conditions, a sequence interruption is not possible. Here, the whole spline sequence has to be in the look-ahead range.

9.1.6 Spline with checkpoint programming (B-spline)

Function

B-splines are created by so-called "checkpoints". In contrast to the C^1/C^2 splines, the spline curve shape does not intersect specified interpolation points but "approximates" the specified "checkpoints" (see the following figure).

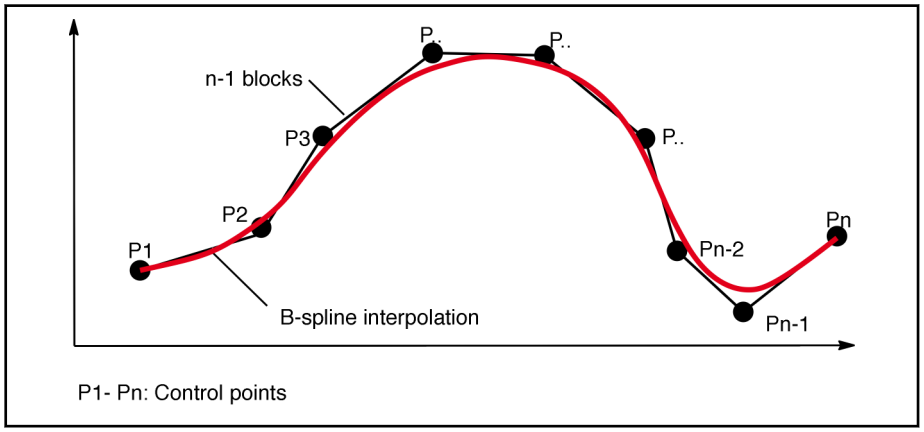


Fig. 9-19: Spline with checkpoint programming (B-spline)

NURBS (Non-Uniform Rational B-Spline) are a special design of B-splines. They are referred to as "B-splines". For the special characteristics of NURBS, refer to the relevant literature.

The B-splines are provided with the following characteristics:

- From n points, i.e. from "n-1" NC blocks, "n-p" splines are generated, "p" describing the degree of B-splines (see the following example).
- Generally, **smaller** curvatures are generated than in case of the global cubic splines, since - except for some special cases - the B-splines do not pass the checkpoints. They overshoot less, but are still provided with a continuity curvature at the transition.
- By programming point weights with the PW function, the splines in the environment of a point can be changed. A point weight >1 "drags" the spline to the point, one point weight <1 "pushes" it away from the point. This is called a rational B-spline or NURBS.
- B-splines have a local effect, i.e. the modification of a point influences a number of splines which depends on the degree of the B-spline.
- As a starting and end boundary condition, type 10 (tangent along the starting-end point connecting line, also see chapter 9.1 "Spline " on page 115) is available.

Checkpoints: For a b-spline curve from degree p, a minimum of "p+1" different checkpoints have to be defined. Within the spline sequence, there have to be at least "p" NC blocks (with end point not equal to the starting point).

Double checkpoints are not supported, i.e. the NC filters them. Thus, a spline without traversing motion is generated.

Spline parameter length: See C^1-continuous cubic spline (see chapter "Motion of the tool orientation" on page 126).

B-splines can be parameterized in an equidistant, chordal and centripetal manner (also see chapter "Motion of the tool orientation" on page 126). The equidistant parameterization is called uniform B-splines. In the other two ca-

ses, non-uniform B-splines. Furthermore, the node vector (see chapter 9.1.6 "Spline with checkpoint programming (B-spline)" on page 135) can specifically be changed by programming the spline parameter length PL.

Point weights: A point weight (**P**oint **W**eight) can be assigned to the checkpoints using the syntax **PW ...**. By different scaling (weighting), it is thus possible to influence the shape of the spline curve at each checkpoint (depending on the degree of the spline, also in the environment of the checkpoint). A point weight of 0 is not permitted.

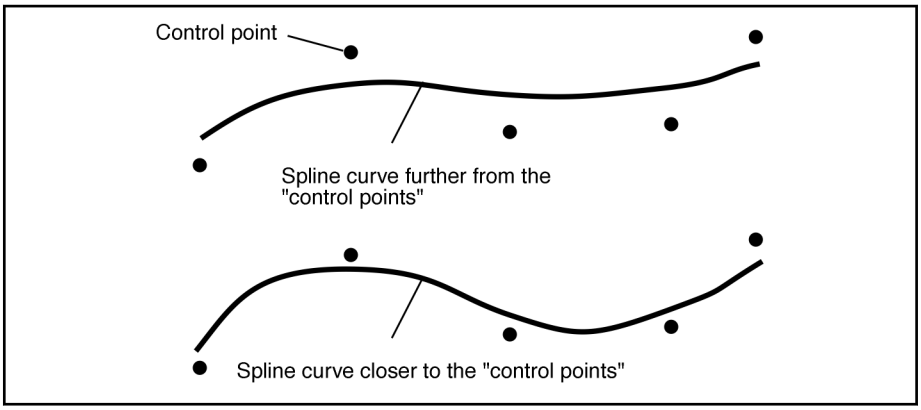


Fig. 9-20: Checkpoints

Example:

Third degree B-spline:

;Selection of third degree B-spline:

N0 SplineDef(3213,x,y,z,O)

; 5-axis transformation with vector orientation:

N0 Coord(1)

N0 x0 y0 z0 phi0 theta0

N0 G06 G108

N1 x10y20z11 O(0.1,0,1.0)

N2 x.. y.. z.. O(..)

;First spline:

N3 x.. y.. z.. O(..)

;Second spline, point weight 2.3:

N4 x.. y.. z.. O(..) PW2.3

... .

... ...

;18th spline:

N20 x,.. y.. z.. O(..)

N21 G1

Explanation:

N0	Selection of a third degree B-spline
N1	Starting point of the spline curve tangentially to the connecting line
	Starting point - End point
N1-N2	Non-traversing block

N1-N20	Specify 21 checkpoints
N3-N20	18 splines which are processed
N20	End point of the spline curve tangentially to the connecting line. Starting point - End point.

Detailed B-spline description (for developers)

In the following, the implementation of a B-spline sequence from a CAD/CAM file into the NC program format is described. Thus, this chapter is especially intended for post-processor developers.

Definition and properties of b-spline curves in the NC

The NC supports B-splines up to degree p = 5. Degree p = 3 is the standard.

From degree p, a B-spline curve (NURBS curve) is defined by

- n+1 checkpoints (for n NC blocks),
- A node vector with n+p+2 elements
- n+1 point weights
- Node range PL = 1 (default)
- Point weight PW = 1 (default)

The node vector is uniform with the node range equal to 1.

It look as follows:

U = {0,...,0,1,2,..,n+1-p,...,n+1-p}

The node vector starts with p+1 zeros; in the end, it has p+1 nodes with the value n+1-p. The number of node elements of the node vector is m = n+p+2.

The number of splines is n+1-p, i.e. it equals the number of node ranges which are unequal to zero.

For a cubic uniform B-spline p = 3, the default node vector is generally as follows:

U = {0,0,0,0,1,2,...,n-3,n-2,n-2,n-2,n-2}

The number of splines (number of usable nodes) is n-2.

Example:

☞ In the following example for an NC program with 5-axis machining, <P0>, <P1> ... stands for a syntax expression as follows: x.. y.. z.. O(..,..,..)

The three linear coordinates x, y, and z describe the position of the tool tip (TCP). The orientation vector of the tool is described by O(..) (also refer to Vector orientation motion).

Thus, the following results:

p = 3:	Degree of the cubic B-spline
n = 7:	Number of spline NC blocks
n+1 = 8:	Number of checkpoints
n+1-p = 5:	Number of splines (usable nodes)
n+p+2 = m = 12:	Number of node elements of the node vector

The node vector is as follows: U = {0,0,0,0,1,2,3,4,5,5,5,5}

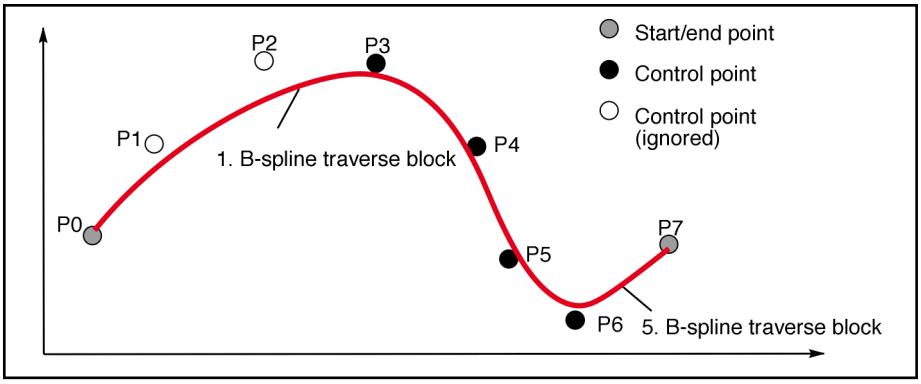


Fig. 9-21: Example

N01 SplineDef(3213,,x,y,z,O)	;Cubic uniform B-spline for the x-, y-, z- and o-co-ordinates
N02 COORD(1)	;5-axis transformation with vector orientation
	;(e.g.. 3232201)
N10 G1 <P0>	;Approaching the checkpoint P0
N11 G06 <P1>	;Spline on, checkpoint P1
N12 <P2>	;Checkpoint P2
N13 <P3>	;Checkpoint P3,
	;First spline traversing block
N14 <P4>	
N15 <P5>	
N16 <P6>	
N17 <P7>	;Checkpoint P7; ;5. Spline traversing block
N18 G1	;Spline off

As the number of the spline NC blocks (N11 to N17) is higher by two than the number of splines, two internal non-traversing blocks are generated at the start of the spline sequence (N11 and N12).

Possible methods to influence the B-spline:

- **Changing the node vector**

  Use the syntax **PL**, to change the node vector (see example 1):

  – A node range has to be programmed only after the first traversing block (in the example below, from N13).
  – Programming PL in the non-traversing blocks N11 and N12 has no effect.
  – Multiple nodes are generated with PL0 (see example 2).

Example 1:

Changed node range with PL.

N01 Spline-Def(3213,,x,y,z,O)	;Cubic B-spline for x, y, z and o
N02 COORD(1)	;5-axis transformation with vector orientation ;(e.g.. 3232201)
N10 G1 <P0>	;Approaching the checkpoint P0
N11 G06 <P1>	;Spline on, first checkpoint

```
N12 <P2>                ;2. Checkpoint
N13 <P3> PL2.3          ;1. Node range is 2.3
N14 <P4>                ;2. Node range is 1.0. (default)
N15 <P5> PL0.8          ;3. Node range is 0.8
N16 <P6> PL1.2          ;4. Node range is 1.2
N17 <P7> PL2.0          ;5. Node range is 2.0
N18 G1                  ;Spline off
```

The node vector is as follows:

U = {0,0,0,0,2.3,3.3,4.1,5.3,7.3,7.3,7.3,7.3}

Example 2:

Changed node range by multiple node with PL0

```
N01 SplineDef(3213,,x,y,z,O)  ;Cubic B-spline for the space curve x, y, z
                              ;And the orientation vector p
N02 Coord(1)                  ;5-axis transformation with vector orientation
                              ;(e.g.. 3232201)
N10 G1 <P0>                   ;Approaching the checkpoint P0
N11 G06 <P1>                  ;Spline on, checkpoint P1
N12 <P2>                      ;Checkpoint P2
N13 <P3> PL2.3                ;1. Node range is 2.3
N14 <P4> PL0                  ;2. Node range is 0.0
                              ;(multiple node)
N15 <P5> PL0.8                ;3. Node range is 0.8
N16 <P6> PL1.2                ;4. Node range is 1.2
N17 <P7> PL2.0                ;5. Node range is 2.0
N18 G1                        ;Spline off
```

This results in the node vector

U = {0,0,0,0,2.3,2.3,3.1,4.3,6.3,6.3,6.3,6.3}

Tab. 9-1: Example 2

Note that a double node reduces the number of the splines by 1. Thus, the number of the generated splines in example 2 equals 4.

Changing of the point weights: A weight is assigned to each checkpoint (except for <P0>) via the syntax **PW**. Default is PW=1.

Example:

```
N01 SplineDef(3213,,x,y,z,O)
N02 Coord(1)
N10 G1 <P0>             ;Approaching the checkpoint P0
N11 G06 <P1> PW1.8      ;P1 weight = 1.8
N12 <P2> PW 2.5         ;P2 weight = 2.5
N13 <P3> PL2.3          ;P3 weight = 1
N14 <P4> PL0            ;P4 weight = 1
N15 <P5> PL0.8          ;P5 weight = 1
```

N16 <P6> PL1.2 **PW0.6** ;P6 weight = 0.6
N17 <P7> PL2.0 ;P7 weight = 1
N18 G1 ;Spline off

Example implementation of a CAD/CAM nurbs sequence into the NC program format of the MTX:

Example: APT nurbs format of Catia version 5 for 5-axis milling.

For a Siemens postprocessor, the APT format structure is as follows:

```
BEGIN NURBS_SIEMENS(D=3,F=xxxx,AXIS=VAR,LENGTH=100.00);

N0,XT=xt0,YT=yt0,ZT=zt0,XH=xh0,YH=yh0,ZH=zh0,DK=dk0,W=w0;
N1,XT=xt1,YT=yt1,ZT=zt1,XH=xh1,YH=yh1,ZH=zh1,DK=dk1,W=w1;
N2,XT=xt2,YT=yt2,ZT=zt2,XH=xh2,YH=yh2,ZH=zh2,DK=dk2,W=w2;

../..

Nn,XT=xtn,YT=ytn,ZT=ztn,XH=xhn,YH=yhn,ZH=zhn,DK=dkn,W=wn;
END NURBS;
```

The parameters of the **postprocessor APT format** can be implemented into the MTX format as follows:

Parameter	Postprocessor APT format elements	MTX program format Bosch Rexroth
Header	BEGIN NURBS_SIEMENS(D=3,F=xxxx,AXIS= VAR,LENGTH=100.00);	SplineDef(3213,x,y,z,O) Coord (1)
Degree of the nurbs	D	p
Path feed	F	F
Axis type	AXIS = VAR	z. B. x, y ,z, O
Checkpoint position and tool orientation	Checkpoint position = position of the tool tip (XTi,YTi,ZTi) and another position along the tool symmetry axis (XHi, YHi, ZHi) LENGTH = distance between XTi,YTi,ZTi and XHi,YHi,ZHi With i = 0 to n	x<xti> y<yti> z<zti> O(<xhi-xti>,<yhi-yti>,<zhi-z) With i = 0 to n.
Node range	dki with i = 0 to n	PL<dki> with i = 0 to n
Point weight	wi with i = 0 to n	PW<wi> with i = 0 to n

Tab. 9-2: MTX format

In the postprocessor, the checkpoint position is represented by the position of the tool tip (XTi,YTi,ZTi) and another position along the tool symmetry axis (XHi, YHi, ZHi).

Converted into MTX format, the postprocessor checkpoint position defines the MTX position of the TCP and an orientation vector belonging to the TCP position:

<Pi> = x<xti> y<yti> z<zti> O(<xhi-xti>,<yhi-yti>,<zhi-zti>)

It is not necessary that the orientation vector is standardized with "One", i.e. LENGTH is not considered. In the MTX format, the postprocessor NC program above is as follows:

```
SplineDef(3203,,x,y,z,O)  ;B-spline for x, y, z and o
COORD(1)                  ;Selection of 5-axis transformation with vector orientation
```

```
G1 <P0>                          ;Approaching the starting point of the nurbs sequence
G06 <P1> PL<dk0> PW<w1> F..      ;Spline on, PL <dk0> can be omitted
<P2> PL<dk1> PW<w2>              ;PL<dk1> can be omitted
<P3> PL<dk2> PW<w3>
..
<Pn> PL<dkn-1> PW<wn>
G1 <Pn+1>                        ;Spline off, possibly Pn+1 = P0 of the next nurbs sequence
```

9.1.7 Approximate B-splines with tolerance specification

The NC program creation (CAD/CAM - postprocessor sequence) to process free-form surfaces does often not result in the optimum NC programs with regard to curve smoothness and point distances. These deficits cannot be compensated easily by the NC. An important reason is the frequently missing or lacking support of the spline processing. Instead of splines, short linear blocks (G1) are programmed for all space curves and orientation motions causing the following problems:

1. The programmed curve points are often inaccurate, e.g. NC block points are only generated with two numbers of decimals.
2. The distance between adjacent points is so small that a reduction of the path velocity is executed by the NC. Thus, the machining time increases unnecessarily. Variable point distances can cause an disturbed path velocity if blocks with and without velocity reduction change multiple times.
3. The simple solution of using spline interpolation (G6) instead of linear interpolation (G1) can often not be used, since a few labeled blocks actually have to be traversed with G1 due to reasons of geometry

Approximating b-splines solves these problems.

These are the properties functions:

The points and orientations given in the NC program are approximated by a smooth b-Spline curve with a degree between 2 and 5. The way of approximation ensures that the distance of each programmed point of the b-Spline curve is smaller than the programmed dimensional tolerance. A "tube" with a radius equal to the dimensional tolerance is located along the curve. This tube crosses all programmed points. All inaccurate programmed points are smoothed and few, long spline blocks are generated according to the required accuracy. The function is mainly used to machine free-form surfaces.

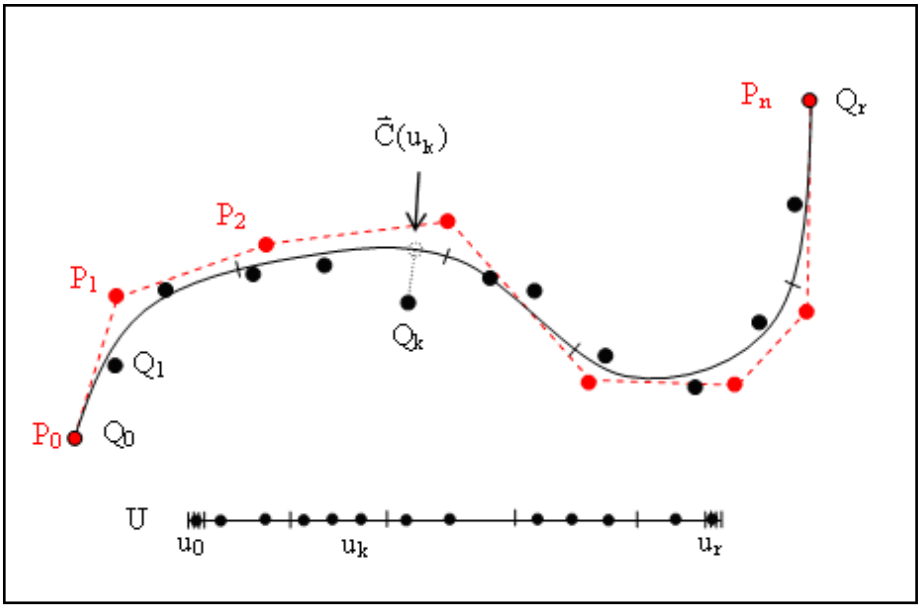


Fig. 9-22: B-spline curve

The programmed points Q_0 to Q_r are approximated by a b-Spline curve using the calculated checkpoints P_0 to P_n (n<<r).

The dimensional tolerance can be specified in b-spline approximation parameters.

They are programmed with the following **syntax**:

BsaPar(E..,OE..) Short form: **BAP(..)**

The individual parameters have the following meaning:

1. **E** (**E**rror)

   Specifies the dimensional tolerance in mm or inch.

   In case of a missing "E", the value in the machine parameter 80070030 is applied.

2. **OE** (**O**rientation**E**rror)

   Dimensional tolerance of the orientation motion in mm or inch.

   If no effective radius factor is programmed for the orientation vector, mm corresponds to degree. "OE" is only important in case of an active vector orientation.

   If "OE" is missing, OE = E is applied as switch-on value. OE = 5*E is a useful value for an inclined tool machining.

Starting and end boundary conditions

Apart from the boundary conditions 1, 10 and 11 (see "Starting and end condition:" on page 124), 0 is possible for the b-Spline approximation:

- SBC(0):

  A new spline sequence starts in the current block (contour knee at the starting point of the block)

- EBC(0):

  A new spline sequence ends in the current block (contour knee at the end point of the block)

The b-Spline approximation is a further spline type 4 of the MTX spline family (tab. 9-4 "Relevant NC functions - Spline with coefficient programming" on page 144).

The variant with spline ID 4233 is a special case. A particularly good smoothening of the curve is generated by an additional geometry filter. The spline ID 4233 is recommended for free form surface milling.

9.1.8 Spline programming

Function

All useable splines can be parameterized in the program with:

SplineDef(<Id>,<Members>)

The table shows an overview on the available options:

Spline-Id = <Type>+<Parameterization>+<Subtype>+<Degree>

Spline type	Parameterization 0 = Without significance 1 = Equidistant 2 = Chordal 3 = Centripetal	Subtype	Degree	Description	Recommended combination
0	0	0 = Irrelevant	1,..,5	Polynomial coefficient programming	SplineDef(3)
1	1, 2, 3	Tangent calculation: 1 = Bessel 2 = Akima 3 = False position 4 = Hermite	3	c1-continuous local cubic spline from specified points with different tangent calculation or specification	SplineDef (1213,...) or SplineDef (1143)
2	1, 2, 3	0 = Irrelevant	3	c2-continuous global cubic spline from specified points	SplineDef(2203,...)
3	1, 2, 3	0 = Irrelevant	1,..,5	B-spline and nurbs from specified checkpoints and weights PW	SplineDef(3203,..)
4	2	0 = Irrelevant	1,..,5	B-spline approximation (BSA)	SplineDef(4203,..) *)
		3	3	BSA with smoothing filter	SplineDef(4233,..)

*) The spline types 4201, 4202, 4203, 4204 and 4205 are not supported anymore in MTX 14VRS and above.

Tab. 9-3: Spline ID

Restrictions



The spline types 4201, 4202, 4203, 4204 and 4205 belonging to the cycle G591 are not supported anymore in MTX 14V22 and above.

*The following functions **cannot** be programmed with splines:*

- Tensor orientation: This affects the tensor syntax Ox(..), Oy(..) and Oz(..) and the Euler angles phi, theta and psi.
- 2D path correction G41/G42 (for 3D path correction use G141/G142)
- Chamfers and roundings (use SCO)
- Tangential tool guidance

Relevant NC functions

- Spline with coefficient programming:

G06 with the modal parameters:	
<CoordName >(..)	Coefficient programming of individual **coordinates**(CoordName) with their polynomial coefficients.
O1(..), O2(..), O3(..)	Coefficient programming of the **vector orientation** (spatial coordinates programming (Coord(..)) have to be activated before)
DN(..)	**Denominator polynomial** with coefficient programming to describe rational monomial splines (Bezier spline, rational B-splines (NURBS), all conics).
PL(..)	Spline parameter length

Tab. 9-4: Relevant NC functions - Spline with coefficient programming

Example:

N10 SplineDef(2) ;Coefficient programming second degree
N20 G1 X0 Y0
N30 G06 X(0,1) Y(0,0,1) PL3.0 ; parabola X=w, Y=w^2, 0<= w <= 3

- C^1-continuous cubic splines:

G06 with the modal parameters:	
<CoordName>	Programming of the coordinates is only considered to calculate the splines if they are listed under the parameter in the NC function Splinedef(..). All the others are included linearly.
<Axes>	For the axis programming, see above <CoordName>
<Orientation coordinates>	Programming the vector orientation (5-axis transformation) as spline
SBC(..)	Starting boundary conditions (type = 1, 10, 11) for the **starting point** of a spline sequence
EBC(..)	End boundary conditions (type = 1, 10, 11) for the **end point** of a spline sequence
PL(..)	Spline parameter length

Tab. 9-5: Relevant NC functions - c1-continuous cubic splines

Example:

N10 SplineDef(1213) ;c^1-continuous spline third degree
N20 G1 X0 Y0
N30 G06 X1 Y1
N40 X2 Y4

..
N50 G1

- Hermite splines

G06 with the modal parameters:	
<CoordName>	Coefficient programming of individual **coordinates**(CoordName) with their position and derivation value
PL(..)	Spline parameter length

Tab. 9-6: Relevant NC functions - Hermite splines

Example:

N10 SplineDef(1143) ;Hermite spline
N20 G1 X0 Y0 Z0 F1000
N30 G06 X(20,1) Y(10,-1) PL3.0
N40 X(30.5,2.1) Y(10.8,-0.8) Z(5,1.3) PL4.2
...
N1000 G1

- C^2-continuous cubic splines:

G06 with the modal parameters:	
<CoordName>	Programming of the coordinates is only considered to calculate the splines if they are listed under the parameter <Members> in the NC function Splinedef(..). All the others are included linearly.
<Axes>	For the axis programming, see above <CoordName>
<Orientation coordinates>	Programming the vector orientation (5-axis transformation) as spline
SBC(..)	Starting boundary conditions (type = 1,2,3,4,10,11)
EBC(..)	End boundary conditions (type = 1,2,3,4,10,11)
PL(..)	Spline parameter length

Tab. 9-7: Relevant NC functions - C2-continuous cubic splines

Example:

N10 SplineDef(2203) ;c^2-continuous spline third degree
N20 G1 X0 Y0
N30 G06 X1 Y1
N40 X2 Y4
..
N50 G1

- B-splines (NURBS):

G06 with the modal parameters:	
<CoordName>	Programming of the coordinates is only considered to calculate the splines if they are listed under the parameter in the NC function Splinedef(..). All the others are included linearly.

<Axes>	For the axis programming, see above <CoordName>
<Orientation coordinates>	Programming the vector orientation (5-axis transformation) as spline
SBC(..)	Starting boundary conditions (type = 10)
EBC(..)	End boundary conditions (type = 10)

Tab. 9-8: Relevant NC functions

- SplineDef(<Id>,<Members>):
  Definition of the spline type

The following applies:

<Id>	Spline type, (see tab. 9-3 "Spline ID" on page 143)
<Members>	Specifies the coordinate or axis names involved in the spline motion. Coordinates/axes not listed here can only be moved linearly. If Members is not programmed, all coordinates/axes of the channel are used for the spline motion. The specification of <Members> is not applicable to type 0.

Tab. 9-9: Relevant NC functions - SplineDef(<Id>,<Members>)

Example:

```
N10 SplineDef(3203) ;B-spline third degree
N20 G1 X0 Y0
N30 G06 X1 Y1
N40 X2 Y4
..
N50 G1
```

B-spline approximation

G06 with the modal parameters	
<CoordName>	Programming coordinates. Only coordinates from the <Members> list may be used.
<Orientation coordinates>	Programming the vector orientation O if contained in <Members>
SBC(..)	Starting boundary condition (type = 0)
EBC(..)	End boundary conditions (type = 0)
BAP(..)	B-spline approximation parameter

Tab. 9-10: Relevant NC functions - B-Spline with approximation

Example:

```
N10 SplineDef(4233) BAP(E0.02,OE0.1) ;B-spline approximation
N20 G1 X0 Y0 O(1,1,1)
N30 G06 X1 Y1 O(1.2,0.8,1)
N40 X2 Y4 O(1,3,0.7,0.95)
```

..

N50 G1

The following table provides an overview on the programming options of the individual spline types:

Spline type	End point programming	PL	PW	SBC/EBC
0	Coefficient programming, denominator polynomial for rational splines	Obligatory	no	no
1	Channel coordinates as data points	Optional	no	Type = 1, 2, 10, 11
2	Channel coordinates as data points	Optional	no	Type = 1, 2, 3, 4, 10, 11
3	Channel coordinates as checkpoints	Optional	Yes	Type =10
4	Channel coordinates (only "Members") as points to be approximated	no	no	Type = 0

Tab. 9-11: Programming options of the spline types

Relevant machine parameters (MP)

7060 00120[3]	Maximum block number for block look-ahead Recommended values: For **spline type 2**: 30 blocks For **spline type 4**: 80 blocks For **all other spline types**: 10 blocks
7060 00130[3]	Percentage distribution look-ahead function
8007 00010	Tolerance value ε_0 in micrometers for the C^0-continuity for splines with coefficient programming. If the starting point is not identical to the end point of the preceding spline, a runtime error is generated if the deviation is greater than the set tolerance value ε_0 . The value is to be as small as possible (depending on the accuracy of the programming system). Otherwise, axis jumps can occur at the segment transitions.
8007 00020	Max. permissible relative path velocity error ε_v in %. The relation (inverse path length function) of the "spline parameter length w" and the "path length of the curve" is approximated by splines in the NC. The path velocity error represents the accuracy of this approximation. Recommended value is 0.1 %. If the machine parameter is set to 100%, the path length function w(s) is approximated by a single fifth degree spline.

8007 00099	Spline configuration (relevant only for C^2-continuous cubic spline) [1] Degree of the spline between two partial sequences:	
	3:	Cubic spline (C^1-continuous only at the starting point)
	4:	Square spline (C^2-continuous to the preceding spline)
	[2] Overlapping (number of NC blocks) of two partial sequences [3] Spline variant for vector orientation:	
	1:	Spline for vector orientation together with spatial spline
	2:	Independent spline for vector orientation
	[4]...[8] not relevant	
800700030	Dimensional tolerance in mm for the b-spline approximation	

Tab. 9-12: Relevant machine parameters

9.1.9 Handling instruction: Applying spline

The following machine parameters are relevant for spline curves:

IW Engineering/configuration: Editing parameters

- **Spline** (7060 00120) "max. number of blocks for spline functions"
  60 NC blocks are recommended
- **Spline** (7060 00130) "percentage number of blocks for spline functions"
  A percentage resulting in 60 blocks is recommended
- **MaxPosDev** (8007 00010)
  "Max. starting/end point deviation allowed"
  Only relevant for spline type 0
- **MaxVelDev** (8007 00020)
  "Max. relative path velocity error allowed"
  0.1 % is recommended.
- **BSplineApproxTol** (8007 00030)
  "Dimensional tolerance in mm for the b-spline approximation"
  Only relevant for spline type 4
  Recommended is 0.02 mm
- **ConSplineType** (8007 00099)
  "Degree of connection spline"
  Only relevant for spline type 2
- **NofOverlapBl** (8007 00099)
  "Overlapping of two partial sequences"
  Only relevant for spline type 2
- **VectOri** (8007 00099)
  "Spline variant for vector orientation"
  Only relevant for spline type 2

The parameters from **MaxPosDev** to **VectOri** have to be activated via the parameter **spl** "splines" in the setup (SUP).

9.1.10 Handling instruction: Activating spline

IW Operation/NC programming: Initialization

Before using G06 in the NC program for the first time, program the initialization **SplineDef** or apply the initialization stored in the machine parameter init string.

SplineDef(<Id>,{<Members>})

- **<Id>**:

  Integer number with up to four for the spline variant:

- **<Members>**:

  List of names of the coordinates/axes to be involved in the spline motion. Non-programmed coordinates/axes are linearly taken along in <Members>. Orientation coordinates can also be programmed as splines. Specify "O" for the orientation or the polar coordinates "phi" and "theta" under <Members>. "Members" is not relevant for spline type 0.

  "Members" can also be omitted. The following default settings apply:

  5-axis transformation on: Members = Coordinates of the transformation (x ,y, z, phi, theta)

  No 5-axis transformation: Members = All channel coordinates/axes

Example: SplineDef(3203,X,Y,Z)

	Documentation	
Documentation:	MTX Machine Parameters	Initializing spline

IW Operation/NC programming: Activating spline

Program **G06**. Depending on the required programming type, **G06** can be programmed for the various spline types with different, modally effective parameters.

	Documentation	
Documentation:	MTX Programming Manual	Activating spline

9.1.11 Handling instruction: Deactivating spline

IW Operation/NC programming: Deactivating spline

To deactivate the path type "Spline", another path type is programmed, for example **G1**, **G2**, **G3**, etc. The corresponding function has to be applied.

	Documentation	
Documentation:	MTX Programming Manual	Deactivating spline

9.2 Additional spline functions

9.2.1 Automatic detection of corners and lines

Function

Background In many cases, CAD/CAM postprocessors generate NC programs to machine free-form surfaces which exclusively contain linear blocks (G1). A high surface quality combined with a short processing time can only be achieved on

smooth curves, that is with splines for example. The simple solution of using spline interpolation (G6) instead of linear interpolation (G1) can often not be used, since a few labeled blocks actually have to be traversed with G1 due to reasons of geometry. The NC function "Automatic detection of corners and straight lines" provides a solution to this problem.

Angle and distance conditions can be specified for each spline type via NC command. Thus, the spline generation automatically generates corners and lines within the otherwise smooth spline sequence. The function cannot be used for spline type 0 (coefficient programming) and is not recommended for spline type 4 (B-spline approximation).

Angle condition:

Absolute angle condition

A maximum permitted contour knee angle α^{max} is specified.

If the knee angle α exceeds $_i$ at the end point $\vec{P}_i$ of the current block, the angle condition is met. If the angle condition is met, the spline sequence is interrupted. Therefore, it acts as if an end boundary condition EBC was programmed in block i and if a corresponding starting boundary condition SBC was programmed in block i+1. This results in a contour knee at the end point of the current block (see fig. 9-23 "Spline reaction if the angle and distance conditions are met" on page 151).

Relative angle condition

An angle factor f_a is specified. If the average knee angle in the look-ahead range is $\bar{\alpha}$. The angle condition at a point $\vec{P}_i$ is $\alpha_i > f_\alpha \cdot \bar{\alpha}$.

When calculating the average knee angle $\bar{\alpha}$ the α_i, where the angle condition is met, are not counted.

The absolute und the relative angle condition are linked by a logic "Or".

Distance condition:

Absolute distance condition

A maximum permitted point distance d^{max} is specified. The distance condition in a block j with starting point $\vec{P}_{j-1}$ and end point $\vec{P}_j$ is met if the point distance $d_j = \left|\vec{P}_j - \vec{P}_{j-1}\right|$ is greater than d^{max}.

If the distance condition is met, the two points are connected with a first degree spline (linear block).

Relative distance condition

A distance factor f_d is specified. If the average point distance in the look-ahead range is $\bar{d}$. the distance condition for a block j is $d_j > f_d \cdot \bar{d}$.

When calculating the average point distance $\bar{d}$ the d_j, where the distance condition is met, are not counted.

The absolute und the relative distance condition are linked by a logic "Or".

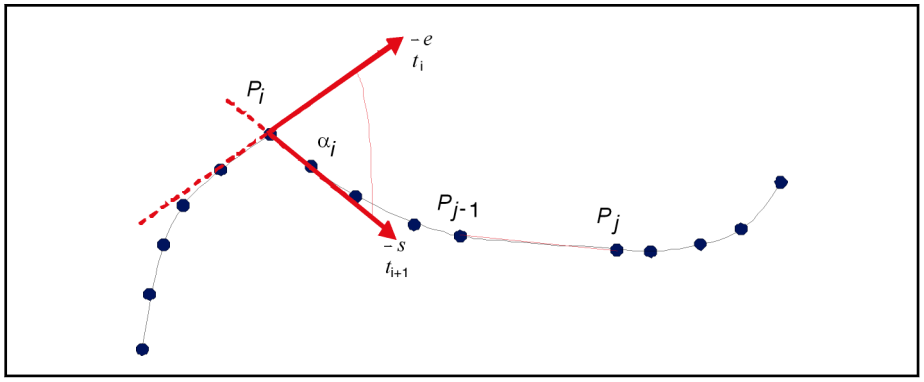


Fig. 9-23: Spline reaction if the angle and distance conditions are met

Syntax

Automatic linear spline switching is activated via the syntax "CornerLineDetection" (short form CLD)

CLD(ANG<a>,AFACT<fa>,DIST<d>,DFACT<fd>)

The parameters have the following meaning:

ANG<a> **a** is the maximum admissible contour knee angle α^{max}.
Knee angles greater than "a" meet the angle condition.

AFACT<fa> **fa** is the angle factor. Knee angles exceeding the average knee angle by the factor fa fulfill the angle condition.

DIST<d> **d** is the maximum admissible point distance d^{max}.
Distances greater than "d" fulfill the distance condition.

DFACT<fd> **fd** is the distance factor.
Point distances exceeding the point distance by the factor fd fulfill the distance condition.

Parameters can be omitted individually.

After programming **CLD** with at least one parameter, the function is modally switched on and is active as soon as G06 is active (exception: spline coefficient programming).

The parameters can be changed at any time by reprogramming.

This function is switched off by programming without the parameters **CLD()**.

Example:

NC program example

c^2-continuous cubic spline

```
SPD(2203,x,y,z,O) CLD(ANG20,DIST5)
...
G06 x.. y.. z.. phi.. theta..        ;Spline 2203 off
                                     absolute distance condition, absolute and
                                     relative angle condition are active
x.. y.. z.. O(..)
...
x.. y.. z.. O() CLD(DIST8)           Disabling the angle condition, new abso-
                                     lute distance condition
...
G01                                  ;Spline off
```

```
...
G06                                   ;Spline on, the last CLD(DIST8) is valid
```

9.3 Automatic linear circular spline conversion

9.3.1 Description

Function

The function converts a contour programmed with straight lines and circles to a spline sequence.

Each motion specified with G1 (straight line interpolation with feed) and with G2/G3/G5 (circle/helical interpolation) is separated into a straight move. The number of points depends on the specified tolerance (BAP(E<Tol>). The generated points are subsequently connected by splines.

Programmed as well as MTX-internally generated straight lines and circles (roundings, chamfers, inserted blocks of the milling path correction) are converted.

During the conversion, the spline type 4233 (B-spline approximation with filters) is used.

Sequence:

1. Read programmed straight lines and circles.
2. Contour changes due to an active chamfer and rounding function.
3. Contour adjustment by active 2D milling path correction.
4. Conversion of the resulting contour in a spline sequence.

Example *Program:*

```
...
N10 T1                                ;select rough tool
N15 G0 G90 G54 X50 Y50 Z100 S12000 M3 ;go to start
N20 LCTS BAP(E0.1)                    ;activate conversion with rough tolerance
N25 P part.npg                        ;call machining subroutine
N30 LCTS()                            ;deactivate conversion
N35 T2                                ;select finish tool
N40 G0 G90 G54 X50 Y50 Z50 S20000 M3  ;go to start
N45 LCTS BAP(E0.01)                   ;activate conversion with finish tolerance
N50 P part.npg                        ;call machining subroutine
N55 LCTS()                            ;deactivate conversion
...
```

Special features

The special features of the used spline variant with the spline ID 4233 “Approximating B-splines with tolerance specification and geometry filter” apply.

Local, motion-generating functions are exempt from the conversion. These are:

- G63, G63.2 "Tapping without compensating chuck"
- G74 "Approach reference point coordinates"
- G74(HOME) "Approach reference point"
- G75, G75.2 "Travel to touch probe"
- G76 "Approach to fixed machine axis position"
- G77 "Reposition individual coordinates"
- AXC/AxCouple "Axis coupling" (approaching motion of coupling)

- FME/FlyMeas "Measurement on-the-fly"
- FSM/FsMove "Move to fixed stop"
- FSP/FsProbe "Measuring at fixed stop"
- FSR/FsReset "Reset fixed stop"
- PTP/PtpMove "PTP traversing motion"
- TCM/TcsMove "Traversing in the TCS"

Modal motions complemented by motion-accompanying functions are exempt from the conversion. Complementary functions:

- MOB "Measuring on a block"
- HSB/HsBlkSwitch "Block switching on-the-fly"
- PHS/PosDepHsOut "Position-dependent HighSpeed output"

Motion with an exact stop function are exempt from the conversion:

- G1(IPS1), G1(IPS2)
- G1, G2, G3 in case of active G61

Restrictions

An NC block containing one of the NC functions specified in the following in addition to a path motion, is not converted to a spline.

- Auxiliary function M, S, T, ...
- An asynchronous axis motion
- A drive communication such as FFW, IPS1, IPS2, KVP, WID3
- Synchronizing functions for the block execution time
  - Bit events REV, SEV, WEV, WREV
  - Bit interface WAITA, WAITO
  - Permanent variables SPV, WPV, WPVE

    Note: The "SPVE" command can often be used as substitute for SPV.
  - SSD system data

    Note: The "SSDE" command can often be used as substitute for SSD.
  - System data queue SSDQ and SSDQInit
- Axis transfer DAX, GAX, RAX, WAX
- All specifications affecting a channel or system spindle

Relevant NC functions

- **LCTS**

  LCTS or LCTS(1) enables the automatic linear circular spline conversion.

  LCTS() or LCTS(0) disables the automatic linear circular spline conversion.
- **BAP(E..)**

  Parameter description:

  "E" specifies the dimensional tolerance in mm or inch.

  In case of a missing "E", the value in the machine parameter BSplineApproxTol (MP 80070030) is applied.

Relevant CPL functions

- **SD(60,11)**
  SD(60,11) returns the spline ID used by LCTS.
- **SDR(60,2)**
  SDR(60,2) returns the dimensional tolerance specified with BAP(E..) of the B-spline approximation in mm or inch.
- **SDR(60,3)**
  SDR(60,3) returns the dimensional tolerance specified with BAP(OE..) of the B-spline approximation in mm, inch or -1.0.

Relevant machine parameters

Refer to chapter 9.3.2 "Applying" on page 154.

9.3.2 Applying

The "Automatic linear circular spline conversion" function requires a lot of memory space. Adjust the startup.xml file accordingly. An appropriate entry is:

```
<AdditionalNCBlockMem>4000000</AdditionalNCBlockMem>
```

Machine parameters used by the B-spline approximation have to be set to an appropriate value. These are:

- NofBlkPrep (7060 00110) "Total number of blocks for block preparation and interpolation"
  A sufficiently large value covering the requirements of all functions (MaxBlkVelCalc, MaxBlkBuffered, MaxBlkSpline, MaxBlkTCorr3D und MaxBlkMiscFunc).
- MaxBlkSpline (7060 00120) "Max. number of blocks for spline functions"
  120 blocks is recommended.
- Spline (7060 00130) "Percentage number of blocks for spline functions"
  A percentage resulting in 120 blocks is recommended.
- MaxVelDevSpline (8007 00020) "Max. permissible relative path velocity error"
  0.1 mm is recommended.
- BSplineApproxTol (8007 00030) "Dimensional tolerance in mm for the B-spline approximation"
  0.02 mm is recommended.

9.3.3 Activating

"LCTS" enables the function.

If required, the tolerance margin can be adjusted by programming of "BAP(E<Tol>)".

9.3.4 Deactivating

LCTS() deactivates the function.

Temporary deactivation

LCTS can be temporarily deactivated within a cycle or a subroutine by using the CPL and NC functions previously described.

Program:

```
;Save parameters and deactivate LCTS
0001 DIM NCFTS$(25)
0002 NCFTS$ = NCF("LCTS")
0003 IF NCFTS$ = "LCTS" THEN
0004   TSETOL! = SDR(60,2)
0005   TSOTOL! = SDR(60,3)
N006   LCTS()
0007 ENDIF
...
...
...
;Activate LCTS again
0990 IF NCFTS$ = "LCTS" THEN
N991   LCTS BAP(E[TSETOL!],OE[TSOTOL!])
0992 ENDIF
```

10 Tool orientation und movement of orientation

10.1 General information

10.1.1 Description

Orientation motions refer to orienting the tool interpolatory in relation to the workpiece coordinate system (WCS or PCS). A corresponding axis transformation (5-/6-axis transformation, hexapod) implementing the orientation motion in axis motions is required.

10.2 Orientation definitions

10.2.1 Tool coordinate system TCS

The tool coordinate system (TCS: **T**ool **C**oordinate **S**ystem) is a Cartesian coordinate system with the coordinates X_t, Y_t and Z_t connected to the tool. Its origin lies in the TCP (**T**ool **C**enter **P**oint). Its Z_t coordinate always points in the direction of the tool holder. In case of rotationally symmetric tools such as cutter, laser beam, etc., the coordinates X_t and Y_t do not have any significance. In case of non-rotationally symmetric tools such as e.g. a gripping tool, X_t and Y_t point in specific symmetric directions vertically to each other and to Z_t. The reference position of the corresponding axis transformation is decisive. In the reference position, the TCS and the basic workpiece coordinate system BCS have to match.

Example:

Orienting the TCS

- Rotationally symmetric tool in a 5-axis kinematics, see chapter 11 Axis transformation → 5-axis transformation - Type 1: LLLRR:

  Milling cutter with three position coordinates (x, y, z) and two orientation coordinates (phi, theta).
- Rotationally symmetric tool in a 6-axis kinematics, see chapter 11 Axis transformation → 6-axis transformation:

  Gripper tool with three position coordinates (x, y, z) and three orientation coordinates (phi, theta, psi).

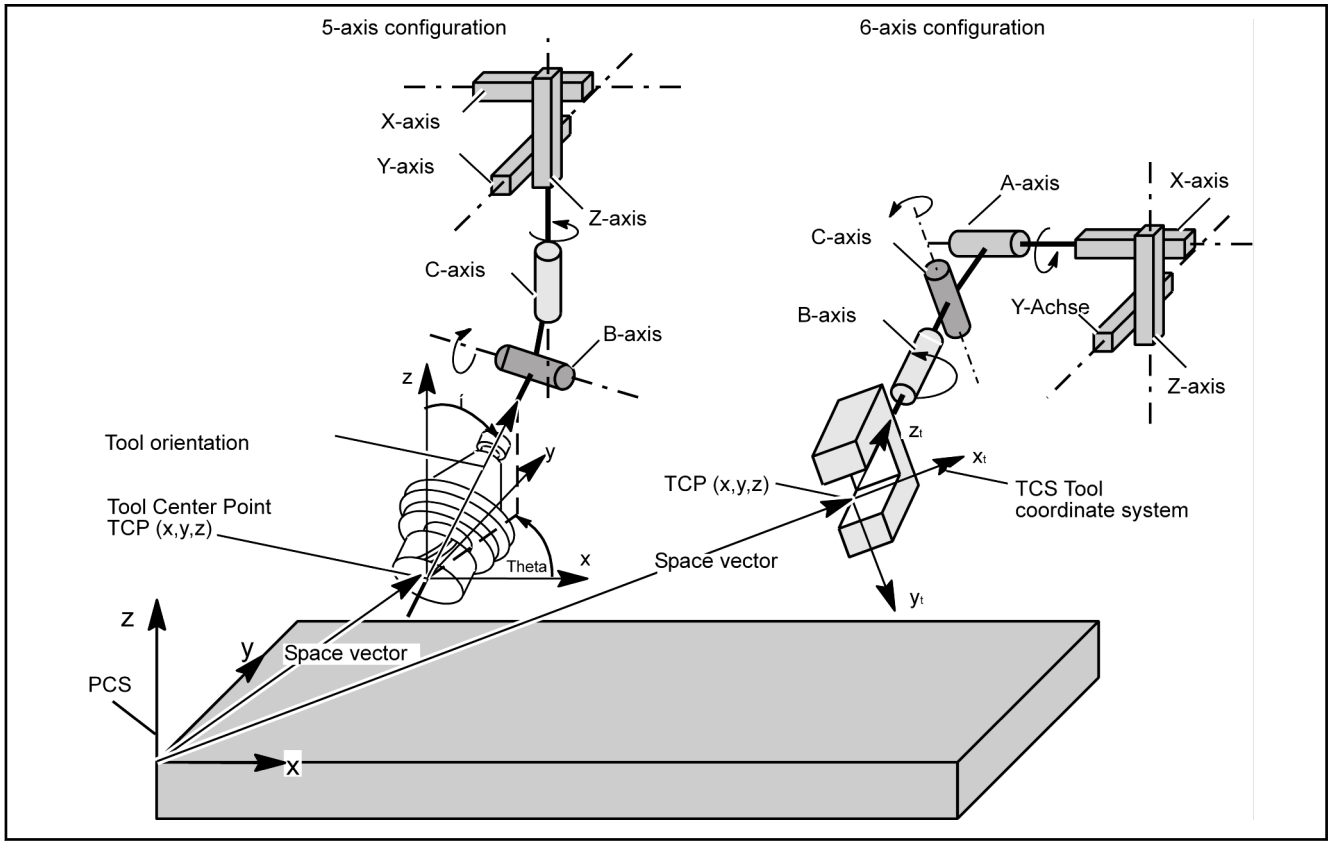


Fig. 10-1: Orienting the TCS with regard to the PCS in a 5- and 6-axis kinematics

10.2.2 Euler Angles

General information

The TCS orientation regarding the WCS is describes using three Euler angles phi, theta and psi. In case of rotationally symmetry, the third angle psi is omitted. To determine the Euler angles of a TCS with specified orientation, TCS(3) (target TCS) is reached by three consecutive basic rotations around the Euler angle, assuming a TCS (0) corresponding to the WCS. There are three equivalent variants with identical (phi, theta, psi) block to reach the target TCS.

Variants for Euler angle orientation with the angles phi, theta, and psi

Variant 1 is the most commonly used variant out of the **three variants**.

Variant 1: **Rotation around the respective new coordinate**

1. D_{Zt}(phi): Rotation around the Zt coordinate of the TCS(0) = WCS with angle phi
2. D_{Yt}(theta): Rotation around the Y_t-coordinate of the TCS(1) with angle theta
3. D_{Zt}(psi): Rotation around the Z_t-coordinate of the TCS(2) with angle psi

Example:

Gripping tool orientation

For a better overview, the tool coordinate system was shifted to the north pole of the program coordinate system.

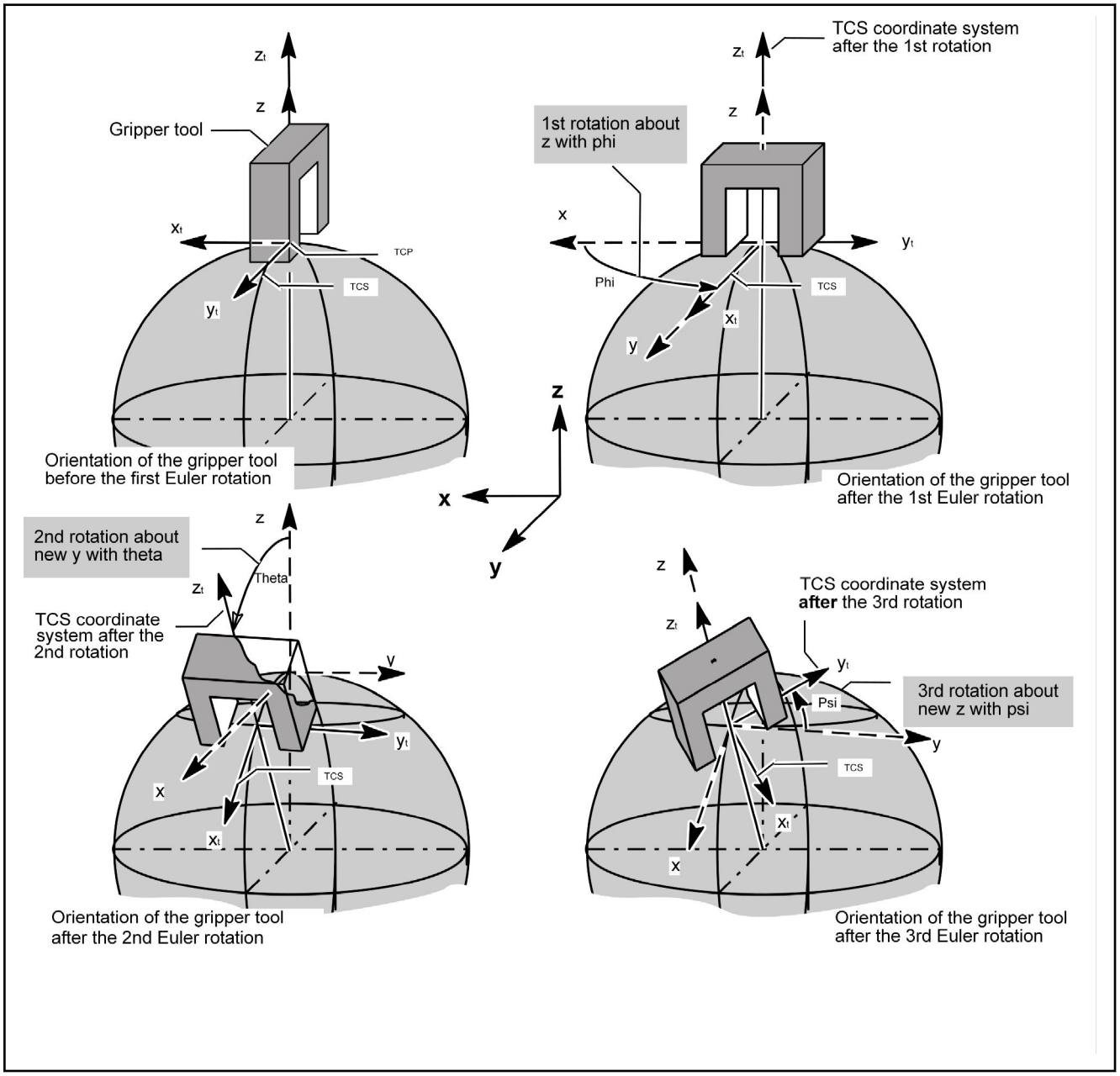


Fig. 10-2: Rotation around the new coordinate of the gripping tool TCS: D_{Zt}(phi), D_{Yt}(theta) and D_{Zt}(psi)

Variant 2: **Rotation around coordinates fixed in space**

1. D_Z(psi): Rotation around the z-coordinate of the fixed WCS with angle psi
2. D_y(theta): Rotation around the y-coordinate of the fixed WCS with angle theta

3. D_Z(phi): Rotation around the z-coordinate of the fixed WCS with angle phi

Example:

Gripping tool orientation

For a better overview, the tool coordinate system was shifted to the north pole of the program coordinate system.

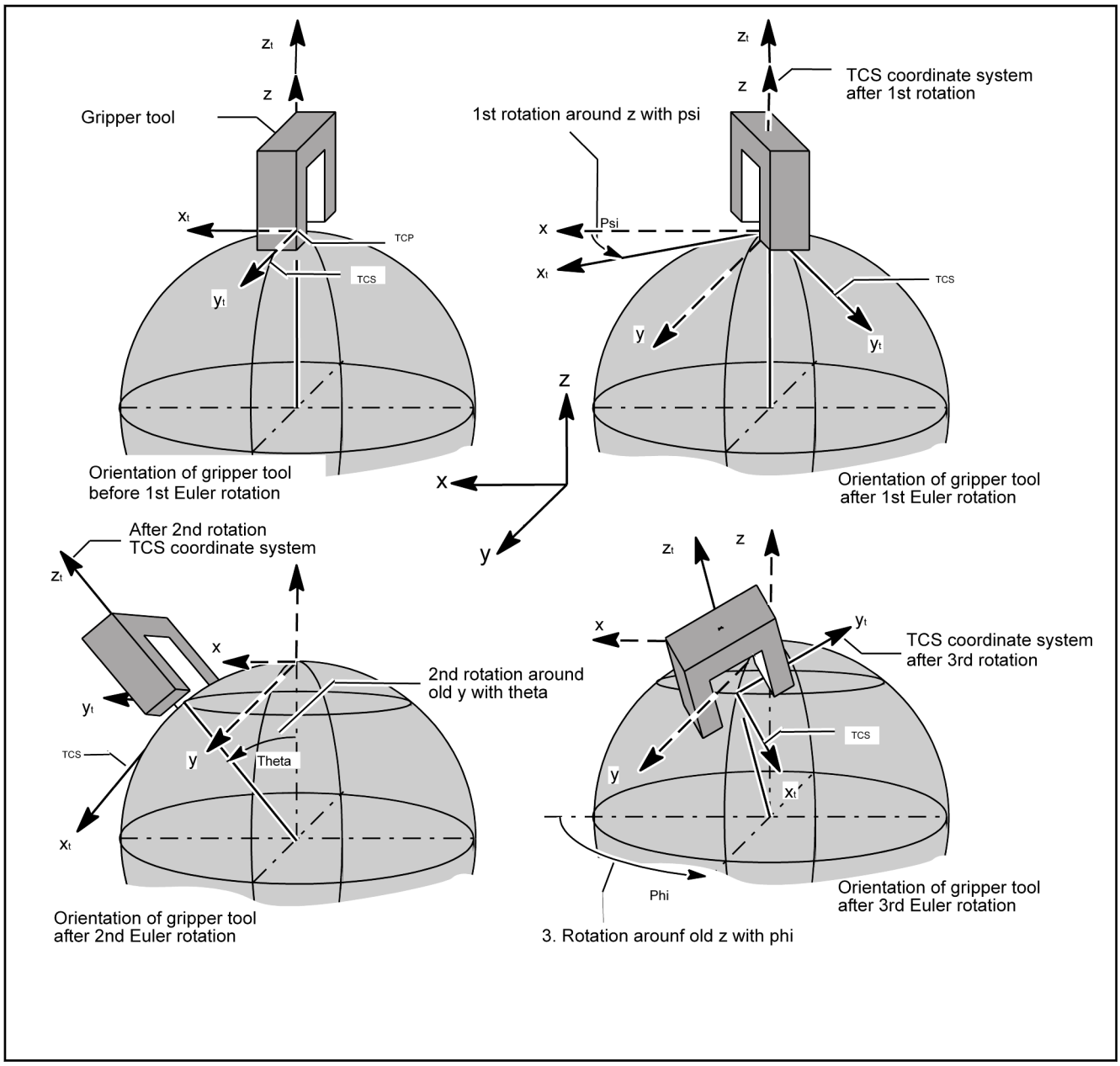


Fig. 10-3: Rotation around fixed WCS coordinates: D_Z(psi), D_y(theta) and D_Z(phi)

Variant 3: **Mixed rotation**

1. D_y(theta): Rotation around the y-coordinate of the fixed WCS with angle theta

2. D_z(phi): Rotation around the z-coordinate of the fixed WCS with angle phi
3. D_{Zt}(psi): Rotation around the Z_t-coordinate of the "TCS(2)" with angle psi

   Now, the target coordinate system is reached.

Example:

Gripping tool orientation

For a better overview, the tool coordinate system was shifted to the north pole of the program coordinate system.

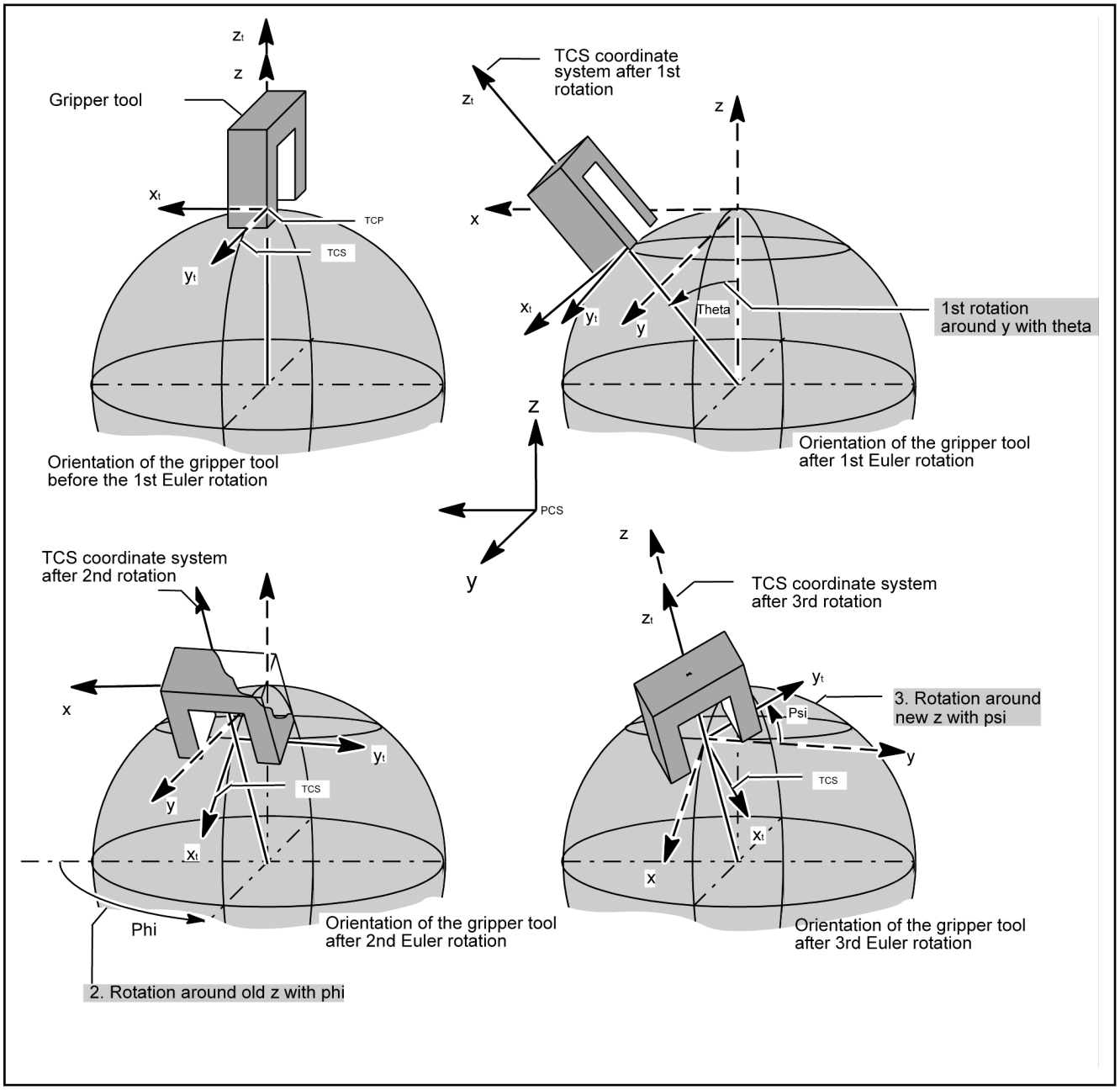


Fig. 10-4: Mixed rotation: D_y(theta), D_z(phi) and D_{Zt}(psi)

Description

The Euler angles can also be subsequently established from the target TCS:

theta: "Latitude" of Z_t (angle between north pole and Z_t)
phi: Latitude of Z_t (latitude of x is 0 degrees)
psi: Deviation of X_t from the southern direction

The following value ranges apply to the Euler angles phi, theta and psi:

phi:	0° < phi < 360°
theta:	0° < theta <180°
psi:	0° < psi < 360°

Due to the limited value ranges, there is a unique value triple for the Euler angles with regard to the original coordinate system for any orientation of a target coordinate system. When programming Euler angles, the user can program any values (e.g. negative theta values). To display these values, they are automatically converted into the valid value ranges.

Euler angles are also used to orient the workpiece coordinate system WCS with regard to the basic workpiece coordinate system (BCS) in any position in space (refer to the manual "MTX Functional Description Basics", chapter "Placement: Inclined plane G152.1...G159.5").



The programming systems interpret Euler angles differently. Euler angles can for example refer to other axes around which the rotation takes place.

Thus, ensure that the programming system supports Euler angles according to the MTX programming instructions!

10.2.3 Orientation vector

The orientation of rotation-symmetrical tools is specified by the two Euler angles phi and theta. They define an **orientation vector** O of length 1 specifying the orientation - with regard to the program coordinate system - along the tool length axis to the tool holder. In the free-form surface milling (5-axis milling), the orientation vector is predominantly used in the NC program.

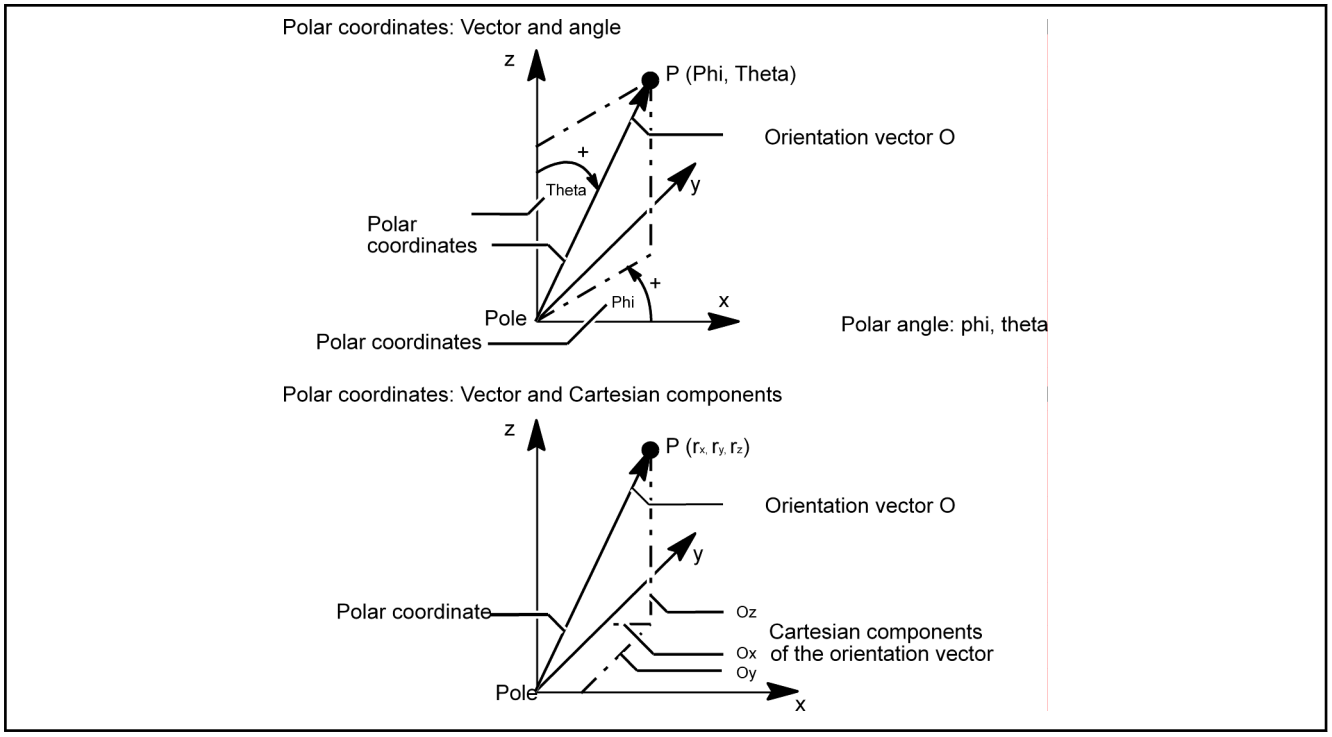


Fig. 10-5: Vector orientation

Relation between the Euler angles and the Cartesian components (O_x, O_y, O_z) of the orientation vector O (with the standardized length 1):

$$O = \begin{vmatrix} O_x \\ O_y \\ O_z \end{vmatrix} = \begin{vmatrix} \sin \text{Theta} * \cos \text{Phi} \\ \sin \text{Theta} * \sin \text{Phi} \\ \cos \text{Theta} \end{vmatrix}$$

Example:

Euler angle and orientation vector

The following figure shows the angle and the vector variant.

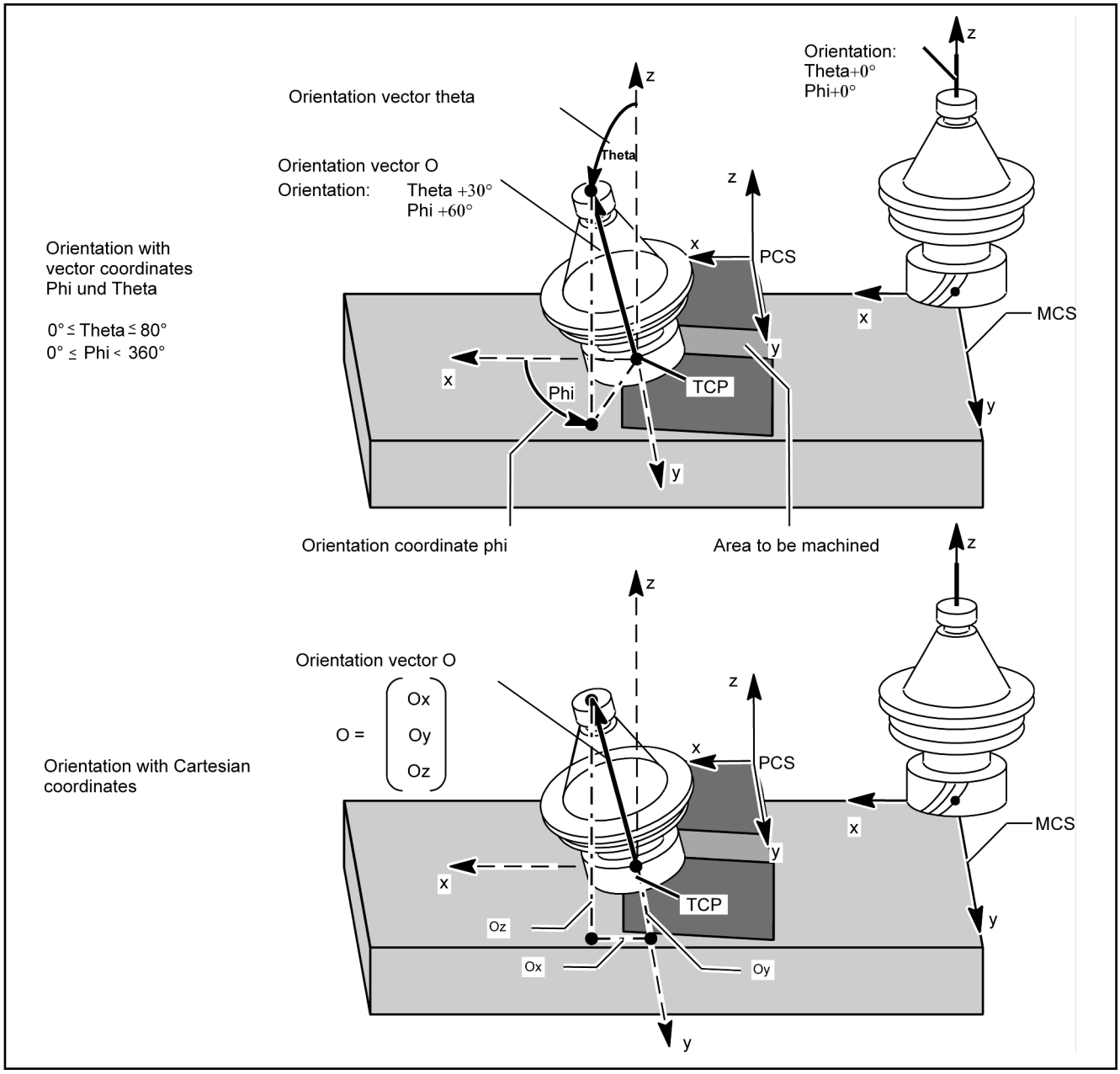


Fig. 10-6: Euler angle and orientation vector

- Tool orientation with regard to the PCS has the Euler angle phi = 60°and theta = 30°. This orientation can be specified with "phi60 theta30" or "O(60,30)" in the NC program.
- As a second option, the orientation of the tool length axis can be specified as **Cartesian position** of the tip of orientation vector O in relation to the PCS moved to the TCP. Inserting the angles in the formula above results in Ox = 0.25, Oy = 0.433 and Oz = 0.866 for O. In the NC program, this orientation can also be specified with "O(0.25,0.433,0.866)".

10.2.4 Orientation tensor

The orientation of non-rotation-symmetrical tools is specified by the three Euler angles phi, psi and theta. These Euler angles specify a 3x3 rotation matrix, the so-called **Euler matrix**. This matrix is also referred to as **Orientation tensor** O.

Structure of the Euler matrix or the orientation tensor:

$$\overleftrightarrow{O} = \begin{bmatrix} O_{11} & O_{12} & O_{13} \\ O_{21} & O_{22} & O_{23} \\ O_{31} & O_{32} & O_{33} \end{bmatrix}$$

Lines and columns of the tensor are provided with the length "1" and located vertically upon each other.

The columns of the tensor are the vector components of the **basic vectors** e_x, e_y, e_z of the rotated TCS with regard to the workpiece coordinate system:

$$\overleftrightarrow{O} = \begin{pmatrix} o_{11} & o_{12} & o_{13} \\ o_{21} & o_{22} & o_{23} \\ o_{31} & o_{32} & o_{33} \end{pmatrix}$$

$$\vec{e}_x = \begin{pmatrix} o_{11} \\ o_{21} \\ o_{31} \end{pmatrix} \quad \vec{e}_y = \begin{pmatrix} o_{12} \\ o_{22} \\ o_{32} \end{pmatrix} \quad \vec{e}_z = \begin{pmatrix} o_{13} \\ o_{23} \\ o_{33} \end{pmatrix}$$

Example:

3x3 orientation tensor of a non-rotationally symmetric tool in space

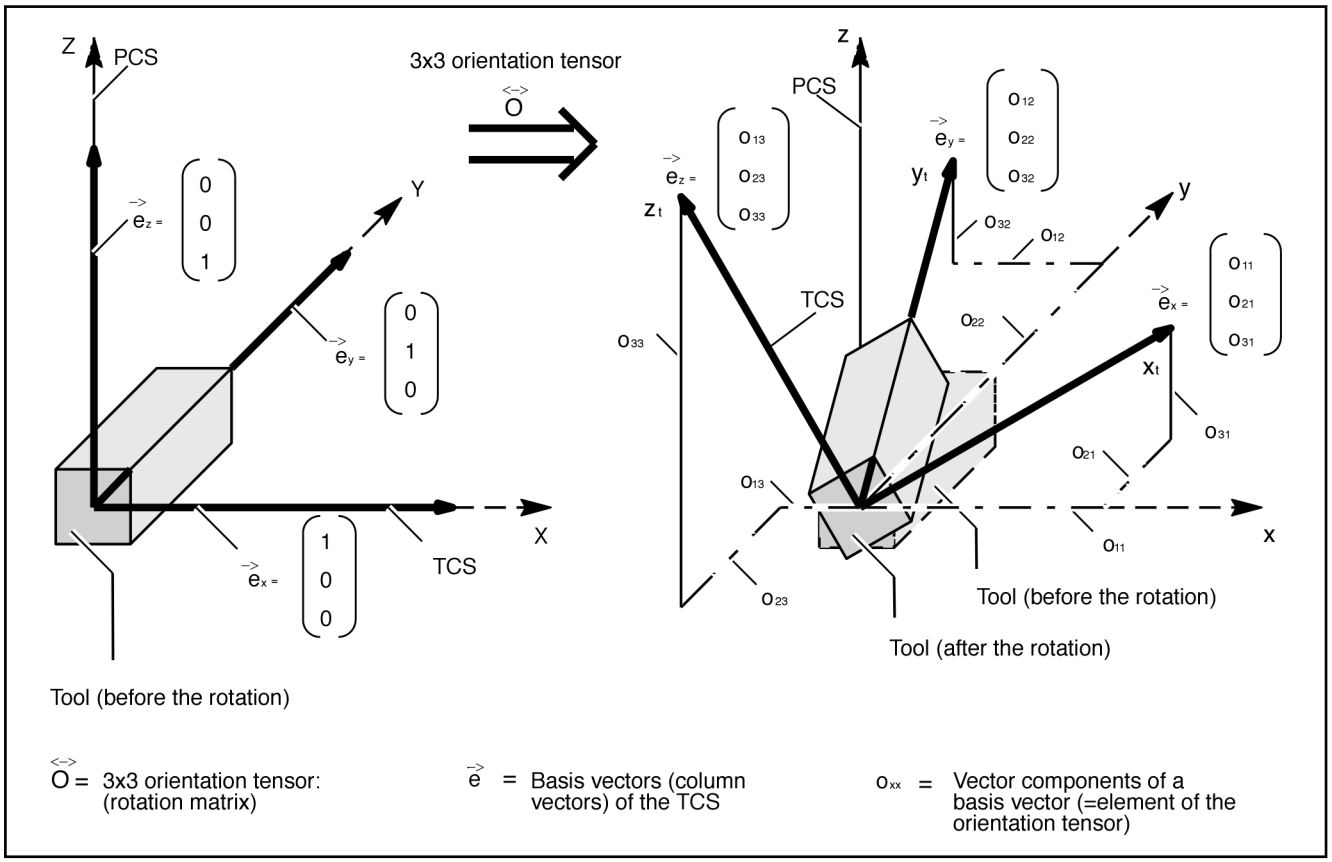


Fig. 10-7: Example - Orientation tensor

☞ The disadvantage of the orientation tensor is that its nine components are not independent of each other, as three coordinates are sufficient for a general orientation. Thus, it is recommended to program the three Euler angles in the NC program. Optionally, the orientation can be programmed as quaternion.

10.3 Orientation motion

10.3.1 Description

General information

Tool orientations are described using Euler angles (see chapter 10.2 "Orientation definitions" on page 157).

The type of the orientation motion is a feature of the respectively used axis transformation. The hundreds place in the axis transformation type specifies which orientation motion type is activated.

Hundreds place of axis transformation type	Orientation motion type
0	No orientation or orientation motion with a single angle coordinate. These special cases are described in the corresponding axis transformation chapters
1	Linear orientation motion with the Euler angles phi, theta
2	Vector orientation motion
3	Tensor orientation motion

Tab. 10-1: Orientation motion type depending on the axis transformation type

Example:

Axis transformation and orientation motion type

The first configured axis transformation in the machine parameters are of type 3232201 (5-axis transformation LLLRR). Using the NC command CORD(1), this axis transformation and the vector orientation motion is activated as the 100th place of the type is equal to 2.

For the same 5-axis kinematics, type 3232101 is available. The linear orientation motion is activated using this 5-axis kinematics.

10.3.2 Vector orientation motion

Function

In case of rotation-symmetrical tools, the interpolatory orientation motion is executed via the two Euler angles phi and theta or via the orientation vector. In the first case, it is a linear orientation motion, in the second case, a vector orientation motion.

Relevant NC functions

Coord(<i>)	Activates axis transformation i. Orientation programming (phi, theta or O(..)) is only possible in connection with a corresponding axis transformation i
Coord(0)	Deactivating axis transformation and thus the orientation programming

Programming the orientation vector with three alternatives:

phi<phi> theta<theta>	Orientation with the Euler angles phi and theta. The names "Phi" and "theta" are defined in the machine parameter CHAN/Ch[x]/Coord/Wcs/OriCoord[1..3] (MP7080 00010). "phi" and "theta" always have to be programmed together. Programming is only possible absolute in degree. Ex: G1 x10 y50 z30 phi90 theta90
O(<phi>,<theta>)	Same as above for "phi, theta" Example: G1 x10 y50 z30 O(90,90)
O(<x>,<y>,<z>)	Orientation with Cartesian components O_x, O_y, O_z of the orientation vector. Programming absolute. Example: G1 x10 y50 z30 O(0,1,0)

Tab. 10-2: Relevant NC functions

Programming the orientation vector with spline:

Within the spline coefficient programming, the orientation vector can be programmed as polynomial.

O1(<o_{10}>,<o_{11}>,..., <o_{1n}>)	x-components of the orientation vector with the spline coefficients o_{10} to o_{1n}
O2(<o_{20}>,<o_{21}>,..., <o_{2n}>)	y-components of the orientation vector with the spline coefficients o_{20} to o_{2n}
O3(<o_{30}>,<o_{31}>,..., <o_{3n}>)	z-components of the orientation vector with the spline coefficients o_{30} to o_{3n}

Tab. 10-3: Programming the orientation vector with spline coefficients



The common denominator polynomial (DN) does not apply to the vector orientation (see chapter 10.3.2 "Vector orientation motion" on page 166).

Programming the orientation motion of the orientation vector:

ROTAX(<R_x>,<R_y>,<R_z>) O(<β>) - or - ROTAX(<phi_R>,<$theta_R$>) O(<β>)	ROTAX(..) defines the orientation of the rotary axis around which the orientation vector rotates. The direction can be specified with Cartesian vector components R_x, R_y,R_z or with the polar coordinates phi_R, $theta_R$. O(β) defines the angle β, around which the initial orientation vector rotates around the rotary axis ROTAX. O(β) can only be programmed incrementally and in degree. β can assume any value, i.e. several rotations are also possible. Example: ROTAX(0,45) O(720)

Motion of the orientation vector:

Except for the spline interpolation, the motion of the orientation vector is always a rotation around the rotary axis through the TCP (TCS origin).

The following variables are involved:

- O_s: Initial orientation vector
- O_e: End orientation vector
- R: Rotary axis
- β: Angle of rotation

The following variables have to be specified to perform an orientation motion from an initial state of the orientation vector O:

Orientation motion by specifying the end orientation vector O_e:

- The end orientation vector O_e is programmed with one of the three alternatives of tab. 10-2 "Relevant NC functions" on page 167.
- The orientation vector rotates from the initial vector O_s around an internally calculated rotary axis R to the end vector O_e.
- Rotary axis R is always perpendicular to the orientation vector O which rotates from the initial orientation to the final orientation. The rotary axis runs through TCP.
- The motion of the orientation vector scans the angle of rotation β ≤ 180° and thus takes the shortest connection between the initial and end orientation.
- The orientation motion is always performed in such a was that the orientation vector remains in the plane spanned by O_s and O_e. The area scanned by the orientation vector is plane. The tip of the orientation vector runs on a great circle of the unit ball. The orientation motion depends on the programming type in tab. 10-2 "Relevant NC functions" on page 167.
- To be able to determine a rotary axis R, O_s and O_e must not be parallel or antiparallel.

Orientation motion with specified rotary axis:

- The orientation vector rotates from initial value O_s around a programmed rotary axis R by a programmed angle of rotation β.

- The rotary axis is programmed with ROTAX(..), the angle of rotation with O(..).

  Example: N10 ROTAX(0,0,1) O(40)
- The programmed angle of rotation can assume any value. The restriction $\beta \leq 180°$ is omitted. Generally, it is interpreted as a incremental value. The angle rotates the initial orientation tensor O_s by exactly one revolution from O(360) to its original position.
- The desired direction of rotation can be generated by positive and negative angles of rotation.
- The orientation vector scans the surface of a cone.

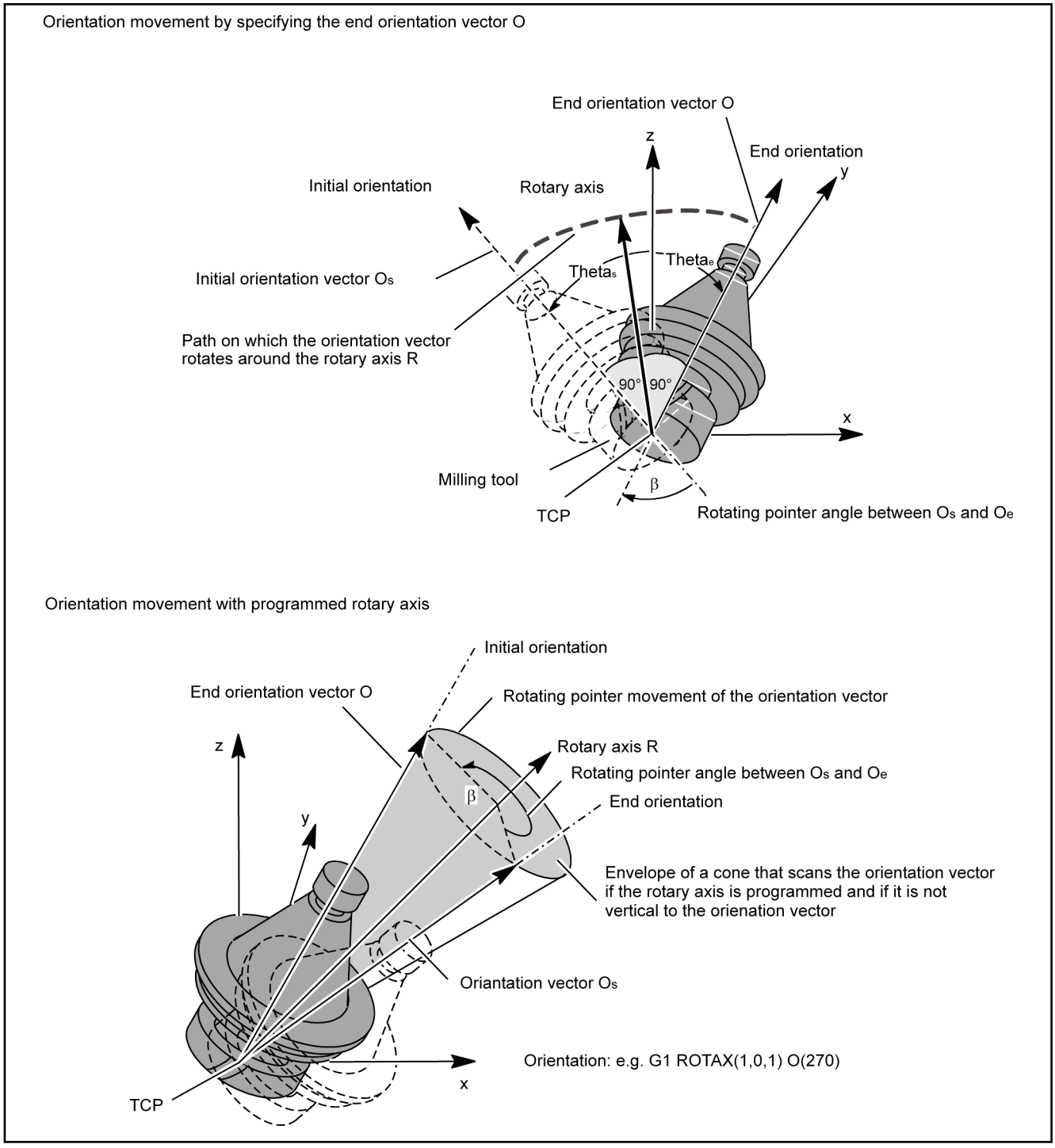


Fig. 10-8: Orientation motion without and with programmed rotary axis

Rotating speed of the orientation vector:

The orientation vector has a length of 57.23 mm (= 360/2π). In a coordinate system parallel to the WCS and with the TCP as origin, the vector tip moved by 1 mm at a rotation around 1 degree. The programmed feed F proportionately refers to the velocity of the vector tip. The following three examples illustrate the topic.

Example:

Pure orientation motion

N10 O(0,0,1)	Initial state theta = 0
N20 O(0,1,0) F180	Motion to phi=90 theta=90 with feed 180 mm/min

Tab. 10-4: Pure orientation motion

The tip of the orientation vector is moved at 180 mm/min. As the tip traverses an arc length of 90 mm, the motion duration is approximately 0.5 min.

Example:

Spatial motion plus orientation motion, orientation vector as member of the feed group

N10 FeedGrp(x, y, z, phi, theta)	Feed group: x, y, z and O
N20 x0 O(1,0,0)	Initial state phi0 theta90
N30 x90 O(0,0,1) F90	Motion in x and O with feed 90 mm/min

Tab. 10-5: Spatial motion plus orientation motion, orientation vector as member of the feed group

In block N30, x and O move by 90 mm. The block path length is $S^2 = 90^2 + 90^2$, consequently S = 1,41*90. The motion thus takes T = S/F = 1.41 min.

Example:

Spatial motion plus orientation motion, orientation vector as member of the feed group

N10 FeedGrp(x, y, z)	Feed group: x, y, z.
N20 x0 phi0 theta90	
N30 x90 phi0 theta0 F90	Motion in x and O with feed 90 mm/min

Tab. 10-6: Spatial motion plus orientation motion, orientation vector as member of the feed group

The path length is generated by the feed group members, i.e. S = 90. Thus, the motion takes T = S/F = 1 min. The orientation motion is moved along accordingly.

Limiting the velocity of the orientation vector (Omega programming)

The velocity of the orientation vector can be limited by programming "Omega". This can be important if a motion with a small spatial ratio and a large orientation ratio is executed at FeedGrp(x,y,z). The orientation motion would then only be limited by the maximum axis velocities.

Omega programming

Omega<Value>	Value > 0: The velocity of the vector tip is limited to value. Value has the same unit as the feed F.
Omega0	Omega programming off: The orientation velocity is limited to feed F. This also applies if Omega is not programmed.

The Omega limitation is only active in case of motions in feed. The limitation is not active in case of rapid traverse (G0).

Example:

Orientation motion with limitation by Omega

N10 Omega180	
N20 G1 O(0,0,1)	Initial state theta = 0
N30 O(0,1,0) F600	Motion to phi=90 theta=90 with feed 180 mm/min

Tab. 10-7: Orientation motion with limitation by Omega

The feed in block N30 is limited to 180 mm/min.

In case of mixed motions, the resulting path velocity can sporadically be limited so that the orientation ratio does not exceed the Omega value.

Special features

Singularity of the orientation vector motion:

For axis configurations like those shown in the following figure, non-constant axis motions result if the orientation vector is moved in a non-tangential manner through the position O = (0, 0, ±1).

Example:

Avoiding a jump of a C-axis from 0 degrees to 90 degrees.

In these cases, the MTX is able to **automatically** insert a "**Singularity block**" which first rotates the C-axis to the required position before continuing positioning. Refer to the following illustration:

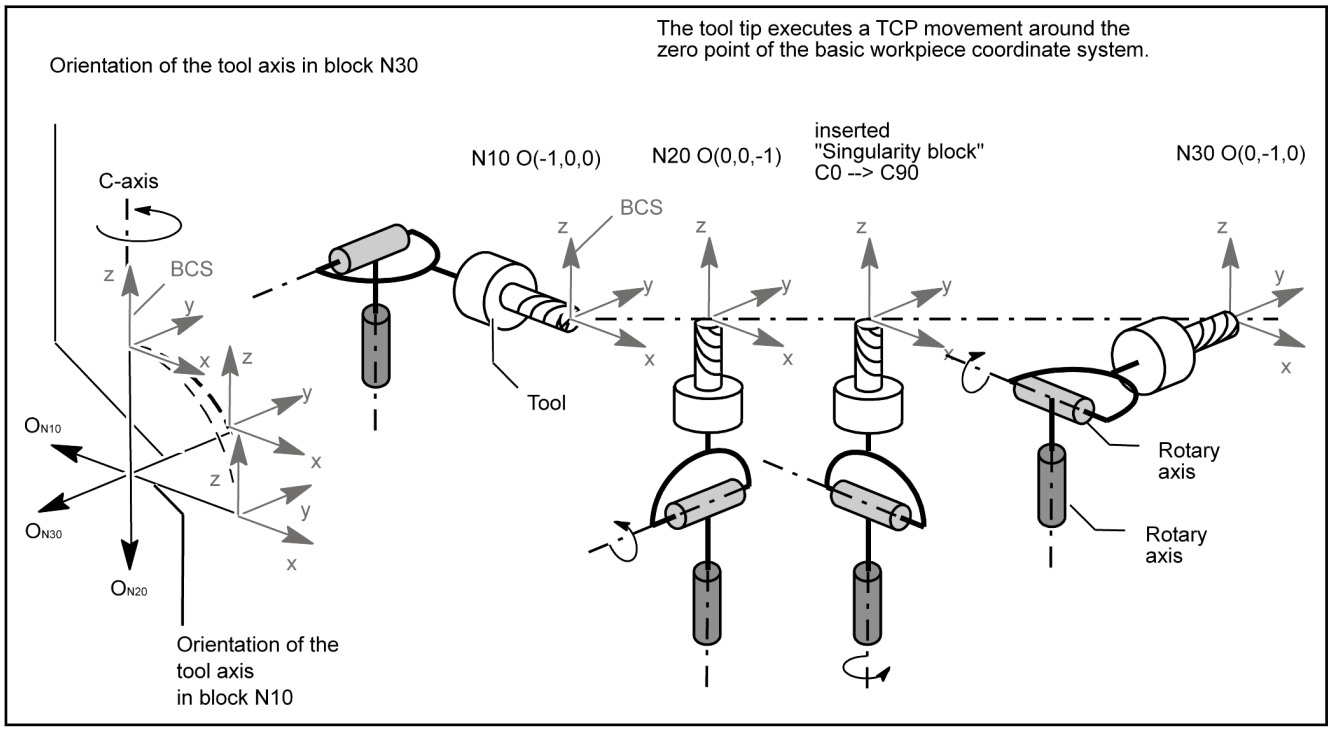


Fig. 10-9: Automatic rotation of the C-axis from 0 degrees to 90 degrees

Motion in the vicinity of O = (0,0,1):

If the orientation vector closely rotates past the pole O = (0, 0, 1) and the O-velocity is constant, the C-axis has to rotate the faster, the closer the orientation vector moves to pole O = (0, 0, 1). The following figure shows the projection of the O-motion to the xy-plane.

At a distance from the pole position, there are small changes in angle of the C-axis and close to the pole there are large changes in angle. Accordingly, the C-axis considerably limits the maximum possible path velocity for a given path segment (a NC block). In path planning, the path velocity is then reduced so that the maximum permitted C-axis velocity is not exceeded. This may possibly result in an extremely slow creeping movement on the whole path segment.

With an exact passage through the pole position, either - depending on the kinematics - no C-axis velocity or as in case of the Cardanic milling head, a constant C-axis velocity. The path velocity is then significantly less limited.

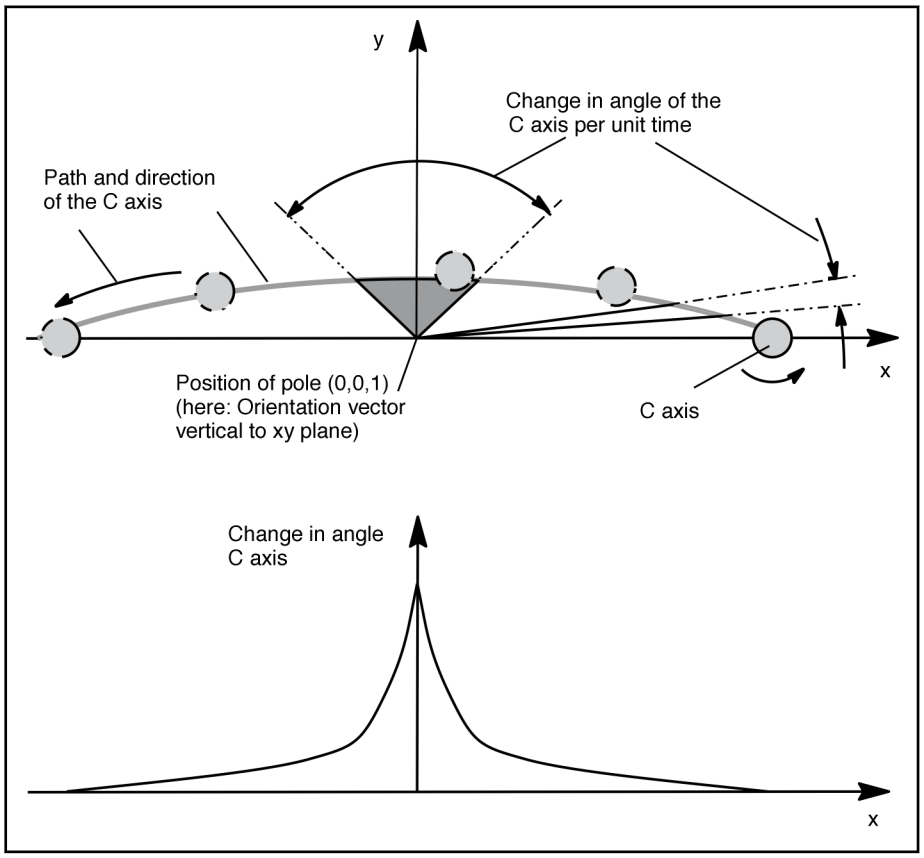


Fig. 10-10: Projection of the O-motion to the xy-plane

In this case, a capture range (in degree) can be defined around the pole (0,0,1) in the machine parameter TRA/AxTrafo[x]/EpsAxTrafo (MP 1030 00 60) "Epsilon environments". Orientation motions through this area are modified so that they run exactly through the pole position. The modification is executed so that the difference between the original and the modified O-path is higher than the parameter value.

Restrictions

Online correction of phi and theta is not possible.

Relevant machine parameters

TRA/AxTrafo[x]/ EpsAxTrafo MP 1030 00160	Epsilon environment Defines a capture range (in degree) around the orientation theta = 0°. Orientation motions through the epsilon environment are then exactly re-routed through the position theta = 0°. The parameter is only significant for axis transformation capable of vector orientation.

Tab. 10-8: Relevant machine parameters

10.3.3 Linear orientation motion

Function

The **linear orientation motion** also uses the coordinates phi and theta. However, the motion is executed as a straight line in a theoretical phi-theta plane.

On the standard ball, the tip of the orientation vector - contrary to vector orientation - does not describe a large circle but a linear motion in the phi and theta coordinates. In the following figure, the difference can be seen taking the example of a motion from "phi0 theta45" to "phi90 theta45".

The linear orientation motion does not require any singularity treatment (see chapter "Special features" on page 172) from the respective axis transformation, as no phi jump can occur.

Find a list of the special characteristics of the linear orientation motions and its differences from the vector orientation in the following:

- The prerequisite is an axis transformation configured in the machine parameters with two orientation coordinates and the orientation ID 1 (e.g. 3232101). Switching on this kind of axis transformation via Coord(<n>) automatically activates the linear orientation motion.
- The orientation does not always refer to the basic coordinate system, i.e. coordinate transformations (inclined plane, placement, fixture adaptation) are **not** supported, contrary to the vector orientation.
- It is programmed via the coordinate names phi and theta or via the vector syntax O(O_x, O_y, O_z). Both programming types are equivalent.
- A special path search logic ensures that no rotation greater than 180° is performed.
- The phi-theta values are not converted into the definition interval

  0° ≤ phi < 360°, 0° ≤ theta < 180°.

  Thus, phi-theta values outside the definition interval are shown in the display.
- Programming a rotary axis with the functions "ROTAX(..) O(..)" is **not** permitted.
- The mapping functions "mirroring" and "rotation" are programmed as for the vector orientation.
- The "online correction" function can be used for phi and theta.

The following illustration shows the "linear orientation motion" compared to the "vector orientation motion":

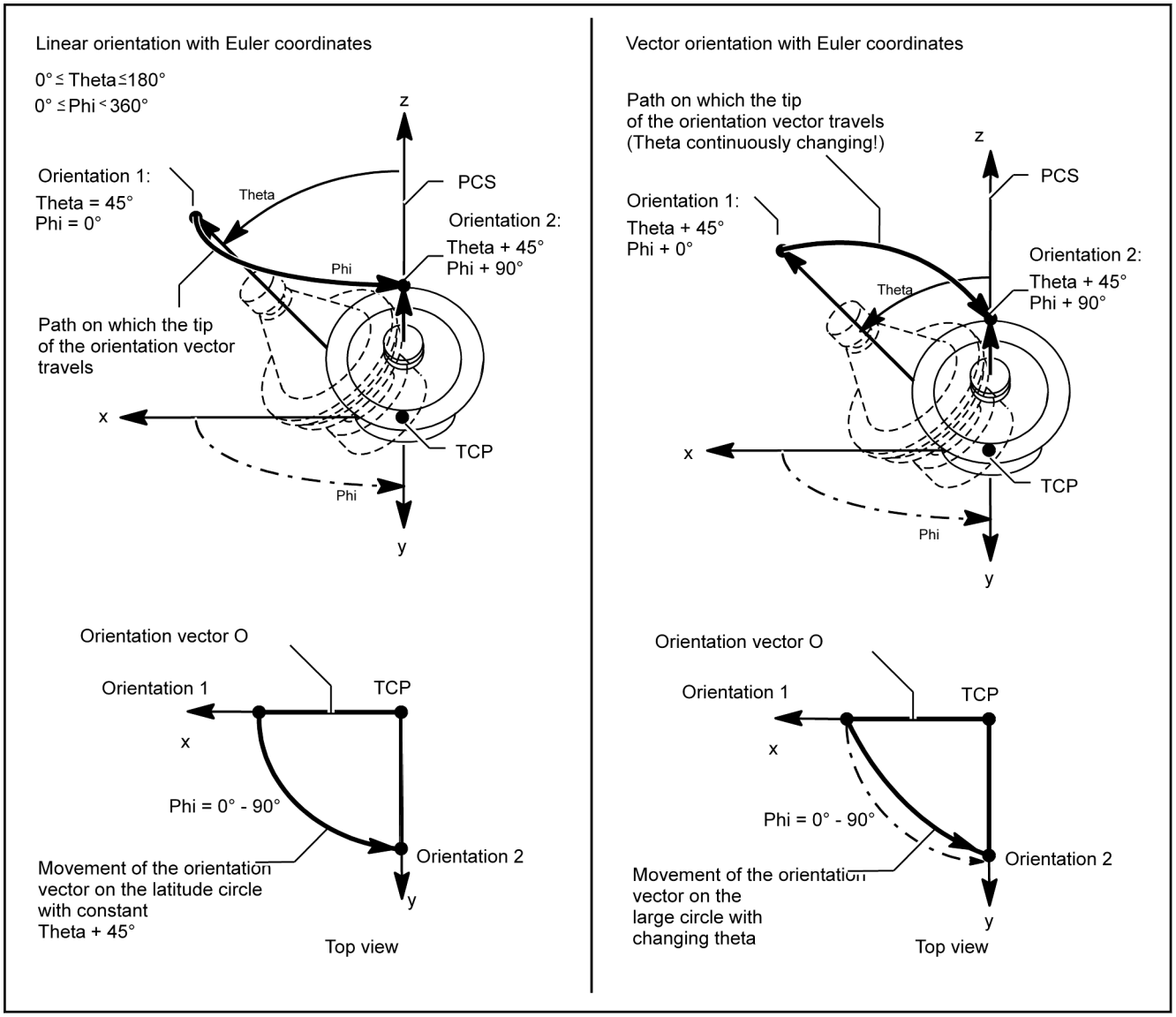


Fig. 10-11: Linear orientation - vector orientation

Relevant NC functions

Coord(<i>)	Activate axis transformation i. Spatial coordinates can only be programmed together with an active axis transformation i. No axis addresses must be programmed which are affected by switching to spatial coordinates. i: Number of the axis transformation to be activated.
Coord(0)	Deactivating axis transformation and thus the spatial coordinate programming.
Programming the linear **orientation** using two alternatives:	

phi<phi> theta<theta>	Orientation using the two first Euler angles phi and theta (angles of direction). The names "Phi" and "theta" are defined in the machine parameter CHAN/Ch[x]/Coord/Wcs/OriCoord[1..3] (MP7080 00010). Programming is only possible absolute in degree. Ex: G1 x10 y50 z30 phi90 theta90
O(<phi>,<theta>)	Refer to the specifications with "phi, theta" above Example: G1 x10 y50 z30 O(90,90)
O(O_x, O_y, O_z)	Orientation with Cartesian components O_x, O_y, O_z of the orientation vector. Programming absolute. Example: G1 x10 y50 z30 O(0,1,0.707)

Tab. 10-9: Relevant NC functions

10.3.4 Tensor orientation motion

Function

Tensor orientation refers to orientation of non non-rotation symmetrical tools (e.g. robot grippers).

A tool coordinate system (TCS) that is permanently connected to the tool has an orientation with regard to the program coordinate system (PCS) defined by Euler angles phi, theta and psi or by the "3x3 orientation tensor" O (see chapter 10.2.4 "Orientation tensor" on page 164).

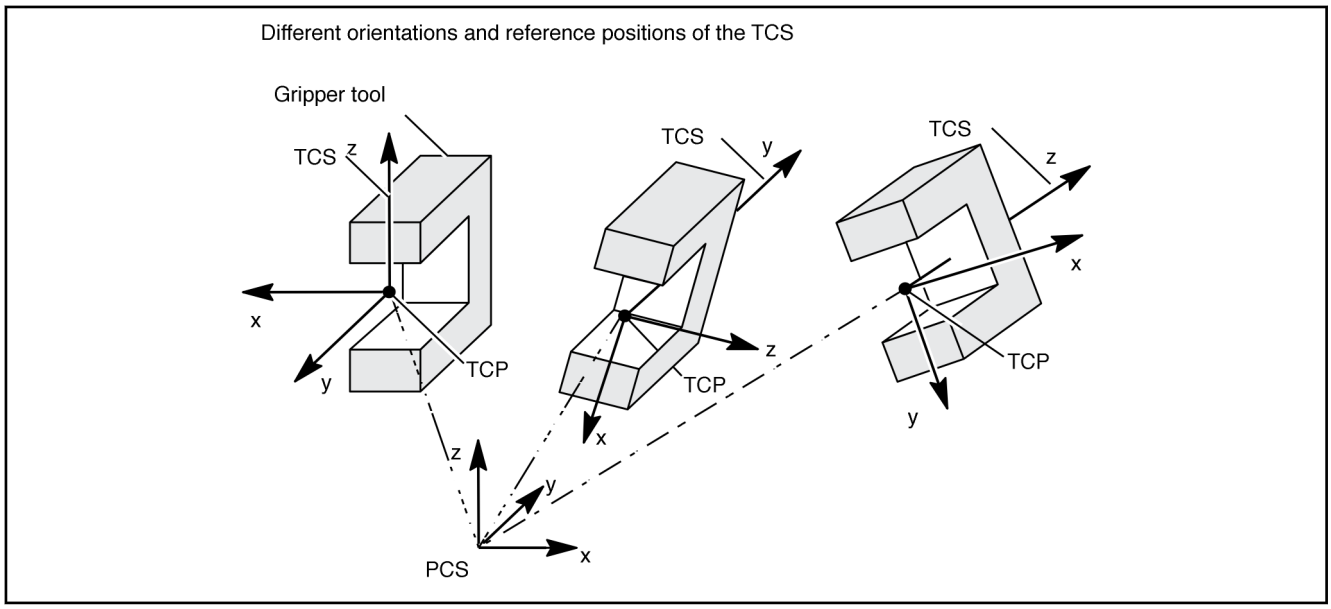


Fig. 10-12: TCS orientations

Relevant NC Function

The tensor orientation is activated using a corresponding axis transformation (see tab. 10-1 "Orientation motion type depending on the axis transformation type" on page 166) (e.g. type 3333301).

It is not necessary to specify the column vectors

$$\vec{e}_x = \begin{pmatrix} o_{11} \\ o_{21} \\ o_{31} \end{pmatrix} \quad \vec{e}_y = \begin{pmatrix} o_{12} \\ o_{22} \\ o_{32} \end{pmatrix} \quad \vec{e}_z = \begin{pmatrix} o_{13} \\ o_{23} \\ o_{33} \end{pmatrix}$$

of the orientation tensor standardized to 1. They are automatically standardized to "1" the NC.

The **TCS orientation** can be programmed with the following alternatives:

phi<phi> theta<theta> psi<psi>	Orientation with the Euler angles phi, theta and psi. Euler angle names are defined in the machine parameter CHAN/Ch[x]/Coord/Wcs/OriCoord[1..3] (MP7080 00010). Example: G1 x10 y50 z30 phi90 theta90 psi45
$O_x(<o_{11}>,<o_{21}>,<o_{31}>)$ $O_y(<o_{12}>,<o_{22}>,<o_{32}>)$ $O_z(<o_{13}>,<o_{23}>,<o_{33}>)$ - or - $O_x(<phi_x>,<theta_x>)$ $O_y(<phi_y>,<theta_y>)$ $O_z(<phi_z>,<theta_z>)$	$O_x(..)$ defines the direction of the x-coordinate of the TCS in the program coordinate system. The direction can be specified with Cartesian vector components $O_x(<o_{11}>,<o_{21}>,<o_{31}>)$ or with polar coordinates $O_x(<phi_x>,<theta_x>)$. The same applies to the definition for the column vectors $O_y(..)$ and $O_z(..)$. Only two of the three TCS coordinates need to be programmed. Example: G1 x10 y50 z30 Ox(1,0,0) Oy(0,0.707,-0.707)

ROTAX(<R_x>,<R_y>,<R_z>) O(<β>) - or - ROTAX(<phi_u>,<$theta_u$>) O(<β>)	ROTAX(..) defines the orientation of the rotary axis around which the orientation tensor rotates. The direction can be specified in Cartesian vector components R_x, R_y,R_z or in polar coordinates phi_x, $theta_x$. ROTAX(..) can only be programmed absolutely. O(..) defines the angle β, around which the initial orientation vector rotates. O(..) can only be programmed incrementally and in degree. β can assume any value, i.e. several rotations are also possible. Example: ROTAX(0,45) O(720) ROTAX(1,0,1) O(720)
O(<w>,<x>,<y>,<z>)	Quaternion programming with w = cosβ/2 x = $R_x$sinβ/2 y = $R_y$sinβ/2 z = $R_z$sinβ/2 Describes a TCS orientation realized by rotating around the rotary axis R = (R_x,R_y, R_z) with angle of rotation β from position TCS = PCS.

Tab. 10-10: Programming alternatives - TCS orientation

☞ For a detailed description of the individual parameters, refer to the "MTX Programming Manual".

Movement of the orientation tensor:

The following dimensions are involved in the motion of the orientation tensor:

- O_s: Initial orientation tensor (initial orientation of the TCS)
- O_e: Final orientation vector (final orientation of the TCS)
- R: Rotary axis
- β: Angle of rotation

To execute an orientation motion, the following variables can be defined:

Orientation motion on definition of O_e

Initial orientation tensor O_s is rotated around an internally calculated rotary axis R to the final orientation tensor. For the internally computed rotary angle β, the following applies: β ≤ 180°.

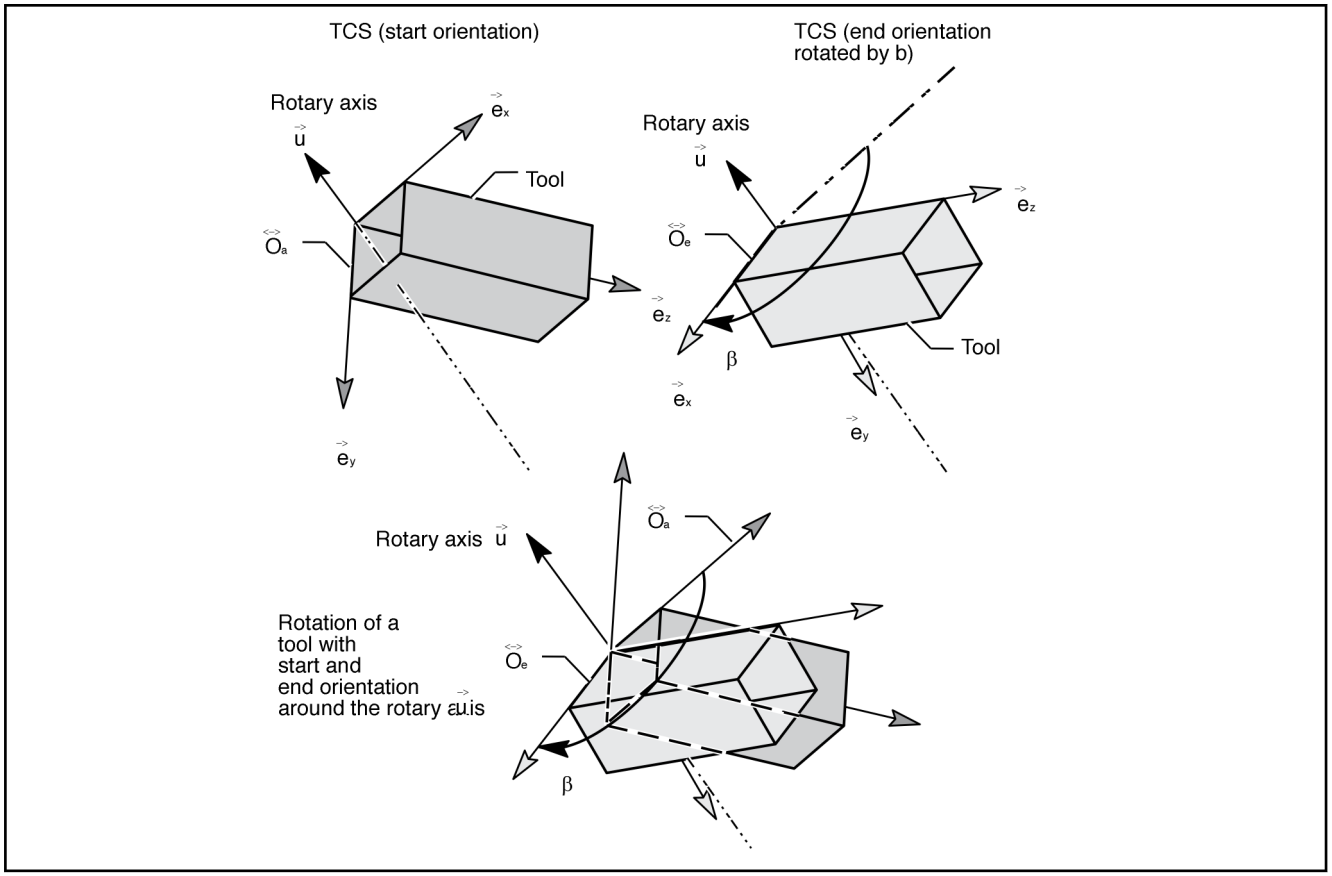


Fig. 10-13: Orientation motion

Orientation motion when specifying the rotary axis R and the angle of rotation β:

- Initial orientation tensor O_s is rotated around the programmed rotary axis R by angle β.
- The orientation of the rotary axis is programmed using "ROTAX(..) ".
- The motions of the orientation tensor is generated with "ROTAX(...) O(..)" generated.

  Example:

  ROTAX(...) O(360) turns the initial orientation tensor O_s for exactly one rotation to its original position. The programmed angle of rotation is generally interpreted as incremental value.
- The desired direction of rotation can be generated by positive and negative angles of rotation.

In general, TCS coordinates e_x, e_y, e_z describe a conical surface around the rotary axis R during the orientation motion.

Exception:

R is parallel to e_x, e_y, e_z. Then, the conical surface turns into a spherical surface.

Rotating velocity of the orientation tensor:

As for the vector orientation motion, the assignment 1 mm ≙ 1 degree applies. In case of an orientation motion, a feed of F=100mm/min would result in an angle velocity of 100 degrees/min of the rotating phasor around the rotary axis R. In case of a spatial plus orientation motion, the rotation speed de-

pends on whether O is a member of the feed group or not. The examples of chapter "Relevant NC functions" on page 167 apply.

The velocity of the orientation motion is limited by Omega (see chapter "Relevant NC functions" on page 167).

Special features: **Singularity of the motion of the orientation tensor:**

The occurrence of singularities depends on the type of axis transformation used. A singularity occurs in 6-axis transformations of type 3333301 if the orientation tensor results in a position of ±90° for the center axis out of the three rotary axes.

Non-constant orientation motions at this position are bridged by an **automatic NC block insertion** (also refer to chapter 10.3.2 "Vector orientation motion" on page 166).

10.4 Tangential tool guidance

10.4.1 Description

Function

The tangential tool guidance guides a tool axis (rotary or endless axis) or a rotary coordinate tangentially to the programmed path in the plane of a channel which is active at the respective time.

Features:

- Parameterizable setting angle (angle between the symmetry axis of the tool and the path tangent).
- The tool symmetry is taken into account. Thus, the control can independently optimize the setting rotation of the tool necessary for contour knees ("shortest way" logic).
- Intermediate blocks to rotate the tool are inserted automatically if the knee angle between two contour elements exceeds a parameterizable value.
- Communication with the PLC if an intermediate block has to be automatically inserted in case of a contour knee. This allows the PLC to take the necessary steps before the intermediate block is processed (e.g. lift the tool), before the NC rotates the tool.
- Complete parameterization of the "tangential tool guidance" via the TangTool (TTL) command in the part program or via machine parameters.
- Read-only access to active parameters of TangTool is possible via CPL-SD function.

Restrictions

- Can only be used together with straight line-circle interpolation, not for splines.
- Function TangToolOri (TTO) "Tangential tool orientation" (for nibbling/punching, see chapter 12.6 "Tangential tool orientation " on page 455) must not be active simultaneously.
- Only rotary or endless axes and rotary coordinates such as phi and theta are permitted as tool axes.
- TangTool must not be programmed in a traversing block.
- Automatic insertion of an intermediate block is impossible if a single WAIT block has been programmed between 2 traveling blocks while "tangential tool guidance" is active.

- Tangent start and end angle of the programmed contour section have to be within the limit switch range of the tool axis (provided the tool axis is a rotary axis).

Relevant NC functions

TangTool(1) -or- TTL(1)	Enabling the "Tangential tool guidance" with initial values of the machine parameters. The tool axis is only set with the next traversing block. Depending on the active intermediate block angle, the tool axis either rotates in an automatically generated intermediate block or with a jump to its new setting position. The TTL parameters are described in the programming manual.
TangTool(0) - or - TTL(0)	Deactivating the "tangential tool guidance".

Tab. 10-11: Relevant NC functions

Relevant IF signals

iCh_TangTRotCmd	"Tool rotation (TangTool)" (channel interface). The high-signal informs the PLC that the knee angle between two contour elements exceeds the currently set intermediate block angle. Use MP 7050 00260 or the TangTool parameter "PLC" to specify whether the NC is to wait for the release by the PLC before executing the intermediate block.
qCh_TangTRotRel	"TangTool (TTL) released" (channel interface) Release for TangTool tool rotation. The high-signal releases the execution of the intermediate block by the NC, provided "1" is assigned to the MP 7050 00260 or the TangTool parameter "PLC". The control only executes the blocks following the intermediate block when the PLC resets this signal.

Tab. 10-12: Relevant IF signals

Relevant SD functions

SD(131,1)	Provides the channel coordinate number (channel axis number) of the rotary axis
SD(131,2)	Provides tool symmetry
SD(131,3)	Provides the setting angle in degree
SD(131,4)	Provides the intermediate block angle in degree

Tab. 10-13: Relevant SD functions

Relevant machine parameters (MP)

The values of the machine parameters apply as initialization values if TangTool is programmed without parameters. If required, overwrite the initialization values in the TangTool block with the parameters given in brackets (see the "MTX Programming Manual" for the syntax):

7050 00210	System axis number of the rotary axis (no coordinate can be specified)	(TAX)
7050 00220	Tool symmetry	(SYM)
7050 00230	Intermediate block angle	(IA)
7050 00240	Setting angle	(ANG)
7050 00250	Initialization of the setting angle	-
7050 00260	NC-PLC communication for the intermediate block	(PLC)

Tab. 10-14: Relevant machine parameters (MP)

11 Axis transformation

11.1 Introduction

11.1.1 Explanations

The transformations currently available in the NC are described in the **Axis transformation** chapter. See chapter 11.1.8 "Overview on axis transformations(in a Table)" on page 192 for a list of all axis transformations.

Additionally, the last subchapter describes the **axis kinematic calibration** which significantly increases the positioning accuracy of the transformed axes.

11.1.2 Axis transformation principle

If the coordinate axes (= program coordinates) of the workpiece-related coordinate systems no longer run parallel to the machine axes, the NC is initially not able to directly derive axis command values for the machine axes. In this case, a programming system is required to generate the axis command values and to store them into the program. Disadvantageously, the program only runs on a machine with one specific tool. The NC avoids these restrictions via **axis transformation**.

The program coordinates are interpreted as "system coordinates" when the axis transformations are active. Furthermore, the system coordinates are divided into spatial coordinates and Cartesian coordinates. These coordinates refer to a machine-independent coordinate system positioned and oriented anywhere in the system (also refer to the manual "MTX Functional Description - Basics", chapter "Axes, coordinates, coordinate systems").

The implementation of the program coordinates into the real axis system of the machine creates a machine-specific axis transformation considering or defining the geometric-kinematic correlations between machine and tool. Here, axis transformation calculates the required portions of all involved real axis for each position in space to be approached.

For machining in an "inclined plane", up to five axes can be required for plain surface milling. In this case, only the block "G1x 100 phi 100 theta 45" is programmed with the transformed coordinates x, phi, and theta. They act upon the linear axes X, Y, and Z as well as the rotary axes B and C.

For the axis transformation, the program coordinates have to be internally "prepared" by "Backward transformation" (also see the following figure):

1. If compensations to the programmed coordinates caused displacement and/or rotation with regard to the program coordinates system, the compensations are first subtracted by means of a "coordinate transformation". This provides spatial coordinates relative to the basic workpiece coordinate system (BCS).
2. These spatial coordinates are input variables for the axis transformation AT2.
3. Backward transformation of these spatial coordinates to Cartesian coordinates after "subtraction" of the axis transformation AT2, the result is the input variable for axis transformation AT1.

   By "subtracting" the axis transformation AT1, these Cartesian coordinates now provide the actual axis command values.

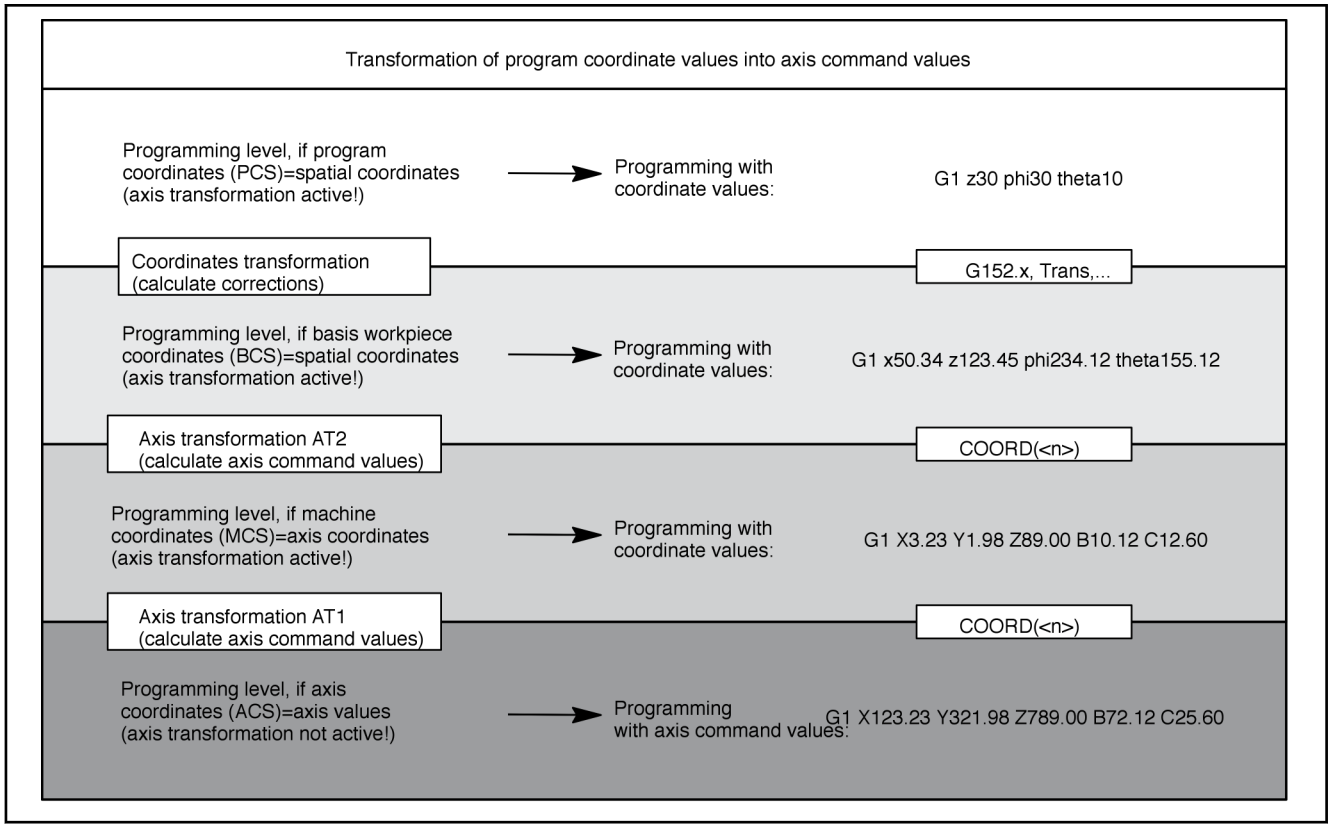


Fig. 11-1: Transformation of the program coordinates values in axis command values

The manner how the axis command values are generated from the program coordinates via axis transformation, indicates that spatial or Cartesian coordinates can have different effects on the physical axis configuration with each axis configuration. This can be carried out by different settings of the axis transformations in the machine parameter.

11.1.3 Definitions

Type ID

All axis transformations are differentiated by a type ID and are entered into the MP 1030 00110.

The type ID is a seven-digit number whose figures have the following meaning:

Digit	**1**	**2**	**3**	**4**	**5**	**6 a. 7**
	A_l	**A_r**	**E_l**	**E_r**	**O**	**NN**

Digit	Abbr.	Description	Options
1	A_l	Forward transformation outputs, linear	**0..3**
2	A_r	Forward transformation outputs, rotary	**0..3**
3	E_l	Forward transformation inputs, linear	**0..3**

Digit	Abbr.	Description	Options
4	E_r	Forward transformation inputs, rotary	**0..3**
5	O	ID of the supported orientation motion	**0**: No orientation motion **1**: Linear orientation motion in the rotary axes **2**: Vector rotation (rotary pointer motion) of the orientation vector **3**: Tensor rotation of the orientation
6 and 7	NN	Continuous numbering for internal purposes	**01 .. 99** Number is type-related

Tab. 11-1: Type ID - Meaning of the digits

The type ID reveals the functions active when selecting the axis transformation:

- The axis transformation with the number **i** (from the i-th parameter block in the MP) is activated via the COORD(<I>) command, provided the configuration in the MP does not contain any incorrect entries.
- If the activated axis transformation supports **vector or tensor orientation**, the respective NC functions are activated for the orientation motion as well.
- The programming of **coordinate names** is enabled, i.e. the axes are switched to coordinates.
- Axes participating in the axis transformation may not be programmed when the axis transformation is active.

  Example:

  Coordinate names: | x| y| z| phi| theta| psi| U| V|

  Axis names: | X| Y| Z| A| B| C| U| V|

  The axis transformation defines how many of the maximum six spatial coordinates can be programmed. The spatial coordinates consist of a maximum of three translatory "position coordinates" and a maximum of three rotary "orientation coordinates". Their names can be freely set in the machine parameter (in the following, "x", "y" and "z" are used for position coordinates and "phi", "theta" and "psi" for orientation coordinates). In the example, U and V are other channel axes not participating in the axis transformation.

☞ Switching to the axis transformation causes the display of the workpiece coordinates to jump from "axis positions" to "coordinate values". The axis names of the axis field change to the coordinate names as shown in the example above.

ID for axis arrangements

For a fast recording of arrangement, type and number of axes participating in the 5-axis and 6-axis transformations, the following ID is defined:

L = Name for linear axes

R = Name for axes of rotation

Example 1:

LLLRR stands for three linear axes followed by two rotary axes

Example 2:

RLLLR stands for one rotary axis rotating around three linear axes followed by one rotary axis (mostly rotary table)

Input variables, output variables

Input variables: Designates the transformed coordinates from which the axis transformation generates real axis/coordinate positions via backward transformation.

Output variables: Designates the axes or coordinates from which the axis transformation generates transformed input variables.

Example:

5-axis transformation

Output variables in the MCS: X,Y,Z,R2,R1

Input variables in the BCS: x, y, z, φ, θ

11.1.4 Points of action

Properties

- The points of actions AT1 and AT2 define the locations in the system at which the axis and/or coordinate values of a reference system are converted (transformed) into coordinate values of another reference system.
- Only specific transformations may be effective at each point of action.
- As a point of action is specified indirectly through the respective transformation, it cannot be programmed.

Point of action AT1: An axis transformation at the point of action AT1 transforms between the axis coordinate system (ACS) and a Cartesian machine coordinate system (MCS) that is permanently connected to the machine frame. An AT1 is required with machine kinematics without Cartesian axis arrangement as e.g. rod kinematics.

Currently, this applies to the following axis transformations at AT1:

- Bipod transformation (type: 2121001, 3131001)
- SCARA transformation (type: 2103011)
- Non-orthogonal axes (type: 2020001, 2020002, 3030001)

When one of the above listed types is activated in the NC program, the respective axis transformation automatically moves to point of action AT1.

An axis transformation at AT1 can be followed by a suitable axis transformation AT2 (between MCS and BCS).

Example:

The gripper coordinates (x, y, z, φ, ϑ) of a SCARA robot (AT1) are programmed with the help of the 5-axis transformation (AT2).

Point of action AT2: An axis transformation at the point of action AT2 transforms between the machine coordinate system MCS and the basic workpiece coordinate system BCS (= WCS(1)).

At AT2, the following axis transformations are used for example:

- Face transformation (type: 2011001)
- Cylinder jacket transformation (type: 3021001, 3021002)
- 5-axis transformation LLLRR (type: 3232201, 3232101, 3032001)
- 5-axis transformation RLLLR (type: 3232202)
- 5-axis transformation RRLLL (type: 3232203)
- 5-axis transformation LLLRR-Cardanic (type: 3232211)
- 6-axis transformation LLLRRR (type: 3333301, 3033001)

If one of the transformation types listed above is activated in the NC program, the respective axis transformation automatically moves to the point of action AT1. Depending on the machine kinematics, an AT1 has to be set ahead of an AT2 (refer to the examples in the manual "MTX Functional Description - Basics", chapter "Axes, coordinates, coordinate systems") providing Cartesian coordinates in the MCS.

11.1.5 Configuring axis transformations

The input variables for the forward transformation are specified via the MP 1030 00120 "System axis/coordinates of the transformation".

Example:

Bipod transformation at point of action 1

MP 1030 00120	
[1] 1	(Axis YL)
[2] 2	(Axis YR)
[3] 3	(Axis Za)
[4] 9	(Axis Sc)

Example:

Axis transformation at point of action AT2, superimposed on bipod transformation

MP 1030 00120	
[1] -6	(System coordinate X)
[2] -5	(System coordinate Y)
[3] -4	(System coordinate Z)
[4] 7	(Axis C)
[5] 6	(Axis B)

Example:

5-axis transformation at point of action AT2, no transformation at point of action AT1

MP 1030 00120	
[1] 1	(Axis Xa)

[2] 2	(Axis Ya)
[3] 3	(Axis Za)
[4] 7	(Axis C)
[5] 6	(Axis B)

In MP 103000130 "Axis and coordinate positions of the reference position", the positions for axis transformations at AT2 refer to the MCS. For axis transformations at AT1, refer to the ACS.

In MP 1030 00140, the lengths and angle parameters for the current axis transformation are set. These can be optimized using the "Axis kinematic calibration "function regarding the **exact** alignment of the axes/coordinates (refer to the manual "MTX Functional Description - Basics", chapter "Axes, coordinates, coordinate systems").

11.1.6 Activating and deactivating axis transformations

Conditions:
- Activation: AT1 has to be activated before AT2
- Deactivation: AT2 has to be deactivated before AT1

Activation of the axis transformation at AT1 with:

COORD(I)	The number I (I=1,..,20) identifies one of the 20 axis transformation blocks in the machine parameters (see the "MTX Machine Parameters" manual).

Programming COORD(I) triggers the following actions:

1. The axis transformation at AT1 with the number i is activated, provided the configuration in the machine parameters does not contain any incorrect entries.
2. Coordinate names and positions in the MCS and in all following coordinate systems are switched accordingly (refer to the examples in the manual "MTX Functional Description - Basics", chapter "Axes, coordinates, coordinate systems").

Activating the axis transformation at AT2 with:

COORD(i)	The number I (I=1,..,20) identifies one of the 20 axis transformation blocks in the machine parameters (see the "MTX Machine Parameters" manual).

Programming COORD(i) triggers the following actions:

1. The axis transformation with the number i is activated, provided the configuration in the machine parameters does not contain any incorrect entries.
2. If the activated axis transformation supports vector or tensor orientation, the corresponding orientation NC function is activated.
3. Coordinate names and positions in the WCs are switched accordingly.

Activating the axis transformation via system date:

As an alternative to the machine parameter number described above, axis transformation can also be activated via system date. Therefore, the control-internal type **SysAxTrafo_t**, is provided comprising all axis transformation-specific machine parameters.

The structure type SysAxTrafo_t is defined in the schema file

feprom/schemas/sdaxtrf.xsd (see chapter 11.1.7 "Displaying active transformation data" on page 190). The MTB defines the respective SD variable of

the type SysAxTrafo_t in the **usrfep/SDDefMTB.xml** file (e.g. with the name MTBAxTrafo). This can also be an array of the type SysAxTrafo_t.

Initialization values can be stored in the file **usrfep/SDDatMTB.xml** .

Example:

Activating the axis transformation via system date

Within an NC program, SD.MTBAxTrafo[3].LenParam[5] can be used to address the fifth length parameter of the third axis transformation parameter block. As this can be either read-only or write access, online changes in the transformation parameters are possible.

If COORD is programmed with the name of a transformation SD variable, the transformation data is taken from the respective system data.

COORD(<[SDTrfName]>)	"SDTrf Name" is the name of the system date of type SysAxTraf_t containing the transformation data to be activated.

Deselecting an axis transformation:

COORD (0,[<TrafoLocation>])	Deactivating the active transformation	
	The second parameter "TrafoLocation" identifies the point of action of the transformation to be deactivated. Thus, it can assume the values 1 or 2.	
	Default: TrafoLocation = 2	
COORD(0)	Short form for the deactivation of the active transformation 2	
	Example:	
	COORD(4)	;Activating an AT1
	COORD(3)	;Activating an AT2
	x.. y.. z..	
	COORD(0)	;Deactivating AT2, identical to COORD(0,2)
	COORD(0,1)	;Deactivating AT1
COORD ([SD.MTBAxTrafo[4]]	;Activating the axis transformation stored in system date SD.MTBSxTrafo[4]	

Programming COORD(0,1) triggers the following actions:

1. The axis transformation previously active at AT1 is deactivated.
2. The names and positions of coordinates in the MCS and in all subsequent coordinate systems are switched back to ACS-names and ACS-positions

Programming COORD(0.2) triggers the following actions:

1. The axis transformation previously active at AT2 is deactivated.
2. Any currently active orientation NC function is deactivated.
3. The names and positions of coordinates in the WCS are switched back to MCS names and MCS positions.

 An axis transformation at AT2 can be directly switched to another axis transformation without previous deactivation.

Example program: At program start, both axis transformations are switched off:

```
N01 G1 YL10 YR10 U20 V30          ;G1 is effective in the WCS.
                                  ;The WCS identifiers YL, YR, U, V apply
N02 COORD(4)                      ;Activating bipod transformation
N03 X20 Y30 Z40 psi0 B120 C120    ;WCS identifier X, Y, Z, y, B, C
N04 G06 YL1.0 YR1.5               ;G06 is effective in the ACS
N05 COORD(3)                      ;Activating 5-axis transformation
N06 x10 y10 z10 phi20 theta30 U80 ;WCS-identifier x, y, z, φ, ϑ
N07 G76 X0 Y0 Z0                  ;MCS-identifier X, Y, Z
N08 COORD(0,2)                    ;Deactivating 5-axis transformation
                                  ;(point of action;AT2)
N09 X0 Y0
N10 COORD(0,1)                    ;Deactivating bipod transformation
                                  ;(point of action;AT1)
N11 YL0 YR10
```

11.1.7 Displaying active transformation data

There are two MTX-internal SD variables in which axis transformation data active at the time of block preparation is stored:

- SysAxTrafo1 and
- SysAxTrafo2

They are defined by the type "SysAxTrafo_t" and in the file "feprom/SDDef.xml". They are channel-specific SD variables.

SysAxTrafo1 contains the axis transformation data active at the point of action 1 (AT1) and SysAxTrafo2 contains the axis transformation data active at the point of action two (AT2). If no axis transformation is active, the variable is nulled completely.

The structure type SysAxTrafo_t of the system data SysAxTrafo1 and SysAxTrafo_2 is defined in the schema file feprom/schemas/sdaxtrf.xsd. It comprises the axis transformation-specific machine parameters 1030 00110, 1030 00120, 1030 00125, 1030 00140, 1030 00150 and 1030 00160.

In detail, SysAxTrafo_t looks as follows:

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified">
<xs:include schemaLocation="basic_ty.xsd"/>
<xs:complexType name="SysAxTrafo_"t>
<xs:annotation>
<xs:documentation>Configuration data of the axis transformation
</xs:documentation> <xs:documentation>Author: KE</xs:documentation>
<xs:documentation>Caution!
If changes are made, "acb1sysd.h" has to be changed at the same time.
</xs:documentation>
</xs:annotation>
<xs:sequence>
```

<xs:element name="AxTrafoNo" type="Int_t">
<xs:annotation>
<xs:documentation>Axis transformation number:
0, No axis transformation enabled
1..20, e.g. active/activate Coord (4)
</xs:documentation>
</xs:annotation>
</xs:element>
<xs:element name="Type" type=$_{Int_t}$> <xs:annotation>
<xs:documentation>Transformation type corresponds to MP 1030 00110
</xs:documentation>
</xs:annotation>
</xs:element>

<xs:element name="AxisAssignment" type="Byte_t" minOccurs="0" maxOccurs="8">
<xs:annotation>
<xs:documentation>System axes/coordinates of the transformation corresponds to MP 1030 00120
</xs:documentation>
</xs:annotation>
</xs:element>

<xs:element name="AxisClassification" type="Int_t" minOccurs="0" maxOccurs="8">
<xs:annotation>
<xs:documentation>Axis classification of the transformation axes corresponds to MP 1030 00125
</xs:documentation>
</xs:annotation>
</xs:element>

<xs:element name=$_{AxZeroPos}$ type="Double_t" minOccurs="0" maxOccurs="8">
<xs:annotation>
<xs:documentation>Axis positions of the reference position corresponds to MP 1030 00130
</xs:documentation>
</xs:annotation>
</xs:element>

<xs:element name="LenParam" type="Double_t" minOccurs="0" maxOccurs="16">
<xs:annotation>
<xs:documentation>Length and angle parameter corresponds to MP 1030 00140
</xs:documentation>
</xs:annotation>
</xs:element>

<xs:element name="TCSOrientation" type="Byte_t" minOccurs="0" maxOccurs="8">
<xs:annotation>
<xs:documentation>Reference orientation of the tool coordinate system corresponds to MP 1030 00150
</xs:documentation>
</xs:annotation>
</xs:element>

<xs:element name="EpsilonRanges" type="Float_t">
<xs:annotation>
<xs:documentation>Epsilon environments corresponds to MP 1030 00160
</xs:documentation>
</xs:annotation>
</xs:element>

</xs:sequence>
</xs:complexType>
</xs:schema>

Example:

SysAxTrafo_t

101 TYPCHAN3%	= SD.SysAxTrafo2[3].Type
102 TYPACTCH%	= SD.SysAxTrafo2.Type
103 AT1NO%	= SD.SysAxTrafo1[2].AxTrafoNo

TYPCHAN3%	Returns the type of the active AT2 of the third channel (corresponds to MP 1030 00110, e.g. 32322201).	
TYPACTCH%	Provides the type of the active AT2 of the current channel.	
AT1NO%	Provides the AxTrafoNo of the active AT1 of the second channel.	
	Note:	
	AT1NO% = 0	No axis transformation enabled
	AT1NO% = 1..20	Coord (AT1NO%) active (e.g. Coord (4)
	AT1NO% = 1000	e.g. Coord (SD.MTBAxTrafo[3]) active. If the axis transformation was activated via an SD variable, "1000" is always displayed

In the NC program, the variables may only be used read-only.

11.1.8 Overview on axis transformations(in a Table)

The following table shows the state of the currently available axis transformations:

Description	Kinematic type	Type	Axis number	Point of action	In chapter...
Bipod rod kinematics	Bipod	3131001	4/3	1	chapter 11.2 "Bipod axis transformations " on page 196
	Bipod	2121001	3/2	1	chapter 11.2 "Bipod axis transformations " on page 196
SCARA	SCARA	2103011	3	1	chapter 11.3 "Scara-axis transformation " on page 205
	SCARA	3113011	4	1	
LLR - Cartesian	LLR	3021005	3	1	
Non-orthogonal axes	Non-orthogonal	3030001	3	1	chapter 11.5.1 "Variant 1" on page 210
	Non-orthogonal	2020001	2	1	chapter 11.5.2 "Variant 2" on page 212
	Non-orthogonal	2020002	2	1	chapter 11.5.3 "Variant 3" on page 213
3-/4-axis transformation SCARA on linear axis	LRR	2112001	3	2	chapter 11.6 "4-axis transformation - Type 1: SCARA on linear axis" on page 219
	LLRR	3122001	4	2	chapter 11.6 "4-axis transformation - Type 1: SCARA on linear axis" on page 219
3-axis transformation LLR (concave milling)	LLR	2121002	3	2	chapter 11.20 "3-axis transformation LLR" on page 337
3-axis transformation LLR - Cartesian	LLR	3021004	3	2	
4-axis transformation with swivel axis	LLLR	3131002	4	2	chapter 11.8 "4-axis transformation - Type 2: LLLR" on page 242
4-axis transformation with swivel table	RLLL	3131003	4	2	chapter 11.9 "4-axis transformation - Type 3: RLLL" on page 248

Description	Kinematic type	Type	Axis number	Point of action	In chapter...
5-axis transformation with swivel/rotary axis LLLRR	LLLRR (Vector orientation)	3232201	5	2	chapter 11.10.2 "Variant 1 (with vector orientation)" on page 257
	LLLRR (Linear orientation)	3232101	5	2	chapter 11.10.3 "Variant 2 (with linear orientation)" on page 258
	LLLRR (Linear motion of the rotary axes)	3032001	5	2	chapter 11.10.4 "Variant 3 (with linear motion of the rotary axes) " on page 258
5-axis transformation with rotary table RLLLR	RLLLR (Vector orientation)	3232202	5	2	chapter 11.11 "5-axis transformation - Type 2: RLLLR (with rotary table) " on page 273
	RLLLR (Linear orientation)	3232102	5	2	chapter 11.11 "5-axis transformation - Type 2: RLLLR (with rotary table) " on page 273
5-axis transformation with swiveling/rotary table RRLLL	RRLLL (Vector orientation)	3232203	5	2	chapter 11.12 "5-axis transformation - Type 3: RRLLL(with swivel/rotary table) " on page 283
	RRLLL (Linear orientation)	3232103	5	2	chapter 11.12 "5-axis transformation - Type 3: RRLLL(with swivel/rotary table) " on page 283
	RRLLL (less-equipped)	3030013	3 (5)	2	chapter 11.16 "5-axis transformation - Type 5: Less-equipped RRLLL (with swivel/rotary table) " on page 308
Cardanic 5-axis transformation with swivel/rotary axis LLLRR	LLLRR (Vector orientation)	3232211	5	2	chapter 11.13 "5-axis transformation LLLCB - Type 4: Cardanic head with vector orientation motion" on page 293
	LLLRR (Linear orientation)	3232111	5	2	chapter 11.14 "5-axis transformation LLLCB - Type 4.1: Cardanic head with linear orientation motion" on page 303

Description	Kinematic type	Type	Axis number	Point of action	In chapter...
Cardanic 5-axis transformation with rotary table RLLLR	RLLLR (Vector orientation)	3232212	5	2	chapter 11.17 "5-axis transformation - Type 6: Cardanic RLLLR" on page 317
	RLLLR (Linear orientation)	3232112	5	2	chapter 11.17 "5-axis transformation - Type 6: Cardanic RLLLR" on page 317
Tripod transformation with wobble plate	LLLLL	3250201	5	2	chapter 11.18 "5-axis transformation - Type 7: Tripod transformation with wobble plate" on page 326
Tripod transformation with wobble plate and path slave axis	LLLLL	3250202	6	2	chapter 11.19 "5-axis transformation - Type 8: Tripod transformation with wobble plate and additional c-axis for the tangential tool guidance" on page 332
6-axis transformation	LLLRRR (Euler angle orientation)	3333301	6	2	chapter 11.21.2 "Variant 1 (with tensor orientation) " on page 344
	LLLRRR (Linear motion of the rotary axes)	3033001	6	2	chapter 11.21.3 "Variant 2 (with linear motion of the rotary axes)" on page 345
3-axis cylinder jacket transformation	Unwinding	3021001	3	2	chapter 11.22.2 "Variant 1 (unwinding)" on page 359
	Project planning	3021002	3	2	chapter 11.22.3 "Variant 2 (plane projection) " on page 361
4-axis cylinder jacket transformation	Unwinding	3131021	4	2	chapter 11.23 "4-Axis Cylinder Jacket Transformation " on page 367
	Project planning	3131022	4	2	chapter 11.23 "4-Axis Cylinder Jacket Transformation " on page 367

Description	Kinematic type	Type	Axis number	Point of action	In chapter...
5-axis cylinder jacket transformation	Unwinding Linear orientation	3232121	5	2	chapter 11.24 "5-axis cylinder jacket transformation " on page 375
	Unwinding Vector orientation	3232221	5	2	chapter 11.24 "5-axis cylinder jacket transformation " on page 375
	Project planning Linear orientation	3232122	5	2	chapter 11.24 "5-axis cylinder jacket transformation " on page 375
	Project planning Vector orientation	3232222	5	2	chapter 11.24 "5-axis cylinder jacket transformation " on page 375
Face transformation	X-C-axis kinematics for face	2011001	2	2	chapter 11.25 "Face transformation" on page 383
	X-C-axis kinematics for face and additional z-axis	3021003	3	2	chapter 11.25 "Face transformation" on page 383

Tab. 11-2: Overview on axis transformations

Currently, 20 different parameter blocks of one axis transformation per machine can be saved **simultaneously** in the MP and called with COORD (<i=1..20>).

11.2 Bipod axis transformations

11.2.1 Description

Explaining 3- and 4-axis bipod

Transformation name	Transformation type	Point of action	Coordinate system
4-axis bipod	3131001	AT1	ACS, MCS
3-axis bipod	2121001	AT1	ACS, MCS

Tab. 11-3: Bipod axis transformation

In the **4-axis bipod** (see following figure), two linear axes (YL and YR) which can slide parallel to the MCS y-axis are each connected by one pivot "A0" and "B0" to elements l1 and l2 (see following figure). The elements l1 and l2 are connected by a third pivot "A". The rotary axes of all three pivots are vertical to the xy-plane of the MCS. At the element l2, a spindle c-axis SC with tool (e.g. gripper) is mounted vertically to the xy-plane of the MCS which is vertical to the spindle c-axis. The tool is moved in the xy-plane by shifting the linear axes YL and YR.

The tool is fed in at the 4-axis bipod by sliding the structure shown above along a linear axis Z running vertical to the xy-plane (z-direction of the MCS).

The **3-axis bipod** does not include the linear axis Z (z-direction of the MCS).

Otherwise, the transformation is identical to the 3131001.

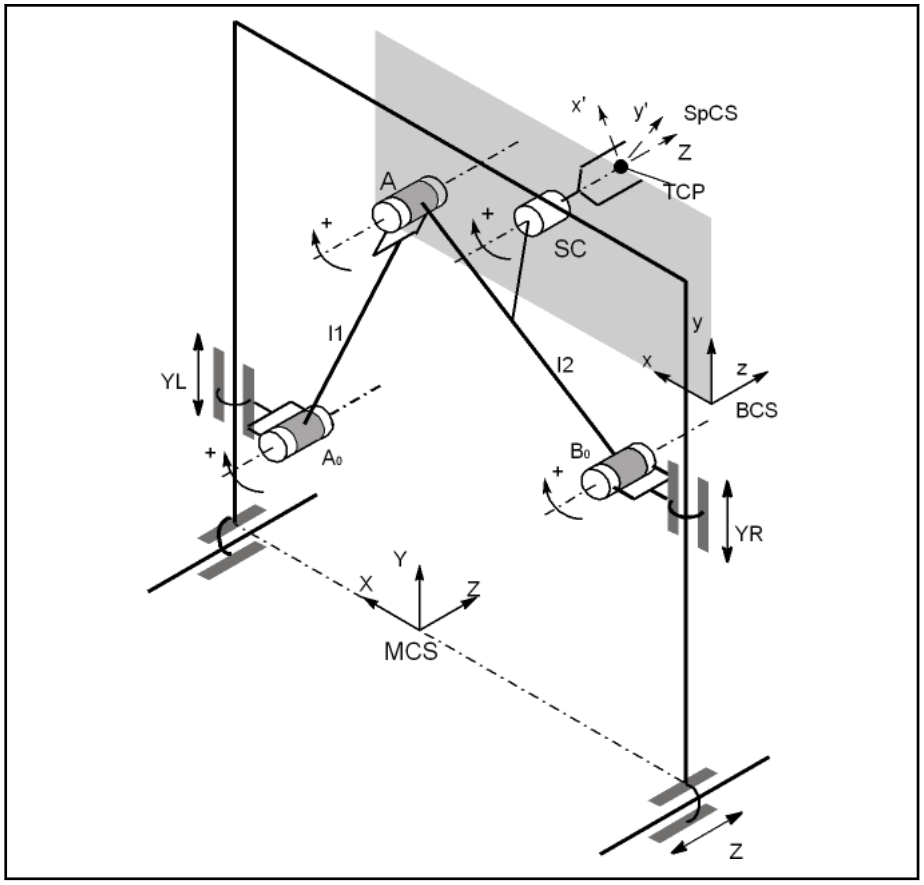


Fig. 11-2: 3- and 4-axis bipod

The bipod transformation has two operating states:

- With spindle C-axis tracking
- Without spindle c-axis tracking

During the motion in the xy-plane, the tool **also** rotates around its longitudinal axis which is parallel to the z-direction.

If the spindle c-axis is tracked, the rotary position of the spindle c-axis is a function of the SC-axis position in the xy-plane. The rotary position of the spindle c-axis is related to the zero point fixed in the machine element I2.

If the spindle is located as c-axis in the interpolation group of the channel, the axial rotation of the tool has to be compensated by an opposite rotation of the spindle c-axis.

If the spindle c-axis is not tracked, the rotary position of the spindle c-axis is defined relative to its fix zero point in the MCS.

The relation between the two zero points is defined in the MP 10300130.

When the bipod transformation is activated, it automatically executes the correct operating state. If the spindle is loaded into or removed from the interpolation group of the channel while the bipod transformation is active, the bipod transformation switches to its operating state accordingly.

Relevant NC functions

COORD(i)	i=1..20 is used to activate a bipod transformation defined in the MP 103000110.
COORD(0,1)	Deactivating the bipod transformation (point of action AT1)

Tab. 11-4: Relevant NC functions

Relevant machine parameters

1030 00110	Defines the transformation type
1030 00120	System axes of the transformation
1030 00130	Information on the axis positions of the reference position (e.g. of the axes YL, YR, (SC), Z)
1030 00140	Defines length and angle parameters (geometric relations of the machine element positions determined while calibrating the reference position)

Tab. 11-5: Relevant machine parameters

The axes YL, YR, (Z), SC should be positioned in such a way that all coordinates (in the BCS) of the transformation reach the value 0 (X = Y = (Z) = C = 0).

Reference position

The axes YL, YR, (Z), SC should be positioned in such a way that all coordinates (in the BCS) of the transformation reach the value 0 (X = Y = (Z) = C = 0).

The reference position is determined together with the calibration of the overall system (see chapter 11.2.2 "Handling Instruction: 3/4-Axis Transformation Bipod" on page 198).

Ambiguous axis positions

Pivot "A" always has to be located above the points "A_0" and "B_0". This avoids any singularities and ambiguities.

Tool correction

No tool correction is calculated for axis transformations at the point of action 1 (AT1). This has to be implemented in a subsequent AT2.

11.2.2 Handling Instruction: 3/4-Axis Transformation Bipod

Applying

First, bring the Bipod into a reference position in which a calibration is performed. Enter the resulting settings for length and angle parameter of the Bipod machine elements into the relevant machine parameters.

Determine data of reference position:

The figure below shows an example of the geometric dependence of the bipod kinematics. It is used as a means of orientation for calibration and for determination of the reference position of the Bipod.

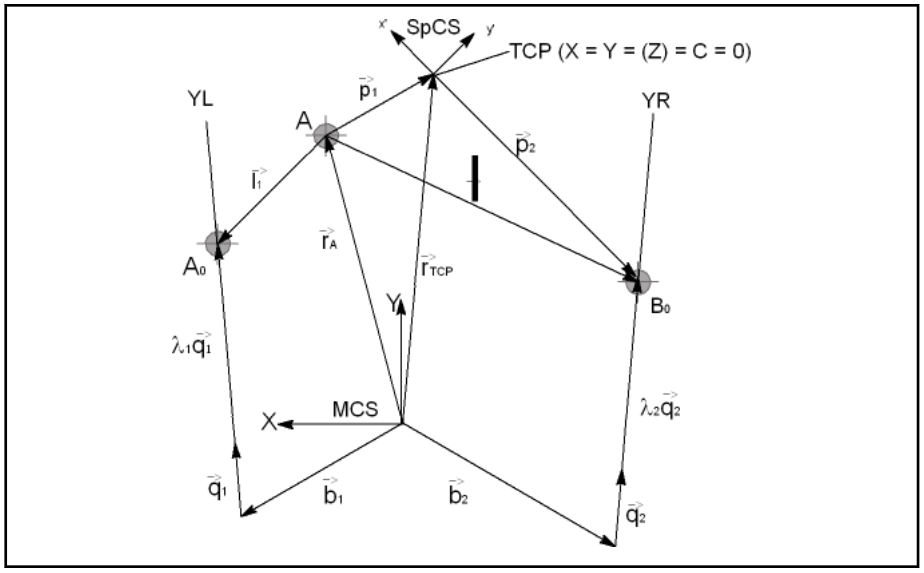


Fig. 11-3: Define data of reference position

1. Use the axis YL and YR to bring the Tool Center Point (TCP) of a suitable tool into the desired reference position. Later, the following should be true at this position when transformation is switched on:

   Coordinates (X = Y = (Z) = C = 0).

2. The axis positions of the axes YL and YR are defined from the length and angle parameters (see in points 3 to 6).

   As a precondition, the axes YL and YR have to take zero point positions.

3. $\vec{q_1}$, $\vec{q_2}$

   Determine the direction vector of the left and right guide (YL and YR) in the MCS (only when the two linear axes are not parallel!)

4. $\vec{b_1}$, $\vec{b_2}$

   Determine the zero points of the motion lines of the joint centers A_0 and B_0 in the MCS = distance MCS zero point to the zero point positions of the YL and YR axes.

   The coordinates of the joint centers A_0 and B_0 in the MCS result from the axis positions λ_1 and λ_2 to $\lambda_1\vec{q_1}$ or $\lambda_2\vec{q_2}$.

   The axis positions $\lambda_1(0)$ and $\lambda_2(0)$ for which the TCP reaches the coordinates zero point (r_{TCP} = 0), are determined internally from the calibration parameters.

5. $\vec{p_1}$, $\overrightarrow{p_{2x'}}$

   Determination of the coordinates of the joint centers A and B_0 relative to the spindle box coordinates system SpCS (x'y'Z coordinates system).

   For $\vec{p_2}$ The y'-component always applies in the SpCS $\overrightarrow{p_{2y'}}$ = 0, thus, only the x'-component $\overrightarrow{p_{2x'}}$ is important.

6. 

   Determine the distance of the joint centers A_0 and A of the couple elements.

7. As the linear axes YL and YR are located on those axis positions for which the coordinates are x = y = 0, determine the position of the z-axis now (only for 4-axis bipod) and the position of the SC-axis for which the coordinates become (X = Y = (Z) = C = 0). To this end, the whole system possibly has to be displaced in z-direction, and the SC axis has to be rotated around the c-axis.

Enter the determined values into the MP (see the example below):

The axis transformation can be applied via the relevant machine parameters. Therefore, the axis transformation function has to be activated via the parameter **tra** "Axis transformations" in the setup (SUP). The path of the machine parameter name is always "TRA/AxTrafo[1..20]/...".

Alternatively, the axis transformation can be applied via a user-defined system date of type SysAxTrafo_t.

The content of both variant is identical.

The axis positions of the reference position have to be determined.

The respectively equivalent machine parameter numbers, machine parameter names and SD elements are specified in the following table. To apply a transformation, one of the variants has to be assigned accordingly

1030 00110	**Transformation type** MP name: TrafoType SD elem. : Type	
	Transformation AT1:	3131001 (4-axis bipod) 2121001 (3-axis bipod)
1030 00120	**System axis number of the 4-axis bipod transformation:** MP name: FwdInCoordIndTrafo[1..8] SD elem. : AxisAssignment[1..8]	
	[1]	System axis index of the YL axis
	[2]	System axis index of the YR axis
	[3]	System axis index of the z-axis
	[4]	System axis index of the spindle c-axis SC
	[5..8]	Not relevant
1030 00120	**System axis number of the 3-axis bipod transformation:** MP name: FwdInCoordIndTrafo[1..8] SD elem. : AxisAssignment[1..8]	
	[1]	System axis index of the YL axis
	[2]	System axis index of the YR axis
	[3]	System axis index of the spindle c-axis
	[4..8]	Not relevant

<table>
<tr><td rowspan="6">1030 00130</td><td colspan="2">Axis positions of the reference position for 4-axis bipod
MP name: RefPosTrafo[1..8]
SD elem. : AxZeroPos[1..8]</td></tr>
<tr><td>[1]</td><td>Irrelevant, as the value results from MP 1030 00140</td></tr>
<tr><td>[2]</td><td>Irrelevant, as the value results from MP 1030 00140</td></tr>
<tr><td>[3]</td><td>Position for Z0</td></tr>
<tr><td>[4]</td><td>Position for SC0</td></tr>
<tr><td>[5..16]</td><td>Not relevant</td></tr>
<tr><td rowspan="5">1030 00130</td><td colspan="2">Axis positions of the reference position for 3-axis bipod
MP name: RefPosTrafo[1..8]
SD elem. : AxZeroPos[1..8]</td></tr>
<tr><td>[1]</td><td>Irrelevant, as the value results from MP 1030 00140</td></tr>
<tr><td>[2]</td><td>Irrelevant, as the value results from MP 1030 00140</td></tr>
<tr><td>[3]</td><td>Position for SC0</td></tr>
<tr><td>[4..16]</td><td>Not relevant</td></tr>
<tr><td>1030 00130</td><td colspan="2">The necessary axis positions YL and YR under [1], [2] are determined by the length and angle parameters described in MP 1030 00140 below, so that [1] and [2] are not evaluated!

4-axis bipod:

Under [3], enter the position of the z-axis and under [4], that of the SC axis the coordinates of which become (X = Y = (Z) = C = 0), provided the YL, YR axes have already assumed the necessary positions, i.e. the x and the y-coordinate are already on zero.

3-axis bipod:

Under [3], enter the position of the SC axis, for which the coordinates become (X = Y = C = 0), provided the YL, YR axes have already assumed the necessary positions, i.e. the x and the y-coordinate are already on zero.</td></tr>
</table>

1030 00140	**Length and angle parameters** MP name: JointParTrafo[1..16] SD elem. : LenParam[1..16] A part of the vectors $\vec{q}_i$ ... shown above are entered here in respect of the coordinates of MCS and SpCS: Values of the parameters q_1, q_2, b_1, b_2, $p_{1x'}$, $p_{1y'}$, $p_{2x'}$ derived from machine calibration of the reference position:	
	[1]	q_{1x}
	[2]	q_{1y}
	[3]	q_{2x}
	[4]	q_{2y}
	[5]	b_{1x}
	[6]	b_{1y}
	[7]	b_{2x}
	[8]	b_{2y}
	[9]	$p_{1x'}$
	[10]	$p_{1y'}$
	[11]	$p_{2x'}$
	[12]	l_1
	[13..16]	Not relevant
7080 00020	**Optional names of the system coordinates with active 4-axis bipod:**	
	[1]	e.g.: X
	[2]	e.g.: Y
	[3]	e.g.: Z
	[4]	e.g.: C
	[5 .. 6]	Not relevant
	Optional names of the system coordinates with active 3-axis bipod:	
	[1]	e.g.: X
	[2]	e.g.: Y
	[3]	Not relevant
	[4]	e.g.: C
	[5 .. 6]	Not relevant

Tab. 11-6: Define data of reference position

Activating

Prerequisites:

The prerequisite for the activation of the Bipod transformation is a calibration to determine the reference position.

Programming theSpindleC-axis tracking:

The Bipod transformation is active already at startup, and the SC axis is executed with spindle c-axis tracking, i.e. it is within the interpolation group of the channel.

Programming example:

```
M3 S5000                ;Spindle in speed mode
                        ;(the SC axis is not tracked)
M5
. . .
SpindleToAxis(SC)       ;Switching of the SC axis into the c-axis mode
WAIT
WaitAxis(SC)            ;Waits for the SC axis to be applied by the channel
. . .
G01 C45 F100            ;Machining is performed
                        ;(SC axis is tracked)
. . .
RemAxis(SC)             ;SC axis enabled from channel
AxisToSpindle(SC)       ;Switching of the SC axis into the spindle mode
. . .                   ;Machining is performed
                        ;(the SC axis is not tracked)
```

☞ Unless the spindle c-axis is in positioning mode (c-axis mode), it cannot transferred to the interpolation group of a channel.

Positioning the spindle (M19):

If the **spindle** is to be positioned with M19, **it** needs spindle tracking even if it is in speed mode.

Example:

```
M19 SC54 G0 X7 Y2 Z9    Approaching to target position X= 7, Y= 2, Z=9, while simultaneously braking the spindle down to ensure deceleration to standstill at angle position C=54 (= 54°).
```

At block start, both processes are started simultaneously; usually, however, they do not end the block together.

In the 4-axis bipod two linear axes which can slide parallel to the y-axis are connected by one pivot with the elements. The elements are connected by a third pivot. The rotary axes of all 3 pivots are vertical to the xy-plane of the MCS. At one element, a spindle c-axis with tool (e.g. gripper) is mounted vertically to the x-y-plane of the MCS which stands vertical to the spindle c-axis. The tool is moved in the XY plane by shifting the linear axes. The tool is fed in at the 4-axis bipod by shifting along a linear axis Z running vertical to the XY-plane (Z-direction of the MCS).

At the 3-axis Bipod, the vertical linear axis is not applicable (z-direction of the MCS). Otherwise, the transformation is identical with the 4-axis Bipod transformation.

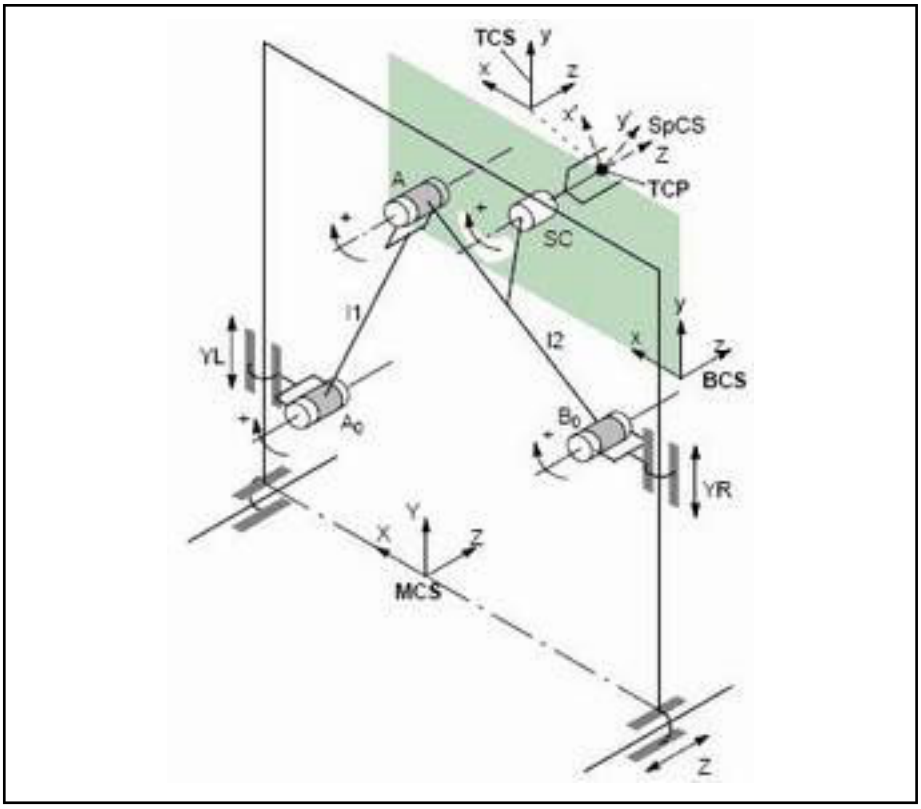


Fig. 11-4: Example of the Bipod configuration

NC: Bring the Bipod into a reference position and perform calibration

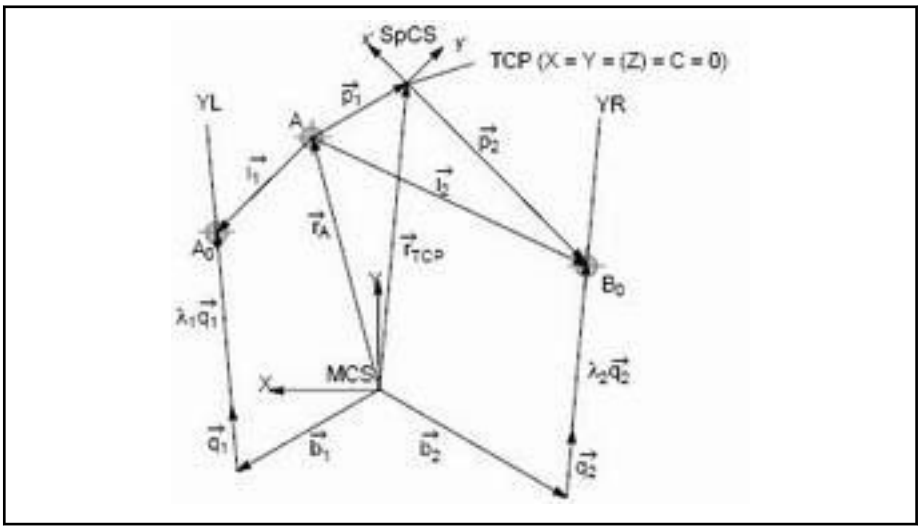


Fig. 11-5: shows an example of the geometric dependence of the Bipod kinematics

- Use the axis YL and YR to bring the Tool Center Point (TCP) of a suitable tool into the desired reference position. Later, the following should be true at this position when transformation is switched on: **Coordinates (X = Y = (Z) = C = 0)**.
- The axis positions of the axes YL and YR are defined from the length and angle parameters. As a precondition, the axes YL and YR have to take zero point positions.
- Determine the direction vector of the left and right guide (YL and YR) in the MCS (only when the two linear axes are not parallel!)

- Determine the zero points of the motion lines of the joint centers A0 and B0 in the MCS = distance MCS zero point to the zero point positions of the YL and YR axes. The coordinates of the joint centers A0 and B0 in the MCS result from the axis positions. The axis positions for which the TCP reaches the coordinates zero point (rTCP = 0), are determined internally from the calibration parameters.
- Determination of the coordinates of the joint centers A and B0 relative to the spindle box coordinates system SpCS (x'y'Z coordinates system).

  For p2 in the SpCS always applies that the y'-component p2Y = 0, therefore the x'-component p2X is the only significant one.
- Determine the distance of the joint centers A0 and A of the couple elements.
- As the linear axes YL and YR are located on those axis positions for which the coordinates are x = y = 0, determine the position of the z-axis now (only for 4-axis bipod) and the position of the SC-axis for which the coordinates become (X = Y = (Z) = C = 0). To this end, the whole system possibly has to be displaced in z-direction, and the SC axis has to be rotated around the c-axis.

☞
- Use **COORD(i)** or **CRD(i)** to activate the bipod-axis transformation. Once a branch is set, this setting is retained until the axis transformation is deactivated
- With the commands **SpindleToAxis (STA)** and **AxisToSpindle (ATS)** the spindle/c-axis in a spindle can be switched back and vice versa.
- The spindle can be positioned via the auxiliary function M19.
- Use **COORD(0,1)** or **CRD(0,1)** to deactivate the bipod-axis transformation.

Deactivating **COORD(0,1)**

switches off the axis transformation in the NC program.

11.3 Scara-axis transformation

11.3.1 Description

Function

Transformation name	Transformation type	Point of action	Coordinate system
3-axis SCARA (without Z-axis)	2103011	AT1	ACS, MCS
4-axis SCARA (with Z-axis)	3113011		

Tab. 11-7: SCARA-axis transformation

The three interconnected rotary axes A, B, and C move a tool in the xy-plane of the machine coordinate system (MCS) to a defined position with a predetermined orientation PHI.

PHI is the angle between l_3-arm and MCS-X.

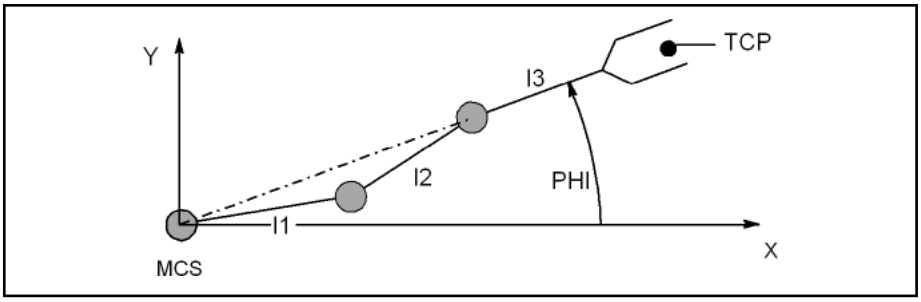


Fig. 11-6: SCARA transformation

The Scara transformation can optionally be used with and without Z-axis (linear axis). In case a Z-axis is available, it is used for the z-coordinate.

Relevant NC functions

COORD(<i>)	With i = 1..20. It activates the SCARA-axis transformation defined in MP 1030 00110.
COORD(0,1)	The active SCARA transformation is deactivated.

Tab. 11-8: Relevant NC functions

Relevant machine parameters

1030 00110	Defines the transformation type
1030 00120	Defines the system axes/coordinates participating in the transformation.
1030 00130	Defines the axis positions of the reference position
1030 00140	Defines the arm lengths of the rotary axes l_1, l_2 and l_3 and the vectorial distance from the first rotary axis to the MCS zero point ($\vec{l}_0$).

Tab. 11-9: Relevant machine parameters

Reference position

The three rotary axes are allocated in parallel to the z-coordinate of the MCS.

The length vector $\vec{l}_0$ and the lengths l_1, l_2 and l_3 define the exact reference positions a_0, a_1, a_2 of the three rotary axes in the xy-plane of the MCS. The vector $\vec{l}_0$ defines the position a_0 of the first rotary axis with respect to the origin in the MCS.

For the lengths, the following applies:

l_1, l_2 and $l_3 > 0$.

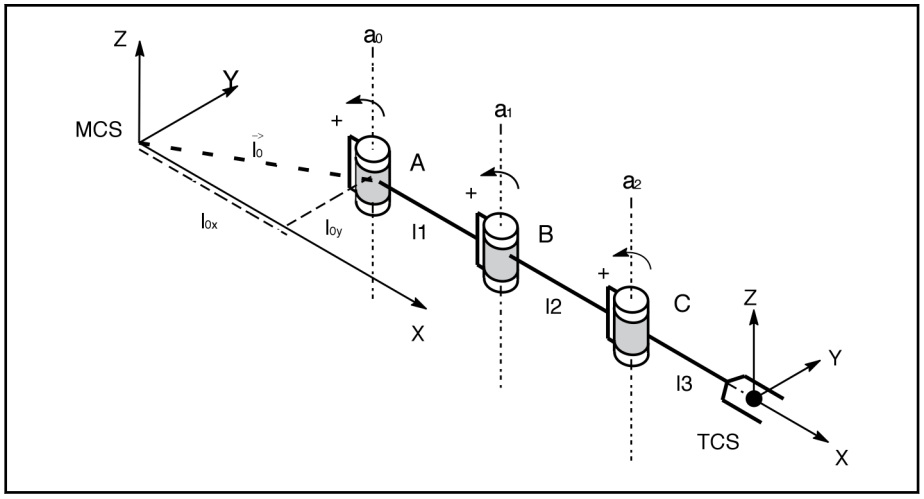


Fig. 11-7: Reference position SCARA

Ambiguous axis positions

A tool position (TCP position) can be reached with two different positions of the rotary axes and their l_1, l_2 arms:

Left bend: Transformation branch -1
Right bend: Transformation branch 1

With the SCARA transformation active, a transition from branch -1 to branch 1 and vice versa is impossible. When activating the transformation, the branch is determined and is retained until deactivation.

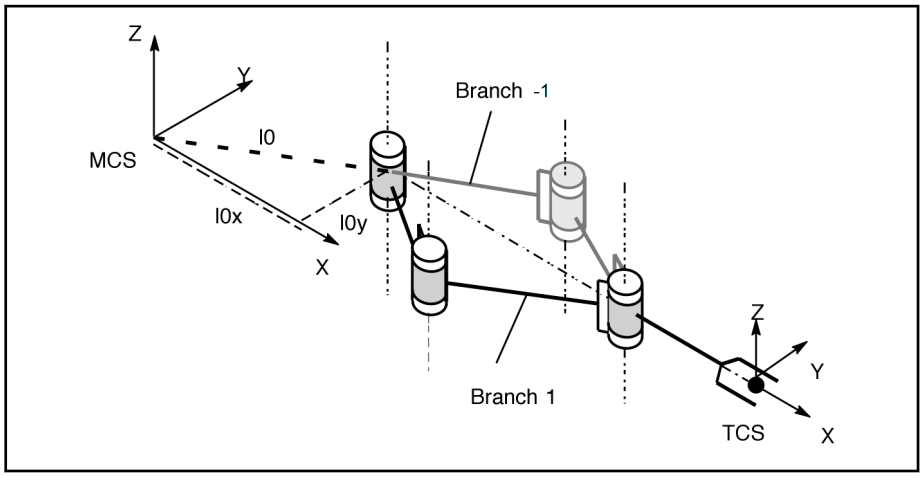


Fig. 11-8: Transition from branch -1 to branch 1

Example:

A right bend (transformation branch 1) causes the axes to collide with the obstacle. In this case, transformation branch -1 has to be activated to approach position 2.

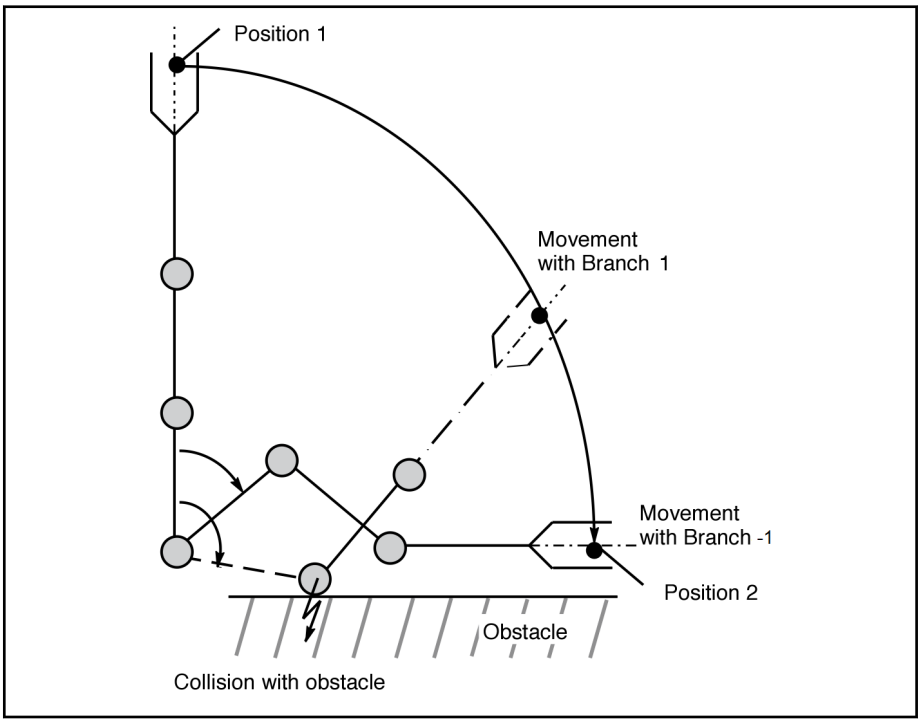


Fig. 11-9: Example - Right bend

Tool correction

No tool correction is calculated for axis transformations at the point of action 1 (AT1). This has to be implemented in a subsequent AT2.

11.4 Handling Instruction: 3-Axis/4-Axis Transformation SCARA

Applying

The axis transformation can be applied via the relevant machine parameters. Therefore, the axis transformation function has to be activated via the parameter **tra** "Axis transformations" in the setup (SUP). The path of the machine parameter name is always "TRA/AxTrafo[1..20]/...".

Alternatively, the axis transformation can be applied via a user-defined system date of type SysAxTrafo_t.

The content of both variant is identical.

The axis positions of the reference position have to be determined.

The respectively equivalent machine parameter numbers, machine parameter names and SD elements are specified in the following table. To apply a transformation, one of the variants has to be assigned accordingly

Transformation-relevant parameters

MP number	Description		
1030 00110	**Transformation type** MP name: TrafoType SD elem. : Type		
	2103011		3-axis SCARA
	3113011		4-axis Scara
1030 00120	**System axes / coordinates of the transformation** MP name: FwdInCoordIndTrafo[1..8] SD elem. : AxisAssignment[1..8]		
	[1]	System axis number of A	
	[2]	System axis number of B	
	[3]	System axis number of C	
	[4]	System axis number of z (if available)	
	[5..8]	Not relevant	
1030 00130	**Axis positions of the reference position** MP name: RefPosTrafo[1..8] SD elem. : AxZeroPos[1..8]		
	[1]	A^0	
	[2]	B^0	
	[3]	C^0	
	[4]	Z^0 (if available)	
	[5..8]	Not relevant	
1030 00140	**Length and angle parameters** MP name: JointParTrafo[1..16] SD elem. : LenParam[1..16]		
	Defines the length vectors l_1, l_2 in [mm].		
	[1]	l_0	
	[2]	l_{0y}	
	[3]	l_1	
	[4]	l_2	
	[5]	l_3	
	[6..16]	Not relevant	

Tab. 11-10: Transformation-relevant parameters

Activating *If the transformation is configured via*

- Machine parameters. Then, the transformation is enabled in the NC program with (i=1,..20):

  COORD(<i>)

- System date. Then, the transformation is enabled in the NC program with (type "SysAxTrafo_t"):

  COORD(<SD.MtbAxTrafo>)

Example Program:

The transformation is defined in the fourth machine parameter block. No axis transformation is active at programming start:

Enable transformation

```
N100  G1 F1000 A10 B10 C30     ;The axes A, B and C are traversed
N110  COORD(4)                 ;Switching on the axis transformation
N130 X20 Y30                   ;Traversing the coordinates X and Y
N140 X10 Y10 PHI45             ;Traversing the coordinates X, Y and PHI
       :
N999  COORD(0,1)                 ;Switching-off the axis transformation
                               ;Axes are again programmed from here
```

Deactivating **COORD(0,1)**

switches off the axis transformation in the NC program.

11.5 Non-orthogonal 2- and 3-axis transformation

11.5.1 Variant 1

Function

Transformation name	Transformation type	Point of action	Coordinate system
3-axes - Non-orthogonal	3030001	AT1	ACS, MCS

Tab. 11-11: Function - Variant 1

The non-orthogonal 3-axis transformation is characterized by the fact that its linear axes Xa, Ya and Za are not orthogonal, i.e., are not or cannot be oriented rectangularly.

Each axis is defined by its polar coordinates φ (phi) and ϑ (theta) in the MCS:

- Xa-axis: of φ_x and ϑ_x
- Ya-axis: of φ_y and ϑ_y
- Za-axis: of φ_z and ϑ_z

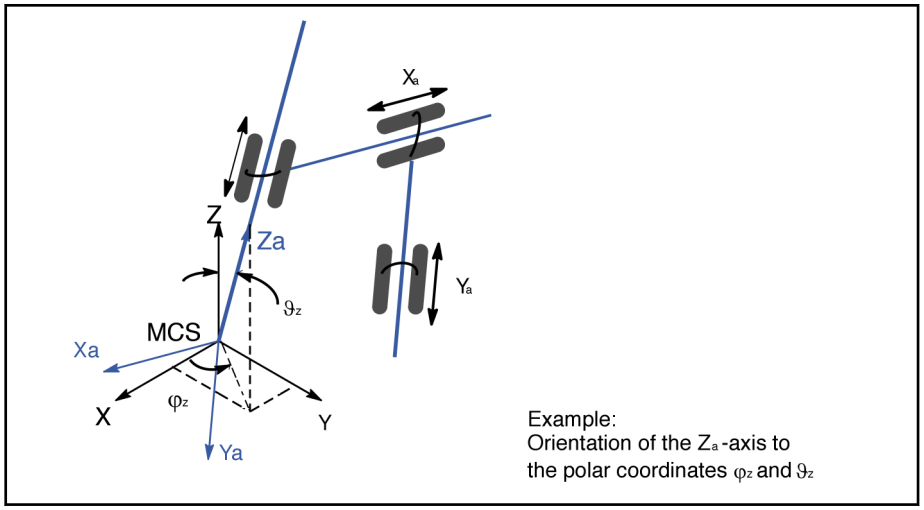


Fig. 11-10: Example

Relevant NC functions

COORD(1) TRFOPT(<φ_x>,<ϑ_x>,<φ_y>,<ϑ_y>,<φ_z>,<ϑ_z>)	
	With i = 1..20, a non-orthogonal 2-axis transformation is activated defined in MP 1030 00110. The axis orientations defined in the machine parameter can be changed on activation using TRFOPT (..) (a non-programmed angle retains its machine parameter value).
COORD(0,1)	The active transformation is deactivated.

Tab. 11-12: Relevant NC functions

Relevant machine parameters

1030 00110	Defines the transformation type.
1030 00120	Defines the system axes/coordinates participating in the transformation.
1030 00130	Defines the axis positions of the reference position.
1030 00140	Defines the axis directions of the non-orthogonal axes in the MCS.

Tab. 11-13: Relevant machine parameters

Reference position

In the reference position of the axis transformation, the TCP is in the zero point of the MCS. In this case, the axis positions are Xa_0, Ya_0 and Za_0.

Tool correction

No tool correction is calculated for axis transformations at the point of action 1 (AT1). This has to be implemented in a subsequent AT2.

11.5.2 Variant 2

Function

Transformation name	Transformation type	Point of action	Coordinate system
2-axes - Non-orthogonal	2020001	AT1	ACS, MCS

Tab. 11-14: Function - Variant 2

The kinematics of the non-orthogonal 2-axis transformation consists of the two linear axes Xa and Ya which are not and cannot be oriented orthogonally, i.e., at right angles to each other.

In view of their polar coordinates φ (phi), the orientation of the linear axes refers to the MCS-X- coordinate:

- Xa-axis: φ_x
- Ya-axis: φ_y

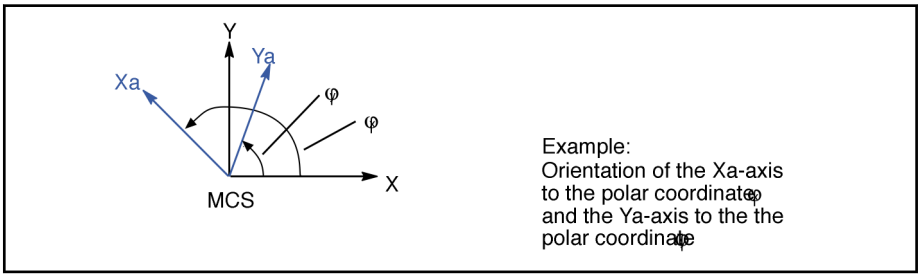


Fig. 11-11: Example

Relevant NC functions

COORD(<i>) TRFOPT(<φ_x>,<φ_y>):	
	With i = 1..20, a non-orthogonal 2-axis transformation is activated defined in MP 1030 00110. The axis orientations defined in the machine parameter can be changed on activation using TRFOPT (..) (a non-programmed angle retains its machine parameter value).
COORD(0,1)	The active transformation is deactivated.

Tab. 11-15: Relevant NC functions

Relevant machine parameters

1030 00110	Defines the transformation type.
1030 00120	Defines the system axes/coordinates participating in the transformation.
1030 00130	Defines the axis positions of the reference position
1030 00140	Defines the axis directions of the non-orthogonal axes in the MCS.

Tab. 11-16: Relevant machine parameters

Reference position

In the reference position of the axis transformation, the TCP is in the zero point of the MCS. In this case, the axis positions are Xa_0 and Ya_0.

Tool correction

No tool correction is calculated for axis transformations at the point of action 1 (AT1). This has to be implemented in a subsequent AT2.

11.5.3 Variant 3

Function

Transformation name	Transformation type	Point of action	Coordinate system
2-axes - Non-orthogonal	2020002	AT1	ACS, MCS

Tab. 11-17: Function - Variant 3

The kinematics of the non-orthogonal 2-axis transformation consists of the two linear axes Xa and Za which are not and cannot be oriented orthogonally, i.e., at right angles to each other.

In view of their polar coordinates ϑ (theta), the orientation of the linear axes refers to the MCS-Z-coordinate:

- Xa-axis: ϑ_x
- Za-axis: ϑ_z

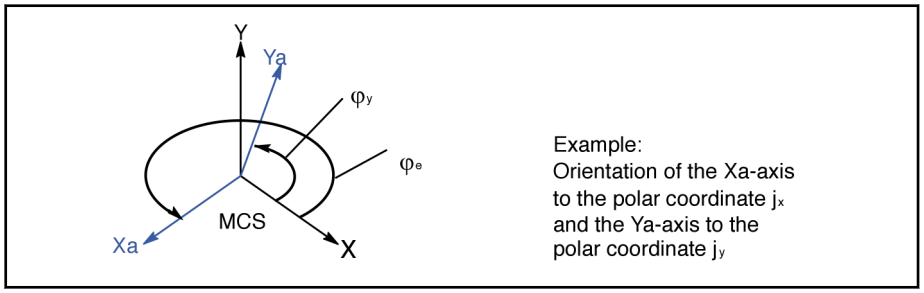


Fig. 11-12: Example

Relevant NC functions

COORD(<i>) TRFOPT(<ϑ_x>,<ϑ_z>):	
	With i = 1..20, a non-orthogonal 2-axis transformation is activated defined in MP 1030 00110. The axis orientations defined in the machine parameter can be changed on activation using TRFOPT (..) (a non-programmed angle retains its machine parameter value).
COORD(0,1)	The active transformation is deactivated.

Tab. 11-18: Relevant NC functions

Relevant machine parameters

1030 00110	Defines the transformation type.
1030 00120	Defines the system axes/coordinates participating in the transformation.
1030 00130	Defines the axis positions of the reference position.
1030 00140	Defines the axis directions of the non-orthogonal axes in the MCS.

Tab. 11-19: Relevant machine parameters

Reference position

In the reference position of the axis transformation, the TCP is in the zero point of the MCS. In this case, the axis positions are Xa_0 and Ya_0.

Tool correction

No tool correction is calculated for axis transformations at the point of action 1 (AT1). This has to be implemented in a subsequent AT2

11.5.4 Handling Instruction: Non-Orthogonal 2/3-Axis Transformation

Applying: The axis transformation can be applied via the relevant machine parameters. Therefore, the axis transformation function has to be activated via the parameter **tra** "Axis transformations" in the setup (SUP). The path of the machine parameter name is always "TRA/AxTrafo[1..20]/...".

Alternatively, the axis transformation can be applied via a user-defined system date of type SysAxTrafo_t.

The content of both variant is identical.

The axis positions of the reference position have to be determined.

The respectively equivalent machine parameter numbers, machine parameter names and SD elements are specified in the following table. To apply a transformation, one of the variants has to be assigned accordingly

There are three variants of non-orthogonal axis transformations whose parameters are described in the following.

Variant 1 (Non-Orthogonal 3-axis Transformation)

The following examples show the application of a non-orthogonal 3-axis transformation (3030001).

1030 00110	**Transformation type** MP name: TrafoType SD elem. : Type	
	3030001	
1030 00120	**System axes / coordinates of the transformation** MP name: FwdInCoordIndTrafo[1..8] SD elem. : AxisAssignment[1..8]	
	[1]	System axis number of the Xa-axis
	[2]	System axis number of the Ya-axis
	[3]	System axis number of the Za axis
	[4..8]	Not relevant
1030 00130	**Axis positions of the reference position** MP name: RefPosTrafo[1..8] SD elem. : AxZeroPos[1..8]	
	[1]	Position for Xa^0
	[2]	Position for Ya^0
	[3]	Position for Za^0
	[4..8]	Not relevant

1030 00140	**Length and angle parameters** MP name: JointParTrafo[1..16] SD elem. : LenParam[1..16] Defines the axis directions in the MCS, unit in degrees.	
	[1]	φ_x
	[2]	ϑ_x
	[3]	φ_y
	[4]	ϑ_y
	[5]	φ_z
	[6]	ϑ_z
	[7..16]	Not relevant

Tab. 11-20: Applying - Variant 1

The following examples show the setting of the angle parameters of a non-orthogonal 3-axis transformation:

Example:

Setting of the Angle Parameters of a Non-Orthogonal 3-Axis Transformation - Example 1

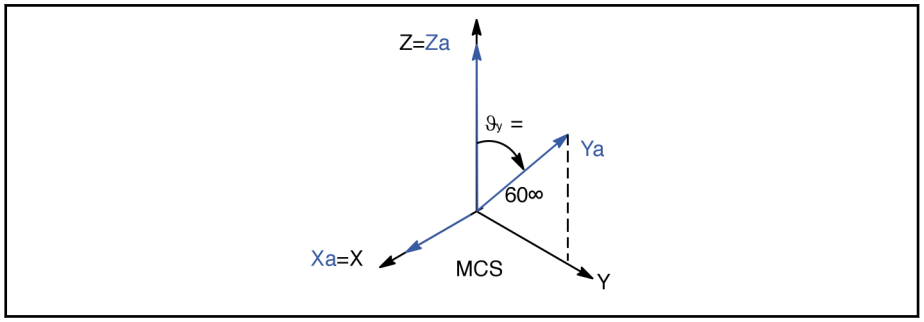


Fig. 11-13: Example 1

The axis directions Xa and Za coincide with the Cartesian MCS coordinates X and Z.

Ya is situated in the YZ plane. The angles of the axis directions in 103000140 are to be set as follows:

1030 00140 Unit in degrees

[1] $\varphi_x = 0$

[2] $\vartheta_x = 90$

[3] $\varphi_y = 90$

[4] $\vartheta_y = 60$

[5] φ_z = any

[6] $\vartheta_z = 0$

Example:

Setting of the Angle Parameters of a Non-Orthogonal 3-Axis Transformation - Example 2

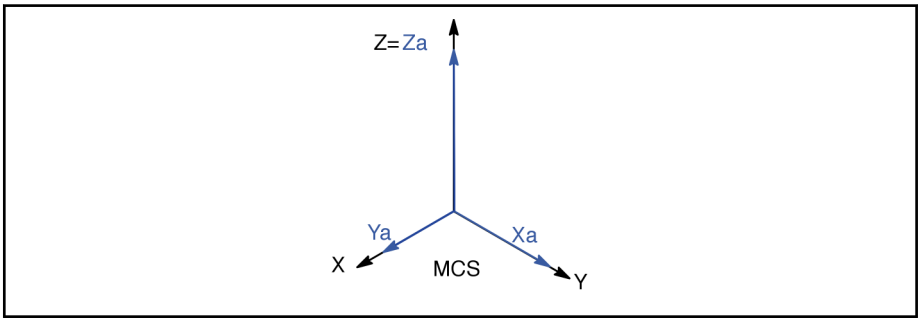


Fig. 11-14: Example 2

Here, axis transformation is used to transform from a left-handed Cartesian axis coordinate system (ACS) to a right-handed Cartesian machine coordinate system (MCS). In this case, the angles are as follows:

1030 00140 Unit in degrees

[1] $\varphi_x = 90$

[2] $\vartheta_x = 90$

[3] $\varphi_y = 0$

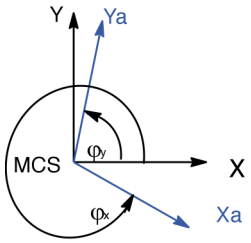
[4] $\vartheta_y = 90$

[5] φ_z = any

[6] $\vartheta_z = 0$

Variant 2 (Non-orthogonal 2-axis Transformation in XY-plane)

The following example shows the application of a non-orthogonal 2-axis transformation (2020001).



Example:
Orientation of the Xa-axis to the polar coordinate φ_x = 330∞ (or φ_x = -30∞) and and the Ya-axis to the polar coordinate φ_y = 80∞

Fig. 11-15: Applying - Variant 2

Determine data of reference position:

As a prerequisite for the reference position, the coordination orientation of all involved axes and the axis positions of the reference position has to be known.

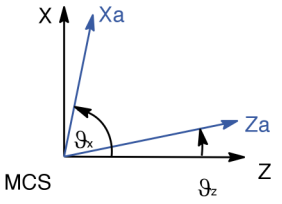
To this end, the following MP has to be set (example):

1030 00110	**Transformation type** MP name: TrafoType SD elem. : Type	
	2020001	
1030 00120	**System axes / coordinates of the transformation** MP name: FwdInCoordIndTrafo[1..8] SD elem. : AxisAssignment[1..8]	
	[1]	System axis number of the Xa-axis
	[2]	System axis number of the Ya-axis
	[3..8]	Not relevant
1030 00130	**Axis positions of the reference position** MP name: RefPosTrafo[1..8] SD elem. : AxZeroPos[1..8]	
	[1]	Position for Xa^0
	[2]	Position for Ya^0
	[3..8]	Not relevant
1030 00140	**Length and angle parameters** MP name: JointParTrafo[1..16] SD elem. : LenParam[1..16] Defines the axis directions in the MCS, unit in degrees.	
	[1]	φ_x
	[2]	φ_y
	[3..16]	Not relevant

Tab. 11-21: Applying - Variant 2

Variant 3 (Non-orthogonal 2-axis Transformation in XZ-plane)

The following example shows the application of a non-orthogonal n2-axis transformation (2020002).



Example:
Orientation of the Xa-axis to the polar coordinate
$\vartheta_x = 80^\infty$ and the Za-axis to the polar coordinate
$\vartheta_z = 10^\infty$

Fig. 11-16: Applying - Variant 3

Determine data of reference position:

As a prerequisite for the reference position, the coordination orientation of all involved axes and the axis positions of the reference position has to be known.

To this end, the following MP has to be set (example):

1030 00110	**Transformation type** MP name: TrafoType SD elem. : Type	
	2020002	
1030 00120	**System axes / coordinates of the transformation** MP name: FwdInCoordIndTrafo[1..8] SD elem. : AxisAssignment[1..8]	
	[1]	System axis number of the Xa-axis
	[2]	System axis number of the Za axis
	[3..8]	Not relevant
1030 00130	**Axis positions of the reference position** MP name: RefPosTrafo[1..8] SD elem. : AxZeroPos[1..8]	
	[1]	Position for Xa^0
	[2]	Position for Za^0
	[3..8]	Not relevant
1030 00140	**Length and angle parameters** MP name: JointParTrafo[1..16] SD elem. : LenParam[1..16] Defines the axis directions in the MCS, unit in degrees.	
	[1]	ϑ_x
	[2]	ϑ_z
	[3..16]	Not relevant

Tab. 11-22: Applying - Variant 3

Activating

If the transformation is configured via

- Machine parameters. Then, the transformation is enabled in the NC program with (i=1,..20):

COORD(<i>)

- System date. Then, the transformation is enabled in the NC program with (type "SysAxTrafo_t"):

  COORD(<SD.MtbAxTrafo>)

Example Program:

The transformation is defined in the fourth machine parameter block. No axis transformation is active at programming start:

Enable transformation

```
N100  G1 F1000 XA10 YA10 ZA5   ;axes XA, YA and ZA are traversed
N110  COORD(4) TRFOPT(...)     ;Switching-on the axis transformation
N130 X20 Y30 Z10               ;Cart. coordinates X,Y and Z are traversed
N140 X10 Y50 Z15               ;Cart. coordinates X,Y and Z are traversed
       :
N999  COORD(0,1)               ;Switching-off the axis transformation
                               ;Axes are again programmed from here
```

Deactivating **COORD(0,1)**

switches off the axis transformation in the NC program.

11.6 4-axis transformation - Type 1: SCARA on linear axis

11.6.1 Description

Function

This transformation is a SCARA arm on a linear axis. It is implemented as 3-axis and as 4-axis variant.

- 3-axis variant in (y,z,φ)
- 4-axis variant in (x,y,z,φ)

The main transformation characteristics of both variants are identical. In the 4-axis variant, the x-coordinate (including x-tool length correction) is additionally taken into consideration. The following sketches and descriptions refer to the 4-axis variant.

Transformation name	Transformation type	Point of action	Coordinate system
SCARA on linear axis LLRR	3122001	AT2	MCS, BCS
SCARA on linear axis LRR	2112001	AT2	MCS, BCS

Tab. 11-23: Function - 3-axis variant SCARA on linear axis

Output variables in the MCS: (X),Z,R_1,R_2

Input variables in the BCS: (x),y,z,φ

The configuration has two (one) linear axes (X) and Z. Two rotary axes R_1 and R_2 are positioned on it. The rotary axis R_1 swivels the axis R_2, R_2 swivels the tool. The axes R_1 and R_2 both rotate around X (type a-axis). The y-coordinate results from simultaneous traveling of the z-axis and R_1-axis. Axis R_2 is used for orientation.

- Orientation vector "ρ" is calculated (in polar coordinates).
- A linear orientation motion in "φ" is possible.
- Online correction is possible for all coordinates.
- The rotary axes can be defined as endless or rotary axes.

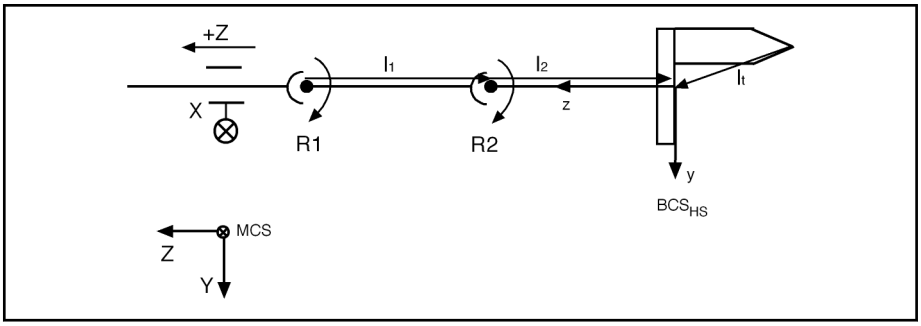


Fig. 11-17: Reference position of the main spindle configuration

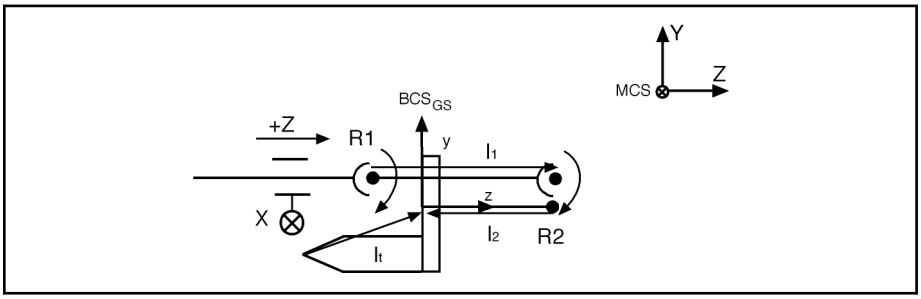


Fig. 11-18: Reference position of the counter spindle configuration

The vectors $\vec{l}_1$ and $\vec{l}_2$ define the geometry of the 4-axis kinematics. They describe the arm lengths of the kinematics and have the form $\vec{l}_1 = (0,0,l_{1x})$ and $\vec{l}_2 = (0,0,l_{2x})$.

$\vec{l}_1$ It is a vector along z from the R_1-axis to the R_2-axis. $\vec{l}_2$ It is the $\vec{l}_1$- Continuation vector of the R_2--axis to the zero point of the BCS (i.e., $l_{1z} \neq 0$, $l_{2z} \neq 0$, $l_{1x} = l_{1y} = l_{2x} = l_{2y} = 0$).

Special features:

Value range:

The orientation angle φ is limited to $\pm 89^0$.

Ambiguities:

During machining, the traversing range of axis R_1 is limited to $\pm 89^0$ for the main spindle and for the counter spindle. Thus, there can be no ambiguities of the axis transformation.

The traversing ranges of the remaining axes are not restricted.

Tool turret:

The tool turret is controlled by an NC axis. A separate axis transformation has to be defined for each tool position. These differ only in the axis positions of the reference position of the axis R_2 (machine parameter 1030 00130 [4]). When changing to a new turret position, axis R_2 has to be traveled to its reference position and then the respective axis transformation is activated.

If R_2 is configured as endless axis, the values from 0..359 have to be entered in the reference position.

Main/counter spindle:

Main and counter spindle can be realized with the same axis transformation. They only differ in their parameterization.

The following applies:

Main spindle:	$l_{1z} < 0$, $l_{2z} < 0$, Z = +3
Counter spindle:	$l_{1z} > 0$, $l_{2z} < 0$, Z = -3

Reference position: The fig. 11-17 "Reference position of the main spindle configuration" on page 220 shows the reference position of the main spindle configuration and fig. 11-18 "Reference position of the counter spindle configuration" on page 220 shows the reference position of the counter spindle configuration. If the machine is in the position shown in the figure, the BCS coordinates have to be (x,y,z,φ) = (0,0,0,0) (in case of l_t=0) for the related MCS coordinates (X,Z,R_1,R_2) = (X^0,Z^0,R_1^0,R_2^0) of MP 1030 00130. The figure also shows the positive directions of rotation/traversing of the axes. Set them accordingly.

In this position, the peak of rotation-symmetric tools (Tool Center Point, TCP) is placed on the origin of basic coordinate system (BCS) and is directed along the z-coordinate. The axis transformation unambiguously determines the orientation of the BCS.

Tool correction: A tool correction is possible. The correction vector $\vec{l}_t = \begin{pmatrix} l_{tx} \\ l_{ty} \\ l_{tz} \end{pmatrix}$ points from the tool point to the tool holder.

Relevant NC functions

COORD ( )	The 4-axis transformation LLRR is activated. This can optionally be done using the machine parameter block (i=1..20) or via the MTB system data (of type SysAxTrafo_t).
COORD (0,2)	The active axis transformation is deactivated.

Tab. 11-24: Relevant NC functions

11.6.2 Handling instruction

Applying The axis transformation can be applied via the relevant machine parameters. Therefore, the axis transformation function has to be activated via the parameter **tra** "Axis transformations" in the setup (SUP). The path of the machine parameter name is always "TRA/AxTrafo[1..20]/...".

Alternatively, the axis transformation can be applied via a user-defined system date of type SysAxTrafo_t.

The content of both variants is identical.

The axis positions of the reference position have to be determined.

The respectively equivalent machine parameter numbers, machine parameter names and SD elements are specified in the two following tables. To apply a transformation, one of the variants has to be assigned accordingly. The first table describes the 3-axis variant LRR and the second table describes the 4-axis variant LLRR.

3-axis variant LRR: Transformation-relevant parameters

MP number	Description	
1030 00110	**Transformation type** MP name: TrafoType SD elem.: Type	
	2112001	Scara on linear axis LRR
1030 00120	**System axes/coordinates of the transformation** MP name: FwdInCoordIndTrafo[1..8] SD elem.: AxisAssignment[1..8]	
	[1]	System axis number of Z
	[2]	System axis number of R_1 (A-axis)
	[3]	System axis number of R_2 (a-axis)
	[4..8]	Not relevant

<table>
<tr><th>MP number</th><th>Description</th></tr>
<tr><td>1030 00125</td><td>Axis classification of transformation axes
MP name: CoordClass[1..8]
MP name: CoordDir[1..8]
SD elem.: AxisClassification[1..8]</td></tr>
<tr><td></td><td>Defines the axes involved in transformation, including their direction of motion.
• MP no. or SD element:
Value ">0" means:
The axis rotates/travels positively as in the figures above.
Value "<0" means:
The axis rotates/travels negatively as in the figures above.
Possible values are:
– +/-1 = x-axis
– +/-2 = y-axis
– +/-3 = z-axis
– +/-100 = a-axis
– +/-200 = b-axis
– +/-300 = c-axis
• MP name:
– Possible values for CoordClass are:
– X,Y,Z (linear)
– A,B,C (rotary)
– Possible values for CoordDir are:
– positive (regarding fig. "Reference position"
– negative (regarding fig. "Reference position"</td></tr>
<tr><td></td><td>If a parameter is not set, the following default settings apply:
<table><tr><td></td><td>MP no /
SD elem.:</td><td>CoordClass:</td><td>CoordDir:</td></tr><tr><td>[1]</td><td>3</td><td>Z</td><td>Positive</td></tr><tr><td>[2]</td><td>100</td><td>A</td><td>Positive</td></tr><tr><td>[3]</td><td>100</td><td>A</td><td>Positive</td></tr><tr><td>[4..8]</td><td>Not relevant</td><td>0</td><td>Positive</td></tr></table></td></tr>
<tr><td>1030 00130</td><td>Axis positions of the reference position
MP name: RefPosTrafo[1..8]
SD elem.: AxZeroPos[1..8]</td></tr>
<tr><td></td><td>[1] Z^0
[2] R_1^0
[3] R_2^0
[4..8] Not relevant</td></tr>
</table>

MP number	Description	
1030 00140	**Length and angle parameters** MP name: JointParTrafo[1..16] SD elem.: LenParam[1..16]	
	Defines the length vectors l_1, l_2 in [mm].	
	[1]	Irrelevant
	[2]	Irrelevant
	[3]	Arm length 1: Main spindle l_{1z}<0, Counter spindle l_{1z}>0
	[4]	Irrelevant
	[5]	Irrelevant
	[6]	Arm length 2: Main spindle and counter spindle l_{2z}<0
	[7..16]	Not relevant
1030 00160	**Epsilon environments**	
	Irrelevant	

Tab. 11-25: Transformation-relevant parameters

4-axis variant LLRR: Transformation-relevant parameters

MP number	Description	
1030 00110	**Transformation type** MP name: TrafoType SD elem.: Type	
	3122001	Scara on linear axis LLRR
1030 00120	**System axes/coordinates of the transformation** MP name: FwdInCoordIndTrafo[1..8] SD elem.: AxisAssignment[1..8]	
	[1]	System axis number of X
	[2]	System axis number of Z
	[3]	System axis number of R_1 (a-axis)
	[4]	System axis number of R_2 (a-axis)
	[5..8]	Not relevant

<table>
<tr><th>MP number</th><th>Description</th></tr>
<tr><td>1030 00125</td><td>Axis classification of transformation axes
MP name: CoordClass[1..8]
MP name: CoordDir[1..8]
SD elem.: AxisClassification[1..8]</td></tr>
<tr><td></td><td>Defines the axes involved in transformation, including their direction of motion.
• MP no. or SD element:
Value ">0" means:
The axis rotates/travels positively as in the figures above.
Value "<0" means:
The axis rotates/travels negatively as in the figures above.
Possible values are:
– +/-1 = x-axis
– +/-2 = y-axis
– +/-3 = z-axis
– +/-100 = a-axis
– +/-200 = b-axis
– +/-300 = c-axis
• MP name:
– Possible values for CoordClass are:
– X,Y,Z (linear)
– A,B,C (rotary)
– Possible values for CoordDir are:
– positive (regarding fig. "Reference position"
– negative (regarding fig. "Reference position"</td></tr>
<tr><td></td><td>If a parameter is not set, the following default settings apply:
<table><tr><th></th><th>MP no /
SD elem.:</th><th>CoordClass:</th><th>CoordDir:</th></tr><tr><td>[1]</td><td>1</td><td>X</td><td>Positive</td></tr><tr><td>[2]</td><td>3</td><td>Z</td><td>Positive</td></tr><tr><td>[3]</td><td>100</td><td>A</td><td>Positive</td></tr><tr><td>[4]</td><td>100</td><td>A</td><td>Positive</td></tr><tr><td>[5..8]</td><td>Not relevant</td><td>0</td><td>Positive</td></tr></table></td></tr>
</table>

MP number	Description	
1030 00130	**Axis positions of the reference position** MP name: RefPosTrafo[1..8] SD elem.: AxZeroPos[1..8]	
	[1]	X^0
	[2]	Z^0
	[3]	$R_1{}^0$
	[4]	$R_2{}^0$
	[5..8]	Not relevant
1030 00140	**Length and angle parameters** MP name: JointParTrafo[1..16] SD elem.: LenParam[1..16]	
	Defines the length vectors l_1, l_2 in [mm].	
	[1]	Irrelevant
	[2]	Irrelevant
	[3]	Arm length 1: Main spindle l_{1z}<0, Counter spindle l_{1z}>0
	[4]	Irrelevant
	[5]	Irrelevant
	[6]	Arm length 2: Main spindle and counter spindle l_{2z}<0
	[7..16]	Not relevant
1030 00160	**Epsilon environments**	
	Irrelevant	

Tab. 11-26: Transformation-relevant parameters

Activating

If the transformation is configured via

- machine parameter, the transformation is enabled in the NC program with (i=1,..20):

  COORD(<i>)

- system date, the transformation is enabled in the NC program with (type "SysAxTrafo_t"):

  COORD(<SD.MtbAxTrafo>)

Example program:

The 3-axis transformation LRR is defined in the fourth machine parameter block. No axis transformation is active at programming start:

Enable transformation

```
N100  G1 F1000 Z10 A1=10 A2=30 ;Axes Z, A1 and A2 are traversed
N110  COORD(4)                 ;Switching on the axis transformation
N130  z20 y30 phi0             ;Coordinates z, y and phi are traversed
        :
N999  COORD(0)                 ;Switching-off the axis transformation
                               ;Axes are again programmed from here
```

Deactivating **COORD(0)** or **COORD(0,2)**

switches off the axis transformation in the NC program.

11.7 2-dimensional delta kinematics with additional rotary axis in the point (RRR)

11.7.1 Description

Function

Transformation name	Transformation type	Point of action
2-dimensional delta kinematics Additional rotary axis in the point (RRR)	2103002	AT2

This transformation is based on a 2-dimensional delta kinematics consisting of the two rotary axes R1 and R2 that are parallel to each other. These two axes are interconnected using the three, also parallel, swivel joints G1, G2 and G3 and the connecting elements e11, e22, e1 and e2:

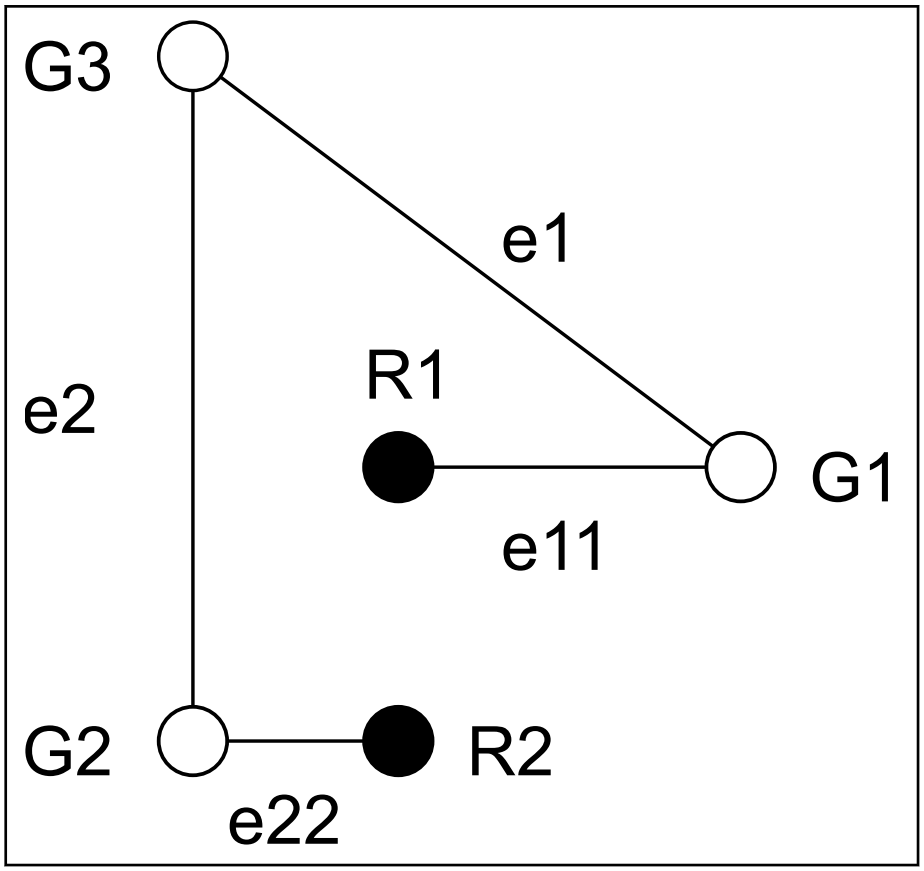


Fig. 11-19: Basic 2-dimensional delta transformation

When R1 and/or R2 are rotated, the center swivel joint G3 can be arbitrarily moved to any position within certain boundaries.

Additionally, there is the other rotary axis R3 concentrically implemented with the swivel joint G3. This allows to rotate the tool. For an example, refer to fig. 11-20 "2-dimensional delta transformation with additional rotary axis in the point" on page 228.

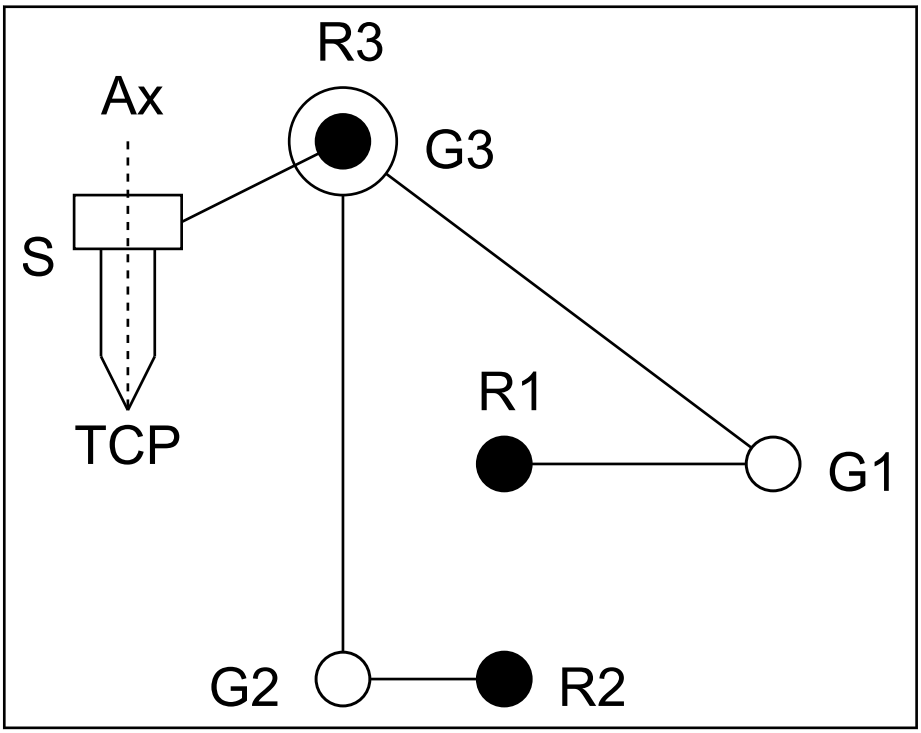


Fig. 11-20: 2-dimensional delta transformation with additional rotary axis in the point

Die axis R3 is positioned mechanically and ensures with its positioning that its housing moves when moving the axes R1 and/or R2, but that it does not rotate.

The spindle nose is labeled with "S" and the tool point (Tool Center Point) with "TCP". The tool reference point is the point of intersection between the tool axis "Ax" and the spindle nose.

fig. 11-21 "Consequence of a rotation of the axes R1 and R2, but without the axis R3 being rotated" on page 229 shows the configuration given in fig. 11-20 "2-dimensional delta transformation with additional rotary axis in the point" on page 228 after a rotation of the axes R1 and R2, but not R3.

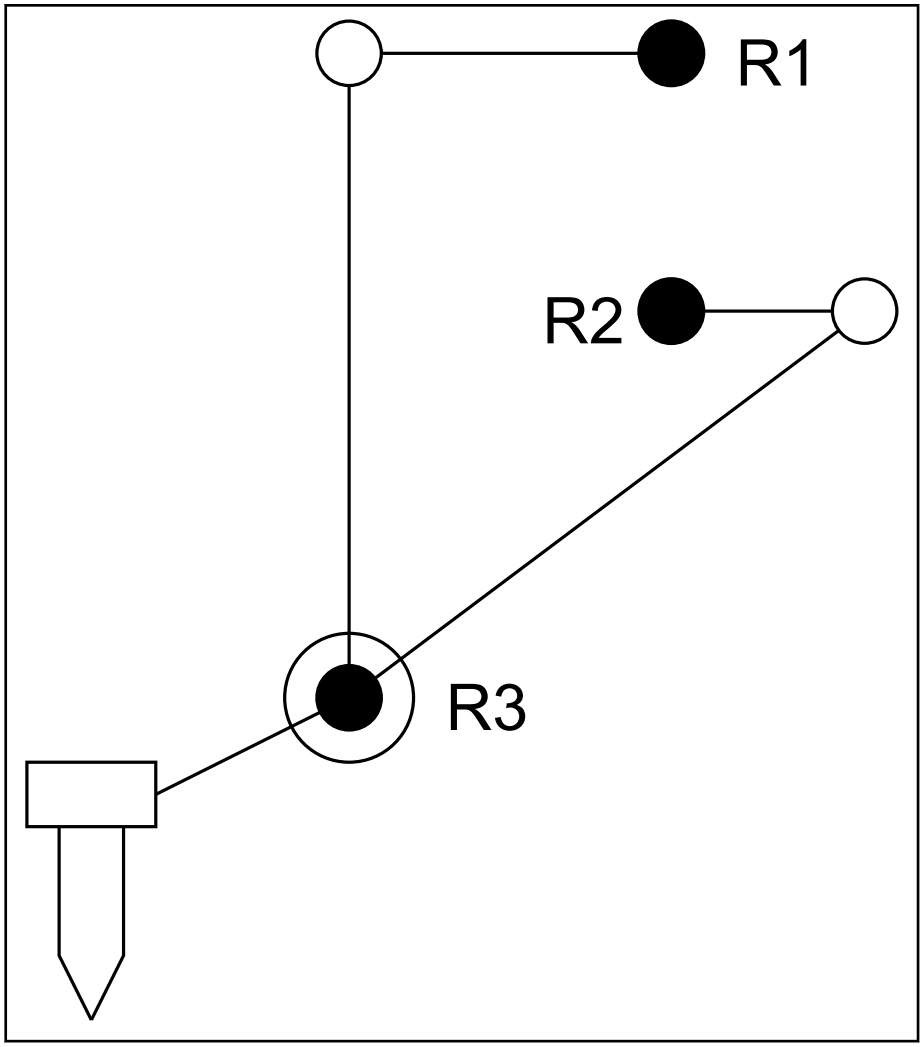


Fig. 11-21: Consequence of a rotation of the axes R1 and R2, but without the axis R3 being rotated

fig. 11-22 "Consequence of an additional rotation of axis R3" on page 230 shows this configuration after an additional rotation of axis R3.

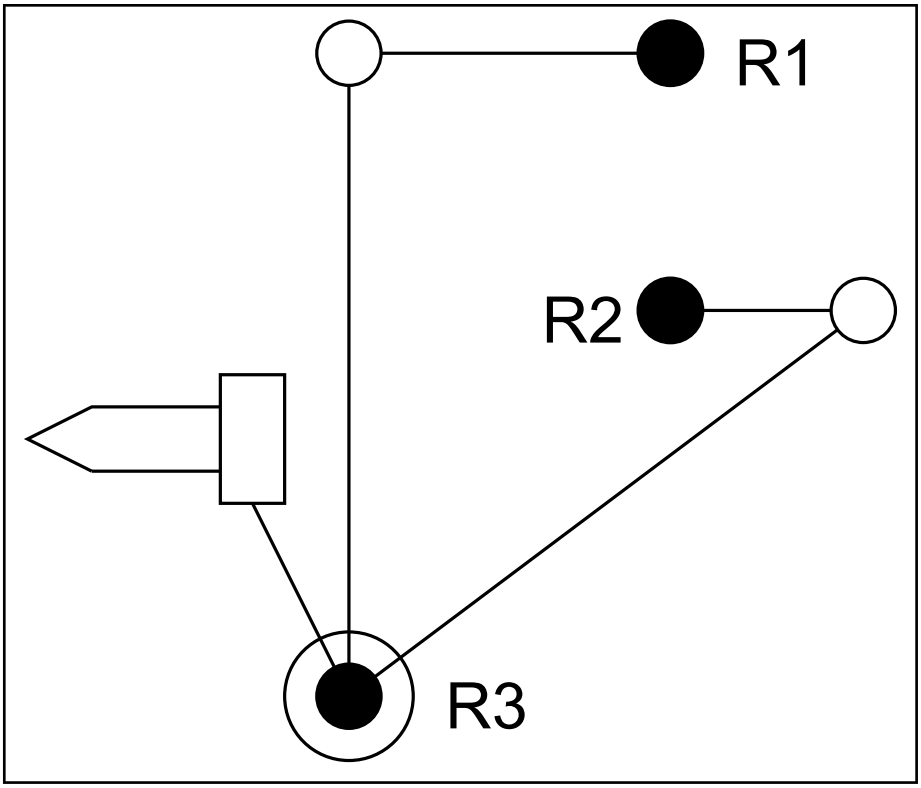


Fig. 11-22: Consequence of an additional rotation of axis R3

The transformation type is specified in machine parameter 103000110 "Transformation type":

MP 103000110 = 2103002

In machine parameter 103000120, the "system axes/system coordinates of the transformation" are specified:

System axis number of the R1-axis MP 103000120 (1)

System axis number of the R2-axis MP 103000120 (2)

System axis number of R3-axis MP 103000120 (3)

The remaining inputs MP 103000120 (4..8) are not relevant.

Variants

As axis transformation at the point of action 2 (AT2), it acts between the basic coordinate system (BCS) as input coordinate system and the machine coordinate system (MCS) = (R1, R2, R3) as output coordinate system.

The input coordinate system (BCS) can be selected as follows by the machine parameter 103000125 "Axis classifications of the transformation axes":

MP 103000125 (1..3) = ±100:

The BCS input coordinates of the transformation are y, z and phi.

For the axis transformation, the axes R1, R2 and R3 have the significance A.

The positive direction of rotation of an axis parameterized with +100 is the direction of rotation to rotate the positive y-direction by 90° to reach the positive z-direction.

The positive direction of rotation of an axis parameterized with -100 is reverse.

MP 103000125 (1..3) = ±200:

The BCS input coordinates of the transformation are z, x and phi.

For the axis transformation, the axes R1, R2 and R3 have the significance B.

The positive direction of rotation of an axis parameterized with +200 is the direction of rotation to rotate the positive z-direction by 90° to reach the positive x-direction.

The positive direction of rotation of an axis parameterized with -200 is reverse.

MP 103000125 (1..3) = ±300:

The BCS input coordinates of the transformation are x, y and phi.

For the axis transformation, the axes R1, R2 and R3 have the significance C.

The positive direction of rotation of an axis parameterized with +300 is the direction of rotation to rotate the positive x-direction by 90° to reach the positive y-direction.

The positive direction of rotation of an axis parameterized with -300 is reverse.

The remaining inputs MP 103000125 (4..8) are not relevant.

Depending on the specification, either the BCS input coordinates x, y and phi or z, x and phi or y, z and phi are programmed.

The control uses them to compute the MCS output coordinates R1, R2 and R3.

Example:

MP 103000125 (1) = -300

MP 103000125 (2) = 300

MP 103000125 (3) = 300

Reference position

The reference position for this axis transformation (as for most of the other axis transformations) is defined by the tool reference point being in the zero point of the input coordinate system (BCS).

For MP 103000125 (1..3) = ±100, it is the position y = 0, z = 0, phi = 0.

For MP 103000125 (1..3) = ±200, it is the position z = 0, x = 0, phi = 0.

For MP 103000125 (1..3) = ±300, it is the position z = 0, y = 0, phi = 0.

In this position, the output coordinate values, i.e. the positions of the three axes in the MCS, have to be read and entered as "Axis positions of the reference position" into the following machine parameters.

103000130 (1) for the R1-axis

103000130 (2) for the R2-axis

103000130 (3) for the R3-axis

Geometry

The individual parameters of MP 103000140 mean the following for MP 103000125 (1..3) = ±300:

MP 103000140 (1): Position of axis R1 in the BCS: x-coordinate

MP 103000140 (2): Position of axis R1 in the BCS: y-coordinate

MP 103000140 (3): Distance between axis R1 and swivel joint G1

MP 103000140 (4): Distance between swivel joint G1 and swivel joint G3

MP 103000140 (5): Position of axis R2 in the BCS: x-coordinate

MP 103000140 (6): Position of axis R2 in the BCS: y-coordinate

MP 103000140 (7): Distance between axis R2 and swivel joint G2

MP 103000140 (8): Distance between swivel joint G2 and swivel joint G3

MP 103000140 (9): Position of axis R3 in the BCS in reference position: x-coordinate

MP 103000140 (10): Position of axis R3 in the BCS in reference position: y-coordinate

MP 103000140 (11): Minimum angle allowed in the swivel joint G1

MP 103000140 (12): Minimum angle allowed in the swivel joint G2

MP 103000140 (13): Minimum angle allowed in the swivel joint G3

MP 103000140 (14):

0 or 1:

The positive direction of rotation is the direction of rotation to rotate the positive x-direction by 90° to reach the positive y-direction.

-1: The positive direction of rotation is the opposite.

For MP 103000125 (1..3) = ±100, thereof differing, the following applies:

MP 103000140 (1): Position of axis R1 in the BCS: z-coordinate

MP 103000140 (2): Position of axis R1 in the BCS: x-coordinate

MP 103000140 (5): Position of axis R2 in the BCS: z-coordinate

MP 103000140 (6): Position of axis R2 in the BCS: x-coordinate

MP 103000140 (9): Position of axis R3 in the BCS in reference position: z-coordinate

MP 103000140 (10): Position of axis R3 in the BCS in reference position: x-coordinate

MP 103000140 (14):

0 or 1:

The positive direction of rotation is the direction of rotation to rotate the positive z-direction by 90° to reach the positive x-direction.

-1: The positive direction of rotation is the opposite.

MP 103000140 (1): Position of axis R1 in the BCS: y-coordinate

MP 103000140 (2): Position of axis R1 in the BCS: z-coordinate

MP 103000140 (5): Position of axis R2 in the BCS: y-coordinate

MP 103000140 (6): Position of axis R2 in the BCS: z-coordinate

MP 103000140 (9): Position of axis R3 in the BCS in reference position: y-coordinate

MP 103000140 (10): Position of axis R3 in the BCS in reference position: z-coordinate

MP 103000140 (14):

0 or 1:

The positive direction of rotation is the direction of rotation to rotate the positive y-direction by 90° to reach the positive z-direction.

-1: The positive direction of rotation is the opposite.

MP 103000140(11..13) is used to protect the mechanics. Enter always the smallest angle allowed by the mechanics.

MP 103000160 is not relevant.

Tool corrections

Axis transformation supports a tool correction in the tool coordinate system (TCS): G47(XTR,YTR,ZTR)

The TCS moves with the tool. In reference position, BCS matches TCS.

The tool correction vector t points from the tool point to the tool reference point. It consists of the length corrections L1, L2 and L3 and of the D-correction table.

Relevant NC functions

COORD(...)	The 2-dimensional delta kinematics is activated with additional rotary axis in the point (RRR) This can optionally be done using the machine parameter block (i=1..20) or via the MTB system data (of type SysAxTrafo_t).
COORD(0,2) - or - COORD(0,1)	The active transformation is deactivated.

Tab. 11-27: Relevant NC functions

Relevant machine parameters

Up to 20 different axis transformations are declared with the following machine parameter block in the machine parameters.

1030 00110	Defines the transformation type.
1030 00120	Defines the system axes/coordinates participating in the transformation.
1030 00125	Defines the axis classification of the transformation axes.
1030 00130	Defines the axis positions of the reference position.
1030 00140	Defines the length and angle parameters.
1030 00160	Epsilon environments

Tab. 11-28: Relevant machine parameters

Relevant system data

The axis transformation can also be defined via system date instead of the machine parameters. Therefore, a system variable of type "SysAxTrafo_t" has to be created (see "MTX Machine Parameters", chapter "System data"). The structure type "SysAxTrafo_t" is control-internally defined in the xml schema files "/feprom/schemas/sdaxtrf.xsd" and contains - analog to the relevant machine parameters - the following transformation-relevant structure elements.

Structural element	Description
Type	Defines the transformation type.
AxisAssignment	Defines the system axes/coordinates participating in the transformation.
AxisClassification	Defines the axis classification of the transformation axes.

AxZeroPos	Defines the axis positions of the reference position.
LenParam	Defines the length and angle parameters.
EpsilonRanges	Epsilon environments

Tab. 11-29: Relevant system data

Examples

Parameterization

It is assumed that the kinematics shown in fig. 11-19 "Basic 2-dimensional delta transformation" on page 227, fig. 11-20 "2-dimensional delta transformation with additional rotary axis in the point" on page 228, fig. 11-21 "Consequence of a rotation of the axes R1 and R2, but without the axis R3 being rotated" on page 229 and fig. 11-22 "Consequence of an additional rotation of axis R3" on page 230 is parameterized as follows:

MP 103000110 = 2103002

MP 103000120 (1) = 3

MP 103000120 (2) = 8

MP 103000120 (3) = 1

The positive direction of rotation of axis R1 should be in clockwise direction.

The positive direction of rotation of axis R2 should be in anti-clockwise direction.

The positive direction of rotation of axis R3 should be on anti-clockwise direction.

The direction of rotation to rotate the positive x-direction by 90° to reach the positive y-direction is the anti-clockwise direction of rotation.

That is:

MP 103000125 (1) = -300

MP 103000125 (2) = 300

MP 103000125 (3) = 300

Before entering further parameters, specify the BCS. The position of the tool reference point has to be specified if the value “zero” is programmed for both the linear coordinates of the BCS, in this example x and y. The orientation of the vertical to the spindle nose has to be specified as well if phi = 0. This is the zero point of the BCS. Traverse the kinematics to this zero point. This reaches the reference position of the axis transformation. For an example, refer to the position shown as reference position in fig. 11-22 "Consequence of an additional rotation of axis R3" on page 230.

In this position, the output coordinate values (MCS) have to be read and entered as "Axis positions of the reference position" into:

MP 103000130 (1) for the axis R1, that is for the system axis 3

MP 103000130 (2) for the axis R2, that is for the system axis 8

MP 103000130 (3) for the axis R3, that is for the system axis 1

The following values are assumed for the example:

MP 103000130 (1) = 17

MP 103000130 (2) = 31

MP 103000130 (3) = 23

The machine parameters also refer to the reference position:

MP 103000140 (9): Position of axis R3 in the BCS in reference position: x-coordinate

MP 103000140 (10): Position of axis R3 in the BCS in reference position: y-coordinate

The requested BCS is drawn in fig. 11-23 "Example of a reference position of the 2-dimensional delta kinematics with additional rotary axis in the point (RRR)" on page 235. The tool reference point is in the zero point of the BCS.

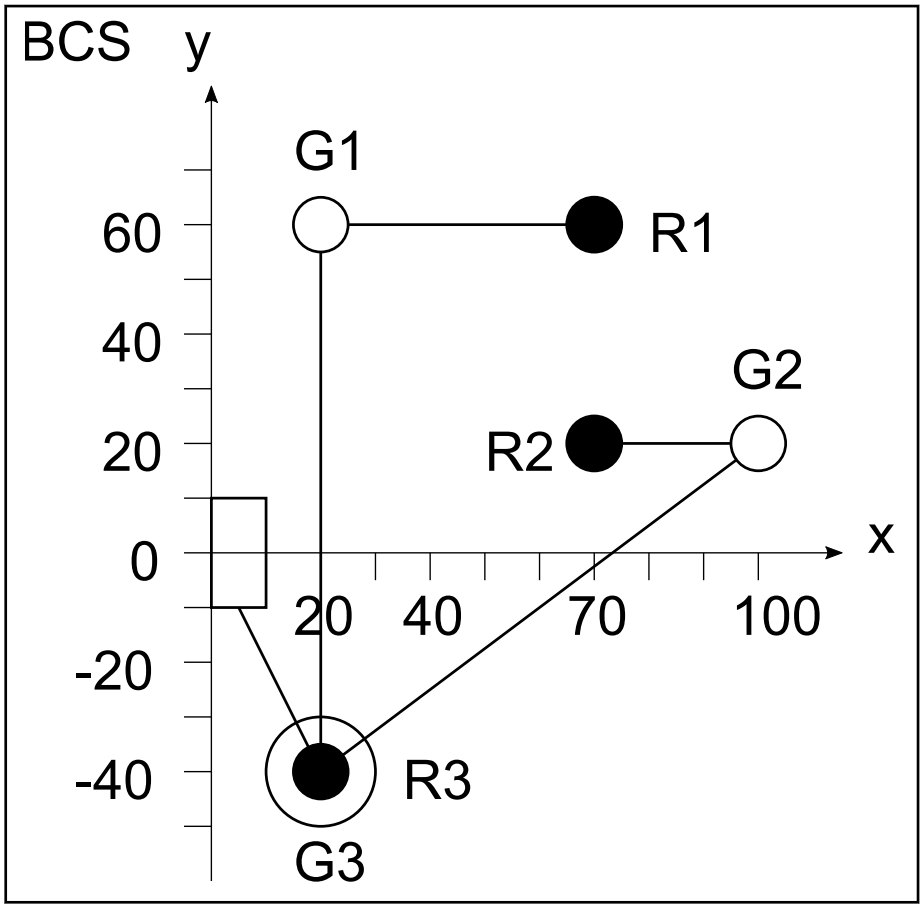


Fig. 11-23: Example of a reference position of the 2-dimensional delta kinematics with additional rotary axis in the point (RRR)

We read the following values:

MP 103000140 (1) = 70

MP 103000140 (2) = 60

MP 103000140 (3) = 50

MP 103000140 (4) = 100

MP 103000140 (5) = 70

MP 103000140 (6) = 20

MP 103000140 (7) = 30

MP 103000140 (8) = 100

MP 103000140 (9) = 20

MP 103000140 (10) = -40

For this example, it is assumed that the mechanics allows the following minimum angles in the swivel joints G1, G2 and G3:

MP 103000140 (11) = 30°

MP 103000140 (12) = 30°

MP 103000140 (13) = 45°

In fig. 11-23 "Example of a reference position of the 2-dimensional delta kinematics with additional rotary axis in the point (RRR)" on page 235, there is an angle of 90° in the swivel joint G1, an angle of approximately 36.87° in the swivel joint G3 and an angle of approximately 51.13° for the swivel joint.

Based on the reference position x = y = 0, phi = 0 shown in fig. 11-23 "Example of a reference position of the 2-dimensional delta kinematics with additional rotary axis in the point (RRR)" on page 235, the vertical should rotate ant-clockwise to the spindle nose if phi is increased. That is the direction of rotation to rotate the positive x-direction by 90° to reach the positive y-direction. Thus, do not change the default setting for this value. That is:

MP 103000140 (14) = 0

Tool correction

If the tool correction is not active, the position shown in fig. 11-24 "Example for BCS position and MCS position with and without tool correction" on page 237 is applied:

BCS: x = 80 , y = 140 , phi = -90°

MCS: R1 = 17° + 180° = 197° , R2 = 31° - 180° = -149° , R3 = 23° - 90° = -67°

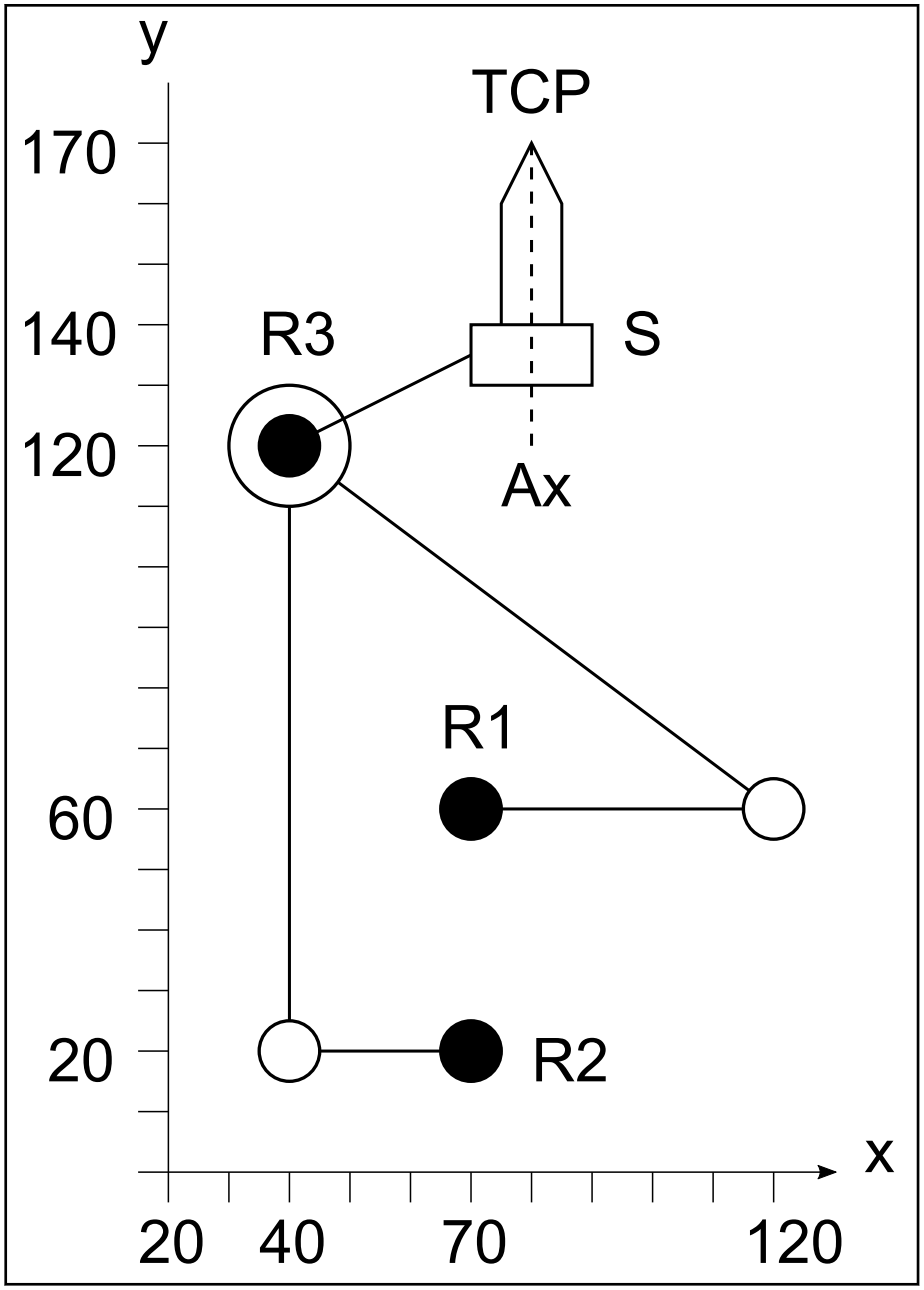


Fig. 11-24: Example for BCS position and MCS position with and without tool correction

If the axis transformation is active, the command G1 x=80 y=140 phi=-90 causes for example that the previously mentioned position is reached.

Activate the tool correction for the tool shown in fig. 11-24 "Example for BCS position and MCS position with and without tool correction" on page 237 as follows:

DCT(1,1,0) = 30

G47(XTR) ED1

In the reference position shown in fig. 11-23 "Example of a reference position of the 2-dimensional delta kinematics with additional rotary axis in the point (RRR)" on page 235, the tool correction vector for this tool would direct towards the positive x-direction. As BCS and TCS match in the reference position, the tool correction vector in the TCS directs towards the positive x-direc-

tion. If this tool correction was active, x = 80 , y = 170 , phi = -90° in fig. 11-24 "Example for BCS position and MCS position with and without tool correction" on page 237.

Parameterization in another BCS

To understand the effect of the BCS on the parameterization, we assume the reference position shown in fig. 11-25 "Example of a reference position in another BCS" on page 238.

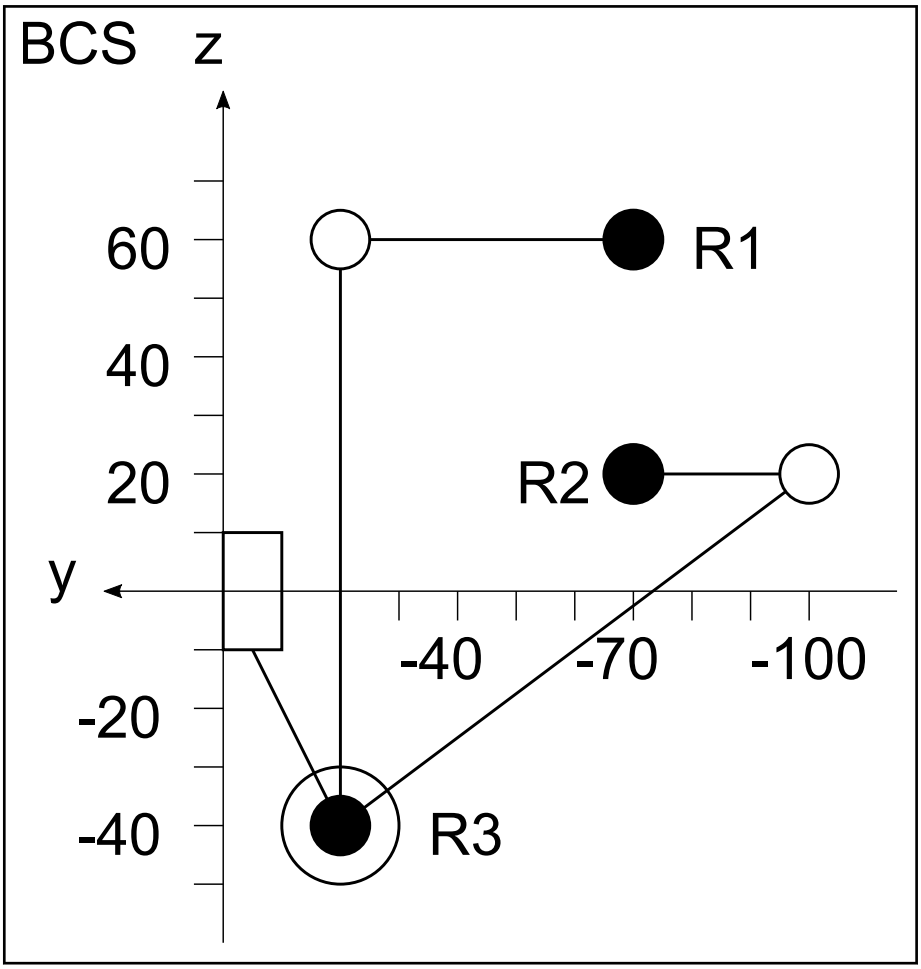


Fig. 11-25: Example of a reference position in another BCS

The following also applies:

The positive direction of rotation of axis R1 should be in clockwise direction.

The positive direction of rotation of axis R2 should be in anti-clockwise direction.

The positive direction of rotation of axis R3 should be on anti-clockwise direction.

Based on the reference position x = y = 0, phi = 0, the vertical should rotate ant-clockwise to the spindle nose if phi is increased.

The direction of rotation to rotate the positive y-direction by 90° to reach the positive z-direction is the clockwise direction of rotation.

This results in the following changes in the parameterization:

MP 103000125 (1) = 100

MP 103000125 (2) = -100

MP 103000125 (3) = -100

MP 103000140 (1) = -70
MP 103000140 (5) = -70
MP 103000140 (9) = -20
MP 103000140 (14) = -1

11.7.2 Handling instruction

Applying

The axis transformation can be applied via the relevant machine parameters. Therefore, the axis transformation function has to be activated via the parameter **tra** "Axis transformations" in the setup (SUP). The path of the machine parameter name is always "TRA/AxTrafo[1..20]/...".

Alternatively, the axis transformation can be applied via a user-defined system date of type SysAxTrafo_t.

The content of both variants is identical.

The axis positions of the reference position have to be determined.

The respectively equivalent machine parameter numbers, machine parameter names and SD elements are specified in the following table. To apply a transformation, one of the variants has to be assigned accordingly.

Transformation-relevant parameters

MP number	Description	
1030 00110	**Transformation type** MP name: TrafoType SD elem.: Type	
	2103002	Point of action AT2
1030 00120	**System axes/coordinates of the transformation** MP name: FwdInCoordIndTrafo[1..8] SD elem.: AxisAssignment[1..8]	
	[1]	System axis number of R1
	[2]	System axis number of R2
	[3]	System axis number of R3
	[4..8]	Not relevant

<table>
<tr><th>MP number</th><th>Description</th></tr>
<tr><td>1030 00125</td><td>Axis classification of transformation axes
MP name: CoordClass[1..8]
MP name: CoordDir[1..8]
SD elem.: AxisClassification[1..8]</td></tr>
<tr><td></td><td>Defines the axes involved in transformation, including their direction of motion.
• MP no. or SD element:
Value ">0" means:
The axis rotates/travels positively as in the figures above.
Value "<0" means:
The axis rotates/travels negatively as in the figures above.
Possible values are:
– ±100 = a-axis
– ±200 = b-axis
– ±300 = c-axis
• MP name:
– Possible values for CoordClass are:
– A,B,C (rotary)
– Possible values for CoordDir are:
– Positive
– Negative</td></tr>
<tr><td></td><td>If a parameter is not set, the following default settings apply:
<table><tr><th></th><th>MP no /
SD elem.:</th><th>CoordClass:</th><th>CoordDir:</th></tr><tr><td>[1]</td><td>300</td><td>C</td><td>Positive</td></tr><tr><td>[2]</td><td>300</td><td>C</td><td>Positive</td></tr><tr><td>[3]</td><td>300</td><td>C</td><td>Positive</td></tr><tr><td>[4..8]</td><td>Not relevant</td><td>Not relevant</td><td>Not relevant</td></tr></table></td></tr>
<tr><td>1030 00130</td><td>Axis positions of the reference position
MP name: RefPosTrafo[1..8]
SD elem.: AxZeroPos[1..8]
<table><tr><td>[1]</td><td>$R1^0$</td></tr><tr><td>[2]</td><td>$R2^0$</td></tr><tr><td>[3]</td><td>$R3^0$</td></tr><tr><td>[4..8]</td><td>Not relevant</td></tr></table></td></tr>
</table>

MP number	Description	
1030 00140	**Length and angle parameters** MP name: JointParTrafo[1..16] SD elem.: LenParam[1..16]	
	Defines the geometry of the kinematics in mm or degrees	
	[1]	Position of axis R1 in the BCS: Main coordinate
	[2]	Position of axis R1 in the BCS: Secondary coordinate
	[3]	Distance between axis R1 and swivel joint G1
	[4]	Distance between swivel joint G1 and swivel joint G3
	[5]	Position of axis R2 in the BCS: Main coordinate
	[6]	Position of axis R2 in the BCS: Secondary coordinate
	[7]	Distance between axis R2 and swivel joint G2
	[8]	Distance between swivel joint G2 and swivel joint G3
	[9]	Position of axis R3 in the BCS in reference position: Main coordinate
	[10]	Position of axis R3 in the BCS in reference position: Secondary coordinate
	[11]	Minimum angle allowed in the swivel joint G1
	[12]	Minimum angle allowed in the swivel joint G2
	[13]	Minimum angle allowed in the swivel joint G3
	[14]	0 or 1: The positive direction of rotation is the direction of rotation to rotate the positive x-direction by 90° to reach the positive y-direction. -1: The positive direction of rotation is the opposite.
	[15..16]	Not relevant
1030 00160	**Epsilon environments** MP name: EpsAxTrafo SD elem.: EpsilonRanges	
	Not relevant	

Tab. 11-30: Transformation-relevant parameters

Activating *If the 2-dimensional delta kinematics is configured with an additional rotary axis in the point (RRR) via*

- machine parameter, the transformation is enabled in the NC program with (i=1,..20):

  COORD(<i>)
- system date, the transformation is enabled in the NC program with (type "SysAxTrafo_t"):

  COORD(<SD.MtbAxTrafo>)

Deactivating **COORD(0)** or **COORD(0,2)**

switches off the axis transformation in the NC program.

11.8 4-axis transformation - Type 2: LLLR

11.8.1 Description

Function

This transformation is a configuration with three linear axes and one swivel axis. Swivel axis R_2 causes the tool to rotate. This transformation is used at point of action 2.

Transformation name	Transformation type	Point of action	Coordinate system
4-axis transformation LLLR	3131002	AT2	MCS, BCS

Output variables in the MCS: X, Y, Z, R_2

Input variables in the BCS: x,y,z,θ

Linear MCS coordinates X, Y, Z span a right-handed Cartesian coordinate system. In the reference position (BCS in parallel to MCS), R_2 optionally rotates around X (type A axis) or around Y (type B axis). Axis R_2 is used for orientation.

The figures below show the LLLR configurations in the reference position with R_2 = A and R_2 = B, respectively. This means for each configuration: If the machine assumes the position shown in the figure, the BCS coordinates have to be (x, y, z, θ) = (0, 0, 0, 0), with associated MCS coordinates (X, Y, Z, R_2) = (X^0, Y^0, Z^0, R_2^0). The figures show the positive directions of rotation/traversing of the axes. Set them accordingly. In this position, the tip (Tool Center Point, TCP) of rotation-symmetric tools is placed on the origin of basic coordinate system (BCS) and is oriented along the z-coordinate. The axis transformation unambiguously determines the orientation of the BCS.

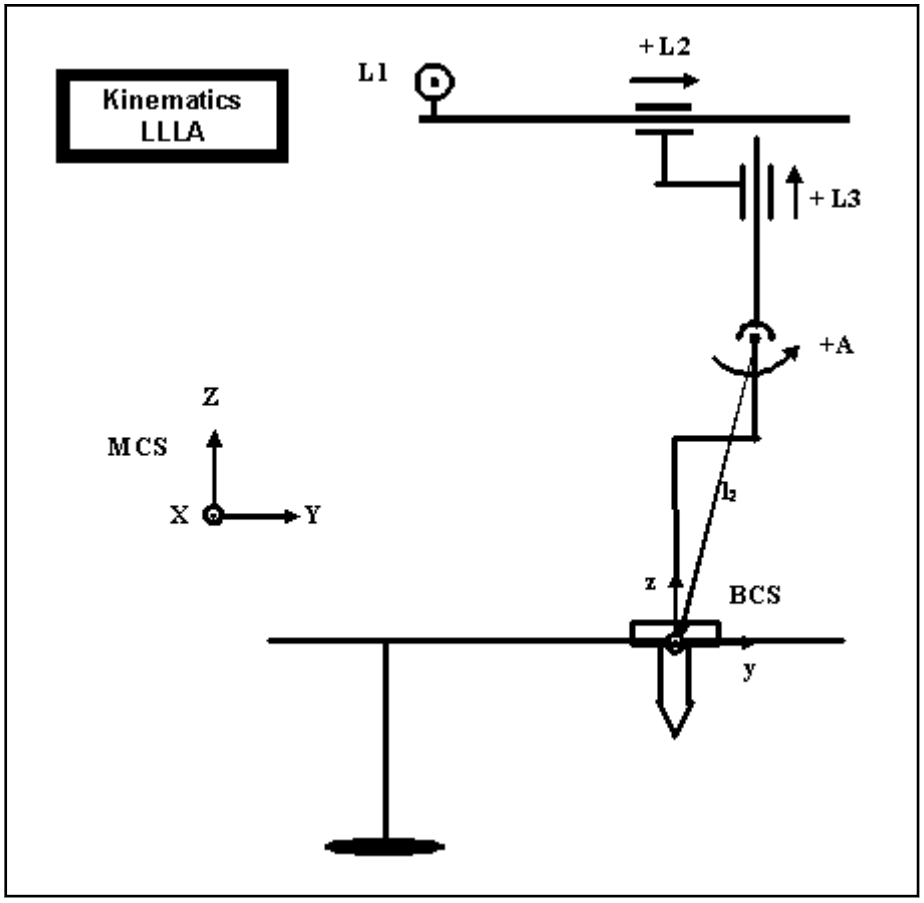


Fig. 11-26: LLLR (type A axis)

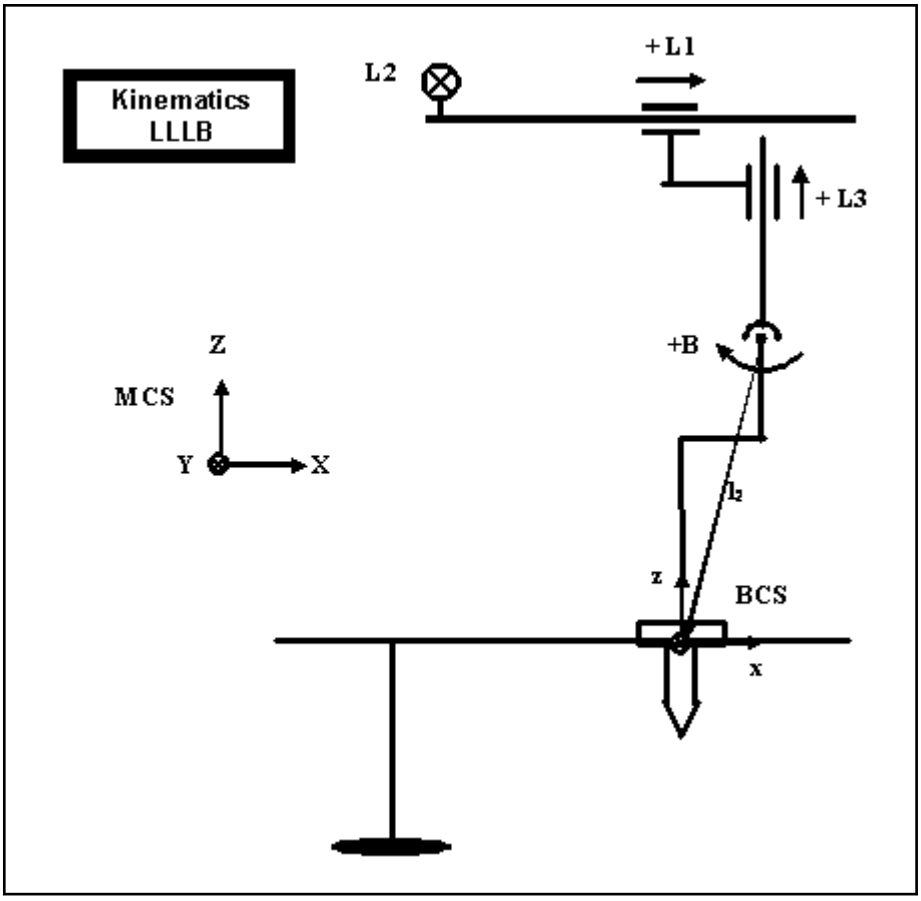


Fig. 11-27: LLLR (type B axis)

Vector l_2 defines the geometry of the 4-axis kinematics. Its form is $l_2=(l_{2x},l_{2y},l_{2z})$. l_2 is the vector extending from axis R_2 (A/B axis) to the zero point of the BCS.

Rotary axis R_2 can be an endless or a rotary axis.

Tool correction: A tool correction is possible. The correction vector points from the tool point to the tool holder.

Relevant NC functions

COORD ( )	The 4-axis transformation LLLR is activated. This can optionally be done using the machine parameter block (i=1..20) or via the MTB system data (of type SysAxTrafo_t).
COORD (0,2)	The active axis transformation is deactivated.

Tab. 11-31: Relevant NC functions

11.8.2 Handling instruction

Applying The axis transformation can be applied via the relevant machine parameters. Therefore, the axis transformation function has to be activated via the parameter **tra** "Axis transformations" in the setup (SUP). The path of the machine parameter name is always "TRA/AxTrafo[1..20]/...".

Alternatively, the axis transformation can be applied via a user-defined system date of type SysAxTrafo_t.

The content of both variants is identical.

The axis positions of the reference position have to be determined.

The respectively equivalent machine parameter numbers, machine parameter names and SD elements are specified in the following table. To apply a transformation, one of the variants has to be assigned accordingly.

4-axis variant LLRR: Transformation-relevant parameters

MP number	Description	
1030 00110	**Transformation type** MP name: TrafoType SD elem.: Type	
	3131002	4-axis transformation LLLR
1030 00120	**System axes/coordinates of the transformation** MP name: FwdInCoordIndTrafo[1..8] SD elem.: AxisAssignment[1..8]	
	[1]	System axis number of X
	[2]	System axis number of Y
	[3]	System axis number of Z
	[4]	System axis number of R_2 (a-/b-axis)
	[5..8]	Not relevant

<table>
<tr><th>MP number</th><th>Description</th></tr>
<tr><td>1030 00125</td><td>Axis classification of transformation axes
MP name: CoordClass[1..8]
MP name: CoordDir[1..8]
SD elem.: AxisClassification[1..8]</td></tr>
<tr><td></td><td>Defines the axes involved in transformation, including their direction of motion.
• MP no. or SD element:
Value ">0" means:
The axis rotates/travels positively as in the figures above.
Value "<0" means:
The axis rotates/travels negatively as in the figures above.
Possible values are:
– +/-1 = x-axis
– +/-2 = y-axis
– +/-3 = z-axis
– +/-100 = a-axis
– +/-200 = b-axis
– +/-300 = c-axis
• MP name:
– Possible values for CoordClass are:
– X,Y,Z (linear)
– A,B,C (rotary)
– Possible values for CoordDir are:
– positive (regarding fig. "Reference position"
– negative (regarding fig. "Reference position"</td></tr>
<tr><td></td><td>If a parameter is not set, the following default settings apply:
<table><tr><td></td><td>MP no /
SD elem.:</td><td>CoordClass:</td><td>CoordDir:</td></tr><tr><td>[1]</td><td>1</td><td>X</td><td>Positive</td></tr><tr><td>[2]</td><td>2</td><td>Y</td><td>Positive</td></tr><tr><td>[3]</td><td>3</td><td>Z</td><td>Positive</td></tr><tr><td>[4]</td><td>200</td><td>B</td><td>Positive</td></tr><tr><td>[5..8]</td><td>Not relevant</td><td>0</td><td>Positive</td></tr></table></td></tr>
</table>

MP number	Description	
1030 00130	**Axis positions of the reference position** MP name: RefPosTrafo[1..8] SD elem.: AxZeroPos[1..8]	
	[1]	X^0
	[2]	Y^0
	[3]	Z^0
	[4]	$R_2{}^0$
	[5..8]	Not relevant
1030 00140	**Length and angle parameters** MP name: JointParTrafo[1..16] SD elem.: LenParam[1..16]	
	Defines the length vectors l_1, l_2 in [mm].	
	[1]	Not relevant
	[2]	Not relevant
	[3]	Not relevant
	[4]	l_{2x}
	[5]	l_{2y}
	[6]	l_{2z}
	[7..16]	Not relevant
1030 00160	**Epsilon environments**	
	Not relevant	

Tab. 11-32: Transformation-relevant parameters

Activating

If the transformation is configured via

- machine parameter, the transformation is enabled in the NC program with (i=1,..20):

  COORD(<i>)

- system date, the transformation is enabled in the NC program with (type "SysAxTrafo_t"):

  COORD(<SD.MtbAxTrafo>)

Example program:

The 4-axis transformation LLLR is defined in the fourth machine parameter block. No axis transformation is active at programming start:

Enable transformation

```
N100  G1 F1000 Z10 X=10 B=30   ;Axes Z, X and B are traversed
N110  COORD(4)                 ;Switching on the axis transformation
N130  z20 y30 theta10          ;Coordinates z, y and theta are traversed
        :
N999  COORD(0)                 ;Switching-off the axis transformation
                               ;Axes are again programmed from here
```

Deactivating

COORD(0) or **COORD(0,2)**

switches off the axis transformation in the NC program.

11.9 4-axis transformation - Type 3: RLLL

11.9.1 Description

Function

This transformation is a configuration with three linear axes and one swivel axis. Swivel axis R_2 causes the workpiece to rotate. This transformation is used at point of action 2.

Transformation name	Transformation type	Point of action	Coordinate system
4-axis transformation RLLL	3131003	AT2	MCS, BCS

Output variables in the MCS: R_2, X, Y, Z

Input variables in the BCS: x,y,z,θ

Linear MCS coordinates X, Y, Z span a right-handed Cartesian coordinate system. In the reference position (BCS in parallel to MCS), R_2 optionally rotates around X (type A-axis) or around Y (type B-axis) or around Z (type C-axis). Axis R_2 is used for orientation.

The following figures show the RLLL configurations in the reference position with R_2 = A and R_2 = B or R_2 = C. This means for each configuration: If the machine assumes the position shown in the figure, the BCS coordinates (x, y, z, θ) = (0, 0, 0, 0), with associated MCS coordinates (R_2, X, Y, Z) = (R_2^0, X^0, Y^0, Z^0). The figures show the positive directions of rotation/traversing of the axes. Set them accordingly. In this position, the tip (Tool Center Point, TCP) of rotation-symmetric tools is placed on the origin of basic coordinate system (BCS) and is oriented along the z-coordinate. The axis transformation unambiguously determines the orientation of the BCS.

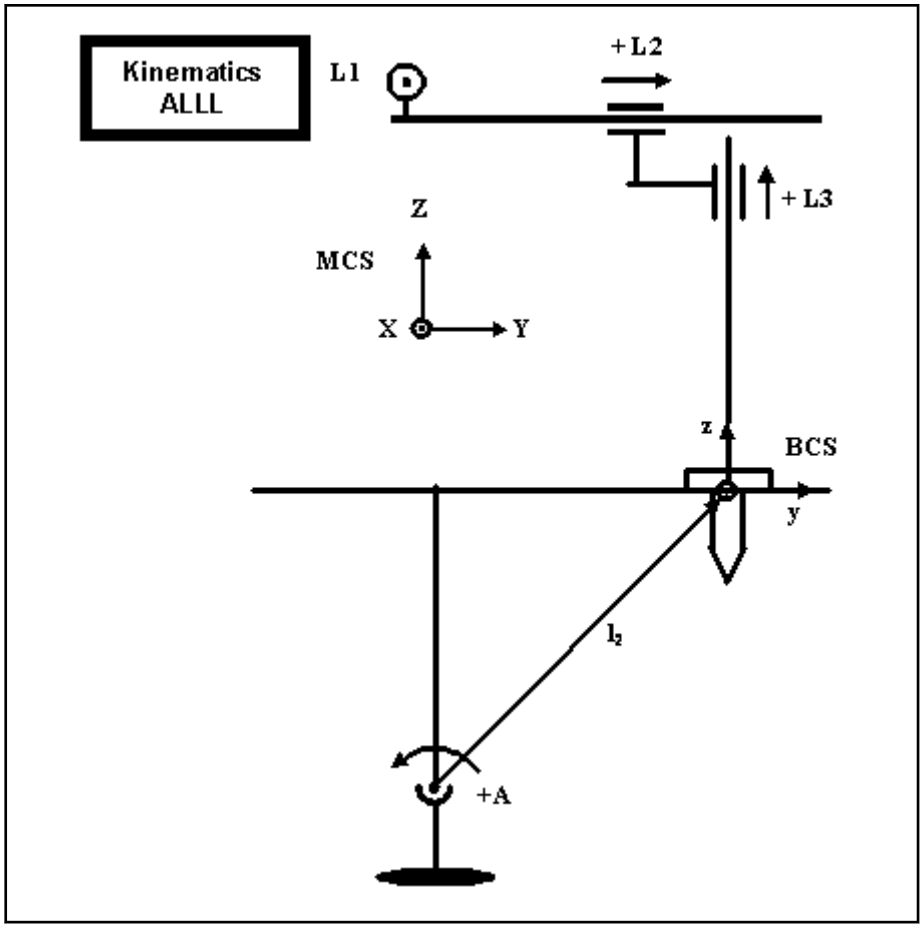


Fig. 11-28: RLLL (type A axis)

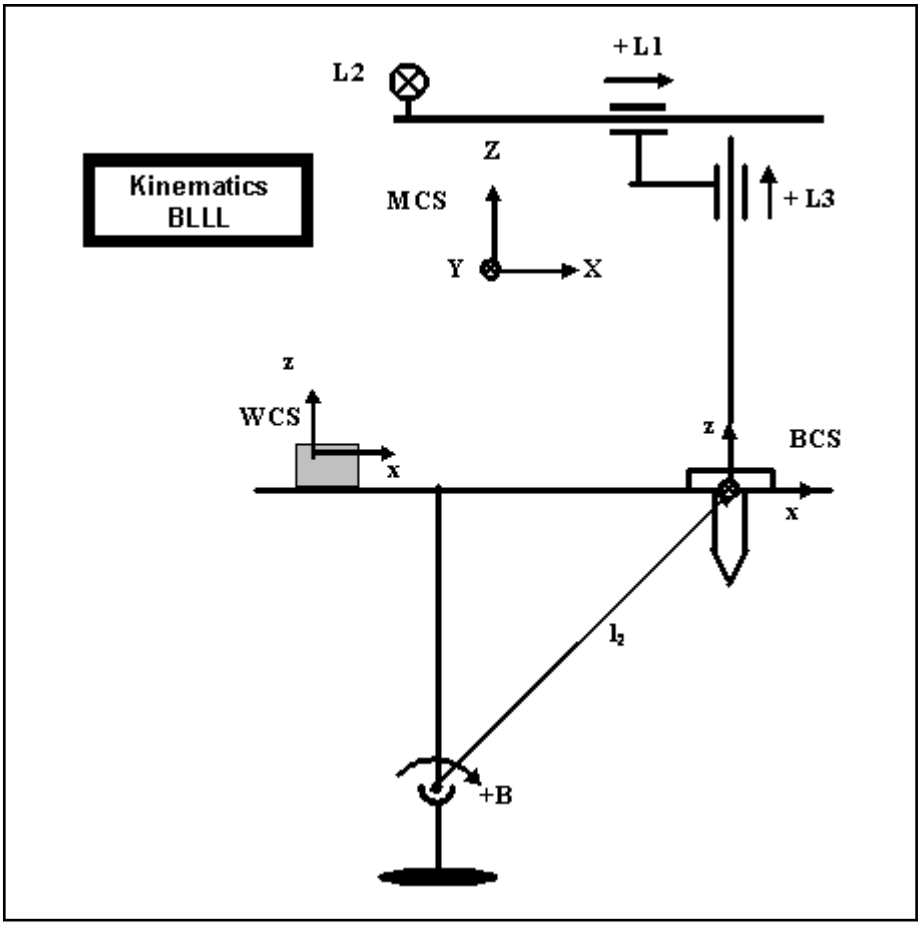


Fig. 11-29: RLLL (type B axis)

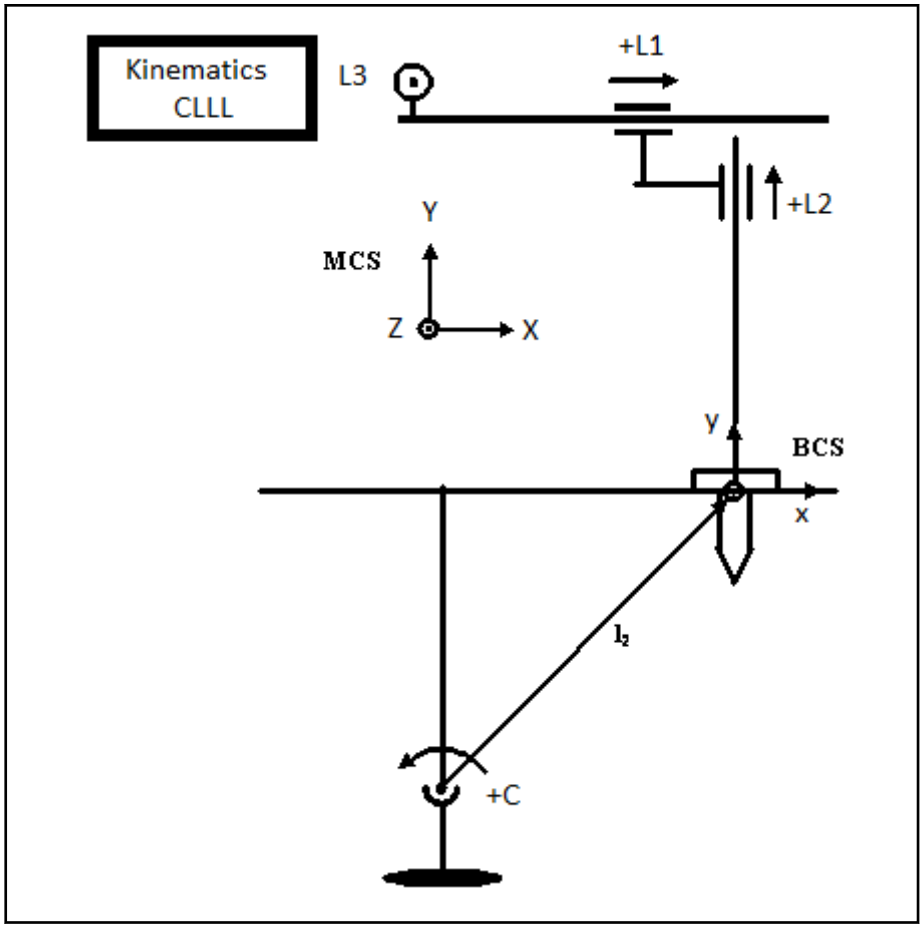


Fig. 11-30: RLLL (type C-axis)

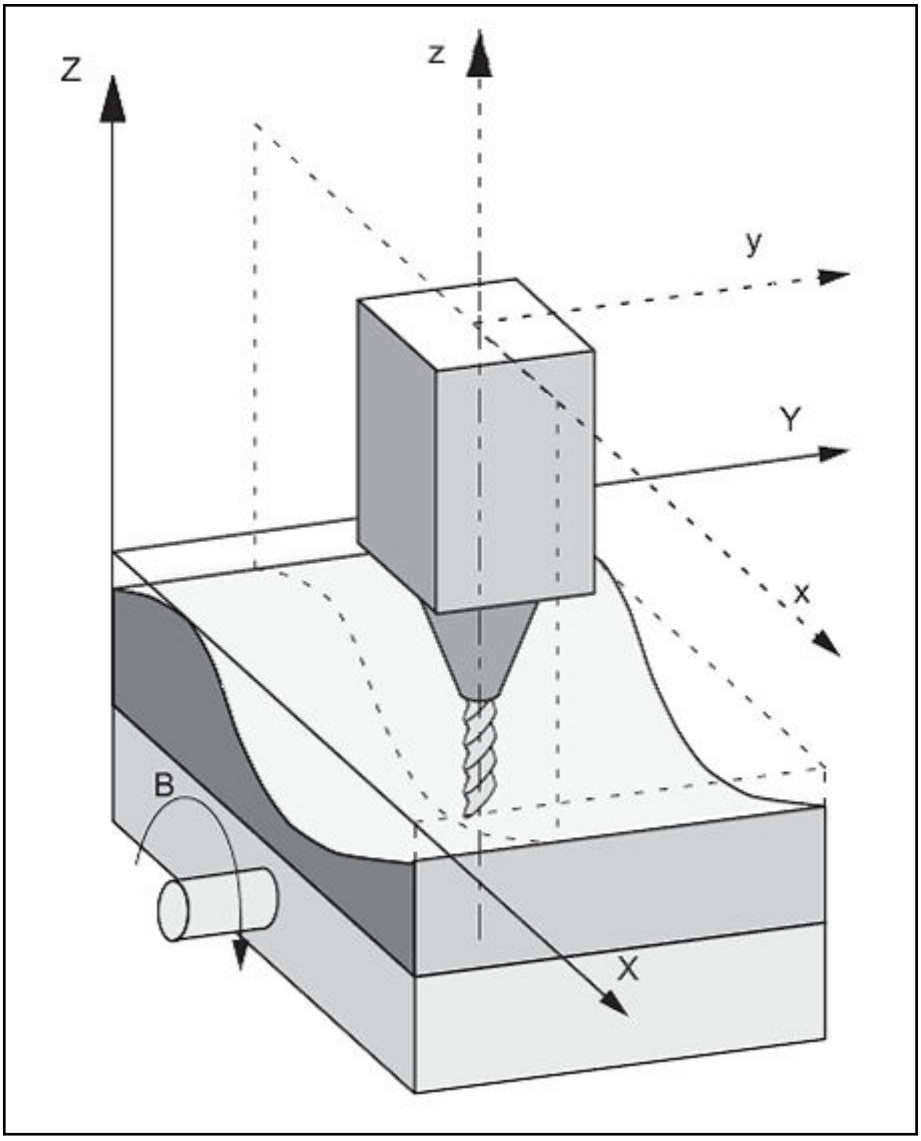


Fig. 11-31: RLLL (type B axis)

Vector l_2 defines the geometry of the 4-axis kinematics. Its form is $l_2=(l_{2x},l_{2y},l_{2z})$. l_2 is the vector extending from axis R_2 (A-/B-/C-axis) to the zero point of the BCS.

Rotary axis R_2 can be an endless or a rotary axis.

Tool correction: A tool correction is possible. The correction vector points from the tool point to the tool holder.

Relevant NC functions

COORD ( )	The 4-axis transformation RLLL is activated. This can optionally be done using the machine parameter block (i=1..20) or via the MTB system data (of type SysAxTrafo_t).
COORD (0,2)	The active axis transformation is deactivated.

Tab. 11-33: Relevant NC functions

11.9.2 Handling instruction

Applying

The axis transformation can be applied via the relevant machine parameters. Therefore, the axis transformation function has to be activated via the parameter **tra** "Axis transformations" in the setup (SUP). The path of the machine parameter name is always "TRA/AxTrafo[1..20]/...".

Alternatively, the axis transformation can be applied via a user-defined system date of type SysAxTrafo_t.

The content of both variants is identical.

The axis positions of the reference position have to be determined.

The respectively equivalent machine parameter numbers, machine parameter names and SD elements are specified in the following table. To apply a transformation, one of the variants has to be assigned accordingly.

4-axis variant LLRR: Transformation-relevant parameters

MP number	Description	
1030 00110	**Transformation type** MP name: TrafoType SD elem.: Type	
	3131003	4-axis transformation RLLL
1030 00120	**System axes/coordinates of the transformation** MP name: FwdInCoordIndTrafo[1..8] SD elem.: AxisAssignment[1..8]	
	[1]	System axis number of R_2 (A-/B-/C-axis)
	[2]	System axis number of X
	[3]	System axis number of Y
	[4]	System axis number of Z
	[5..8]	Not relevant

<table>
<tr><th>MP number</th><th>Description</th></tr>
<tr><td>1030 00125</td><td>Axis classification of transformation axes
MP name: CoordClass[1..8]
MP name: CoordDir[1..8]
SD elem.: AxisClassification[1..8]</td></tr>
<tr><td></td><td>Defines the axes involved in transformation, including their direction of motion.
• MP no. or SD element:
Value ">0" means:
The axis rotates/travels positively as in the figures above.
Value "<0" means:
The axis rotates/travels negatively as in the figures above.
Possible values are:
– +/-1 = x-axis
– +/-2 = y-axis
– +/-3 = z-axis
– +/-100 = a-axis
– +/-200 = b-axis
– +/-300 = c-axis
• MP name:
– Possible values for CoordClass are:
– X,Y,Z (linear)
– A,B,C (rotary)
– Possible values for CoordDir are:
– positive (regarding fig. "Reference position"
– negative (regarding fig. "Reference position"</td></tr>
<tr><td></td><td>If a parameter is not set, the following default settings apply:
<table><tr><td></td><td>MP no /
SD elem.:</td><td>CoordClass:</td><td>CoordDir:</td></tr><tr><td>[1]</td><td>200</td><td>B</td><td>Positive</td></tr><tr><td>[2]</td><td>1</td><td>X</td><td>Positive</td></tr><tr><td>[3]</td><td>2</td><td>Y</td><td>Positive</td></tr><tr><td>[4]</td><td>3</td><td>Z</td><td>Positive</td></tr><tr><td>[5..8]</td><td>Not relevant</td><td>0</td><td>Positive</td></tr></table></td></tr>
</table>

MP number	Description	
1030 00130	**Axis positions of the reference position** MP name: RefPosTrafo[1..8] SD elem.: AxZeroPos[1..8]	
	[1]	R_2^0
	[2]	X^0
	[3]	Y^0
	[4]	Z^0
	[5..8]	Not relevant
1030 00140	**Length and angle parameters** MP name: JointParTrafo[1..16] SD elem.: LenParam[1..16]	
	Defines the length vectors l_1, l_2 in [mm].	
	[1]	Not relevant
	[2]	Not relevant
	[3]	Not relevant
	[4]	l_{2x}
	[5]	l_{2y}
	[6]	l_{2z}
	[7..16]	Not relevant
1030 00160	**Epsilon environments**	
	Not relevant	

Tab. 11-34: Transformation-relevant parameters

Activating

If the transformation is configured via

- machine parameter, the transformation is enabled in the NC program with (i=1,..20):

  COORD(<i>)

- system date, the transformation is enabled in the NC program with (type "SysAxTrafo_t"):

  COORD(<SD.MtbAxTrafo>)

Example program:

The 4-axis transformation RLLL is defined in the 4th machine parameter set. No axis transformation is active at programming start:

Enable transformation

```
N100  G1 F1000 Z10 X=10 B=30   ;Axes Z, X and B are traversed
N110  COORD(4)                 ;Switching on the axis transformation
N130  z20 y30 theta10          ;Coordinates z, y and theta are traversed
        :
N999  COORD(0)                 ;Switching-off the axis transformation
                               ;Axes are again programmed from here
```

Deactivating

COORD(0) or **COORD(0,2)**

switches off the axis transformation in the NC program.

11.10 5-axis transformation - Type 1: LLLR

11.10.1 General information on variants 1-3

The 5-axis transformation of type 1 has three variants of type LLLCB (see chapter 11.10.2 "Variant 1 (with vector orientation)" on page 257, chapter 11.10.3 "Variant 2 (with linear orientation)" on page 258 and chapter 11.10.4 "Variant 3 (with linear motion of the rotary axes) " on page 258). They all describe the kinematics LLLCB and differ only in the type of orientation programming.

The 5-axis transformation (LLLCB) comprises three linear axes (X,Y,Z) forming a right-handed Cartesian coordinate system, and two rotary axes R1 (type C-axis) and R2 (type B-axis) which orientate the tool in space.

The rotary axes R1 and R2 can be endless axes or rotary axes.

With regard to the involvement of the axes in the tool and workpiece motion, the **axis arrangement** of the 5-axis transformation has the following arrangement:

- The tool is moved via the linear axes X, Y, Z. The rotary axis R1 (type C-axis) rotates the tool around the Z-axis and the rotary axis R2 (type B-axis) swivels the tool around the Y-axis.

The workpiece is connected with the BCS. In reference position, the position of the TCS coincides with that of the BCS.

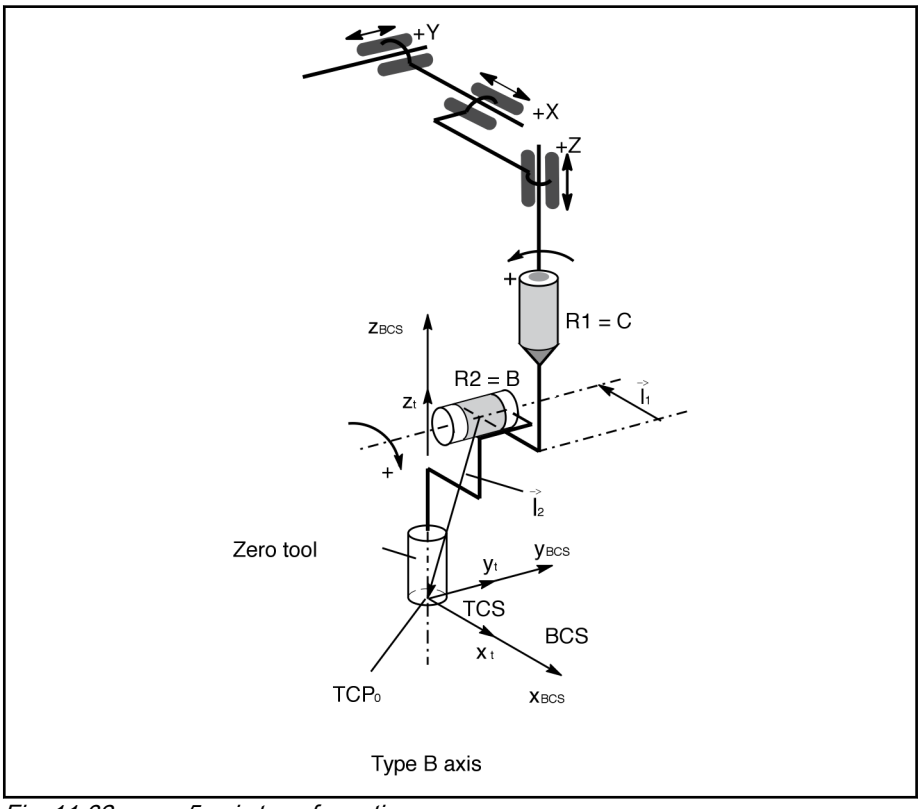


Fig. 11-32: 5-axis transformation

11.10.2 Variant 1 (with vector orientation)

Function

Transformation name	Transformation type	Point of action	Coordinate system
5-axis transformation with tool rotating/ swiveling axis	Variant 1: 3232201 with vector orientation	AT2	MCS, BCS

Tab. 11-35: Function - Variant 1

Output variables in the MCS: X,Y,Z,R2,R1

Input variables in the BCS: x, y, z, phi, theta

The transformation is used at point of action 2, i.e. it acts between the machine coordinate system MCS = (X,Y,Z,R2,R1) and the basic coordinate system BCS = (x, y, z, phi, theta).

The axis transformation supports the vector orientation motion (rotation of the orientation vector $\vec{\rho}$) around a rotary axis fixed in space, large circle motion of the vector point. If a different value is programmed for phi and/or theta, the orientation of the tool changes.

Orientation always rotates around the tool point (TCP, Tool Center Point). This means that it is assumed that the TCP is fixed and that the tool axis rotates around this fixed point.

Under certain circumstances, all five axes can be involved in the change of orientation. Orientation always points at the direction of the tool holder.

The orientation vector $\vec{\rho}$ can be programmed as a spline.

Restriction

Online correction of φ and θ is **not** possible.

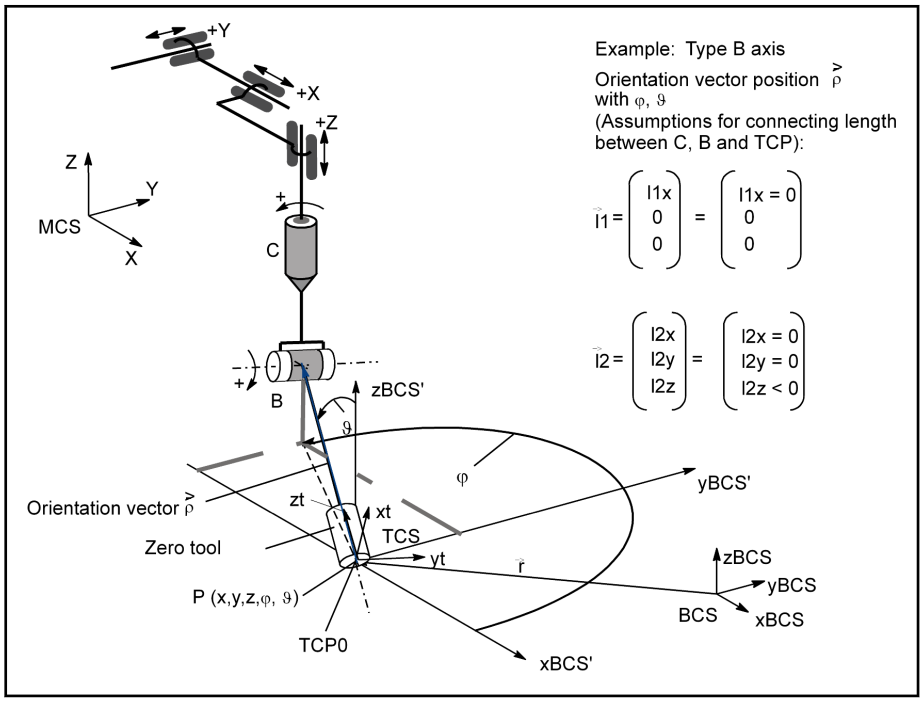


Fig. 11-33: Restriction - Example

11.10.3 Variant 2 (with linear orientation)

Function

Transformation name	Transformation type	Point of action	Coordinate system
5-axis transformation with tool rotating/ swiveling axis	Variant 2: 3232101 with linear orientation	AT2	MCS, BCS

Tab. 11-36: Function - Variant 2

Output variables in the MCS: X,Y,Z,R2,R1

Input variables in the BCS: x, y, z, φ, ϑ

The transformation is used at point of action 2, i.e. it acts between the machine coordinate system MCS = (X,Y,Z,R2,R1) and the basic coordinate system BCS = (x, y, z, phi, theta).

The axis kinematics is identical with type 3232201. However, the axis transformation supports the linear orientation motion (straight line in a theoretical phi-theta plane), i.e., **no** vector orientation. In all orientation motions, a constant phi is expected.

Both polar coordinates phi, theta can also be programmed a splines.

Online correction is possible for all coordinates.

11.10.4 Variant 3 (with linear motion of the rotary axes)

Function

Transformation name	Transformation type	Point of action	Coordinate system
5-axis transformation with tool rotating/ swiveling axis	Variant 3: 3032001 with linear motion of the rotary axes	AT2	MCS, BCS

Tab. 11-37: Function - Variant 3

Output variables in the MCS: X,Y,Z,R2,R1

Input variables in the BCS: x, y, z, C, B

The transformation is used at point of action 2, i.e. it acts between the machine coordinate system MCS = (X,Y,Z,R2,R1) and the basic coordinate system BCS = (x,y,z,B,C).

This transformation is mostly identical with that of the type 3232201; however, it does not support any programmable orientation coordinates but orients the tool by direct programming of the rotary axes B and C.

The orientation motion is executed in a linear fashion in the rotary axis positions.

11.10.5 Common characteristics of the variants 1-3

Relevant NC functions

COORD(<i>)	i=1..20 is used to activate a 5-axis transformation which is defined in MP 1030 00110.
COORD(0,2) - or - COORD(0)	The active transformation is deactivated.

Tab. 11-38: Relevant NC functions

Relevant CPL functions

SDR(300)	Active Epsilon environment. The active value can be greater than the value set in EpsAx-Trafo (MP 1030 00160).

Tab. 11-39: Relevant CPL functions

Relevant machine parameters

1030 00110	Defines the transformation type.
1030 00120	Defines the system axes/coordinates participating in the transformation.
1030 00125	Defines the axis classification of the transformation axes.
1030 00130	Defines the axis positions of the reference position.
1030 00140	Defines the length and angle parameters.
1030 00160	Epsilon environments

Tab. 11-40: Relevant machine parameters

Reference position

In the reference position of all 5-axis configurations (in the MCS: X, Y, Z, R1 = C and R2 = B), the point (TCP) of the rotation-symmetric tool (cutter or laser), based on the basic coordinate system (BCS) is oriented along the z-coordinate. The C-axis rotates the B-axis body and the connected tool along the z-coordinate. The B-axis rotates the tool around the y-coordinate of the BCS. The direction arrows at the axis bodies indicate the respective positive direction of rotation.

The axis transformation unambiguously determines the orientation of the BCS.

The axis vector $\vec{r}_m$ corresponds to the distance between the MCS zero point and the reference position (zero point of the BCS).

The angles to be rotated by the rotary axes so that the reference orientation of phi, theta = 0 in the reference space position x,y,z = 0 can be reached are also brought into relation with the MCS zero point.

Axis connection vectors

The axis connection vectors $\vec{l}_1$ and $\vec{l}_2$ define the geometry of the 5-axis kinematics in the reference position (see the following illustration).

- $\vec{l}_1$ is a vector along the x-axis from the R1 axis (C-axis) to the R2 axis (b-axis); thus, it only has the component l_{1x}.

$$\vec{l}_1 = \begin{bmatrix} l_{1x} \\ 0 \\ 0 \end{bmatrix}$$

- $\vec{l}_2$ is the continuation vector of the R2-axis (b-axis/a-axis) to the zero point of the BCS (in reference position) and consists of up to 3 components l_{2x}, l_{2y} and l_{2z}. The components take the opposite direction in respect of the BCS coordinate direction.

$$\vec{l}_2 = \begin{bmatrix} l_{2x} \\ l_{2y} \\ l_{2z} \end{bmatrix}$$

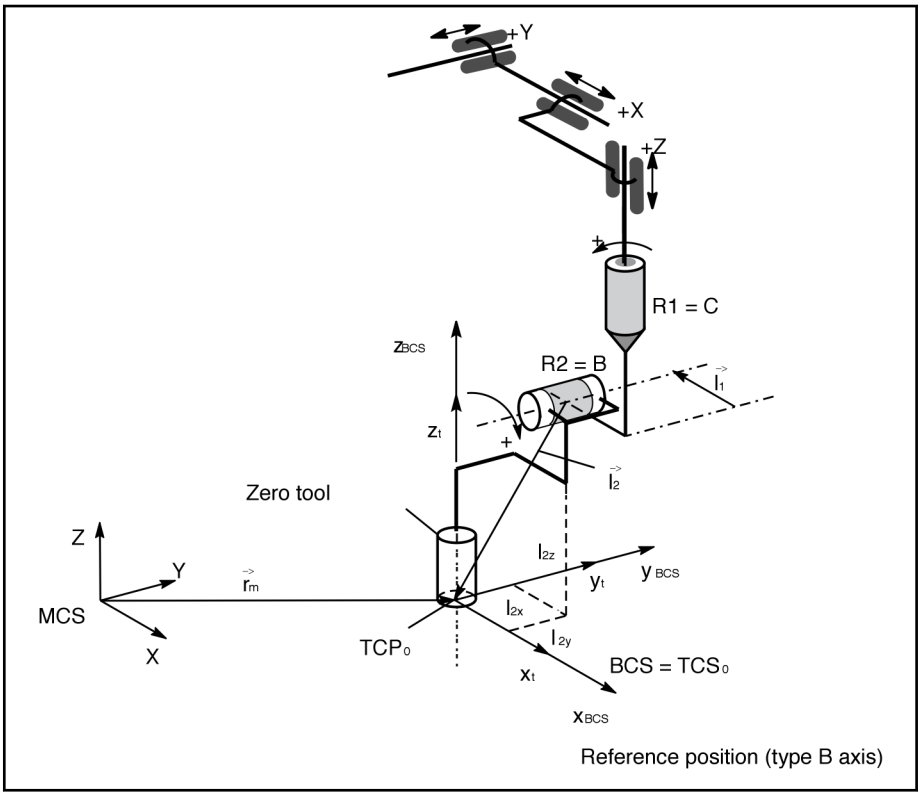


Fig. 11-34: Reference position (type b-axis)

Example:

5-axis rotation swivel head with and without offset of the B-axis

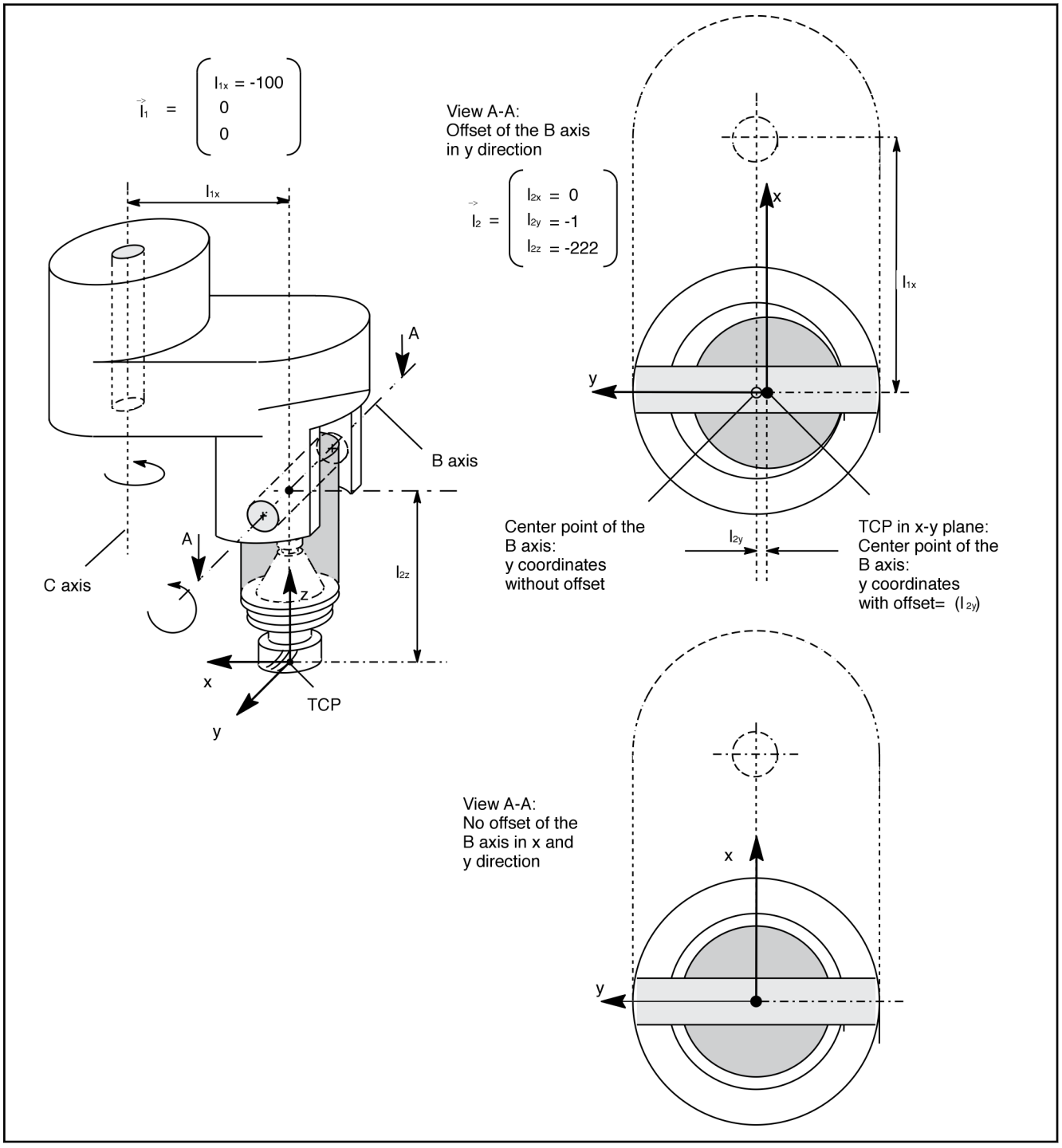


Fig. 11-35: Example

Special features

- **Ambiguous axis positions:**

In general, a TCP-position $\vec{r}$ and a tool orientation $\vec{\rho}$ can be achieved by two different axis positions (transformation branches). The branches differ in their b-axis positions.

Which one of the two poses (transformation branches) is used is decided on activation of the axis transformation. Accordingly, a transition from branch -1 to 1 and vice versa is only possible when traveling through the orientation location theta = 0° (north pole singularity).

- **North pole singularity:**

  Singularities of the orientation motion are taken into consideration in 5-axis transformation. In case of non-tangential traveling of $\vec{\rho}$ through the places $\vec{\rho}$ = (0,0,0), i.e. theta = 0 (north pole singularity), the NC automatically generates a special rotary block for the C-axis.

  If the orientation vector closely passes the singularity, this results in a high C-axis velocity, i.e. the path velocity has to be greatly reduced to prevent the c-axis from exceeding its maximum permissible velocity. In order to avoid the resulting creeping movements, an ε environment is introduced around the location theta = 0°. If the orientation vector passed the ε environment, the orientation motion is guided through the north pole. The value for ε is recorded in the MP.

- **Feed:**

  After activation of the 5-axis transformation, the system is switched to spatial coordinate programming. The programmed feed (F) now only relates to the programmable position coordinates, i.e. the path velocity of the tool center point (TCP) is programmed with the F-word. This path velocity is not changed by additional orientation and pseudo coordinate motions. The orientation and pseudo coordinate motion is included synchronously, i.e. the end position is reached simultaneously for all coordinates. However, the included motion of the orientation and pseudo coordinates may result in an additional limitation of path kinematics (maximum path velocity and acceleration) as the limits for all axes involved in the motion are monitored.

TCS, TCP

TCS stands for Tool Coordinate System.

TCP_0/TCP_1 stands for the Tool Center Point **without/with** correction.

Tool correction

Axis transformation supports a tool correction in the tool coordinate system (TCS). In reference position, BCS conforms with TCS_0. The tool correction vector $\vec{l}_t$ points from the corrected tool center point TCP_1 to the uncorrected TCP_0. It comprises the length corrections L1, L2 and L3 from the D-correction table.

Example:

---

for the tool correction

A milling cutter tool correction (length, radius) means a spatial offset (by L3 and L1) of the TCP along a correction vector $\vec{l}_t$ from TCP_1 to TCP_0 (positive L3 value extends the tool).

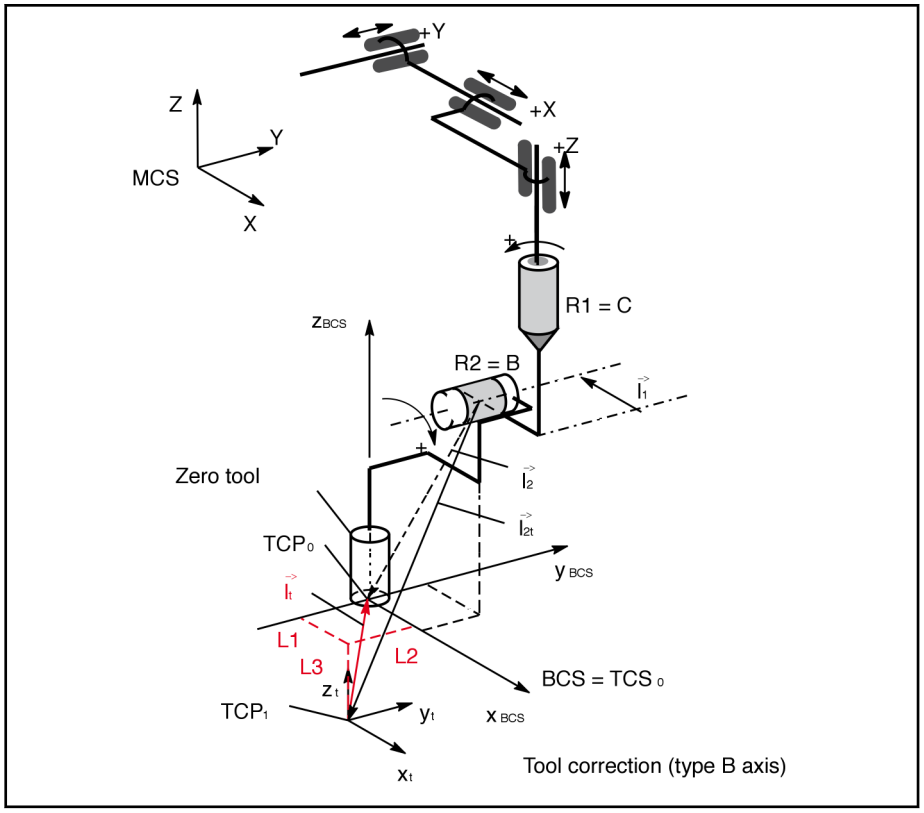


Fig. 11-36: Tool correction: type b-axis

11.10.6 Handling instruction: 5-axis transformation LLLRR

Applying

The axis transformation can be applied via the relevant machine parameters. Therefore, the axis transformation function has to be activated via the parameter **tra** "Axis transformations" in the setup (SUP). The path of the machine parameter name is always "TRA/AxTrafo[1..20]/...".

Alternatively, the axis transformation can be applied via a user-defined system date of type SysAxTrafo_t.

The content of both variants is identical.

The axis positions of the reference position have to be determined.

The respectively equivalent machine parameter numbers, machine parameter names and SD elements are specified in the following table. To apply a transformation, one of the variants has to be assigned accordingly.

Transformation-relevant parameters

MP number	Description	
1030 00110	**Transformation type** MP name: TrafoType SD elem.: Type	
	3232201	LLLRR with vector orientation
	3232101	LLLRR with linear orientation
	3032001	LLLRR with linear motion of the rotary axes

MP number	Description	
1030 00120	**System axes/coordinates of the transformation** MP name: FwdInCoordIndTrafo[1..8] SD elem.: AxisAssignment[1..8]	
	[1]	System axis number of X
	[2]	System axis number of Y
	[3]	System axis number of Z
	[4]	System axis number of C
	[5]	System axis number of B
	[6..8]	Not relevant

<table>
<tr><th>MP number</th><th>Description</th></tr>
<tr><td>1030 00125</td><td>
Axis classification of transformation axes
MP name: CoordClass[1..8]
MP name: CoordDir[1..8]
SD elem.: AxisClassification[1..8]
</td></tr>
<tr><td></td><td>
Defines the axes involved in transformation, including their direction of motion.
• MP no. or SD element:
Value ">0" means:
The axis rotates/travels positively as in the figures above.
Value "<0" means:
The axis rotates/travels negatively as in the figures above.
Possible values are:
– +/-1 = x-axis
– +/-2 = y-axis
– +/-3 = z-axis
– +/-100 = a-axis
– +/-200 = b-axis
– +/-300 = C-axis
• MP name:
– Possible values for CoordClass are
– X,Y,Z (linear)
– A,B,C (rotary)
– Possible values for CoordDir are:
– positive (regarding fig. "Reference position"
– negative (regarding fig. "Reference position"
</td></tr>
<tr><td></td><td>If a parameter is not set, the following default settings apply:</td></tr>
<tr><td></td><td>
<table>
<tr><td></td><td>MP no /
SD elem.:</td><td>CoordClass:</td><td>CoordDir:</td></tr>
<tr><td>[1]</td><td>1</td><td>X</td><td>Positive</td></tr>
<tr><td>[2]</td><td>2</td><td>Y</td><td>Positive</td></tr>
<tr><td>[3]</td><td>3</td><td>Z</td><td>Positive</td></tr>
<tr><td>[4]</td><td>300</td><td>C</td><td>Positive</td></tr>
<tr><td>[5]</td><td>200</td><td>B</td><td>Positive</td></tr>
<tr><td>[6..8]</td><td>Not relevant</td><td>0</td><td>Positive</td></tr>
</table>
</td></tr>
</table>

MP number	Description	
1030 00130	**Axis positions of the reference position** MP name: RefPosTrafo[1..8] SD elem.: AxZeroPos[1..8]	
	[1]	X^0
	[2]	Y^0
	[3]	Z^0
	[4]	C^0
	[5]	B^0
	[6..8]	Not relevant
1030 00140	**Length and angle parameters** MP name: JointParTrafo[1..16] SD elem.: LenParam[1..16]	
	Defines the length vectors l_1, l_2 in [mm].	
	[1]	l_{1x}
	[2]	Not relevant
	[3]	Not relevant
	[4]	l_{2x}
	[5]	l_{2y}
	[6]	l_{2z}
	[7..16]	Not relevant
1030 00160	**Epsilon environments** MP name: EpsAxTrafo SD elem.: EpsilonRanges	
	0.01	

Tab. 11-41: Transformation-relevant parameters

Activating

If the 5-axis transformation is configured via

- machine parameter, the transformation is enabled in the NC program with (i=1,..20):

  COORD(<i>)

- system date, the transformation is enabled in the NC program with (type "SysAxTrafo_t"):

  COORD(<SD.MtbAxTrafo>)

Example program:

The 5-axis transformation is defined in the fourth machine parameter block. No axis transformation is active at programming start:

Enable transformation

```
N100  G1 F1000 XA10 YA10 C30   ;Axes XA, YA and C are traversed
N110  COORD(4)                 ;Switching on the axis transformation
N130  x20 y30 phi0             ;the coordinates x,y,phi are traversed
N140  x10 z10 theta45          ;the coordinates x,z,theta are traversed
        :
```

```
N999  COORD(0)                ;Switching-off the axis transformation
                              ;Axes are again programmed from here
```

Deactivating **COORD(0)** or **COORD(0,2)**

switches off the axis transformation in the NC program.

11.10.7 Variant 4: 5-axis transformation LLLAB

Function

Transformation name	Transformation type	Point of action
LLLAB		
Vector orientation	3232201	AT2
Linear orientation	3232101	AT2

The kinematics of the 5-axis transformation LLLAB consists of the three linear axes x, y and z and the two rotary axes R_1 and R_2. The transformation is used at the point of action 2, i.e. it is applied between the machine coordinate system MCS = MCS = (X,Y,Z,R_1,R_2) and the basic coordinate system BCS = (x, y, z, phi, theta).

- The x, y, z, phi, theta BCS input variables are programmed
- The MCS output variables X, Y, Z, R_1, R_2 are calculated

The linear axes form a right-handed coordinate system. The rotary axis R_1 (type A-axis) rotates the tool around the X-axis and the rotary axis R_2 (type B-axis) swivels the tool around the Y-axis (see the following figure).

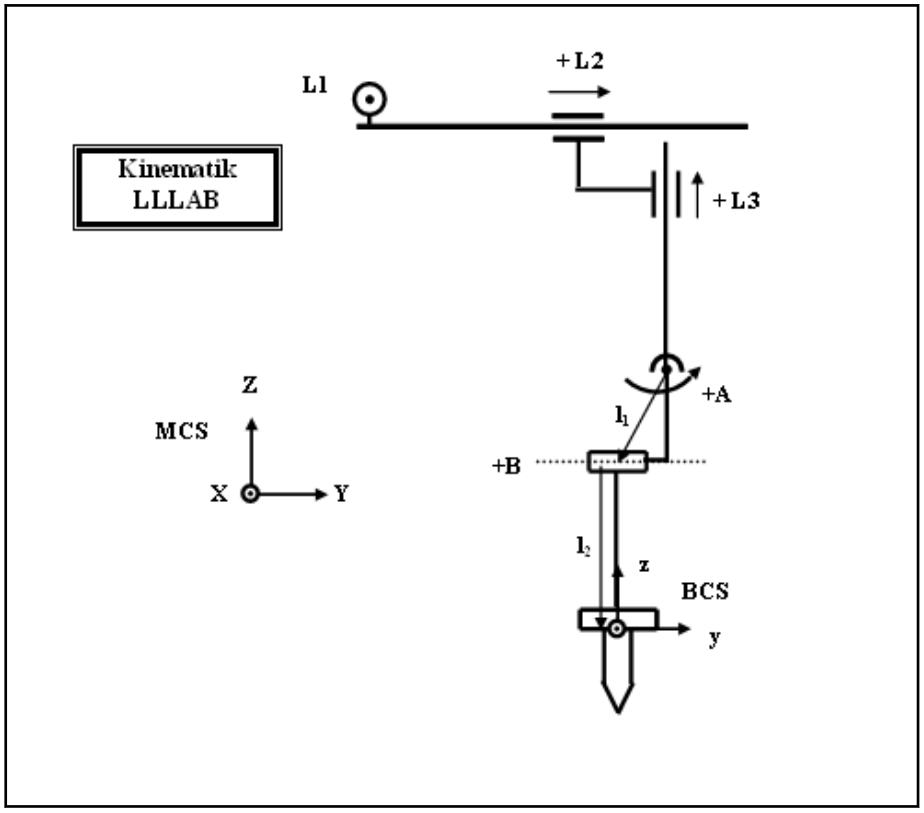


Fig. 11-37: 5-axis transformation LLLAB function

Reference position

In the reference position, the tool holder (spindle nose) in the zero point of the basic coordinate system (BCS) and is aligned along the z-coordinate, i.e. the tool points down vertically (cf. figure).

In this position, the BCS coordinates (x, y, z, phi, theta) = (0,0,0,0,0). To achieve this, the MCS coordinates (X^0,Y^0,Z^0,R_1^0,R_2^0) as reference position in this position have to be entered in the machine parameters (see below).

The figure shows the positive directions of rotation/traversing of the axes. Set them accordingly. In the reference position, the A-axis rotates around the x-coordinate and the B-axis rotates around the y-coordinate.

The vectors l_1 and l_2 define the geometry of the 5-axis kinematics. They have the form

$$\vec{l}_1 = \begin{pmatrix} 0 \\ l_{1y} \\ l_{1z} \end{pmatrix} \quad \text{und} \quad \vec{l}_2 = \begin{pmatrix} l_{2x} \\ l_{2y} \\ l_{2z} \end{pmatrix}$$

Fig. 11-38: Form of vectors l_1 and l_2

l_1 is the vector from the A-axis to the B-axis.

l_2 is the l_1 continuation vector from the B-axis to the zero point of BCS.

Orientation

When switching on the 5-axis transformation LLLAB, either the vector orientation movement (type 3232202) or the linear orientation movement (type 3232102) is activated. There are different properties depending on the orientation type.

Vector orientation

For the vector orientation mode, each change in orientation by programming of phi, theta or a corresponding optional syntax (see spec. rotation of the orientation vector) around a de-spun rotary axis, large circle movement of the vector point.

Phi and theta can assume values from the following ranges:

0≤phi<360 ; 0≤theta<180

The rotary axes can be endless or rotary axes.

Linear orientation

The angles phi and theta are interpolated linearly from the starting to the end point. The behavior of the "shortest way"-logic depends on the programming start and the orientation motion. In case of the phi and theta programming, the "shortest way" logic is switched off. In case of the O(...) programming, the "shortest way" logic is switched off.

Phi and theta can assume values from the following ranges:

-indefinite<phi<definite ; -180≤theta<180

The rotary axes have to be configured as rotary axes (no modulo endless axis).

☞ The linear orientation is not suitable for the "inclined plane". Programmed orientations always refer to the basic coordinate system BCS and not - as usual - to the workpiece coordinate system WCS.

Special features

B-axis limitation

The traversing range of the B-axis is limited to $|B| \leq 90^0$ degrees. With this limitation, the backwards transformation can be solved uniquely.

Tool correction

Axis transformation supports a tool correction in the tool coordinate system TCS. In the reference position, BCS corresponds to TCS_0 (spindle nose). The tool correction vector $_t$ points from the corrected tool center point TCP_1 to the uncorrected TCP_0. It comprises the length corrections L1, L2 and L3.

Restrictions

An online correction is exclusively possible for x, y, z not for phi, theta.

The traversing range of the B-axis is limited to $|B| \leq 90^0$ degrees.

Relevant NC functions

COORD(...)	The 5-axis transformation LLLAB is enabled. This can optionally be done using the machine parameter block (i=1..20) or via the MTB system data (of type SysAxTrafo_t).
COORD(0) - or - COORD(0,2)	The active transformation is deactivated.

Tab. 11-42: Relevant NC functions

Relevant machine parameters

Up to 20 different axis transformations are declared with the following machine parameter block in the machine parameters.

1030 00110	Defines the transformation type
1030 00120	Defines the system axes/coordinates participating in the transformation
1030 00125	Defines the axis classification of the transformation axes
1030 00130	Defines the axis positions of the reference position
1030 00140	Defines the length and angle parameters
1030 00160	Epsilon environments

Tab. 11-43: Relevant machine parameters

Relevant system data

The axis transformation can also be defined via system date instead of the machine parameters. Therefore, a system variable of type "SysAxTrafo_t" has to be created (see "MTX Machine Parameters", chapter "System data"). The structure type "SysAxTrafo_t" is control-internally defined in the xml schema file "/feprom/schemas/sdaxtrf.xsd" and contains - analog to the relevant machine parameters - the following transformation-relevant structure elements.

Structural element	Description
Type	Defines the transformation type
AxisAssignment	Defines the system axes/coordinates participating in the transformation

AxisClassification	Defines the axis classification of the transformation axes
AxZeroPos	Defines the axis positions of the reference position
LenParam	Defines the length and angle parameters
EpsilonRanges	Epsilon environments

Tab. 11-44: Relevant system data

Handling Instruction: 5-axis transformation LLLAB

Applying

The axis transformation can be applied via the relevant machine parameters. Therefore, the axis transformation function has to be activated via the parameter **tra** "Axis transformations" in the setup (SUP). The path of the machine parameter name is always "TRA/AxTrafo[1..20]/...".

Alternatively, the axis transformation can be applied via a user-defined system date of type SysAxTrafo_t.

The content of both variant is identical.

The axis positions of the reference position have to be determined.

The respectively equivalent machine parameter numbers, machine parameter names and SD elements are specified in the following table. To apply a transformation, one of the variants has to be assigned accordingly

Transformation-relevant parameters

MP number	Description	
1030 00110	**Transformation type** MP name: TrafoType SD elem. : Type	
	3232201	5-axis transformation LLLAB with vector orientation
	3232101	5-axis transformation LLLAB with linear orientation
1030 00120	**System axes / coordinates of the transformation** MP name: FwdInCoordIndTrafo[1..8] SD elem. : AxisAssignment[1..8]	
	[1]	System axis number of X
	[2]	System axis number of Y
	[3]	System axis number of Z
	[4]	System axis number of A
	[5]	System axis number of B
	[6..8]	Not relevant

<table>
<tr><th>MP number</th><th>Description</th></tr>
<tr><td>1030 00125</td><td>
Axis classification of transformation axes
MP name: CoordClass[1..8]
MP name: CoordDir[1..8]
SD elem. : AxisClassification[1..8]

Defines the axes involved in the transformation, including their direction of motion
• MP no. or SD element:
Value ">0" means:
The axis rotates/travels positively as in the figures above.
Value "<0" means:
The axis rotates/travels negatively as in the figures above.
Possible values are:
– +/-1 = x-axis
– +/-2 = y-axis
– +/-3 = z-axis
– +/-100 = a-axis
– +/-200 = b-axis
–
• MP name:
– Possible values for CoordClass are
– X,Y,Z (linear)
– A,B,C (rotary)
– Possible values for CoordDir are:
– positive (regarding Fig. "Reference position"
– negative (regarding Fig. "Reference position"

If the traversing directions do not comply with the figure, set the parameters as follows:
<table>
<tr><th></th><th>MP no /
SD elem.:</th><th>CoordClass:</th><th>CoordDir:</th></tr>
<tr><td>[1]</td><td>1</td><td>X</td><td>Positive</td></tr>
<tr><td>[2]</td><td>2</td><td>Y</td><td>Positive</td></tr>
<tr><td>[3]</td><td>3</td><td>Z</td><td>Positive</td></tr>
<tr><td>[4]</td><td>100</td><td>A</td><td>Positive</td></tr>
<tr><td>[5]</td><td>200</td><td>B</td><td>Positive</td></tr>
<tr><td>[6..8]</td><td>Not relevant</td><td>0</td><td>Positive</td></tr>
</table>
</td></tr>
</table>

MP number	Description	
1030 00130	**Axis positions of the reference position** MP name: RefPosTrafo[1..8] SD elem. : AxZeroPos[1..8]	
	[1]	X^0
	[2]	Y^0
	[3]	Z^0
	[4]	A^0
	[5]	B^0
	[6..8]	Not relevant
1030 00140	**Length and angle parameters** MP name: JointParTrafo[1..16] SD elem. : LenParam[1..16]	
	Defines the length vectors l_1, l_2 in [mm].	
	[1]	Not relevant
	[2]	l_{1y}
	[3]	l_{1z}
	[4]	l_{2x}
	[5]	l_{2y}
	[6]	l_{2z}
	[7..16]	Not relevant
1030 00160	**Epsilon environments** MP name: EpsAxTrafo SD elem. : EpsilonRanges	
	Not relevant	

Tab. 11-45: Transformation-relevant parameters

Activating

If the 5-axis transformation is configured via

- Machine parameters. Then, the transformation is enabled in the NC program with (i=1,..20):

  COORD(<i>)

- System date. Then, the transformation is enabled in the NC program with (type "SysAxTrafo_t"):

  COORD(<SD.MtbAxTrafo>)

Example Program:

The 5-axis transformation is defined in the fourth machine parameter block. No axis transformation is active at programming start:

Enable transformation

```
N100  G1 F1000 XA10 YA10 A30   ;Axes XA,YA and A are traversed
N110  COORD(4)                 ;Activating axis transformation
N130  x20 y30 phi0             ;Coordinates x,y,phi are traversed
N140  x10 z10 theta45          ;Coordinates x,z,theta are traversed :
N999  COORD(0)                 ;Deactivating axis transformation
                               ;from here, axes are programmed
```

Deactivating **COORD(0)** or **COORD(0,2)**
switches off the axis transformation in the NC program.

11.11 5-axis transformation - Type 2: RLLLR (with rotary table)

11.11.1 Description

Function

Transformation name	Transformation type	Point of action	Coordinate system
5-axis transformation with rotary table			
Vector orientation	3232202	AT2	MCS, BCS
Linear orientation	3232102	AT2	MCS, BCS

Tab. 11-46: Function

Output variables in the MCS:

X,Y,Z,R2,R1

Input variables in the BCS:

x, y, z, φ, ϑ

The distributed 5-axis transformation (RLLLR) has three linear axes (X, Y, Z) and two rotary axes (R1, R2).

The rotary axes R1 and R2 can be endless axes or rotary axes.

It is used at point of action 2, i.e. it acts between the machine coordinate system MCS = (X,Y,Z,R2,R1) and the basic coordinate system BCS = (x, y, z, φ, ϑ).

The three linear axes form a right-handed coordinate system. The rotary axis R_2 swivels the tool and the rotary axis R_1 rotates the workpiece (mounted on the axis).

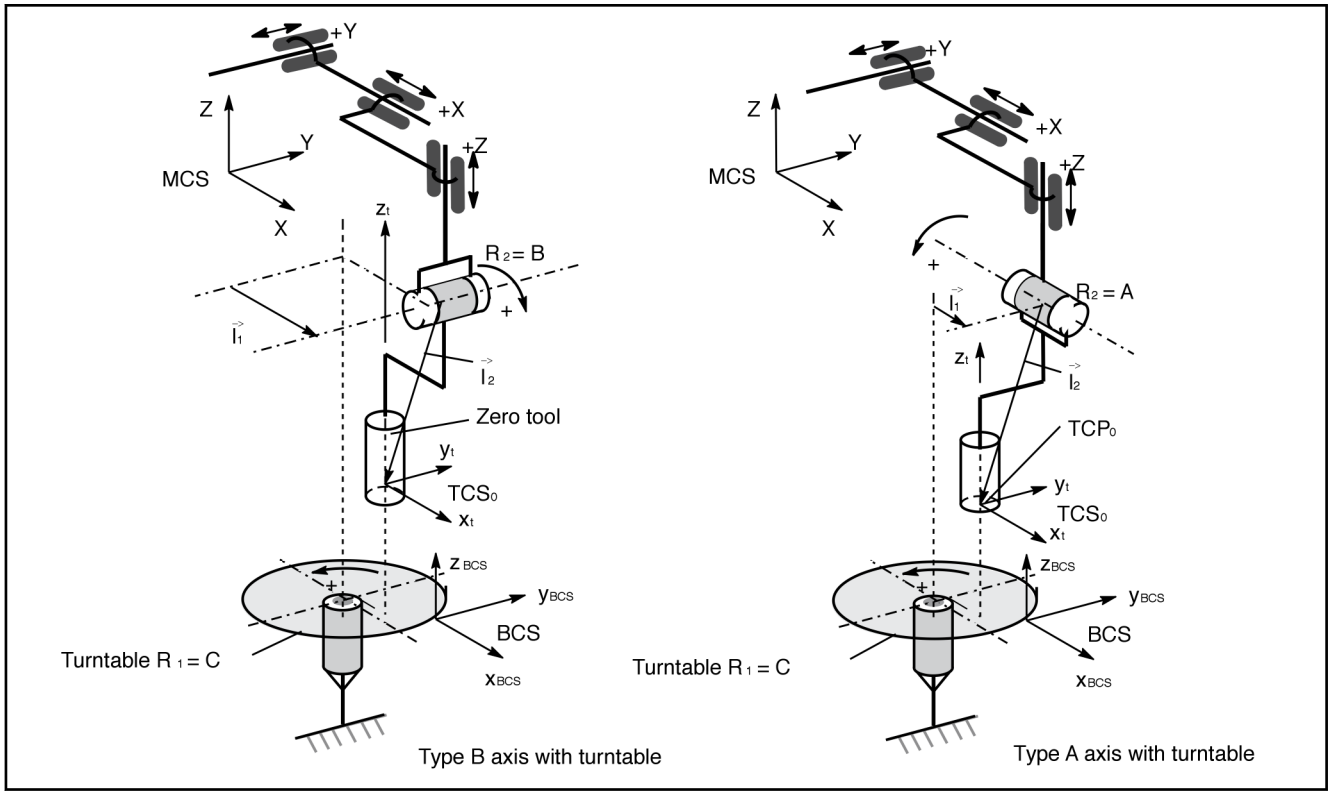


Fig. 11-39: RLLLR (with Rotary Table)

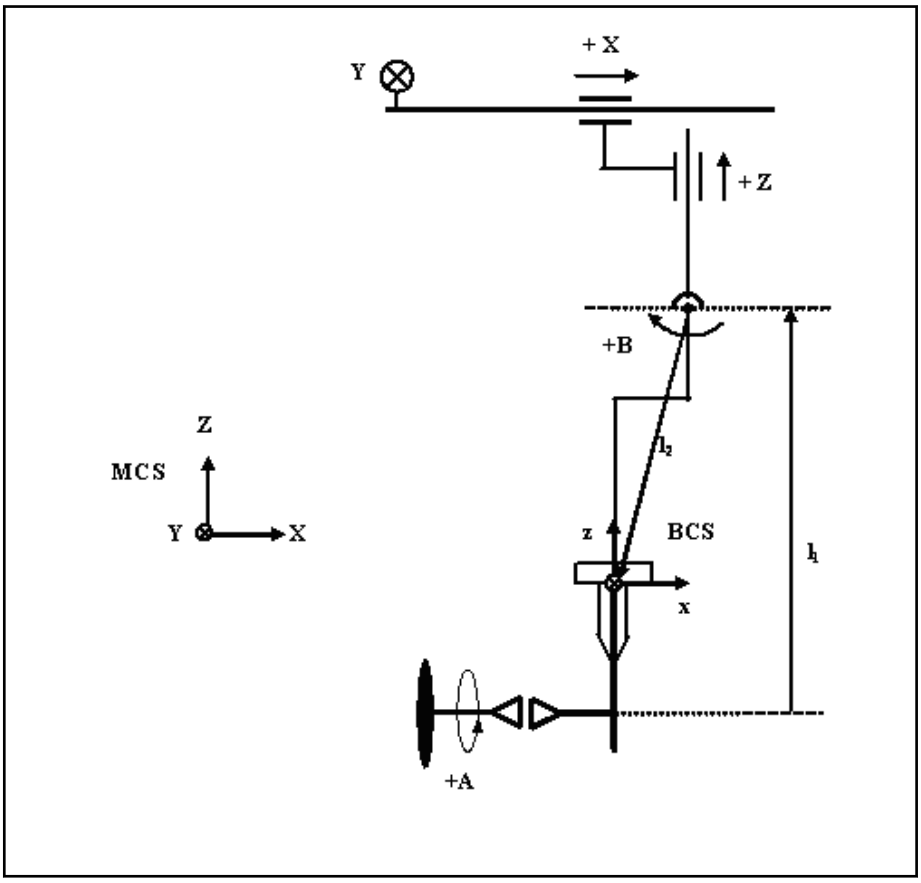


Fig. 11-40: ALLLB

The vectors $\vec{l}_1$ and $\vec{l}_2$ define the geometry of the 5-axis kinematics. $\vec{l}_1$ is a vector from the R_1-axis (C-/A axis) to the R_2-axis (B-/A-axis), i.e. R_1=C $\vec{l}_1$ is a vector along x_{BCS} and in case R_1=A, along z_{BCS}. $\vec{l}_2$ is the continuation vector from the R2 axis (B-/A-axis) to the zero point of the BCS (in the reference position).

There are **three different variants** of the distributed 5 axis transformation RLLLR. These are distinguished by the rotary axis type (the type depends on which BCS-coordinates x, y, z the rotary axis rotates in the reference position):

- **R_1 = C-axis, R_2= A-axis:**

  R_1 rotates around the z-coordinate in the reference position and R_2 rotates around the x-coordinate in the reference position.

- **R_1 = C-axis, R_2 = B-axis:**

  R_1 rotates around the z-coordinate in the reference position and R_2 rotates around the y-coordinate in the reference position.

- **R_1 = A-axis, R_2 = B-axis:**

  R_1 rotates around the x-coordinate in the reference position and R_2 rotates around the y-coordinate in the reference position.

Apart from these axis arrangements, the following **subvariants** also exist. The rotary table R_1 is mounted on the linear axis X or Y for these subvariants. The subvariants are always assigned to one of the three variants mentioned above and differ only in the traveling direction of the R_1-moving linear axis.

The following figure exemplarily shows the variant R_1=C-axis on x-axis, R_2=B-axis is shown. As the rotary table R_1 moves on the linear x-axis, the positive traversing direction of the x-axis executes a U turn.

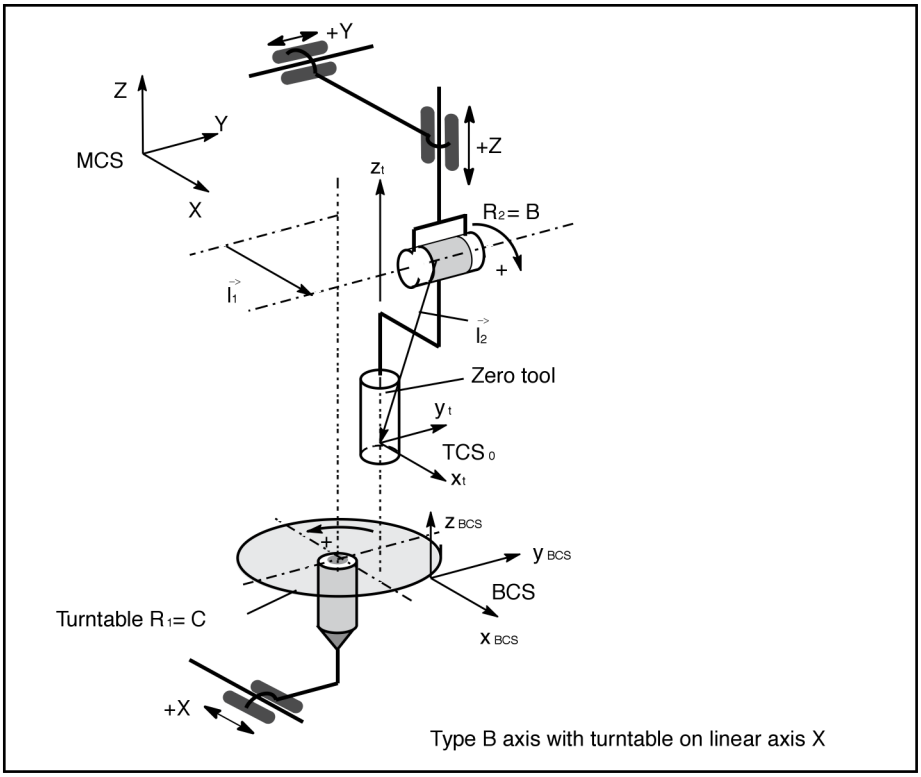


Fig. 11-41: Type b-axis with rotary table on linear axis X

Axis transformation properties:

When the 5-axis transformation is active, the coordinates x, y, z, φ, θ of the basic coordinate system BCS are transformed into the machine coordinate system MCS.

Input variables are the linear coordinates x, y, z and the orientation coordinates φ, θ. When switching on the axis transformation, either the vector orientation movement (type 3232202) or the linear orientation movement (type 3232102) is activated.

The orientation vector $\vec{\rho}$ can be programmed as spline (see chapter 9.1.8 "Spline programming " on page 143).

There are different properties depending on the orientation type

Vector orientation:

In case of the vector orientation motion, every orientation change is caused by programming φ, θ or a respective alternative syntax (s. spec. orientation motion) by rotating the orientation vector $\vec{\rho}$(φ,θ) around a rotary axis which is fixed in space (large circle motion of the vector point). The rotary axes can be endless or rotary axes.

Linear orientation:

The angles φ and θ are interpolated linearly from the starting to the end point. The behavior of the "shortest way"-logic depends on the programming start and the orientation motion. In case of the φ-, θ-programming, the "shortest way"-logic is switched off. In case of the O(...)-programming, the "shortest way" is switched on. φ, θ can be provided with values from the following ranges:

φ ε [-infinite, +infinite] , θ ε [-180, +180].

The rotary axes have to be configured as rotary axes (no modulo endless axis).

Restriction

Online correction of φ and ϑ is not possible.

Relevant NC functions

COORD(<i>)	i=1..20 is used to activate a 5-axis transformation which is defined in MP 1030 00110.
COORD(0,2) - or - COORD(0)	The active transformation is deactivated.

Tab. 11-47: Relevant NC functions

Relevant CPL functions

SDR(300)	Active Epsilon environment. The active value can be greater than the value set in EpsAx-Trafo (MP 1030 00160).

Tab. 11-48: Relevant CPL functions

Relevant machine parameters

1030 00110	Defines the transformation type.
1030 00120	Defines the system axes/coordinates participating in the transformation.
1030 00125	Defines the axis classification of the transformation axes.
1030 00130	Defines the axis positions of the reference position.
1030 00140	Defines the length and angle parameters.
1030 00160	Epsilon environments

Tab. 11-49: Relevant machine parameters

Reference position

In the reference position of the following configuration (in the MCS: X, Y, Z, R1 = C and R2 = B), the point (TCP) of the rotation-symmetric tool (cutter or laser), based on the basic coordinate system (BCS) is oriented along the z-coordinate. The axis transformation unambiguously determines the orientation of the BCS.

In this position, the BCS- coordinates (x,y,z,φ,ϑ) have to be (0,0,0,0,0), with the respective MCS-coordinates (X, Y, Z, R2, R1) = (X^0, Y^0, Z^0, $R2^0$, $R1^0$) of MP 1030 00130.

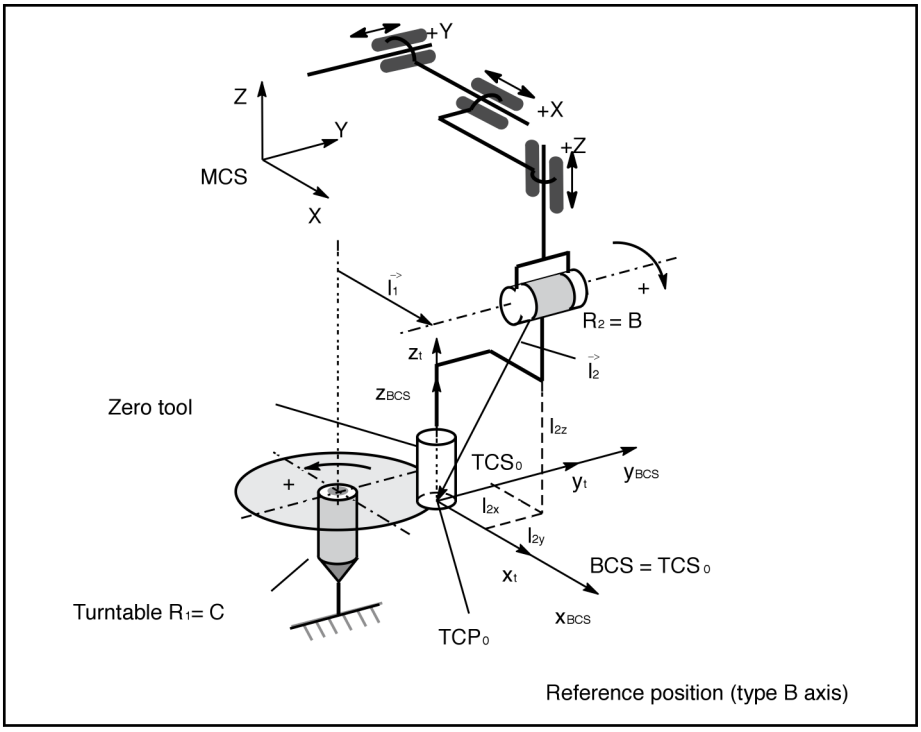


Fig. 11-42: Reference position

TCS, TCP

TCS stands for Tool Coordinate System.

TCP_0/TCP_1 refers to the Tool Center Point **without/with** correction.

Tool correction

Axis transformation supports a tool correction in the tool coordinate system (TCS). In reference position, BCS conforms with TCS_0. The tool correction vector $\vec{l}_t$ points from the corrected tool center point TCP_1 to the uncorrected TCP_0. It comprises the length corrections L1, L2 and L3 from the D-correction table.

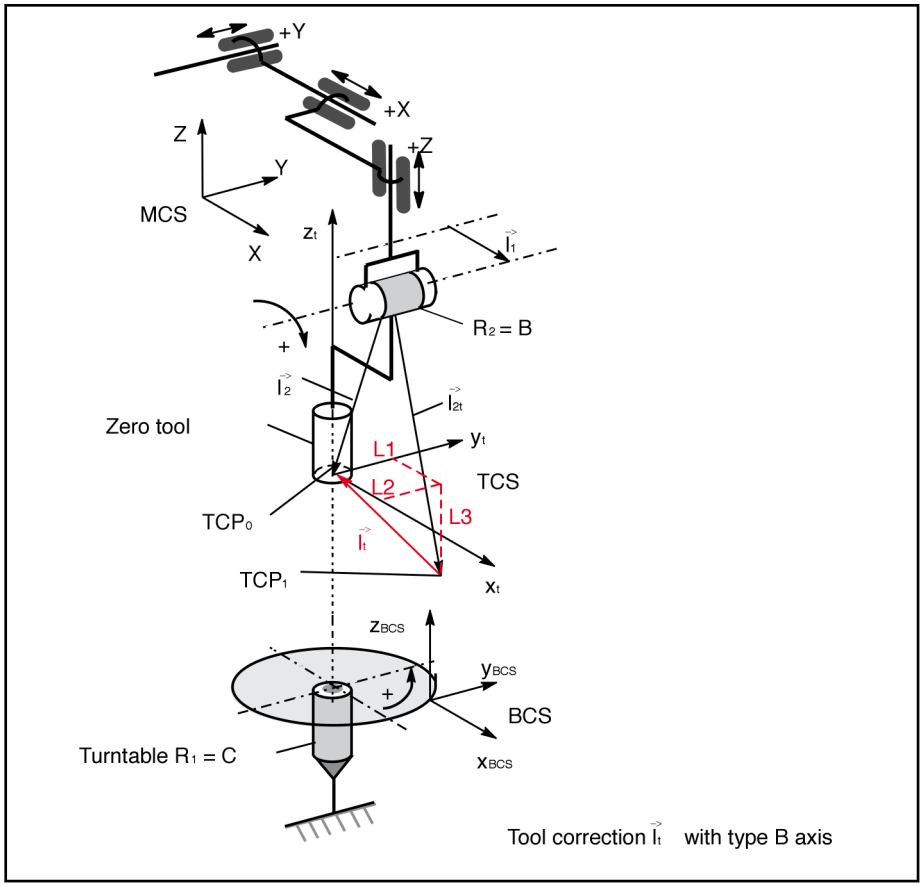


Fig. 11-43: Tool correction

Special features

- **Ambiguous axis positions:**

  Generally, a TCP position and a tool orientation can be realized by two different axis poses (transformation branches). The branches differ in their b-axis positions (or a-axis position).

  Vector orientation:

  Which one of the two poses (transformation branches) is used is decided on activation of the axis transformation. Accordingly, a transition from branch 1 to -1 and vice versa is only possible when traveling through the orientation location θ =0 (north pole singularity).

  Linear orientation:

  The transformation branch is always chosen in a way that the b-axis (a-axis) and the orientation angle θ have the same sign, i.e. the following applies:

  – θ≥0 ≙ B≥0 (branch 1)
  – θ≤0 ≙ B≤0 (branch -1)

  Therefore, a backwards transformation can be solved uniquely. Accordingly, a transition from branch 1 to -1 and vice versa is possible when traveling through the orientation location θ =0 (north pole singularity).

- **North pole singularity:**

  Vector orientation:

  The north pole is a singularity.

In the north pole, the machine parameter 103000160 is effective in the Epsilon environments; i.e. orientation motions through the ε-environment are lead through θ =0.

If θ =0, φ can be programmed, but no motion takes place.

Linear orientation:

The north pole is no singularity.

The north pole can be traveled without problems. The machine parameter 103000160 Epsilon environments is not effective, i.e. orientation motions through the ε-environment are not imperatively lead through point θ =0.

If θ =0, φ can be programmed and the desired motion (rotation of the c-axis) is performed.

11.11.2 Handling Instruction: 5-Axis Transformation RLLLR

Applying

The axis transformation can be applied via the relevant machine parameters. Therefore, the axis transformation function has to be activated via the parameter **tra** "Axis transformations" in the setup (SUP). The path of the machine parameter name is always "TRA/AxTrafo[1..20]/...".

Alternatively, the axis transformation can be applied via a user-defined system date of type SysAxTrafo_t.

The content of both variant is identical.

The axis positions of the reference position have to be determined.

The respectively equivalent machine parameter numbers, machine parameter names and SD elements are specified in the following table. To apply a transformation, one of the variants has to be assigned accordingly

Transformation-relevant parameters

MP number	Description	
1030 00110	**Transformation type** MP name: TrafoType SD elem. : Type	
	3232202	RLLLR with vector orientation
	3232102	RLLLR with linear orientation
1030 00120	**System axes / coordinates of the transformation** MP name: FwdInCoordIndTrafo[1..8] SD elem. : AxisAssignment[1..8]	
	[1]	System axis number of R_1 (c-/a-axis)
	[2]	System axis number of X
	[3]	System axis number of Y
	[4]	System axis number of Z
	[5]	System axis number of R_2 (a-/b-axis)
	[6..8]	Not relevant

<table>
<tr><th>MP number</th><th>Description</th></tr>
<tr><td>1030 00125</td><td>
Axis classification of transformation axes
MP name: CoordClass[1..8]
MP name: CoordDir[1..8]
SD elem. : AxisClassification[1..8]

Defines the axes involved in transformation, including their direction of motion.
• MP no. or SD element:
Value ">0" means:
The axis rotates/travels positively as in the figures above.
Value "<0" means:
The axis rotates/travels negatively as in the figures above.
Possible values are:
– +/-1 = x-axis
– +/-2 = y-axis
– +/-3 = z-axis
– +/-100 = a-axis
– +/-200 = b-axis
– +/-300 = c-axis
• MP name:
– Possible values for CoordClass are
– X,Y,Z (linear)
– A,B,C (rotary)
– Possible values for CoordDir are:
– positive (regarding Fig. "Reference position"
– negative (regarding Fig. "Reference position"

If a parameter is not set, the following default settings apply:
<table>
<tr><th></th><th>MP no /
SD elem.:</th><th>CoordClass:</th><th>CoordDir:</th></tr>
<tr><td>[1]</td><td>300</td><td>C</td><td>Positive</td></tr>
<tr><td>[2]</td><td>1</td><td>X</td><td>Positive</td></tr>
<tr><td>[3]</td><td>2</td><td>Y</td><td>Positive</td></tr>
<tr><td>[4]</td><td>3</td><td>Z</td><td>Positive</td></tr>
<tr><td>[5]</td><td>200</td><td>B</td><td>Positive</td></tr>
<tr><td>[6..8]</td><td>Not relevant</td><td>0</td><td>Positive</td></tr>
</table>
</td></tr>
</table>

MP number	Description	
1030 00130	**Axis positions of the reference position** MP name: RefPosTrafo[1..8] SD elem. : AxZeroPos[1..8]	
	[1]	R_1^0
	[2]	X^0
	[3]	Y^0
	[4]	Z^0
	[5]	R_2^0
	[6..8]	Not relevant
1030 00140	**Length and angle parameters** MP name: JointParTrafo[1..16] SD elem. : LenParam[1..16]	
	Defines the length vectors l_1, l_2 in [mm].	
	[1]	l_{1x}
	[2]	l_{1y}
	[3]	0
	[4]	l_{2x}
	[5]	l_{2y}
	[6]	l_{2z}
	[7..16]	Not relevant
1030 00160	**Epsilon environments** MP name: EpsAxTrafo SD elem. : EpsilonRanges	
	0.01	

Tab. 11-50: Transformation-relevant parameters

Activating If the 5-axis transformation is configured via

- Machine parameters. Then, the transformation is enabled in the NC program with (i=1,..20):

  COORD(<i>)

- System date. Then, the transformation is enabled in the NC program with (type "SysAxTrafo_t"):

  COORD(<SD.MtbAxTrafo>)

Example Program:

The 5-axis transformation is defined in the fourth machine parameter block. No axis transformation is active at programming start:

Enable transformation

```
N100  G1 F1000 XA10 YA10 C30   ;Axes XA, YA and C are traversed
N110  COORD(4)                 ;Switching on the axis transformation
N130  x20 y30 phi0             ;the coordinates x,y,phi are traversed
N140  x10 z10 theta45          ;the coordinates x,z,theta are traversed
        :
```

```
N999  COORD(0)                ;Switching-off the axis transformation
                              ;Axes are again programmed from here
```

Deactivating **COORD(0)** or **COORD(0,2)**

switches off the axis transformation in the NC program.

11.12 5-axis transformation - Type 3: RRLLL(with swivel/rotary table)

11.12.1 Description

Function

Transformation name	Transformation type	Point of action	Coordinate system
5-axis transformation with swivel-ing/rotary table			
Vector orientation	3232203	AT2	MCS, BCS
Linear orientation	3232103	AT2	MCS, BCS

Tab. 11-51: Function

Coordinates in the machine coordinate system (MCS): R1, R2, X, Y, Z

Coordinates in the basic coordinate system BCS: x, y, z, φ, ϑ

The distributed 5-axis transformation (RRLLL) has three linear axes (X, Y, Z) and two rotary axes (R1, R2).

This transformation is used at point of action 2. Thus, the transformation is active between the MCS = (R1, R2, X, Y, Z) and the BCS = (x, y, z, φ, ϑ).

The rotary axes R1 and R2 can be operated as endless or rotary axes.

The tool is moved via the linear axes X, Y, Z. The tool longitudinal axis is parallel to the Z-axis. The workpiece table (rotary-swivel table) containing the workpiece is positioned opposite the workpiece by the rotary axes R1 and R2.

There are 3 variants:

- Variant CBLLL: The BCS connected to the workpiece table is rotated by rotary axis R1 = C-axis. The workpiece table together with the C-axis is swiveled by the B-axis (parallel to the MCS-Y-axis). The C-axis is in reference position, parallel to the MCS-Z-axis but is swiveled by the B-axis from this orientation.
- Variant CALLL: The BCS connected to the workpiece table is rotated by rotary axis R1 = C-axis. The workpiece table together with the C-axis is swiveled by the A-axis (parallel to the MCS-X-axis). The C-axis is in reference position, parallel to the MCS-Z-axis but is swiveled by the A-axis from this orientation.
- Variant BALLL: The BCS connected to the workpiece table is rotated by rotary axis R1 = B-axis. The workpiece table together with the B-axis is swiveled by the A-axis (parallel to the MCS-X-axis). The B-axis is in reference position, parallel to the MCS-Y-axis but is swiveled by the A-axis from this orientation.

The BCS is solidly connected to the workpiece table. In reference position, the x-direction of the BCS is parallel to the MCS X-axis, the y-direction of the BCS is parallel to the MCS-Y-axis and the z-direction of the BCS is parallel to the MCS-Z-direction.

The vectors $\vec{l}_1$ and $\vec{l}_2$ refer to the reference position. These vectors define the geometry of the 5-axis kinematics for the different variants as follows:

Variant CBLLL: $\vec{l}_1$ is the vector along the MCS-X-axis from the R1-axis (C-axis) to the R2-axis (B-axis) and thus only has the component l_{1x}. $\vec{l}_2$ is the continuation vector of the R2-axis (B-axis) to the zero point of the BCS and has the three components l_{2x}, l_{2y} and l_{2z}.

Variant CALLL: $\vec{l}_1$ is the vector along the MCS-Y-axis from the R1-axis (C-axis) to the R2-axis (A-axis) and thus only has the component l_{1y}. $\vec{l}_2$ is the continuation vector of the R2-axis (A-axis) to the zero point of the BCS and has the three components l_{2x}, l_{2y} and l_{2z}.

Variant BALLL: $\vec{l}_1$ is the vector along the MCS-Z-axis from the R1-axis (B-axis) to the R2-axis(A-axis) and thus only has the component l_{1z}. $\vec{l}_2$ is the continuation vector of the R2-axis (A-axis) to the zero point of the BCS and has the three components l_{2x}, l_{2y} and l_{2z}.

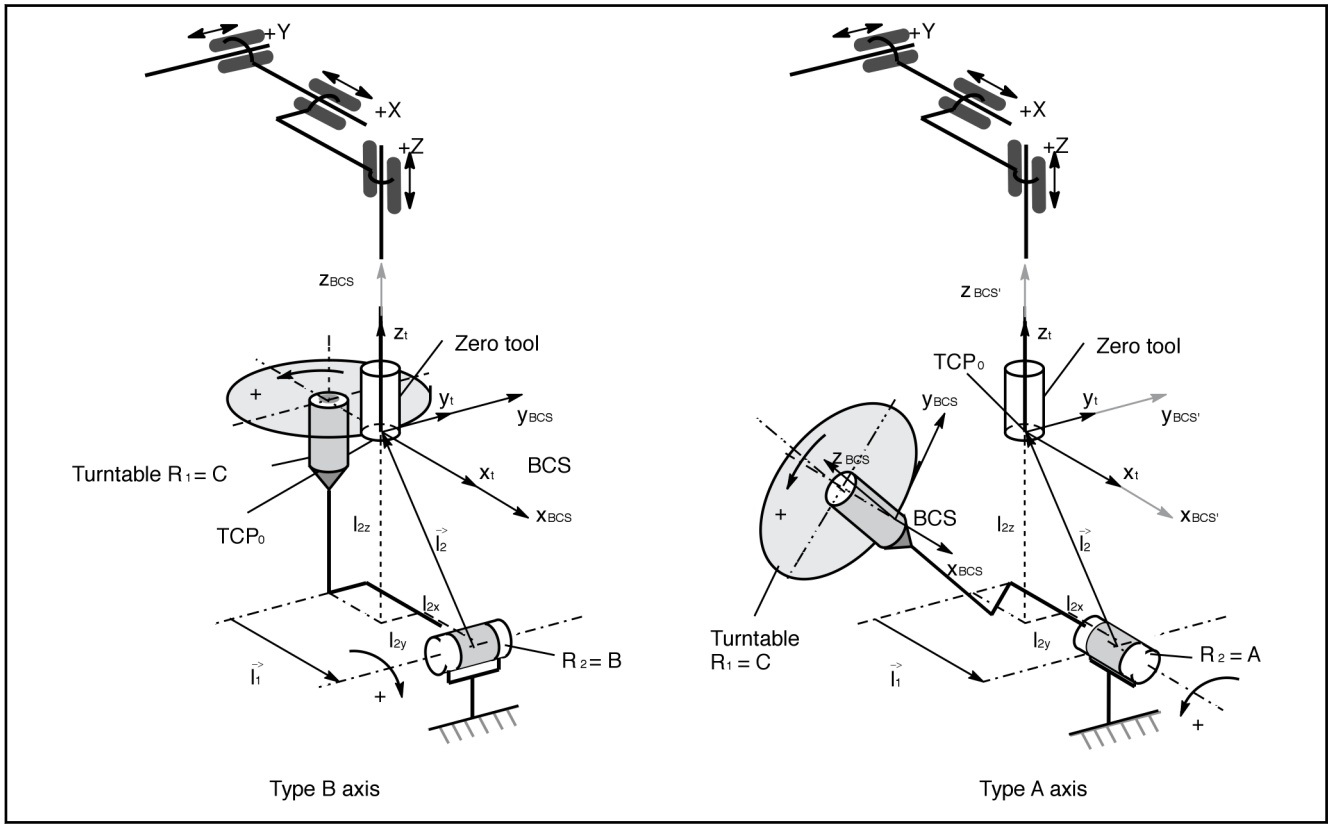


Fig. 11-44: Variants CBLLL and CALLL of the 5-axis transformation RRLLL

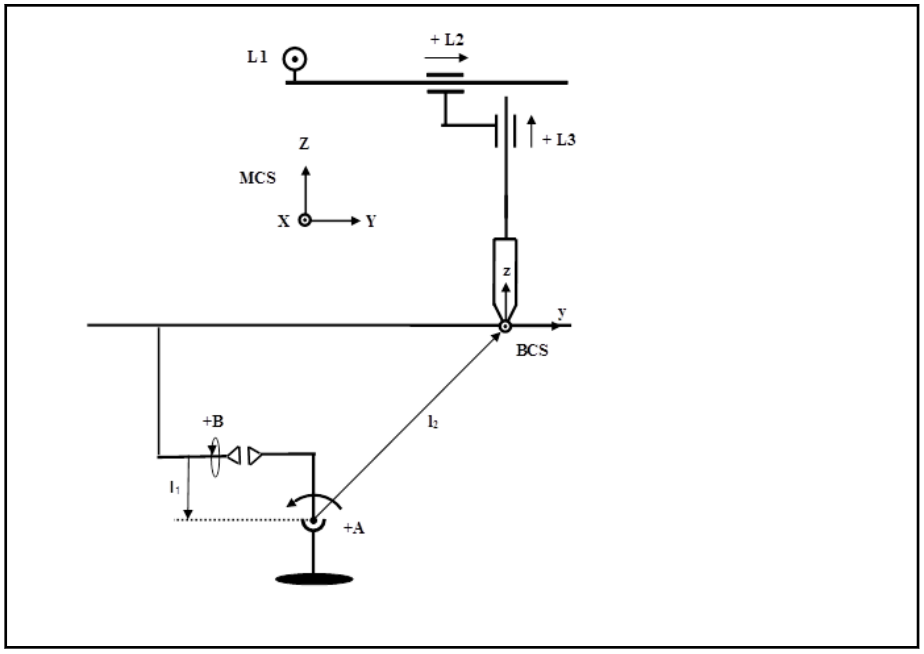


Fig. 11-45: Variant BALLL of the 5-axis transformation RRLLL

Restriction

Online correction of φ and ϑ is not possible.

Relevant NC functions

COORD(<i>)	i=1..20 is used to activate a 5-axis transformation which is defined in MP 1030 00110.
COORD(0,2)	The active transformation is deactivated.

Tab. 11-52: Relevant NC functions

Relevant CPL functions

SDR(300)	Active Epsilon environment. The active value can be greater than the value set in EpsAx-Trafo (MP 1030 00160). The Epsilon environment is not relevant for the BALLL variant.

Tab. 11-53: Relevant CPL functions

Relevant machine parameters (MP)

1030 00110	Defines the transformation type
1030 00120	Defines the system axes/coordinates participating in the transformation.
1030 00125	defines the axis classification of the transformation axes
1030 00130	Defines the axis positions of the reference position
1030 00140	defines the length and angle parameters
1030 00160	Epsilon environment (not relevant for the BALLL variant)

Tab. 11-54: Relevant machine parameters

Axis transformation properties

When enabling the 5-axis transformation, the MCS coordinates R1, R2, X, Y, Z are transformed into the BCS coordinates x, y, z, φ, θ (forward transformation).

When enabling the 5-axis transformation, the BCS coordinates x, y, z, φ, θ are transformed into the MCS coordinates R1, R2, X, Y, Z (backward transformation).

The BCS coordinates are composed of the three linear coordinates x, y, z ad the two orientation coordinates φ, ϑ.

When switching on the axis transformation, either the vector orientation movement (type 3232203) or the linear orientation movement (type 3232103) is activated. However the latter is only valid for the CBLLL and CALLL variants.

The orientation coordinates can also be programmed as spline (see chapter 9.1.8 "Spline programming " on page 143).

There are different properties depending on the orientation type

Vector orientation:

In case of the vector orientation motion, every orientation change is caused by programming φ, θ or a respective alternative syntax (s. spec. orientation motion) by rotating the orientation vector $\vec{o}$ (φ,θ) around a rotary axis which is fixed in space (large circle motion of the vector point). The rotary axes can be endless or rotary axes.

Linear orientation:

The angles φ and θ are interpolated linearly from the starting to the end point. The behavior of the "shortest way"-logic depends on the programming start and the orientation motion. In case of the φ-, θ-programming, the "shortest way"-logic is switched off. In case of the O(...)-programming, the "shortest way" is switched on. φ, θ can be provided with values from the following ranges:

φ ε [-infinite, +infinite] , θ ε [-180, +180].

The rotary axes have to be configured as rotary axes (no modulo endless axis).

Reference position

In the reference position, the tool center point (TCP) of the rotation-symmetrical tool (milling tool or laser) is oriented to the BCS origin along the z-coordinate of the BCS. In this position, the BCS coordinates (x,y,z,φ,ϑ) have to be (0,0,0,0,0). The respective MCS-coordinates (R1,R2,X, Y, Z) are identified with ($R1^0$, $R2^0$, X^0, Y^0, Z^0). Enter them as axis positions of the reference position into MP 1030 00130.

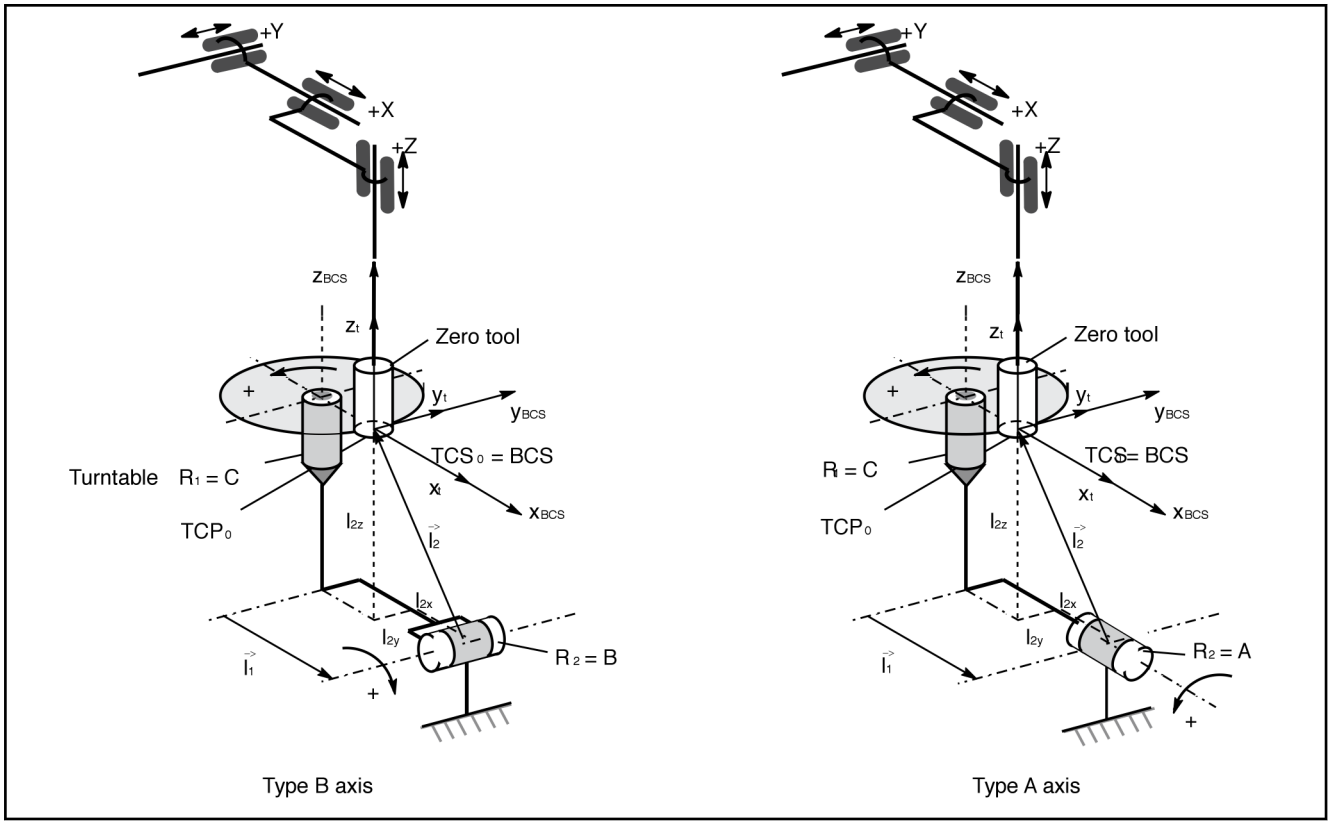


Fig. 11-46: Reference position

TCS, TCP

TCS stands for Tool Coordinate System.

TCP_0/TCP_1 refers to the Tool Center Point **without/with** correction.

Tool correction

Axis transformation supports a tool correction in the tool coordinate system (TCS). In reference position, TCS_0 conforms with BCS. The tool correction vector $\vec{l}_t$ points from the corrected tool center point TCP_1 to the uncorrected TCP_0. It comprises the length corrections L1, L2 and L3 from the D-correction table.

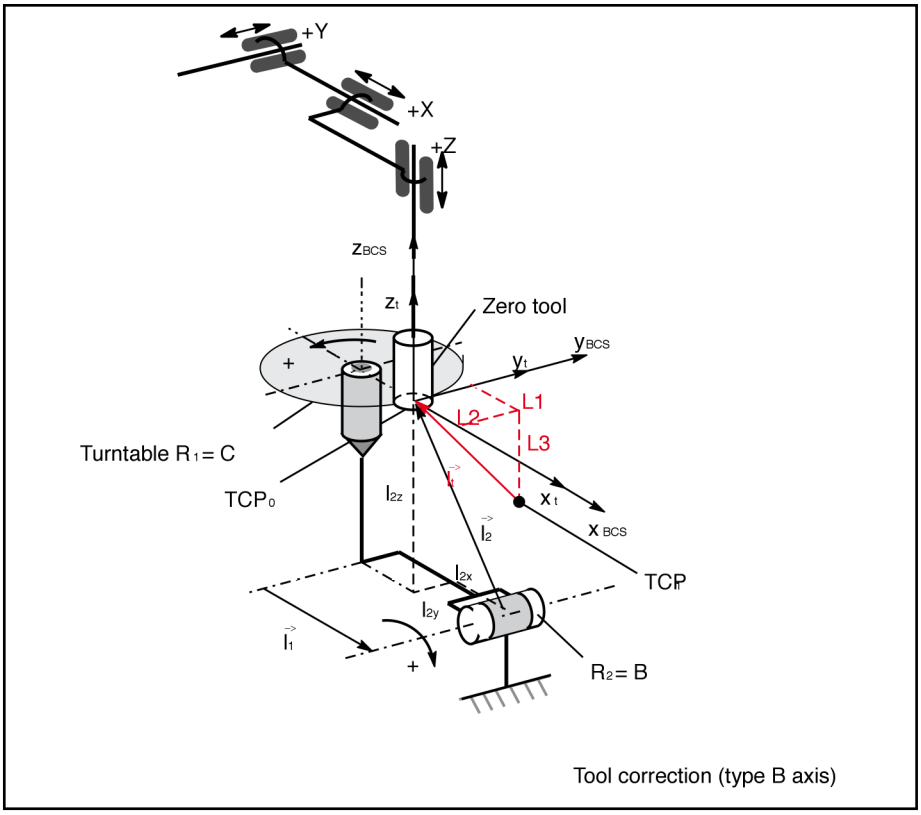


Fig. 11-47: Tool correction

Special features

- **Ambiguous axis positions:**

  In case of the CBLLL and CALLL variants, a TCP position and a tool orientation can be realized by two different axis positions (transformation branches). The branches differ in their b-axis positions (or a-axis position). There is no ambiguity for the BALLL variant.

  Vector orientation:

  Which one of the two poses (transformation branches) is used is decided on activation of the axis transformation. Accordingly, a transition from branch 1 to -1 and vice versa is only possible when traveling through the orientation location θ =0 (north pole singularity).

  Linear orientation:

  The transformation branch is always chosen in a way that the b-axis (a-axis) and the orientation angle θ have the same sign, i.e. the following applies:

  – θ≥0 ≙ B≥0 (branch 1)
  – θ≤0 ≙ B≤0 (branch -1)

  Therefore, a backwards transformation can be solved uniquely. Accordingly, a transition from branch 1 to -1 and vice versa is possible when traveling through the orientation location θ =0 (north pole singularity).

- **North pole singularity:**

  Vector orientation in variants CBLLL and CALLL:

  The north pole is a singularity.

In the north pole, the machine parameter 103000160 is effective in the Epsilon environments; i.e. orientation motions through the ε-environment are lead through θ =0.

If θ =0, φ can be programmed, but no motion takes place.

Linear orientation in variants CBLLL and CALLL:

The north pole is no singularity.

The north pole can be traveled without problems. The machine parameter 103000160 Epsilon environments is not effective, i.e. orientation motions through the ε-environment are not imperatively lead through point θ =0.

If θ =0, φ can be programmed and the desired motion (rotation of the c-axis) is performed.

Vector orientation in variant BALLL:

The north pole is no singularity.

The north pole can be traveled without problems.

However, the orientation φ = 90°, θ = 90° must not be programmed and must not be reachable on different ways.

11.12.2 Handling instruction: 5-Axis Transformation RRLLL

Applying

The axis transformation can be applied via the relevant machine parameters. Therefore, the axis transformation function has to be activated via the parameter **tra** "Axis transformations" in the setup (SUP). The path of the machine parameter name is always "TRA/AxTrafo[1..20]/...".

Alternatively, the axis transformation can be applied via a user-defined system date of type SysAxTrafo_t.

The content of both variants is identical.

The axis positions of the reference position have to be determined.

The respectively equivalent machine parameter numbers, machine parameter names and SD elements are specified in the following table. To apply a transformation, one of the variants has to be assigned accordingly.

Transformation-relevant parameters

MP number	Description	
1030 00110	**Transformation type** MP name: TrafoType SD elem.: Type	
	3232203	RRLLL with vector orientation
	3232103	RRLLL with linear orientation (Invalid for BALLL)

MP number	Description	
1030 00120	**System axes/coordinates of the transformation** MP name: FwdInCoordIndTrafo[1..8] SD elem.: AxisAssignment[1..8]	
	[1]	System axis number of R_1 (C-axis for CBLLL, C-axis: for CALLL, B-axis for BALLL)
	[2]	System axis number of R_2 (B-axis for CBLLL, A-axis: for CALLL, A-axis for BALLL)
	[3]	System axis number of X
	[4]	System axis number of Y
	[5]	System axis number of Z
	[6..8]	Not relevant

<table>
<tr><th>MP number</th><th>Description</th></tr>
<tr><td>1030 00125</td><td>
Axis classification of transformation axes
MP name: CoordClass[1..8]
MP name: CoordDir[1..8]
SD elem.: AxisClassification[1..8]
</td></tr>
<tr><td></td><td>
Defines the axes involved in transformation, including their direction of motion.
• MP no. or SD element:
Value ">0" means:
The axis rotates/travels positively as in the figures above.
Value "<0" means:
The axis rotates/travels negatively as in the figures above.
Possible values are:
– +/-1 = x-axis
– +/-2 = y-axis
– +/-3 = z-axis
– +/-100 = a-axis
– +/-200 = b-axis
– +/-300 = c-axis
• MP name:
– Possible values for CoordClass are
– X,Y,Z (linear)
– A,B,C (rotary)
– Possible values for CoordDir are:
– positive (regarding fig. "Reference position"
– negative (regarding fig. "Reference position"
</td></tr>
<tr><td></td><td>If a parameter is not set, the following default settings apply:</td></tr>
<tr><td></td><td>
<table>
<tr><td></td><td>MP no /
SD elem.:</td><td>CoordClass:</td><td>CoordDir:</td></tr>
<tr><td>[1]</td><td>300</td><td>C</td><td>Positive</td></tr>
<tr><td>[2]</td><td>200</td><td>B</td><td>Positive</td></tr>
<tr><td>[3]</td><td>1</td><td>X</td><td>Positive</td></tr>
<tr><td>[4]</td><td>2</td><td>Y</td><td>Positive</td></tr>
<tr><td>[5]</td><td>3</td><td>Z</td><td>Positive</td></tr>
<tr><td>[6..8]</td><td>Not relevant</td><td>0</td><td>Positive</td></tr>
</table>
</td></tr>
</table>

MP number	Description	
1030 00130	**Axis positions of the reference position** MP name: RefPosTrafo[1..8] SD elem.: AxZeroPos[1..8]	
	[1]	R_1^0
	[2]	R_2^0
	[3]	X^0
	[4]	Y^0
	[5]	Z^0
	[6..8]	Not relevant
1030 00140	**Length and angle parameters** MP name: JointParTrafo[1..16] SD elem.: LenParam[1..16]	
	Defines the length vectors l_1, l_2 in [mm].	
	[1]	l_{1x} (only relevant for CBLLL)
	[2]	l_{1y} (only relevant for CALLL)
	[3]	l_{1z} (only relevant for BALLL)
	[4]	l_{2x}
	[5]	l_{2y}
	[6]	l_{2z}
	[7..16]	Not relevant
1030 00160	**Epsilon environments** MP name: EpsAxTrafo SD elem.: EpsilonRanges	
	0.01	

Tab. 11-55: Transformation-relevant parameters

Activating

If the 5-axis transformation is configured via

- machine parameter, the transformation is enabled in the NC program with (i=1,..20):

  COORD(<i>)

- system date, the transformation is enabled in the NC program with (type "SysAxTrafo_t"):

  COORD(<SD.MtbAxTrafo>)

Example program:

The 5-axis transformation is defined in the fourth machine parameter block. No axis transformation is active at programming start:

Enable transformation

```
N100  G1 F1000 XA10 YA10 C30   ;Axes XA, YA and C are traversed
N110  COORD(4)                 ;Switching on the axis transformation
N130  x20 y30 phi0             ;the coordinates x,y,phi are traversed
N140  x10 z10 theta45          ;the coordinates x,z,theta are traversed
        :
```

```
N999  COORD(0)                ;Switching-off the axis transformation
                              ;Axes are again programmed from here
```

Deactivating **COORD(0)** or **COORD(0,2)**

switches off the axis transformation in the NC program.

11.13 5-axis transformation LLLCB - Type 4: Cardanic head with vector orientation motion

11.13.1 Description

Function

Transformation name	Transformation type	Point of action	Coordinate system
Cardanic 5-axis transformation	3232211	AT2	MCS, BCS

Tab. 11-56: Function

Output variables in the MCS: X,Y,Z,C,B

Input variables in the BCS: x, y, z, φ, θ

The axis kinematics of the Cardanic axis transformation comprises three linear axes (e.g., B, Y, and Z) which form a right-handed Cartesian coordinate system, as well as a Cartesian machining head which orients the tool in space with two rotary axes (B and C).

The c-axis is arranged parallel to the z-axis of the MCS. The b-axis is inclined in relation to the xy-plane of the MCS. The angle of inclination α varies in the range of 0° ≤ α ≤ 89°. In most practical cases, α = 45°. If α = 0°, the Cardanic axis transformation is identical to the 5-axis transformation RLLLR of type 3232201.

$\vec{l}_1$ is a vector along x_{BCS} from the c-axis to the b-axis. That means, $\vec{l}_1$ only has one component l_{1x}.

$\vec{l}_2$ is the continuation of the b-axis to the zero point of the BCS and consists of up to three components l_{2x}, l_{2y} and l_{2z}.

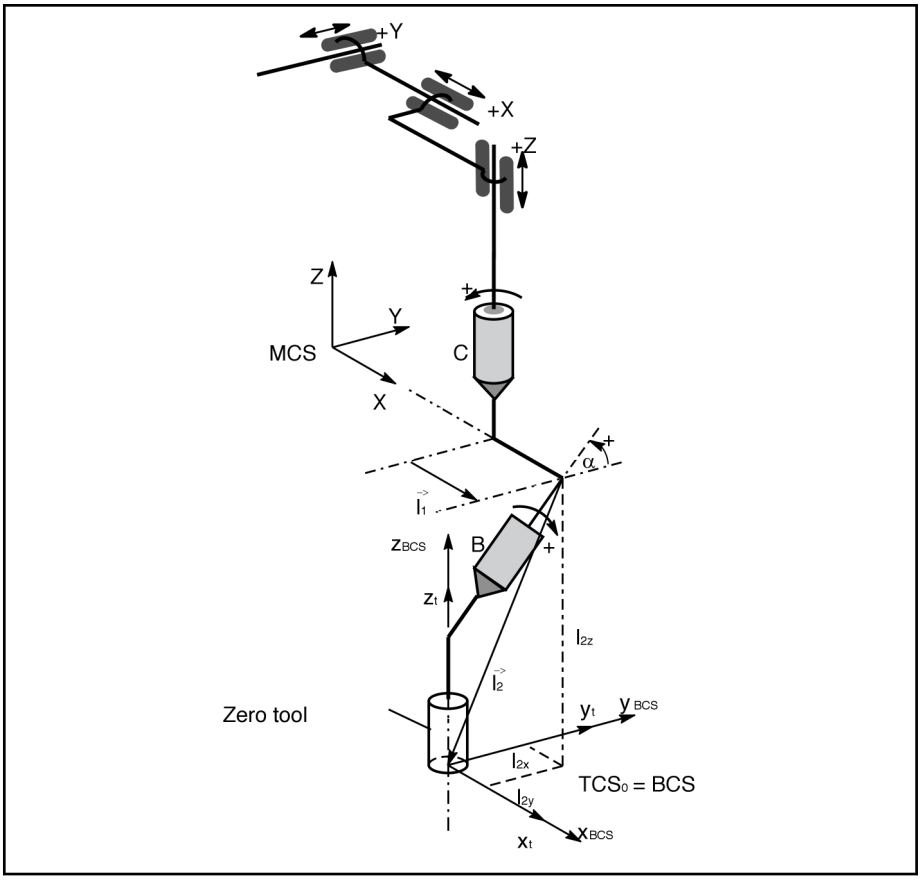


Fig. 11-48: Function type 4: Cardanic head

Restriction

Online correction of φ and ϑ is not possible.

Relevant NC functions

COORD(<i>)	I=1..20 is used to activate a 5-axis transformation which is defined in MP 1030 00110.
COORD(0,2)	The active transformation is deactivated.

Tab. 11-57: Relevant NC functions

Relevant CPL functions

SDR(300)	Active Epsilon environment. The active value can be greater than the value set in EpsAx-Trafo (MP 1030 00160).

Tab. 11-58: Relevant CPL functions

Relevant machine parameters

1030 00110	Defines the transformation type
1030 00120	Defines the system axes/coordinates of the transformation which are involved in the transformation.
1030 00130	Defines the axis positions of the reference position

1030 00140	defines the length and angle parameters
1030 00160	Epsilon environments

Tab. 11-59: Relevant machine parameters

Properties of the cardanic axis transformation

Input variables are the linear coordinates x, y and z and the orientation coordinates φ and ϑ. When the axis transformation is activated, the vector orientation movement is activated automatically, i.e. each change in orientation by programming φ and ϑ or the optional syntax (see DIN programming instruction) is implemented by a rotation of the orientation vector $\vec{\rho}$ (φ, ϑ) around a rotary axis which is fixed in space (large circle motion of the vector point). The orientation vector $\vec{\rho}$ can be programmed as a spline.

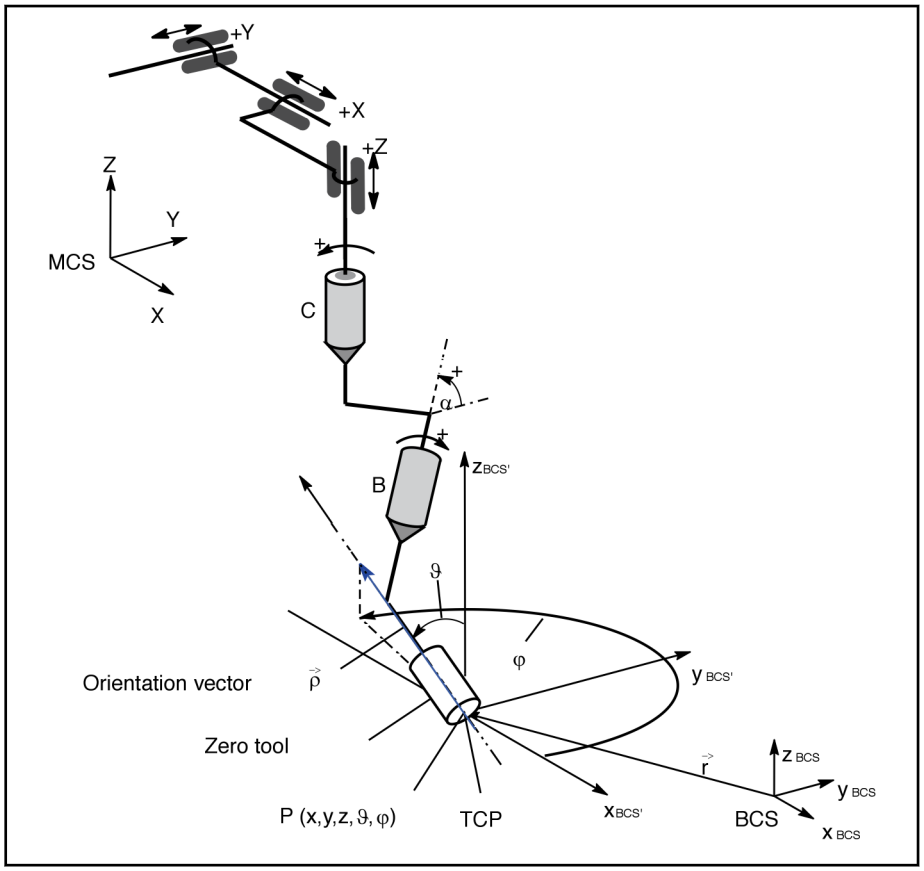


Fig. 11-49: Cardanic axis transformation

Due to the inclined b-axis, the angle ϑ between z_{BCS} and the tool symmetry axis is not larger than

$\vartheta_{max} = 2(90° - \alpha)$

Thus, the z-component of the orientation vector $\vec{\rho}_z$ is not smaller than

$$\vec{\rho}_z^{\,min} = \cos \vartheta_{max}$$

For ϑ_{max} or $\vec{\rho}_z^{\,min}$ using the reference position, the result is an orientation, rotating B by ±180° (assuming the reference position).

Example:

α = 45°: ϑ_{max} → = 90°; therefore: $\vec{\alpha}_{min}$ = 0 in b-axis position ±180°

Reference position

In reference position, the axis positions of all axes are defined by the following characteristics:

- The tool zero point (tool holder or TCP of a zero tool) is in the origin of the basic coordinate system (BCS)
- The tool symmetry axis runs along Z
- The b-axis is in the yz-plane
- Angle of inclination α of B-axis $\vec{l}_1$ is the vector from C-axis to B-axis.

  $\vec{l}_2$ is the continuation vector to the BCS.

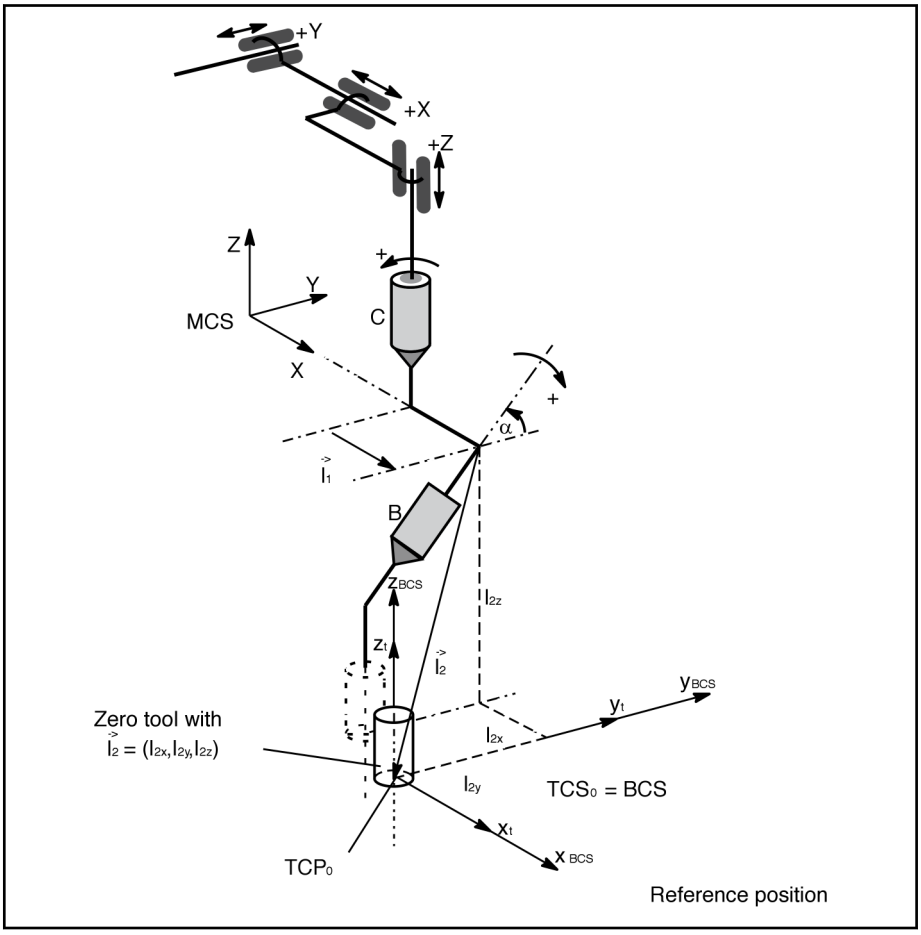


Fig. 11-50: Reference position

TCS, TCP

TCS stands for Tool Coordinate System.

TCP_0/TCP_1 refers to the Tool Center Point **without/with** correction.

Tool correction

Axis transformation supports a tool correction in the tool coordinate system (TCS). In the reference position, the TCS matches the BCS. The tool correction vector $\vec{l}_t$ points from the corrected tool center point TCP_1 to the uncorrected TCP_0. It comprises the length corrections L1, L2 and L3 from the D-correction table.

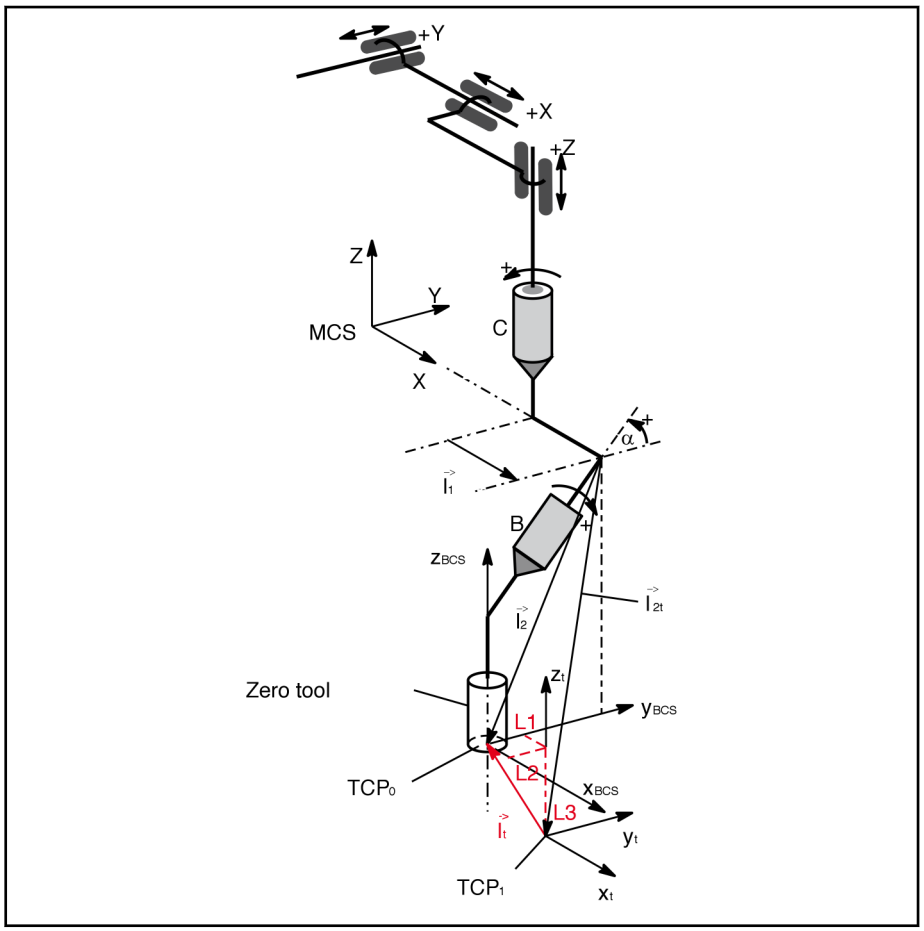


Fig. 11-51: Tool correction

Special features

With the Cardanic axis transformation, singularities of the orientation motion are taken into consideration, i.e., non-tangential traveling of $\vec{\rho}$ through the places $\vec{\rho}$ = (0,0,1) (=north pole singularity) and $\vec{\rho}$ =(*,*, $\vec{\rho}_{xy}^{min}$) (=Equator singularity).

- **Ambiguous axis positions:**

  In general, a TCP position $\vec{r}$ and a tool orientation $\vec{\rho}$ can be achieved by two different axis positions (transformation branches). The branches differ in their b-axis positions.

  The following applies:

  $B \in [n \cdot 180°, (n + 1) \cdot 180°]$

  with

n = even for branch 1 and

n = odd for branch -1.

Which one of the two transformation branches is used is decided on activation of the axis transformation. Accordingly, a transition of branch 1 to -1 and vice versa is only possible when the orientation location $\vartheta = 0$ (north pole singularity) or $\vartheta = \vartheta_{max}$(equator singularity) is traveled.

- **North pole singularity:**

  For the singularity ϑ= 0°, the same applies as to the 5-axis transformation of the type 3232201. Specifically with non-tangential passage of $\vec{\rho}$ through the places $\vec{\rho}$ = (0,0,0) (north pole singularity) the NC automatically generates a special rotary block for the c-axis.

  If the orientation vector closely passes the singularity, this results in a high C-axis velocity, i.e. the path velocity has to be greatly reduced to prevent the c-axis from exceeding its maximum permissible velocity. In order to avoid the resulting creeping motions, an ε-environment is introduced around the ϑ = 0° location. If the orientation vector passed the ε-environment, the orientation motion is guided through the north pole. The value for ε is recorded in the MP.

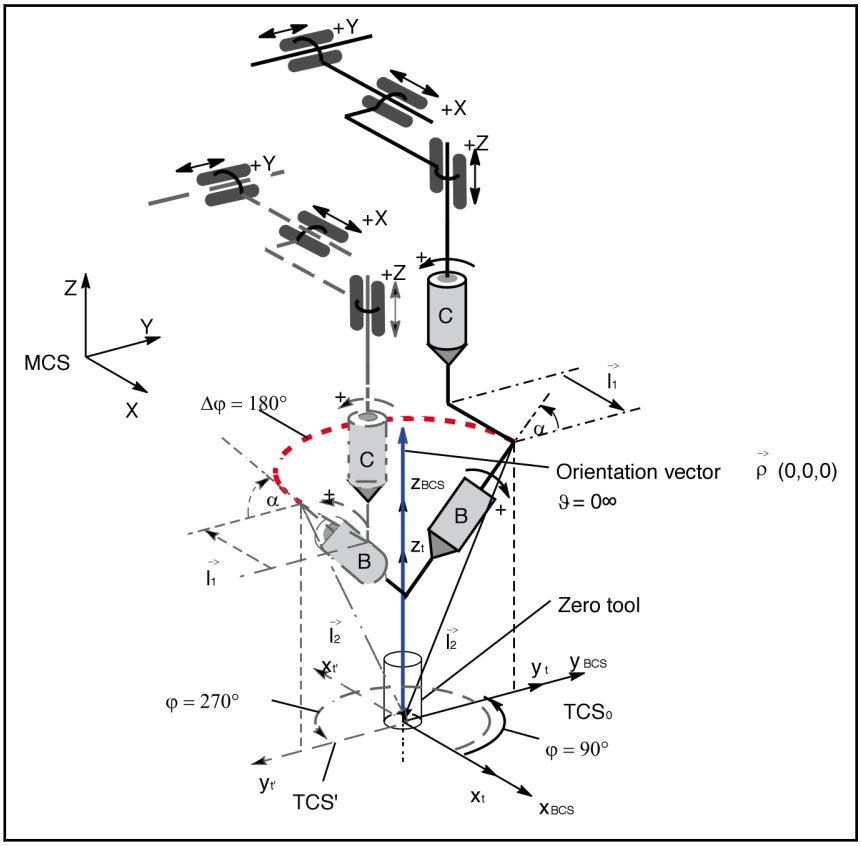


Fig. 11-52: North pole singularity

- **Equator singularity:**

  In case of orientation motions which go tangentially through the parallel of latitude $\vartheta = \vartheta_{max}$, the transformation branch is only changed if the b-axis has a traversing range of at least Bε [-360°, + 360°]. The traversing range is defined by the negative and positive first limit switch of the b-axis.

This is not affected by hiding of the software limit switch or changing to the second limit switch.

Motions in $\vartheta = \vartheta_{max}$ result in high b-axis velocities. In these cases, the path velocity profile is changed internally in the NC in such a way to ensure a smooth motion of the b-axis.

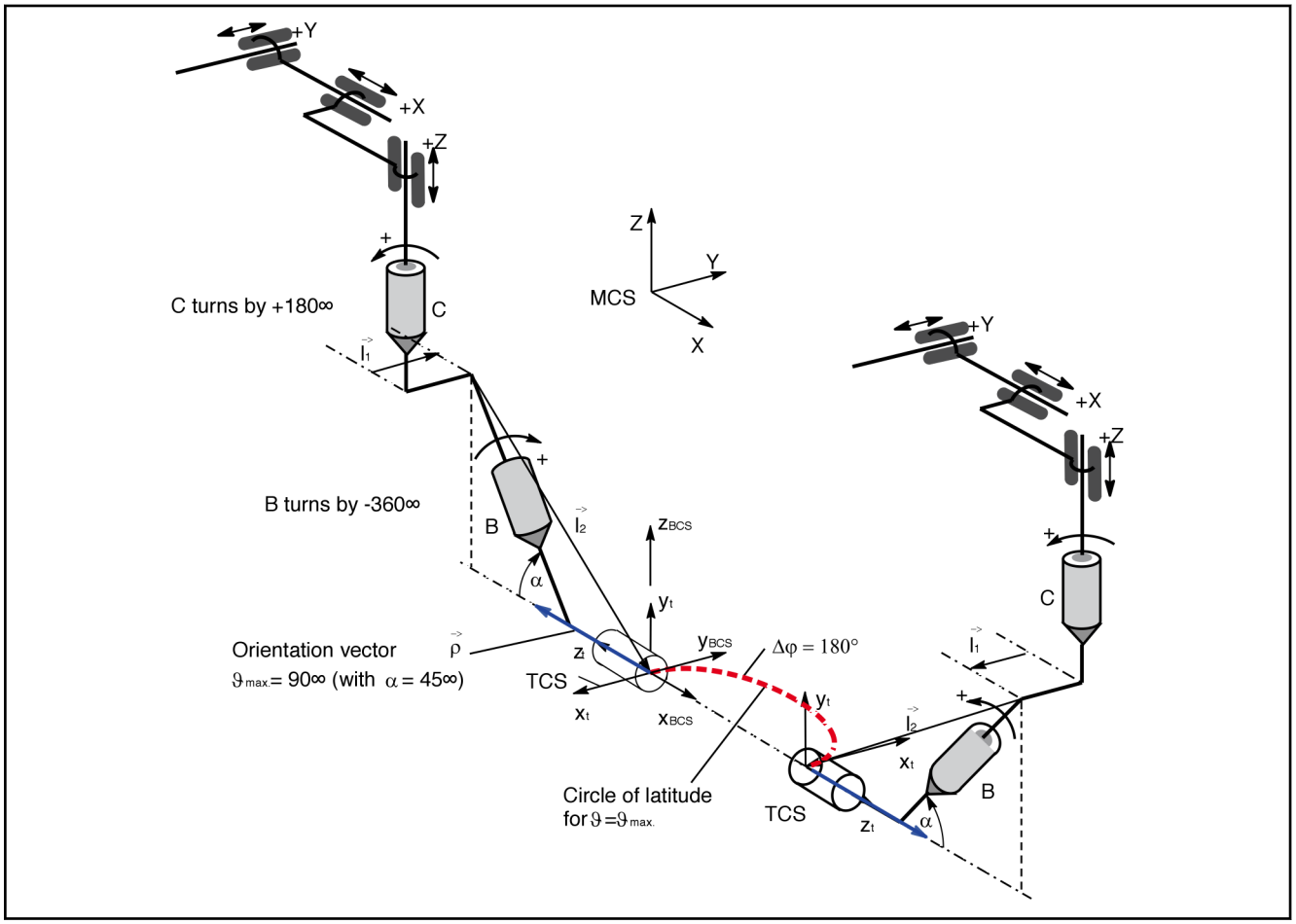


Fig. 11-53: Equator singularity

- **Feed:**

  After activation of the 5-axis transformation, the system is switched to spatial coordinate programming. The programmed feed (F) now only relates to the programmable position coordinates, i.e. the path velocity of the tool center point (TCP) is programmed with the F-word.

  This path velocity is not changed by additional orientation and pseudo coordinate motions. The orientation and pseudo coordinate motion is included synchronously, i.e. the end position is reached simultaneously for all coordinates. However, the included motion of the orientation and pseudo coordinates may result in an additional limitation of path kinematics (maximum path velocity and acceleration) as the limits for all axes involved in the motion are monitored.

11.13.2 Handling instruction: 5-axius transformation cardanic LLLRR

Applying The axis transformation can be applied via the relevant machine parameters. Therefore, the axis transformation function has to be activated via the param-

eter **tra** "Axis transformations" in the setup (SUP). The path of the machine parameter name is always "TRA/AxTrafo[1..20]/...".

Alternatively, the axis transformation can be applied via a user-defined system date of type SysAxTrafo_t.

The content of both variants is identical.

The axis positions of the reference position have to be determined.

The respectively equivalent machine parameter numbers, machine parameter names and SD elements are specified in the following table. To apply a transformation, one of the variants has to be assigned accordingly.

Transformation-relevant parameters

MP number	Description	
1030 00110	**Transformation type** MP name: TrafoType SD elem.: Type	
	3232211	Cardanic LLLRR with vector orientation
	3232111	Cardanic LLLRR with vector orientation
1030 00120	**System axes/coordinates of the transformation** MP name: FwdInCoordIndTrafo[1..8] SD elem.: AxisAssignment[1..8]	
	[1]	System axis number of X
	[2]	System axis number of Y
	[3]	System axis number of Z
	[4]	System axis number of C
	[5]	System axis number of B
	[6..8]	Not relevant

<table>
<tr><th>MP number</th><th>Description</th></tr>
<tr><td>1030 00125</td><td>
Axis classification of transformation axes
MP name: CoordClass[1..8]
MP name: CoordDir[1..8]
SD elem.: AxisClassification[1..8]
</td></tr>
<tr><td></td><td>
Defines the axes involved in transformation, including their direction of motion.
• MP no. or SD element:
Value ">0" means:
The axis rotates/travels positively as in the figures above.
Value "<0" means:
The axis rotates/travels negatively as in the figures above.
Possible values are:
– +/-1 = x-axis
– +/-2 = y-axis
– +/-3 = z-axis
– +/-100 = a-axis
– +/-200 = b-axis
– +/-300 = c-axis
• MP name:
– Possible values for CoordClass are
– X,Y,Z (linear)
– A,B,C (rotary)
– Possible values for CoordDir are:
– positive (regarding fig. "Reference position"
– negative (regarding fig. "Reference position"
</td></tr>
<tr><td></td><td>If a parameter is not set, the following default settings apply:</td></tr>
<tr><td></td><td>
<table>
<tr><th></th><th>MP no /
SD elem.:</th><th>CoordClass:</th><th>CoordDir:</th></tr>
<tr><td>[1]</td><td>1</td><td>X</td><td>Positive</td></tr>
<tr><td>[2]</td><td>2</td><td>Y</td><td>Positive</td></tr>
<tr><td>[3]</td><td>3</td><td>Z</td><td>Positive</td></tr>
<tr><td>[4]</td><td>300</td><td>C</td><td>Positive</td></tr>
<tr><td>[5]</td><td>200</td><td>B</td><td>Positive</td></tr>
<tr><td>[6..8]</td><td>Not relevant</td><td>0</td><td>Positive</td></tr>
</table>
</td></tr>
</table>

MP number	Description	
1030 00130	**Axis positions of the reference position** MP name: RefPosTrafo[1..8] SD elem.: AxZeroPos[1..8]	
	[1]	X^0
	[2]	Y^0
	[3]	Z^0
	[4]	C^0
	[5]	B^0
	[6..8]	Not relevant
1030 00140	**Length and angle parameters** MP name: JointParTrafo[1..16] SD elem.: LenParam[1..16]	
	Defines the length vectors l_1, l_2 in [mm].	
	[1]	l_{1x}
	[2]	Not relevant
	[3]	Not relevant
	[4]	l_{2x}
	[5]	l_{2y}
	[6]	l_{2z}
	[7]	α
	[8..16]	Not relevant
1030 00160	**Epsilon environments** MP name: EpsAxTrafo SD elem.: EpsilonRanges	
	0.01	

Tab. 11-60: Transformation-relevant parameters

Activating If the 5-axis transformation is configured via

- machine parameter, the transformation is enabled in the NC program with (i=1,..20):

  COORD(<i>)

- system date, the transformation is enabled in the NC program with (type "SysAxTrafo_t"):

  COORD(<SD.MtbAxTrafo>)

Example program:

The 5-axis transformation is defined in the fourth machine parameter block. No axis transformation is active at programming start:

Enable transformation

```
N100  G1 F1000 XA10 YA10 C30   ;Axes XA, YA and C are traversed
N110  COORD(4)                 ;Switching on the axis transformation
N130  x20 y30 phi0             ;the coordinates x,y,phi are traversed
N140  x10 z10 theta45          ;the coordinates x,z,theta are traversed
```

```
      :
N999  COORD(0)                ;Switching-off the axis transformation
                              ;Axes are again programmed from here
```

Deactivating **COORD(0)** or **COORD(0,2)**

switches off the axis transformation in the NC program.

11.14 5-axis transformation LLLCB - Type 4.1: Cardanic head with linear orientation motion

11.14.1 Description

Function

Transformation name	Transformation type	Point of action	Coordinate system
Cardanic 5-axis transformation with linear orientation motion	3232111	2	MCS, BCS

Tab. 11-61: Function

The axis configuration of the Cardanic axis transformation with linear orientation motion and that of the Cardanic axis transformation with vector orientation motion comply mechanically. Both have the structure that is shown in the illustration.

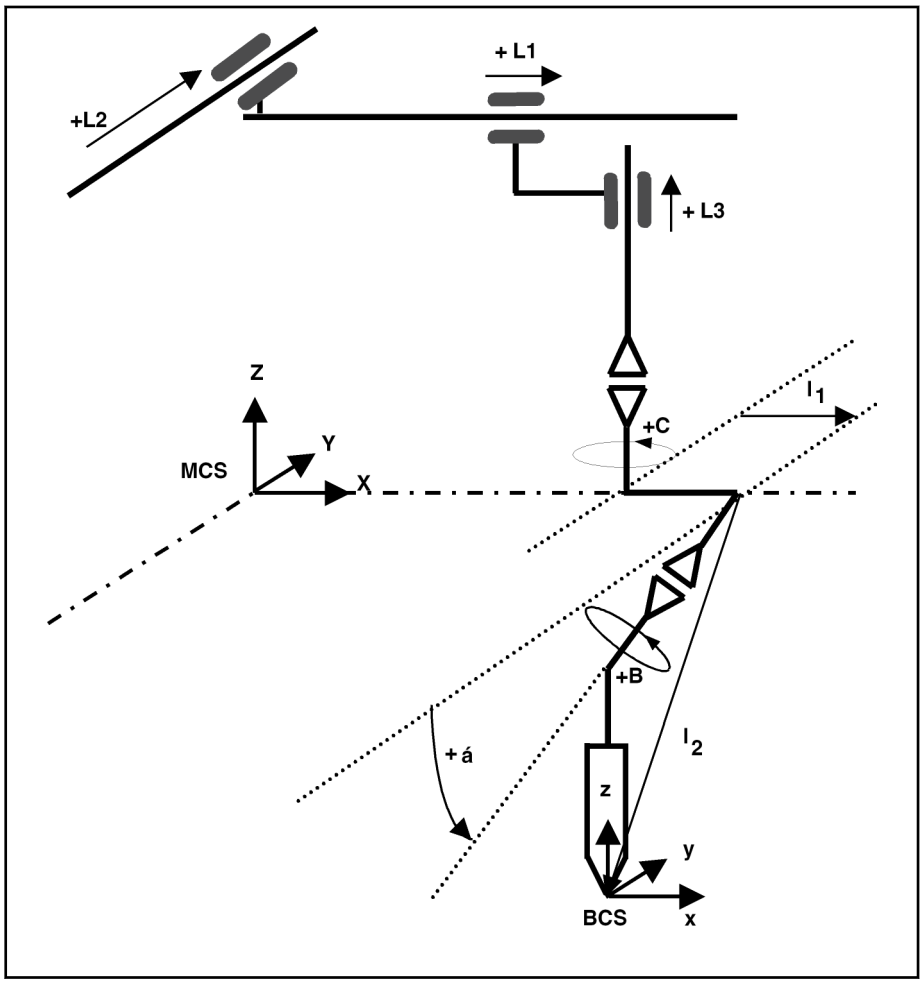


Fig. 11-54: Structure

This chapter is going to show only the differences based on the different orientation motions. Otherwise, go to chapter 11.13 "5-axis transformation LLLCB - Type 4: Cardanic head with vector orientation motion" on page 293.

The Cardanic axis transformation with linear orientation has transformation type 3232111.

When the axis transformation is activated, the linear orientation motion is activated automatically, i.e., each change of orientation by programming of φ and ϑ is realized by linear interpolation of φ and ϑ.

During the orientation motion the "shortest way" logic can be deactivated via MP length parameter [8].

φ and ϑ can assume values from the following ranges:

$$\vartheta \in [-\vartheta_{max}, \vartheta_{max}] \quad \text{and} \quad \varphi \in [-\infty, +\infty]$$

The rotary axes B and C have to be configured as rotary axes (no modulo endless axes).

Ambiguities

In general, a TCP position $\vec{r}$ and a tool orientation $\vec{o}$ can be achieved by two different axis positions (transformation branches). The branches differ in their b-axis positions. The transformation branch is always chosen in a way that the b-axis and the orientation angle ϑ have the same sign, i.e. the following applies:

ϑ ≥ 0 ⇔ B ≥ 0 (branch 1)

ϑ < 0 ⇔ B < 0 (branch -1)

Therefore, a backwards transformation can be solved uniquely. Accordingly, a transition from branch 1 to -1 and vice versa is possible when traveling through the orientation location ϑ = 0° (north pole singularity).

North pole

The north pole can be traversed without problems. The machine parameter 103000160 Epsilon environments is not effective, i.e. orientation motions through the ε-environment are not imperatively lead through point ϑ = 0°. In case of ϑ = 0°, φ can be programmed and the desired motion (rotation of the c-axis) is executed.

Restrictions

Inclined plane:

The linear orientation motion is not suitable for workpiece position correction (BCR) and the placement functionality, if it causes a rotation of the coordinate system WCS. The programmed positions x,y,z correctly refer to the WCS, orientations φ and ϑ , but always refer to the BCS, i.e. only machining with constant orientation can be performed well. If BCR and placement constitute a mere offset then the effect is correct.

Relevant NC functions

Coord (...)

Relevant machine parameters (MP)

The Cardanic axis transformation with linear orientation motion is stored in the parameters 103000110 to 103000160.

1030 00110 transformation type:	
3232111	(Cardanic axis transformation with linear orientation motion)
1030 00160 Epsilon environments	
	Irrelevant

Tab. 11-62: Relevant machine parameters

11.14.2 Handling instruction: Cardanic head with linear orientation motion

Configure cardanic 5-axis transformation in the machine parameters or via system date.

In the NC-program, activate the 5-axis transformation with Coord (...).

11.15 5-axis transformation LLLCA - Type 4/4.1: Cardanic head with vector or linear orientation motion

11.15.1 Description

Function

The 5-axis transformation LLLCA with a cardanic head differs from the previously described 5-axis transformation LLLCB with cardanic head only by the rotated coordinate systems BCS and MCS.

In the following figure, variant LLLCB is displayed in reference position. In this position, the C-axis is rotated so that the B-axis is in the yz-plane of the coordinate system BCS. The B-axis is rotated so that the zero tool is in parallel to the z-direction of the BCS. The linear axes X, Y, Z are positioned so that the tip of the zero tool is positioned in the zero point of the BCS.

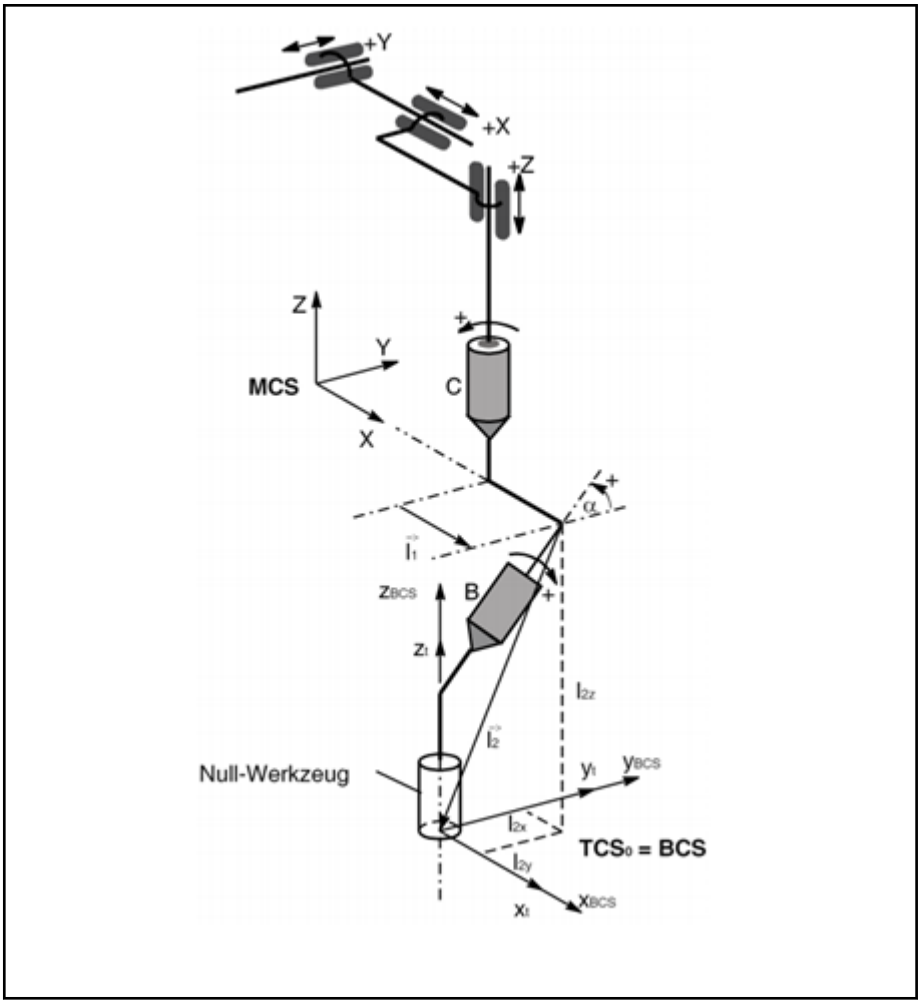


Fig. 11-55: Axis transformation LLLCB with cardanic head, reference position

Based on this position, the BSC is rotated by +90° around the BCS-z-direction. Based on this position, the MSC is rotated by +90° around the MCS-z-direction. Thus, the following reference position of variant LLLCA can be seen is shown in the figure. As the previous B-axis is not positioned in the yz-plane anymore but in the zx-plane of the BCS, this axis is referred to as A-axis.

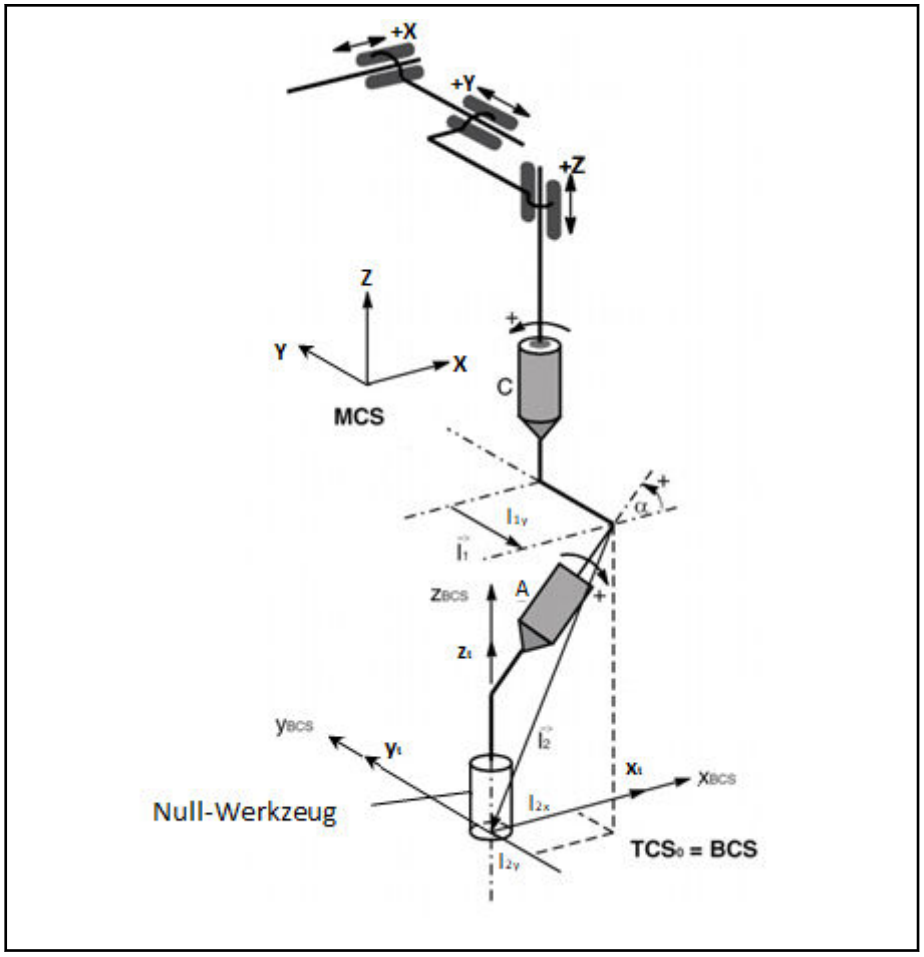


Fig. 11-56: Axis transformation LLLCA with cardanic head, reference position

The l_1 and l_2 vector as well as the axis positions of the reference position refer to the rotated coordinate systems BCS and MCS for variant LLLCA. Amongst others, this means $l_{1y} = l_{1z} = 0$ for the LLLCB variant while $l_{1x} = l_{1z} = 0$ for the LLLCA variant.

In the example shown in the first figure for variant LLLCB, the following applies: $l_{1x} > 0$, $l_{1y} = 0$, $l_{1z} = 0$, $l_{2x} < 0$, $l_{2y} < 0$, $l_{2z} < 0$,

In the example shown in the second figure for variant LLLCA, the following applies: $l_{1x} = 0$, $l_{1y} < 0$, $l_{1z} = 0$, $l_{2x} < 0$, $l_{2y} > 0$, $l_{2z} < 0$,

In the machine parameters, there are the following differences between variants LLLCB and LLLCA:

	LLLCB	LLLCA
FwdInCoordIndTrafo[5]	System axis number of B-axis	System axis number of A-axis
CoordClass[5]	B or no entry	A
RefPosTrafo[5]	B^0	A^0
JointParTrafo[1]	l_{1x} related to LLLCB-BCS	Not relevant
JointParTrafo[2]	Not relevant	l_{1y} related to LLLCA-BCS
JointParTrafo[3]	Not relevant	Not relevant

JointParTrafo[4]	l_{2x} related to LLLCB-BCS	l_{2x} related to LLLCA-BCS
JointParTrafo[5]	l_{2y} related to LLLCB-BCS	l_{2y} related to LLLCA-BCS
JointParTrafo[6]	l_{2z} related to LLLCB-BCS	l_{2z} related to LLLCA-BCS

Tab. 11-63: Differences between the 5-axis transformations LLLCB and LLLCA with cardanic head

11.16 5-axis transformation - Type 5: Less-equipped RRLLL (with swivel/rotary table)

11.16.1 Description

Function

Transformation name	Transformation type	Point of action	Coordinate system
Less-equipped 5-axis transformation with swiveling/rotary table	3030013	AT2	MCS, BCS

Tab. 11-64: Function - less-equipped RRLLL

Output variables in the MCS:

X,Y,Z, R_{2fix},R_{1fix}

Input variables in the BCS:

x,y,z, φ_{fix}, ϑ_{fix}

The less-equipped 5-axis transformation RRLLL has three linear axes (X, Y, Z) and two rotary axes (R1, R2). The two rotary axes are not activated by the NC but are exclusively under the control of the PLC. These axes do not exist for the NC.

The positions to be approached are generally in the workpiece coordinates (x, y ,z, φ , ϑ), i.e., they are specified by five degrees of freedom (three positions and two orientations).

During the actual machining, the rotary axes cannot be moved mechanically. It is only possible to position them before processing (from the PLC).

For a special processing cycle, φ and ϑ are always constant. Of the rotary axes R_1, R_2 resulting from φ, ϑ in order to be able to approach the starting position before the machining. Therefore, the less-equipped 5-axis transformation is used.

The transformation is used at point of action 2, i.e. it acts between the machine coordinate system MCS = (X,Y,Z,R_{2fix}, R_{1fix}) and the basic coordinate system BCS = (x,y,z,φ_{fix},ϑ_{fix}).

The rotary axes can be defined as endless or rotary axes.

The basic setup concerns the configuration RRLLL (transformer type 3232203). The tool is moved by means of the linear axes X,Y,Z. The workpiece is rotated by the rotary axis R_1 (rotary table). The swiveling axis R_2 swivels the rotary table.

In the reference position (BCS parallel to MCS) R_1 rotates around Z (type c-axis). The less-equipped 5-axis transformation differentiates between two variants of the R_2 axis arrangement (see the following illustrations):

- In the reference position, R_2 rotates around X (type a-axis)
- In the reference position, R_2 rotates around Y (type b-axis)

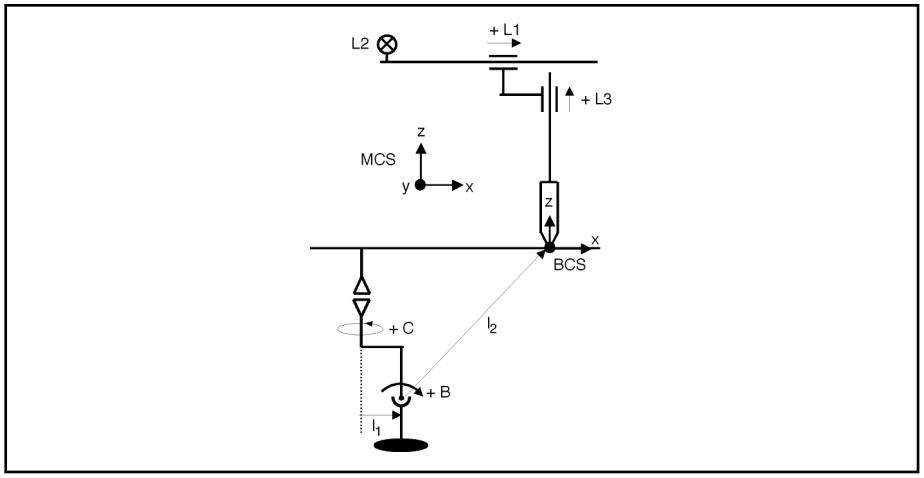


Fig. 11-57: RRLLL (type b-axis)

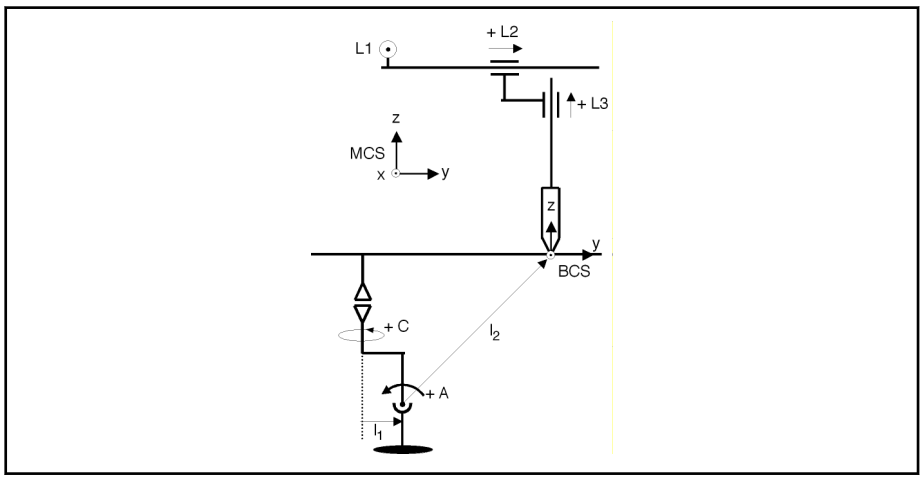


Fig. 11-58: RRLLL (type a-axis)

The vectors $\vec{l}_1$ and $\vec{l}_2$ define the geometry of the less-equipped 5-axis kinematics. They have the form $\vec{l}_1 = \begin{pmatrix} l_{1x} \\ l_{1y} \\ 0 \end{pmatrix}$ and $\vec{l}_2 = \begin{pmatrix} l_{2x} \\ l_{2y} \\ l_{2z} \end{pmatrix}$.

For type B-axis, there is a vector along x from the C-axis to the B-axis (that means $l_{1x} \neq 0$, $l_{1y} = 0$). For type A-axis, there is a vector along y from the c-axis to the A-axis (that means $l_{1x} = 0$, $l_{1y} \neq 0$). $\vec{l}_2$ It is the $\vec{l}_1$-the continuation vector of the R_2-axis (b-/a-axis) to the zero point of the BCS.

Optional transformation parameters:

Input variables for the less-equipped 5-axis transformation are the linear coordinates x, y and z. When activating, either the orientation coordinates (φ, ϑ) or the axis positions (R_1, R_2) including the transformation branch are transmitted as fix transformation parameters. This is the case during activation using three optional transformation parameters via TRFOPT.

- Coord ( ) TRFOPT ([PHI!], [THETA!], 1) or
- Coord ( ) TRFOPT ([C!], [B!], 0)

In the first case, the axis transformation is activated with a specified (φ, ϑ) and a preset transformation branch.

In the second case, a fixed axis position of the rotary axes (R_1, R_2) is transferred.

Using the third parameter, it is control-internally detected if the orientation angle (parameter value = ±1) or the axis positions (parameter value = 0) is transferred.

The five corresponding values (φ, ϑ, R_1, R_2, transformation branch), are kept in the length parameters (MP 1030 00140) and - after activation of the less-equipped 5-axis transformation - can be read out within an NC program via the SD variable of the active axis transformation 2.

SD.SysAxTrafo2.LenParam[8]	φ_{fix}
SD.SysAxTrafo2.LenParam[9]	ϑ_{fix}
SD.SysAxTrafo2.LenParam[10]	R_{1fix} (c-axis)
SD.SysAxTrafo2.LenParam[11]	R_{2fix} (A-/B-axis)
SD.SysAxTrafo2.LenParam[12]	Transformation branch (1 or -1)

Tab. 11-65: SD variable

Special features:

Ambiguities:

Generally, a TCP position and a tool orientation can be realized by two different axis poses (transformation branches). Which one of the two poses (transformation branches) is used is decided on activation of the axis transformation.

If the orientation coordinates (φ, ϑ) are also transferred, the third optional parameter corresponds to the transformation branch.

Note:

- Transformation branch = +1

  There is always a solution selected in which the R_2 axis (a-/b-axis) has a positive value.
- Transformation branch = -1

  There is always a solution selected in which the R_2 axis (a-/b-axis) has a negative value.

If the axis positions (R_1, R_2) are also transferred, the third optional parameter has to be set to the value 0. The transformation branch then automatically results from the position of the axes. In the singularity (i.e. R_2 = 0), the transformation branch +1 is determined.

After activation of the axis transformation, a transition from branch 1 to –1 and vice versa is no longer possible.

Value ranges:

The orientation angle (φ, ϑ) are internally limited to:

$0 \le \varphi \le 360$

$0 \le \vartheta \le 180$,

For axes always result values from the main area, that is for endless axes

$0 \le R_1 \le 360$

$0 \le R_2 \le 360$

and in rotary axes

Branch 1: $-180 \le R_1 \le 180$

Branch -1: $-360 \leq R_1 \leq 0$

$0 \leq R_2 \leq 180$

$-180 \leq R_2 \leq 0$.

Reference position: The images above show the two RRLLL configurations, type b-axis with R_1=C, R_2=B and type a-axis with R_1=C, R2=A in each case in the reference position. I.e. if the machine takes the position shown in the illustration, the BCS coordinates are (x, y, z, φ, ϑ) = (0,0,0,0,0), in case of MCS coordinates (X,Y, Z, R_1, R_2) = (X^0,Y^0, Z^0, R_1^0 , R_2^0) of machine parameter 1030 00130. The figure also shows the positive directions of rotation/traversing of the axes. Set them accordingly.

In this position, the peak of rotation-symmetric tools (Tool Center Point, TCP) is placed on the origin of basic coordinate system (BCS) and is directed along the z-coordinate. The axis transformation unambiguously determines the orientation of the BCS.

Tool correction: As in all 5-axis transformations, tool correction is possible. The correction vector $\vec{l}_t = \begin{pmatrix} l_{tx} \\ l_{ty} \\ l_{tz} \end{pmatrix}$ points from the tool point to the tool holder.

Relevant NC functions

COORD ( ) TRFOPT (.., ..,..)	
	The less-equipped 5-axis transformation is activated using three optional parameters. This can optionally be done using the machine parameter block (i=1..20) or via the MTB system data (of type SysAxTrafo_t).
COORD (0,2)	The active axis transformation is deactivated.

Tab. 11-66: Relevant NC functions

Relevant machine parameters (MP)

103000110	Transformation type
103000120	System axes/coordinates of the transformation
103000125	Axis classification of transformation axes
103000130	Axis positions of the reference position
103000140	Length and angle parameters
103000150	Reference position of tool coordinate system
103000160	Epsilon environments

Tab. 11-67: Relevant machine parameters (MP)

11.16.2 Applying

Determine data of reference position:

For setting the values, the axes have to be moved into the reference position.

Setting machine parameters:

103000110	Enter type of transformation 3030013 (less-equipped RRLLL)	
103000120	System axes / coordinates of the transformation: Defines the system axes involved in the transformation. The following applies: [1] System axis number of X [2] System axis number of Y [3] System axis number of Z [4..8] Not relevant	
103000125	Axis classification of transformation axes Defines the traveling direction of linear axes and the rotary axes involved in the transformation, including the direction of rotation. Value ">0" means: The axis travels positively as in the figure above. Value "<0" means: The axis travels negatively as in the figure above. Value ">0" means: The axis travels positively as in the figure above. Value "<0" means: The axis travels negatively as in the figure above. The following are defined: +/-1 = x-axis +/-2 = y-axis +/-3 = z-axis +/-100 = A-axis (rotary, no NC axis) +/-200 = b-axis (rotary, no NC axis) +/-300 = C-axis (rotary, no NC axis) +/-111 = A-axis (endless, no NC axis) +/-222 = B-axis (endless, no NC axis) +/-333 = C-axis (endless, no NC axis)	
103000125	If a parameter is not set, the following default settings apply:	
	[1] 1	(optionally -1)
	[2] 2	(optionally -2)
	[3] 3	(optionally -3)
	[4] 300	(optionally -300, +/-333)
	[5] 200	(optionally -200, +/-222, +/-100, +/-111)
	[6..8]	Not relevant

103000130	Axis positions of the reference position Defines the axis positions of the reference position. [1] X^0 [2] Y^0 [3] Z^0 [4] C^0 [5] R^2_0 (A^0 or B^0) [6..8] Not relevant	
103000140	Length and angle parameters Defines the length vectors $\vec{l_1}$, $\vec{l_2}$ in [mm], coupling factor, default orientation angle and rotary axis positions in [degree].	
	[1] l_{1x}	(Type b-axis)
	[2] l_{1y}	(Type a-axis)
	[3]	Irrelevant
	[4] l_{2x}	
	[5] l_{2y}	
	[6] l_{2z}	
	[7] K_{BC}	Coupling factor from b-axis to c-axis
	[8] φ	Default value for orientation angle φ is probably overwritten by TRFOPT or calculated from (R_1, R_2).
	[9] ϑ	Default value for orientation angle is probably overwritten by TRFOPT or calculated from (R_1, R_2).
	[10] R_1	Default value for position of c-axis is probably overwritten by TRFOPT or calculated from (φ, ϑ)
	[11] R_2	Default value for position of R_2-axis (a-axis or b-axis) is probably overwritten from TRFOPT or calculated from (φ, ϑ).
	[12] TrfBr	Default value for the transformation branch (value 1 or -1) is probably overwritten from TRFOPT or calculated from (R_1, R_2).
	[13..16]	Irrelevant
103000150	Reference position of tool coordinate system Irrelevant	
103000160	Epsilon environments Irrelevant	

Tab. 11-68: Setting the machine parameters

11.16.3 Activating

Prerequisite: The less-equipped 5-axis transformation is configured via MP 1030 00110 to MP 1030 00160 or via the MTB system data of type SysAxTrafo_t.

In the NC program, the less-equipped 5-axis transformation is activated for a special PHI!, THETA!, BRANCH! using Coord (...) TRFOPT ([PHI!], [THETA!], [BRANCH!]) activated and remains activated until it is switched off (COORD(0)).

Example:

1

Orientation angle (φ, ϑ) specified

Task: Determine corresponding axis positions (B, C) and move in

; workpiece coordinates.

;_____

; Activate axis transformation with requested phi, theta, transformation branch = -1

;_____

001 PHI! = 45

002 THETA! = 30

N001 **Coord(1) TRFOPT ([PHI!], [THETA!], -1)** ; less-equipped 5-axis transformation is activated with phi=45, theta=30, transformation branch=-1 (i.e. B-axis is always calculated negatively)

; the positions of the c-axis and b-axis resulting from (phi=45, theta=30)

; are read in

020 WAIT

021 C! = SD.SysAxTrafo2.LenParam [10]

022 B! = SD.SysAxTrafo2.LenParam[11]

;_____

; now the calculated positions of b-axis and c-axis have to

; be approached via PLC

; only then will the displayed WCS coordinates correspond to

; the machine

;

; only then will the displayed WCS coordinates correspond to

; the machine

;_____

N310 G1 F1000 x100 z200 ; x, z are WCS coordinates, e.g. the values

; relate to the workpiece coordinate system

:

: other traversing blocks, all regarding orientation (phi=45, theta=30)

:

N380 x50 y60 z70

N399 Coord(0) ; Deactivate the axis transformation

;_____end of cycle with (phi=45, theta=30) _____

Example:

2

Axis positions (B, C)

specified task: determine corresponding orientations (φ, ϑ)

; and move in workpiece coordinates

;_____

; approach specified position of b-axis and c-axis via PLC.

; Activate axis transformation with specified B, C

; read corresponding phi, theta

;_____

401 B! = 5

402 C! = 20

N403 Coord(1) TRFOPT([C!], [B!], 0) ; less-equipped 5-axis transformation is activated with B=5, C=20, "0"=identification "B,C are programmed"

; the orientations phi, theta resulting from (B=5, C=20) are

; read in

404 WAIT

405 PHI! = SD.SysAxTrafo2.LenParam [8]

406 THETA! = SD.SysAxTrafo2.LenParam [9]

;_____

; now it can be programmed and

; moved directly in workpiece coordinates.

; the displayed WCS coordinates correspond with the machine

;_____

N410 G1 F1000 x100 z200 ; x, z are WCS coordinates, e.g. the values relate to the workpiece coordinate system

:

: other traversing blocks, all regarding orientation (B=5, C=20)

:

N480 x50 y60 z70 N499 Coord(0)

; Deactivate axis transformation

;_____end of cycle with (B=5, C=20)_____

Example:

3

Inclined drillings

The following procedure is recommended for inclined drillings:

Assumption: $\varphi \neq 0$, $\vartheta \neq 0$.

It should be drilled in the point (x_{pos}, $y_{pos,}$, 0) of the basic workpiece coordination system BCS. In the point (xpos, ypos, 0) set an inclined level with (φ, ϑ). From this point of time, it can be programmed and moved in the coordinates (x', y', z') of WCS_1 as shown in the following illustration. The position (0,0,10) is now exactly 10 [mm] above the drill hole. The position (0,0,-100) corresponds e.g. to the deepest drill position.

```
;_____________________________________________________________
; Activate axis transformation with requested phi, theta, transformation branch = -1,
; normal procedure as in example 2 "Axis positions (B, C) specified"
;_____________________________________________________________
N601 Coord(1) TRFOPT ([PHI!], [THETA!], -1)
;_____________________________________________________________
; With the following command G152, an inclined plane in the point (100,200,0) is switched on.
;There is no traversing motion.
; The new program coordination system corresponds now to the WCS1 system
; of the figure.
;The zero point is now on the requested drill location.
;_____________________________________________________________
1000 XPOS! = 100
1001 YPOS! = 200
N602 G152 ([XPOS!], [YPOS!], 0, [PHI!], [THETA!], 0)
N603 G1 F1000 N604 x0 y0 z10 ; Approach the position 10 [mm] above the drill position
:
N650 z-100 ; Move (drill) 100 [mm] into the component
:
N690 G153 ; deactivate the inclined plane
N699 Coord(0) ; Deactivate axis transformation
```

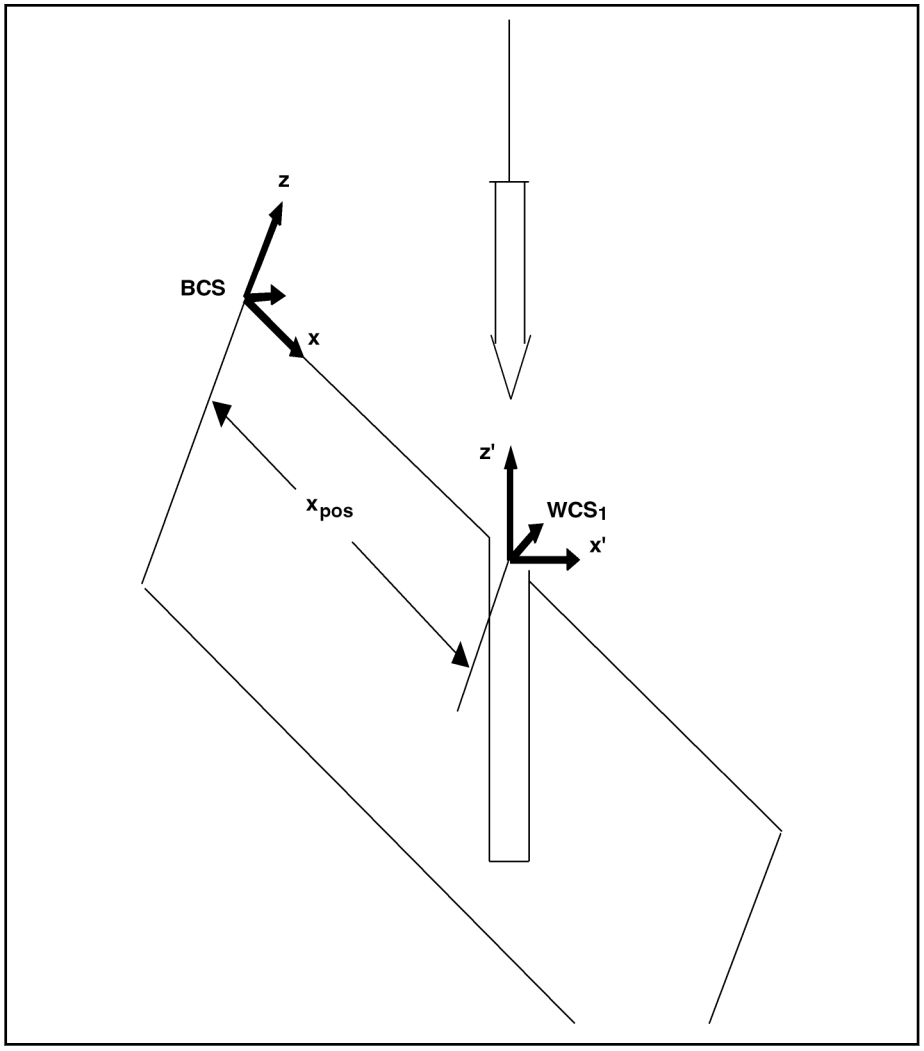


Fig. 11-59: Inclined drilling

11.17 5-axis transformation - Type 6: Cardanic RLLLR

11.17.1 Description

Function

Transformation name	Transformation type	Point of action	Coordinate system
Cardanic 5-axis transformation RLLLR			
Vector orientation	3232212	AT2	MCS, BCS
Linear orientation	3232112	AT2	MCS, BCS

Tab. 11-69: Function

Output variables in the MCS: X,Y,Z,R_2,R_1

Input variables in the BCS: x, y, z, φ, θ

The Cardanic 5-axis transformation (RLLLR) has three linear axes (X, Y, Z) and two rotary axes (R_1,R_2).

The rotary axes R_1 and R_2 can be endless axes and rotary axes.

The axis transformation is used at point of action 2, i.e., it acts between the machine coordinate system MCS = (X,Y,Z,R_2,R_1) and the basic coordinate system BCS = (x, y, z, φ, θ).

The three linear axes form a right-handed coordinate system. The rotary axis R_2 swivels the tool and the rotary axis R_1 rotates the workpiece (mounted on the axis).

The R_1-axis is a c-axis that means that is rotates in the reference position around the z-axis. R_2 is a b-axis and rotates around the y-axis. The b-axis is inclined in relation to the xy-plane of the MCS. The angle of inclination α varies in the range of 0° ≤ α ≤ 89°. In most practical cases, α = 45°. If α = 0°, the Cardanic axis transformation is identical with the 5-axis transformation RLLLR of type 3232202.

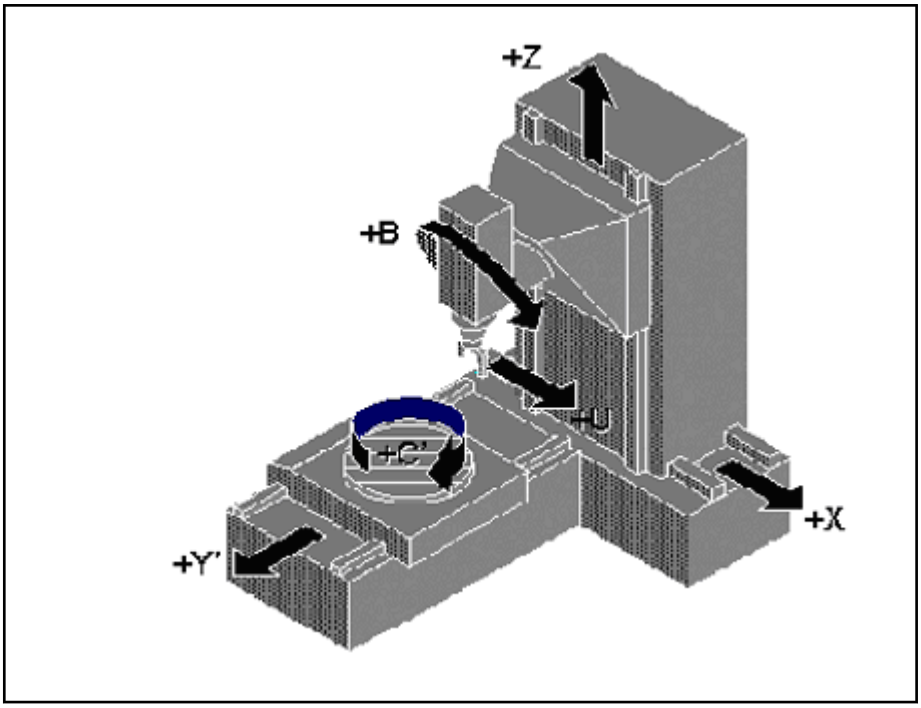


Fig. 11-60: Cardanic CLLLB (schematically)

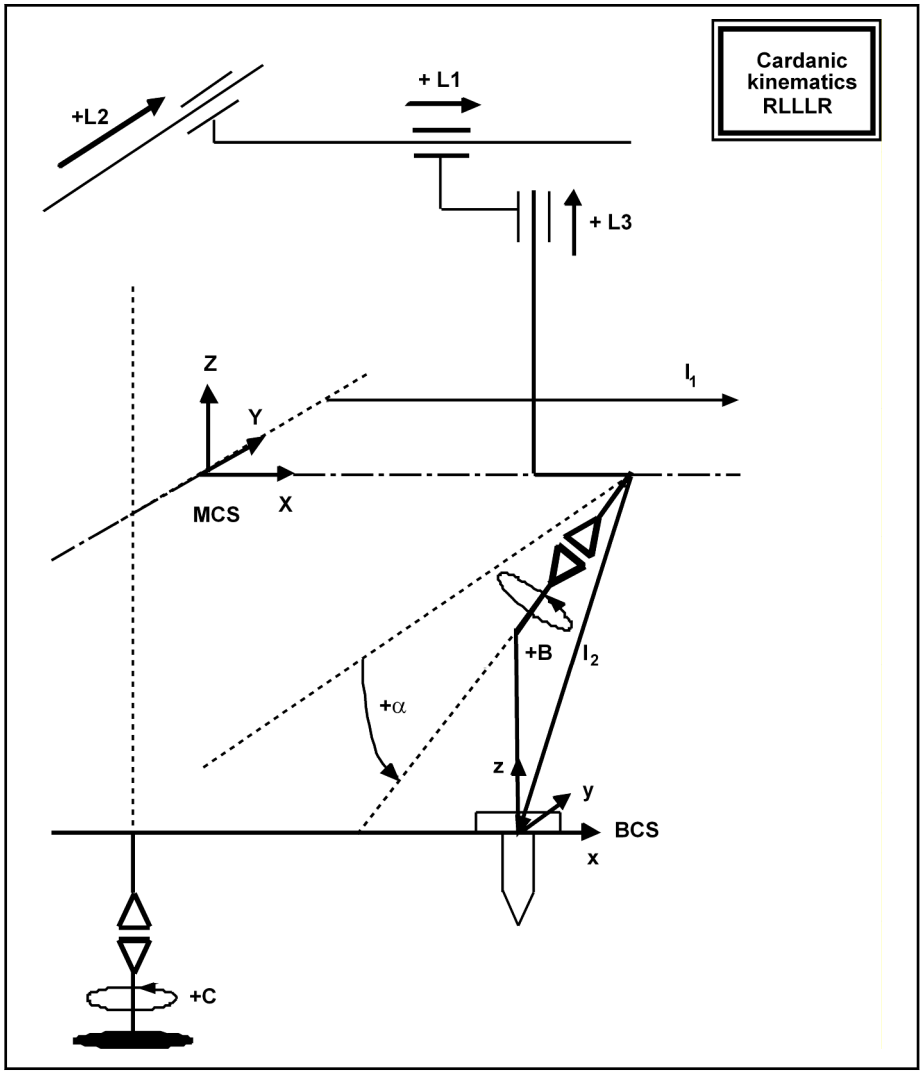


Fig. 11-61: Cardanic CLLLB (reference position)

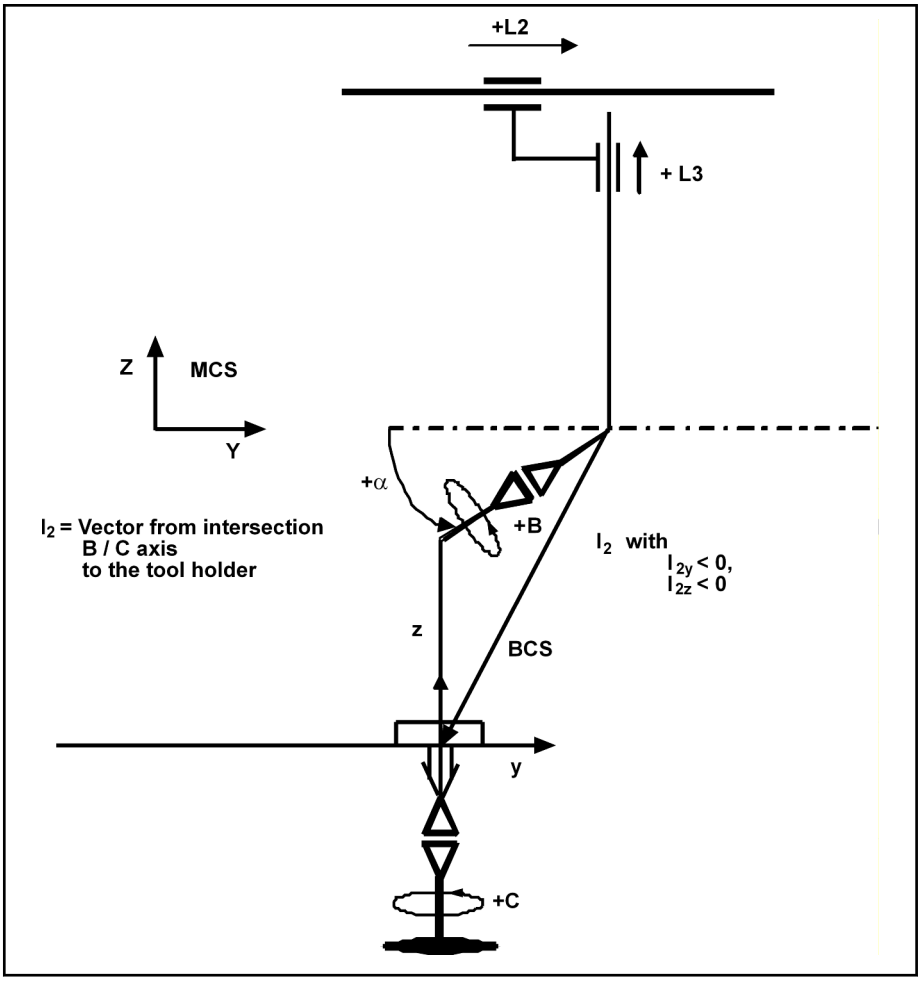


Fig. 11-62: Cardanic CLLLB (side view: reference position)

The angle of inclination α and the vectors $\vec{i}_1$ and $\vec{i}_2$ define the geometry of the 5-axis kinematics.

$\vec{i}_1$ is a vector along X; from the R_1 axis (C-axis) to the R_2 axis (B-axis) (that means $I_{1x} \neq 0$, $I_{1y} = I_{1z} = 0$).

$\vec{i}_2$ is the $\vec{i}_1$ continuation vector of the R_2 axis (B-axis) to the zero point of the BCS (that means $I_{1x} \neq 0$, $I_{1y} \neq 0$, $I_{1z} \neq 0$).

Due to the inclined b-axis, the angle θ between z_{BCS} and the tool symmetry axis is not larger than $\theta_{max} = 2(90^0-\alpha)$

Properties of the axistransformations:

When the 5-axis transformation is active, the coordinates x, y, z, φ, θ of the basic coordinate system BCS are transformed into the machine coordinate system MCS.

Input variables are the linear coordinates x, y, z and the orientation coordinates φ and θ. When switching on the axis transformation, either the vector orientation movement (type 3232212) or the linear orientation movement (type 3232112) is activated.

The orientation vector $\vec{o}$ can be programmed as spline (see chapter 9.1.8 "Spline programming " on page 143).

There are different properties depending on the orientation type

Vector orientation:

In case of the vector orientation motion, every orientation change is caused by programming φ, θ or a respective alternative syntax (s. spec. orientation motion) by rotating the orientation vector $\vec{\rho}$ (φ,θ) around a rotary axis which is fixed in space (large circle motion of the vector point). φ,θ can assume values of the following range:

φ ε [0,+360] , θ ε [0,θ_{max}].

The rotary axes B and C can be endless or rotary axes.

Linear orientation:

The angles φ and θ are interpolated linearly from the starting to the end point. The behavior of the "shortest way"-logic depends on the programming start and the orientation motion. In case of the φ-, θ-programming, the "shortest way"-logic is switched off. In case of the O(...) programming, the "shortest way" is switched on. φ,θ can be provided with values of the following range:

φ ε [-infinite, +infinite] , θ ε [-θ_{max}, θ_{max}].

The rotary axes B and C have to be configured as rotary axes (no modulo endless axis).

Restriction

Online correction of φ and θ is not possible.

Relevant NC functions

COORD(<i>)	i=1..20 is used to activate an axis transformation which is defined in MP 1030 00110.
COORD(0,2)	The active transformation is deactivated.

Tab. 11-70: Relevant NC functions

Relevant CPL functions

SDR(300)	Active Epsilon environment. The active value can be greater than the value set in EpsAx-Trafo (MP 1030 00160).

Tab. 11-71: Relevant CPL functions

Relevant machine parameters

1030 00110	Defines the transformation type.
1030 00120	Defines the system axes/coordinates of the transformation which are involved in the transformation.
1030 00125	Defines the rotary axes involved in the transformation, including the direction of rotation and the traveling direction of the linear axes.
1030 00130	Defines the axis positions of the reference position
1030 00140	Defines the length and angle parameters
1030 00160	Epsilon environments

Tab. 11-72: Relevant machine parameters

Reference position

The reference position of the Cardanic 5-axis transformation RLLLR is shown in fig. 11-61 "Cardanic CLLLB (reference position) " on page 319 and in fig. 11-62 "Cardanic CLLLB (side view: reference position) " on page 320. In the reference position of the configuration (MCS: X, Y, Z, R_1 = C and R_2 = B), the tip (TCP) of the rotation-symmetric tool (cutter or laser), based on the basic coordinate system (BCS) is oriented along the z-coordinate. The axis transformation unambiguously determines the orientation of the BCS (see illustration)

In this position, the BCS coordinates (x,y,z,φ,θ) have to be equal to (0,0,0,0,0), with the respective MCS coordinates (X, Y, Z, R_2, R_1) = (X^0, Y^0, Z^0, R_2^0, R_1^0) of MP 1030 00130.

Tool correction

The axis transformation supports a tool correction in the tool coordinate system (TCS). In the reference position, the TCS matches the BCS. The tool correction vector $\vec{l}_t$ points from the corrected tool center point TCP_1 to the uncorrected TCP_0. It comprises the length corrections L1, L2 and L3 from the D-correction table.

Special features

- **Ambiguous axis positions:**

  Generally, a TCP position and a tool orientation can be realized by two different axis positions (transformation branches). The branches differ in their b-axis positions.

  Vector orientation:

  Which one of the two poses (transformation branches) is used is decided on activation of the axis transformation. Accordingly, a transition of branch 1 to -1 and vice versa is only possible when the orientation location θ = 0 (north pole singularity) or $\theta=\theta_{max}$(equator singularity) is traveled.

  Linear orientation:

  The transformation branch is always chosen in a way that the b-axis and the orientation angle θ have the same sign, i.e. the following applies:

  – θ≥0 ≙ B≥0 (branch 1)

  – θ≤0 ≙ B≤0 (branch -1)

  Therefore, a backwards transformation can be solved uniquely. Accordingly, a transition from branch 1 to -1 and vice versa is possible when traveling through the orientation location θ =0 (north pole singularity).

- **North pole singularity:**

  Vector orientation:

  The north pole is a singularity.

  In the north pole, the machine parameter 103000160 is effective in the Epsilon environments; i.e. orientation motions through the ε-environment are lead through θ =0.

  If θ =0, φ can be programmed, but no motion takes place.

  Linear orientation:

  The north pole is no singularity.

The north pole can be traveled without problems. The machine parameter 103000160 Epsilon environments is not effective, i.e. orientation motions through the ε-environment are not imperatively lead through point θ =0.

If θ =0, φ can be programmed and the desired motion (rotation of the c-axis) is performed.

- **Equator singularity:**

  In case of orientation motions which go tangentially through the parallel of latitude $\theta=\theta_{max}$, the transformation branch is only changed if the b-axis has a traversing range of at least B ε [-360°, + +360°]. The traversing range is defined by the negative and positive first limit switch of the b-axis.

11.17.2 Handling Instruction: 5-Axis Transformation Cardanic RLLLR

Applying

The axis transformation can be applied via the relevant machine parameters. Therefore, the axis transformation function has to be activated via the parameter **tra** "Axis transformations" in the setup (SUP). The path of the machine parameter name is always "TRA/AxTrafo[1..20]/...".

Alternatively, the axis transformation can be applied via a user-defined system date of type SysAxTrafo_t.

The content of both variant is identical.

The axis positions of the reference position have to be determined.

The respectively equivalent machine parameter numbers, machine parameter names and SD elements are specified in the following table. To apply a transformation, one of the variants has to be assigned accordingly

Transformation-relevant parameters

MP number	Description	
1030 00110	**Transformation type** MP name: TrafoType SD elem. : Type	
	3232212	Cardanic RLLLR with vector orientation
	3232112	Cardanic RLLLR with linear orientation
1030 00120	**System axes / coordinates of the transformation** MP name: FwdInCoordIndTrafo[1..8] SD elem. : AxisAssignment[1..8]	
	[1]	System axis number of C
	[2]	System axis number of X
	[3]	System axis number of Y
	[4]	System axis number of Z
	[5]	System axis number of B
	[6..8]	Not relevant

<table>
<tr><th>MP number</th><th>Description</th></tr>
<tr><td>1030 00125</td><td>Axis classification of transformation axes
MP name: CoordClass[1..8]
MP name: CoordDir[1..8]
SD elem. : AxisClassification[1..8]</td></tr>
<tr><td></td><td>Defines the axes involved in transformation, including their direction of motion.
• MP no. or SD element:
Value ">0" means:
The axis rotates/travels positively as in the figures above.
Value "<0" means:
The axis rotates/travels negatively as in the figures above.
Possible values are:
– +/-1 = x-axis
– +/-2 = y-axis
– +/-3 = z-axis
– +/-100 = a-axis
– +/-200 = b-axis
– +/-300 = c-axis
• MP name:
– Possible values for CoordClass are
– X,Y,Z (linear)
– A,B,C (rotary)
– Possible values for CoordDir are:
– positive (regarding Fig. "Reference position"
– negative (regarding Fig. "Reference position"</td></tr>
<tr><td></td><td>If a parameter is not set, the following default settings apply:
<table><tr><td></td><td>MP no /
SD elem.:</td><td>CoordClass:</td><td>CoordDir:</td></tr><tr><td>[1]</td><td>300</td><td>C</td><td>Positive</td></tr><tr><td>[2]</td><td>1</td><td>X</td><td>Positive</td></tr><tr><td>[3]</td><td>2</td><td>Y</td><td>Positive</td></tr><tr><td>[4]</td><td>3</td><td>Z</td><td>Positive</td></tr><tr><td>[5]</td><td>200</td><td>B</td><td>Positive</td></tr><tr><td>[6..8]</td><td>Not relevant</td><td>0</td><td>Positive</td></tr></table></td></tr>
</table>

MP number	Description	
1030 00130	**Axis positions of the reference position** MP name: RefPosTrafo[1..8] SD elem. : AxZeroPos[1..8]	
	[1]	C^0
	[2]	X^0
	[3]	Y^0
	[4]	Z^0
	[5]	B^0
	[6..8]	Not relevant
1030 00140	**Length and angle parameters** MP name: JointParTrafo[1..16] SD elem. : LenParam[1..16]	
	Defines the length vectors l_1, l_2 in [mm].	
	[1]	l_{1x}
	[2]	Not relevant
	[3]	Not relevant
	[4]	l_{2x}
	[5]	l_{2y}
	[6]	l_{2z}
	[7]	α
	[8..16]	Not relevant
1030 00160	**Epsilon environments** MP name: EpsAxTrafo SD elem. : EpsilonRanges	
	0.01	

Tab. 11-73: Transformation-relevant parameters

Activating If the 5-axis transformation is configured via

- Machine parameters. Then, the transformation is enabled in the NC program with (i=1,..20):

  COORD(<i>)

- System date. Then, the transformation is enabled in the NC program with (type "SysAxTrafo_t"):

  COORD(<SD.MtbAxTrafo>)

Example Program:

The 5-axis transformation is defined in the fourth machine parameter block. No axis transformation is active at programming start:

Enable transformation

```
N100  G1 F1000 XA10 YA10 C30   ;Axes XA, YA and C are traversed
N110  COORD(4)                 ;Switching on the axis transformation
N130  x20 y30 phi0             ;the coordinates x,y,phi are traversed
N140  x10 z10 theta45          ;the coordinates x,z,theta are traversed
```

```
        :
N999  COORD(0)                ;Switching-off the axis transformation
                              ;Axes are again programmed from here
```

Deactivating **COORD(0)** or **COORD(0,2)**

switches off the axis transformation in the NC program.

11.18 5-axis transformation - Type 7: Tripod transformation with wobble plate

11.18.1 Description

Function

Transformation name	Transformation type	Point of action	Coordinate system
Tripod transformation with wobble plate	3250201	AT2	MCS,BCS

Tab. 11-74: Function

Output variables in the MCS: X, Y, Z1, Z2, Z3

Input variables in the BCS: x, y, z, φ, θ

The axis kinematics of the tripod transformation with wobble plate consists of five linear axes (x, y, z1, z2, z3).

The axis transformation is used at the point of action 2, i.e., it is applied between the machine coordinate system MCS = (X, Y, Z1, Z2, Z3) and the basic coordinate system BCS = (x, y, z, φ, θ).

When activating the axis transformation, the vector orientation is enabled.

The x-axis and the y-axis and the three z-axes (parallely to each other) are positioned like a right-handed Cartesian coordinate system.

The three z-axes move the plane parallel workpiece tabletop (called worktop in the following) using a tripod rod kinematics. The three rods of the tripod are attached to the worktop at their respective lower side using ball joints and attached to the respective z-axes using hinge joints.

There is an xy-sliding unit above the tripod. It moves the rotation-symmetric tool vertically towards the bottom, (i.e. in negative z-direction).

The zero point of the basic coordinate system (BCS) is on the upper side of the worktop and is fixedly connected to it. x- and y-direction of the BCS are in the plane of the upper side. The z-direction of the BCS points away upwards from the worktop.

The motion of the three z-axes moves the worktop in ±x-, ±y- and ±z-direction and rotates relatively to the tool tip around these directions.

Moving the axes x and y can compensate the worktop motion in ±x- und ±y-direction (wobbling)

When moving the five axes x, y, z1, z2 and z3, the programmed position (x, y, z) and the orientation (φ, θ) of the tool in the BCS is reached within the geometric system limits.

Restriction

Online correction of φ and θ is not possible.

Relevant NC functions

COORD(<I>)	I=1..20 is used to activate an axis transformation which is defined in MP 1030 00110.
COORD(0,2)	The active transformation is deactivated.

Tab. 11-75: Relevant NC functions

Relevant machine parameters

1030 00110	Defines the transformation type.
1030 00120	Defines the system axes/coordinates of the transformation which are involved in the transformation.
1030 00125	Defines the linear axes involved in the transformation; including their traversing direction
1030 00130	Defines the axis positions of the reference position
1030 00140	Defines the length and angle parameters
1030 00160	Epsilon environments

Tab. 11-76: Relevant machine parameters

Reference position

In reference position, the worktop is horizontal. The basic coordinate system in reference position is called BCS_{ref}.

The x-direction of the BCS_{ref} is parallel to the direction of the x-axis.

The y-direction of the BCS_{ref} is parallel to the direction of the y-axis.

The z-direction of the BCS_{ref} is parallel to the direction of the three z-axis.

The three ball joints WG1, WG2 and WG3 are attached at the bottom side of the worktop. Their center points are at the following BCS_{ref} positions:

P_{BCSref}(WG1) = (-WR sqrt(3)/2, -WR/2, -WD)

P_{BCSref}(WG2) = (0, WR, -WD)

P_{BCSref}(WG3) = (WR sqrt(3)/2, -WR/2, -WD)

Specify the parameters WR and WD in the transformation parameters.

WR = Worktop form radius

WD = Effective worktop thickness

The three ball joints form an equilateral triangle. The WCP center point of this triangle has the BCS_{ref} position:

P_{BCSref}(WCP) = (0, 0, -WD)

Each of the three ball joints has the distance WR from WCP.

Rod 1 is attached to the worktop at one end using the ball joint WG1. The same applies to rod 2 and ball joint WG2 and to rod 3 and ball joint WG3. All three rods have the length SL. Specify it in the transformation parameters.

The other ends EG1, EG2 and EG3 of the three rods are at the following BCS_{ref} positions:

P_{BCSref}(EG1) = (-ER sqrt(3)/2, -ER/2, -ED)

P_{BCSref}(EG2) = (0, ER, -ED)

P_{BCSref}(EG3) = (ER sqrt(3)/2, -ER/2, -ED)

Specify the parameter ER in the transformation parameters.

ER = Frame radius

The parameter ED cannot be specified. It is better specified by:

$ED = WD + sqrt[SL^2-(ER-WR)^2]$

Thus, the following condition has to be met when specifying the parameters:

WR < ER < WR + SL

The points EG1, EG2 and EG3 form an equilateral triangle. The ECP center point of this triangle has the BCS_{ref} position:

$P_{BCSref}(ECP) = (0, 0, -ED)$

Each of the three points EG1, EG2 and EG3 has the distance ER from ECP.

The following four points are located in one plane: $P_{BCSref}(WG1)$, $P_{BCSref}(EG1)$, $P_{BCSref}(WCP)$, $P_{BCSref}(ECP)$

The following four points are located in one plane: $P_{BCSref}(WG2)$, $P_{BCSref}(EG2)$, $P_{BCSref}(WCP)$, $P_{BCSref}(ECP)$

The following four points are located in one plane: $P_{BCSref}(WG3)$, $P_{BCSref}(EG3)$, $P_{BCSref}(WCP)$, $P_{BCSref}(ECP)$

The three planes are called the rod planes 1, 2 and 3 and the intersection of all three rod planes, i.e. the straight line through the two points ECP and WCP, are called the symmetry axis of the system. The rods remain in their respective rod planes while moving the axes x, y, z1, z2 and z3.

There is a floating hinge pivot at the point EG1 that can be rotated around the following direction:

(-1/2, sqrt(3)/2, 0)

Rod 1 is attached to a sliding unit with this swivel joint. This sliding unit can thus be traversed by the z-axis z1 which is parallel to the BCS_{ref} z-direction.

Thus, there are hinge swivel joints at the points EG2 and EG3 which can be rotated around the following directions:

(1,0,0)

(½, sqrt(3)/2, 0)

They can be traversed through the axes z2 and z3.

The directions (around which the swivel joints can be rotated), ensure that the rods 1, 2 and 3 remain in their respective rod plane.

In the reference position, the tool reference point (spindle nose) is on the BCS_{ref} position (0, 0, 0). As the workpiece table is horizontal, an exchanged tool had the orientation (0, 0, 1).

The required positions of the axes, x, y, z1, z2 and z3 are called axis positions of the reference position: X_{ref}, Y_{ref}, $Z1_{ref}$, $Z2_{ref}$, $Z3_{ref}$

Specify them in the transformation parameters.

Geometric limits

Specified geometric limits of the system:

The rods always have to be inclined to the inside directed towards the symmetry axis of the system. A vertically positioned rod specifies the limit.

The angle between a rod and a vertical to the worktop has to be lower than 90°.

A smaller critical angle might be especially required for the second angle. Thus, you can specify the angle in the transformation parameters. Only values below 90° are allowed.

Tool correction

Axis transformation supports a tool correction in the tool coordinate system (TCS). In the reference position, the TCS matches the BCS: TCS = BCS_{ref}

The tool correction consists of the length corrections L1, L2 and L3 from the D-correction table and the basic tool correction TCB to be specified as parameter of the transformation.

The basic tool correction becomes active automatically when activating the axis transformation.

The additional tool correction with L1, L2 and L3 is activated using the NC command: G47(XTR,YTR,ZTR)

If the axes are on their reference positions, the tool tip is on the following BCS_{ref} position when the tool correction is active:

P_{BCSref}(TCP) = (-L1, -L2, -L3-TCB)

As the worktop is horizontal for these axis positions, the tool orientation is (0, 0, 1).

The tool correction vector (L1, L2, L3+TCB) shows the tip of the tool to the tool reference point.

11.18.2 Handling instruction: Tripod transformation with wobble plate

Applying

The axis transformation can be applied via the relevant machine parameters. Therefore, the axis transformation function has to be activated via the parameter **tra** "Axis transformations" in the setup (SUP). The path of the machine parameter name is always "TRA/AxTrafo[1..20]/...".

Alternatively, the axis transformation can be applied via a user-defined system date of type SysAxTrafo_t.

The content of both variants is identical.

The axis positions of the reference position have to be determined.

The respectively equivalent machine parameter numbers, machine parameter names and SD elements are specified in the following table. To apply a transformation, one of the variants has to be assigned accordingly.

Transformation-relevant parameters

MP number	Description	
1030 00110	**Transformation type** MP name: TrafoType SD elem.: Type	
	3250201	

MP number	Description			
1030 00120	**System axes/coordinates of the transformation** MP name: FwdInCoordIndTrafo[1..8] SD elem.: AxisAssignment[1..8]			
	[1]	System axis number of X		
	[2]	System axis number of Y		
	[3]	System axis number of Z1		
	[4]	System axis number of Z2		
	[5]	System axis number of Z3		
	[6..8]	Not relevant		
1030 00125	**Axis classification of transformation axes** MP name: CoordClass[1..8] MP name: CoordDir[1..8] SD elem.: AxisClassification[1..8]			
	If the positive traversing directions of all five axes correspond to the previously described reference position, you do not have to enter anything. The preceding zeroes can remain. • –			
	If this is not the case, enter the following first:			
		MP no / SD elem.:	CoordClass:	CoordDir:
	[1]	1	X	Positive
	[2]	2	Y	Positive
	[3]	3	Z1	Positive
	[4]	3	Z2	Positive
	[5]	3	Z3	Positive
	[6..8]	Not relevant	0	Positive
	For each axis with a positive traversing direction not corresponding to the previously described reference position, that is in opposite direction, enter a minus for the MP no./SE elem. or “negative” for CoordDir. Example: x and z2 run in reverse direction.			
		MP no / SD elem.:	CoordClass:	CoordDir:
	[1]	-1	X	Negative
	[2]	2	Y	Positive
	[3]	3	Z	Positive

MP number	Description			
	[4]	-3	Z	Negative
	[5]	3	Z	Positive
1030 00130	**Axis positions of the reference position** MP name: RefPosTrafo[1..8] SD elem.: AxZeroPos[1..8]			
	[1]	X_{ref}		
	[2]	Y_{ref}		
	[3]	$Z1_{ref}$		
	[4]	$Z2_{ref}$		
	[5]	$Z3_{ref}$		
	[6..8]	Not relevant		
1030 00140	**Length and angle parameters** MP name: JointParTrafo[1..16] SD elem.: LenParam[1..16]			
	Defines the geometry of the kinematics in mm or °.			
	[1]	Rod length		
	[2]	Worktop form radius		
	[3]	Frame radius		
	[4]	Effective worktop thickness		
	[5]	Maximum angle between the rod and the vertical on the worktop, to the top		
	[6]	Basic tool correction		
	[7..16]	Not relevant		
1030 00160	**Epsilon environments** MP name: EpsAxTrafo SD elem.: EpsilonRanges			
	Not relevant			

Tab. 11-77: Transformation-relevant parameters

Activating

If the 5-axis transformation is configured via

- machine parameter, the transformation is enabled in the NC program with (i=1,..20):

  COORD(<i>)

- system date, the transformation is enabled in the NC program with (type "SysAxTrafo_t"):

  COORD(<SD.MtbAxTrafo>)

Example program:

The 5-axis transformation is defined in the fourth machine parameter block. No axis transformation is active at programming start:

Enable transformation

Program:

```
N100  G1 F1000 X10 Y11 Z1=20 Z2=21 Z3=23    ; Move to starting positions
N110  COORD(4)                             ; Transformation
N130  x20 y30 z15 phi0  theta17
N140  x10 y20 z10 phi11 theta45
        :
N999  COORD(0)                             ; Transformation off
```

Program:

Deactivating

COORD(0) or **COORD(0,2)**

switches off the axis transformation in the NC program.

11.19 5-axis transformation - Type 8: Tripod transformation with wobble plate and additional c-axis for the tangential tool guidance

11.19.1 Description

Function

Transformation name	Transformation type	Point of action	Coordinate system
Tripod transformation with wobble plate and additional c-axis for a tangential tool guidance	3250202	AT2	MCS,BCS

Tab. 11-78: Function

Output variables in the MCS: X, Y, Z1, Z2, Z3

Input variables in the BCS: x, y, z, φ, θ

This transformation completely corresponds to the additional c-axis for the tangential tool guidance of the tripod transformation with wobble plate (without path slave axis) as described in the previous section (without additional c-axis).

The additional c-axis can be used to rotate the tool.

After activating the transformation, the c-axis responds like every other axis that does not correspond to the transformation. It can be programmed directly.

Activate the calotte transformation, as the tangential tool guidance is not possible without the calotte transformation. The c-axis can still be programmed directly. The “MTX Functional Description - Basics” describes the calotte transformation.

With the NC command TTS(1), the tangential tool guidance and the c-axis becomes part of the transformation. It cannot be programmed directly anymore, but the transformation always rotates by the transformation that it follows the tangent of the contour on the workpiece programmed in the BCS.

The tangential tool guidance rotates the tool to the correct starting direction during the activation. If the programmed contour contains corners, the tangential tool guidance rotates the tool in the corner from the direction at the end of the pre-corner block to the direction of the beginning of the post-corner block. In order not to damage the workpiece, it is reasonable to round corners using the NC functions CHL and/or SCO described in the programming manual.

The position of the c-axis required to rotate the tool to the direction of the x-axis is called as c-axis position of the reference position: C_{ref}

Specify it in the transformation parameters.

A driven cutting wheel can thus be used as tool for example if it can cut a contour into the workpiece like a circular saw. The transformation always rotates the cutting wheel passing along the programmed contour towards the current tangent of the contour.

With the NC command TTS(0), tracking the cutting wheel using the transformation is disabled. The c-axis can now be programmed again directly when the transformations are active.

11.19.2 Handling instruction: Tripod transformation with wobble plate and additional c-axis for a tangential tool guidance

Applying

The axis transformation can be applied via the relevant machine parameters. Therefore, the axis transformation function has to be activated via the parameter **tra** "Axis transformations" in the setup (SUP). The path of the machine parameter name is always "TRA/AxTrafo[1..20]/...".

Alternatively, the axis transformation can be applied via a user-defined system date of type SysAxTrafo_t.

The content of both variants is identical.

The axis positions of the reference position have to be determined.

The respectively equivalent machine parameter numbers, machine parameter names and SD elements are specified in the following table. To apply a transformation, one of the variants has to be assigned accordingly.

Transformation-relevant parameters

MP number	Description	
1030 00110	**Transformation type** MP name: TrafoType SD elem.: Type	
	3250202	
1030 00120	**System axes/coordinates of the transformation** MP name: FwdInCoordIndTrafo[1..8] SD elem.: AxisAssignment[1..8]	
	[1]	System axis number of X
	[2]	System axis number of Y
	[3]	System axis number of Z1
	[4]	System axis number of Z2
	[5]	System axis number of Z3
	[6]	System axis number of C
	[7..8]	Not relevant

<table>
<tr><th>MP number</th><th colspan="4">Description</th></tr>
<tr><td rowspan="11">1030 00125</td><td colspan="4">Axis classification of transformation axes
MP name: CoordClass[1..8]
MP name: CoordDir[1..8]
SD elem.: AxisClassification[1..8]</td></tr>
<tr><td colspan="4">If the positive traversing directions of all six axes correspond to the previously described reference position, you do not have to enter anything. The preceding zeroes can remain.
• –</td></tr>
<tr><td colspan="4">If this is not the case, enter the following first:</td></tr>
<tr><td></td><td>MP no /
SD elem.:</td><td>CoordClass:</td><td>CoordDir:</td></tr>
<tr><td>[1]</td><td>1</td><td>X</td><td>Positive</td></tr>
<tr><td>[2]</td><td>2</td><td>Y</td><td>Positive</td></tr>
<tr><td>[3]</td><td>3</td><td>Z1</td><td>Positive</td></tr>
<tr><td>[4]</td><td>3</td><td>Z2</td><td>Positive</td></tr>
<tr><td>[5]</td><td>3</td><td>Z3</td><td>Positive</td></tr>
<tr><td>[6]</td><td>30</td><td>C</td><td>Positive</td></tr>
<tr><td>[7..8]</td><td>Not relevant</td><td>0</td><td>Positive</td></tr>
<tr><td rowspan="10"></td><td colspan="2">For each axis with a positive traversing direction not corresponding to the previously described reference position, that is in opposite direction, enter a minus for the MP no./SE elem. or “negative” for CoordDir.
Example: y, z1 and c run in reverse direction.</td><td colspan="2"></td></tr>
<tr><td></td><td>MP no /
SD elem.:</td><td>CoordClass:</td><td>CoordDir:</td></tr>
<tr><td>[1]</td><td>1</td><td>X</td><td>Positive</td></tr>
<tr><td>[2]</td><td>-2</td><td>Y</td><td>Negative</td></tr>
<tr><td>[3]</td><td>-3</td><td>Z</td><td>Negative</td></tr>
<tr><td>[4]</td><td>3</td><td>Z</td><td>Positive</td></tr>
<tr><td>[5]</td><td>3</td><td>Z</td><td>Positive</td></tr>
<tr><td>[6]</td><td>-30</td><td>C</td><td>Negative</td></tr>
</table>

MP number	Description	
1030 00130	**Axis positions of the reference position** MP name: RefPosTrafo[1..8] SD elem.: AxZeroPos[1..8]	
	[1]	X_{ref}
	[2]	Y_{ref}
	[3]	$Z1_{ref}$
	[4]	$Z2_{ref}$
	[5]	$Z3_{ref}$
	[6]	C_{ref}
	[7..8]	Not relevant
1030 00140	**Length and angle parameters** MP name: JointParTrafo[1..16] SD elem.: LenParam[1..16]	
	Defines the geometry of the kinematics in mm or °.	
	[1]	Rod length
	[2]	Worktop form radius
	[3]	Frame radius
	[4]	Effective worktop thickness
	[5]	Maximum angle between the rod and the vertical on the worktop, to the top
	[6]	Basic tool correction
	[7..16]	Not relevant
1030 00160	**Epsilon environments** MP name: EpsAxTrafo SD elem.: EpsilonRanges	
	Not relevant	

Tab. 11-79: Transformation-relevant parameters

Activating

If the 5-axis transformation is configured via

- machine parameter, the transformation is enabled in the NC program with (i=1,..20):

  COORD(<i>)

- system date, the transformation is enabled in the NC program with (type "SysAxTrafo_t"):

  COORD(<SD.MtbAxTrafo>)

The c-axis tracking is enabled by: TTS(1)

The c-axis tracking is disabled by: TTS(0)

Example program:

The 5-axis transformation is defined in the fourth machine parameter block. No axis transformation is active at programming start:

Enable transformation

Program:

```
N005  X0 Y0 Z0=-500 Z1=-500 Z2=-500 C10 G1 F1000
; Traverse the tripod to a horizontal position
; and the tool is positioned above the calotte apex.

N010  Coord(1)
; Activate the tripod transformation with wobble plate
; and additional c-axis for the tangential tool guidance.

N020  G1 x0 y0 z100 O(0,0,1)
; Move to a safe position vertically above the apex
; of the calotte with orientation along the BCS-z-direction.

N030  CaloGeo(200,50,100,20)
; Specifies the geometric calotte properties.

N040  CaloCrd(0,0,0)
; Activates the calotte transformation with a programming plane,
; touching the calotte in its apex
; without an additional angle of rotation.

N050  z0
; On the symmetry axis of the calotte, move from the safe position
; down to the calotte.

N055  TTS(1)
; Activate the tangential tool guidance. Thus, the tool
; rotates towards te starting tangent of the next block.

N060  x-30 y50
; Move to any position on the calotte. The tools is oriented
; along the tangent on this path.
; At the end of this block, the tool rotates towards
; the starting tangent of the next block.

N070  x80  y70
; Move to another position on the calotte. The tool is oriented
; along the tangent on this path.

N075  TTS(0)
; Deactivate the tangential tool guidance.

N080  CaloCrd(40,50,30)
; Change the programming plane. This includes
; a deactivation of the current calotte transformation
; and a subsequent re-activation with the new programming plane.

N085  TTS(1)
; Activate the tangential tool guidance. Thus, the tool
; rotates towards the starting tangent of the next block.

N090  x-50 y20
; Move to another position on the calotte. The tool is oriented
; along the tangent on this path.

N095  TTS(0)
; Deactivate the tangential tool guidance.

N100  CaloCrd(0,0,0)
; Change to the proramming plane that touches
; the calotte in its apex
; without an additional twisting angle.

N110  x0 y0
; Move to the calotte apex.

N120  z100
; Return to the safe position.

N130  G153
; Deactivate the calotte transformation.

N140  Coord(0)
; Deactivate the 5-axis transformation.
```

```
N150  M30
```

Program:

Deactivating **COORD(0)** or **COORD(0,2)**
switches off the axis transformation in the NC program.

11.20 3-axis transformation LLR

11.20.1 Description

Function

This transformation is used at point of action 2. This is to say it acts between the machine coordinate system (MCS) and the basic coordinate system (BCS).

Two linear coordinates and a rotary coordinate are required from the MCS. In the following, they are described as X_{MCS}, Y_{MCS}, and C_{MCS}, but they can optionally be given any other names by machine parameter.

It follows that there are 2 linear coordinates and one rotary coordinate in BCS which are called x_{BCS} , y_{BCS} , φ_{BCS} in the following. These coordinates can likewise be given any other names by machine parameters. (x_{BCS},y_{BCS}) is the position of the tool tip (Tool Center Point = TCP), and φ_{BCS} is the tool orientation, i.e. the angle between the x_{BCS} direction and the vector pointing from the tool tip along the tool axis in the direction of the tool holder.

The BCS is solidly connected with the workpiece.

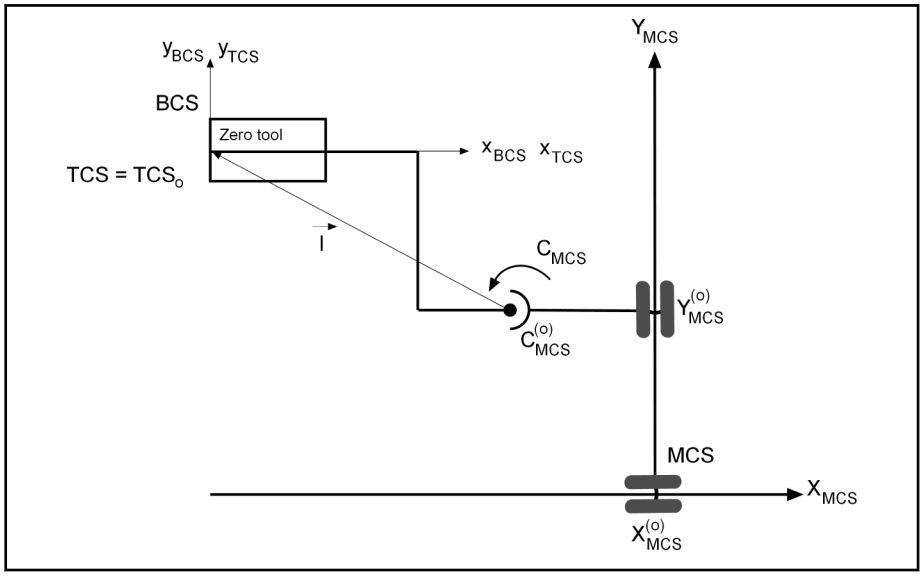


Fig. 11-63: 3-axis transformation - reference position with zero tool

In reference position (fig. 11-63 "3-axis transformation - reference position with zero tool" on page 337) the following applies:

- The tool orientation is defined by $\varphi_{BCS} = 0$.
- The vector from the center of rotation of the rotary MCS coordinate C_{MCS} to position $x_{BCS} = y_{BCS} = 0$ is specified by vector **l**.

- The components l_x and l_y of **l** in the BCS have to be entered in machine parameter JointParTrafo (1030 00 140). The above illustration shows $l_x<0$, $l_y>0$.
- The MCS positions are specified by $X_{MCS}^{(0)}$, $Y_{MCS}^{(0)}$, $C_{MCS}^{(0)}$, these values have to be entered in machine parameter RefPosTrafo (1030 00 130).
- The tip of the zero tool stands at $x_{BCS} = y_{BCS} = 0$ (definition of the zero tool). This case is shown in the illustration. For a tool with other tool lengths, the tool tip has a different position.

The tool coordinate system (TCS) has its zero point in the tool tip; it is solidly connected with the tool. For the zero tool, the TCS in reference position is congruent with the BCS; it is described as TCS_0. For a tool with other tool lengths, the TCS in its reference position is shifted in relation to the BCS.

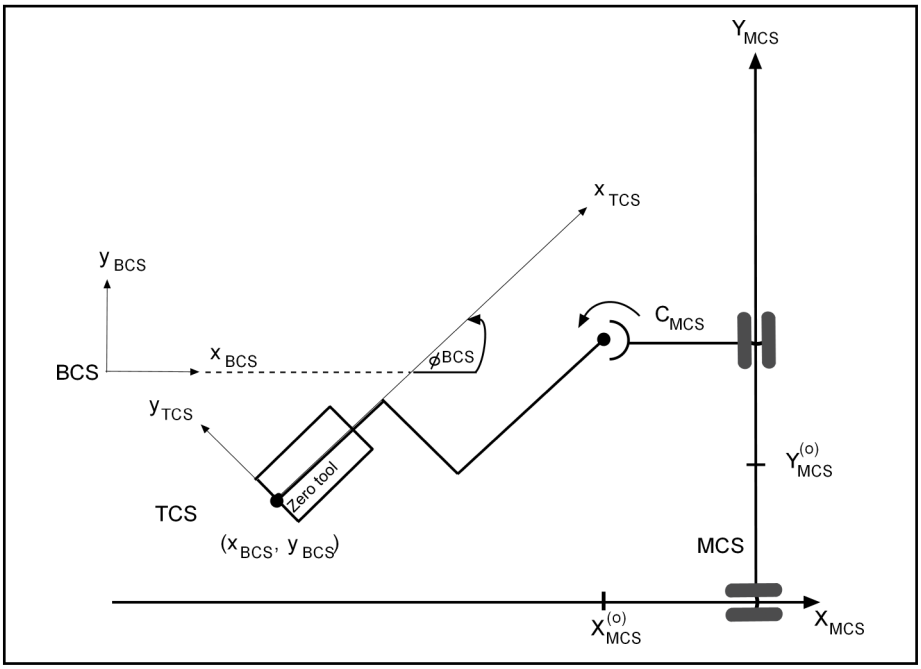


Fig. 11-64: 3-axis transformation - any position with zero tool

The figure fig. 11-64 "3-axis transformation - any position with zero tool" on page 338 shows any position of the axis transformation with the zero tool.

Axis transformation supports a tool correction in the (TCS). The tool correction vector $\mathbf{l_t}$ points from the current tool tip (TCP) to the tool tip of the zero tool (TCP_0). Its components l_{tx} , l_{ty} , l_{tz} are specified in the TCS. See figure fig. 11-65 "3-axis transformation - reference position with any tool" on page 339.

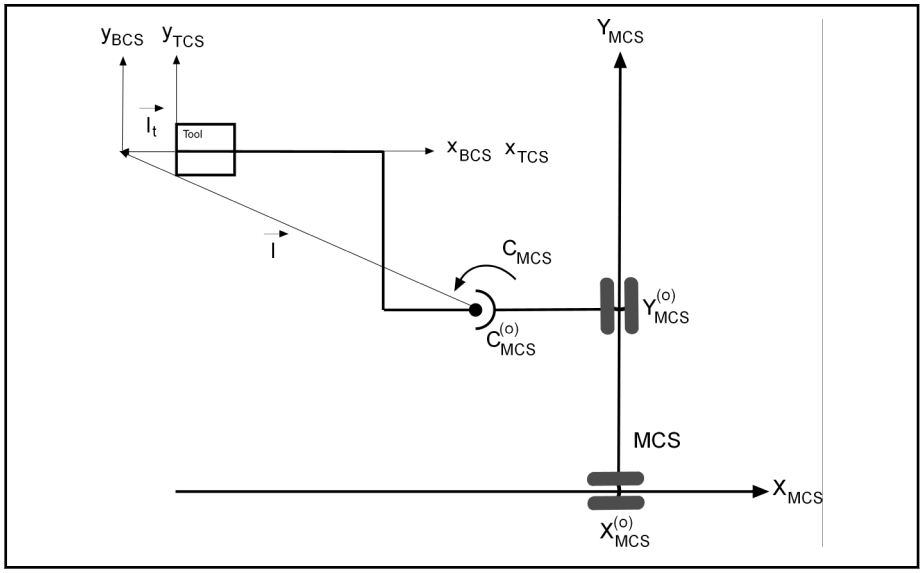


Fig. 11-65: 3-axis transformation - reference position with any tool

As the axis transformation includes only 2 linear coordinates of the BCS, generally only one of the 3 levels G17, G18, and G19 can be activated in the BCS. Whether G17, G18 or G19 is activated does not depend on the names of the BCS coordinates, which are specified here by way of example as x_{BCS} , y_{BCS} , φ_{BCS}; this is rather specified by the machine parameter FwdInCoord[1..3]/CoordClass (1030 00 125).

Relevant NC functions

Activation and deactivation of axis transformations is carried out via the NC command "Coord" which is described in the table below.

Coord(i)	The axis transformation with the number i (i=1,...,20) is activated. If this is to be the axis transformation described here, it has to be parameterized under number i in the machine parameters.
Coord(0)	If a transformation is active at point of action 2, it is deactivated.
Coord(0,2)	If a transformation is active at point of action 2, it is deactivated.

Tab. 11-80: 3-axis transformation: Enabling and disabling

Activation of the tool correction in TCS via the NC command G47 ({<TCS-Coordinate referring to L1>},{<TCS-Coordinate referring to L2>},{<TCS-Coordinate referring to L3>}). "XTR" stands for the coordinate named by the MP LinCoord[1] "YTR" for "LinCoord[2]" and "ZTR" for "LinCoord[3"].

The tool correction can also be related to the WCs coordinates. For the description, refer to the manual "MTX Functional Description - Basics", chapter "Tool corrections".

Relevant machine parameters

CHAN	Ch[k]	Coord	Mcs	CartCoord[1...6]
			Wcs	LinCoord[1..3]
				OriCoord[1...3]
TRA	AxTrafo[1...20]	TrafoType		

FwdInCoord	FwdInCoordIndTrafo[1...8]	
JointPar	JointParTrafo[1...8]	
RefPos	RefPosTrafo[1...8]	
FwdInCoordClass	FwdInCoord[1...8]	CoordClass
		CoordDir
	EpsAxTrafo	
TcsRefOri	TcsX	CoordTcsXBcs
		DirTcsXBcs
	TcsY	CoordTcsYBcs
		DirTcsYBcs
	TcsZ	CoordTcsZBcs
		DirTcsZBcs

Tab. 11-81: Tree structure of the machine parameters relevant for 3-axis transformation

7080 00020	Mcs	Machine coordinates
7080 00010	Wcs	Workpiece coordinates
1030 00110	TrafoType	Axis transformation type
1030 00120	FwdInCoord	Input axes/coordinates of the forward transformation
1030 00140	JointPar	Joint parameters (length/angle parameters)
1030 00130	RefPos	Axis/machine positions of the reference position
1030 00125	FwdInCoordClass	Processing Function of the input axes/ coordinates
1030 00150	TcsRefOri	Reference orientation of the tool coordinate system TCS

Tab. 11-82: Macoda numbers of the relevant machine parameters

11.20.2 Applying

To set the machine parameters, the reference position of the axis transformation has to be approached.

First, specify the number i (i=1...20) under which the axis transformation is to be activated with Coord(i). With this the following machine parameters are to be entered in the registry /TRA/AxTrafo[i]. "k" always designates the channel number:

TrafoType (1030 00 110)		
	For this axis transformation, 2121002 has to be entered here.	
FwdInCoordIndTrafo[1] (1030 00 120)		
	Number of the first MCS coordinate of the transformation (above, named X_{MCS})	
FwdInCoordIndTrafo[2] (1030 00 120)		
	Number of the second MCS coordinate of the transformation (above, named Y_{MCS})	
FwdInCoordIndTrafo[3] (1030 00 120)		
	Number of the third MCS coordinate of the transformation (above, named C_{MCS})	
For FwdInCoordIndTrafo[1] and FwdInCoordIndTrafo[2] (1030 00 120) the following numerical values are permissible::		
	-6:	This describes the MCS coordinate of an active axis transformation at point of action 1 which is named CartCoord[1] (7080 00 020).
	-5:	This describes the MCS coordinate of an active axis transformation at point of action 1 which is named CartCoord[2] (7080 00 020).
	-4:	This describes the MCS coordinate of an active axis transformation at point of action 1 which is named CartCoord[3] (7080 00 020).
	1...99:	The MCS coordinate corresponds to 1:1 (with the exception of the ZO) of a system axis, and defines the system axis number of this system axis. This axis has to be a linear axis.
For FwdInCoordIndTrafo[3] (1030 00 120) the following numerical values are permissible::		
	-3:	This describes the MCS coordinate of an active axis transformation at point of action 1 which is named CartCoord[4] (7080 00 020).
	-2:	This describes the MCS coordinate of an active axis transformation at point of action 1 which is named CartCoord[5] (7080 00 020).
	-1 :	This describes the MCS coordinate of an active axis transformation at point of action 1 which is named CartCoord[6] (7080 00 020).
	1...99:	The MCS coordinate corresponds to 1:1 (with the exception of the ZO) of a system axis, and defines the system axis number of this system axis. This axis has to be a rotary or endless axis.

Tab. 11-83: Applying

For the machine parameter FwdInCoord[1..3]/CoordClass (1030 00125), only the following settings (tab. 11-84 "Connection between the selectable plane and the parameters permissible for the TCS tool correction, and the machine parameters RefPosTrafo and LinCoord" on page 342) are permitted. Each column corresponds to one setting option. The effects on the function of the machine parameters LinCoord and OriCoord and on the selectable plane are shown below each entry.

Selectable plane and permissible parameter for the TCS tool correction

FwdInCoord[1]/CoordClass (1030 00 125)	X	Z	Y
FwdInCoord[2]/CoordClass (1030 00 125)	Y	X	Z
FwdInCoord[3]/CoordClass (1030 00 125)	C	B	A
Name of the lin. BCS main coordinate	LinCoord[1]	LinCoord[3]	LinCoord[2]
Name of the lin. BCS secondary coordinate	LinCoord[2]	LinCoord[1]	LinCoord[3]
Name of the rotary BCS coord.	LinCoord[4]	LinCoord[4]	LinCoord[4]
Single selectable plane	G17	G18	G19
Parameter of the TCS tool correction	XTR, YTR	ZTR, XTR	YTR, ZTR

Tab. 11-84: Connection between the selectable plane and the parameters permissible for the TCS tool correction, and the machine parameters RefPosTrafo and LinCoord

FwdInCoord[1]/CoordDir (1030 00 125)	Sign which considers the first MCS coordinate of the transformation.
FwdInCoord[2]/CoordDir (1030 00 125)	Sign which considers the second MCS coordinate of the transformation.
FwdInCoord[3]/CoordDir (1030 00 125)	Sign which takes the third MCS coordinate of the transformation into consideration.

Tab. 11-85: Sign

The illustrations in the above description of the basics of axis transformation proceed on the assumption that "positive" is entered in each case for FwdInCoord[1..3]/CoordDir (1030 00125). For a MCS coordinate which takes the opposite direction, "negative" has to be entered.

RefPosTrafo[1] (1030 00 130)	Reference position of the first MCS coordinate of the transformation (above, named $X_{MCS}^{(0)}$)
RefPosTrafo[2] (1030 00 130)	Reference position of the second MCS coordinate of the transformation (above, named $Y_{MCS}^{(0)}$)
RefPosTrafo[3] (1030 00 130)	Reference position of the third MCS coordinate of the transformation (above, named $C_{MCS}^{(0)}$)
JointParTrafo[1] (1030 00 140)	1. Length and angle parameter of the transformation (above, named l_x)
JointParTrafo[2] (1030 00 140)	2. Length and angle parameter of the transformation (above, named l_y)
CoordTcsXBcs, DirTcsXBcs (1030 00 150)	CoordTcsXBcs is the coordinate of the BCS (x, y or z) whose direction - specified by DirTcsXBcs - (positive or negative) coincides with the x-coordinate of the TCS in its reference position with zero tool.

CoordTcsYBcs, DirTcsYBcs (1030 00 150)	CoordTcsYBcs is the coordinate of the BCS (x, y or z) whose direction - specified by DirTcsYBcs - (positive or negative) coincides with the y-coordinate of the TCS in its reference position with zero tool.
CoordTcsZBcs, DirTcsZBcs (1030 00 150)	CoordTcsZBcs is the coordinate of the BCS (x, y or z) whose direction - specified by DirTcsZBcs - (positive or negative) coincides with the z-coordinate of the TCS in its reference position with zero tool.

Tab. 11-86: Applying

In the 3 illustrations above, the following case:

CoordTcsXBcs = x,

CoordTcsYBcs = y,

CoordTcsZBcs = z,

DirTcsXBcs = positive,

DirTcsYBcs = positive,

DirTcsZBcs = positive

.

11.20.3 Activation

1. Enter the machine parameters for the axis transformation.
2. NC restart.
3. Activate axis transformation with Coord(i).
4. Check all NC functions used in the NC program for their applicability when transformation 2121002 is active.

11.21 6-axis transformation

11.21.1 Description

General information

6-axis transformation is used in machines the tool of which

- has to simultaneously execute translatory and rotary path motions in space
- executes orientation motions which align the tool in each path point, in addition to the path motion

By means of the 6-axis transformation, the program coordinates are transformed into axis positions of the axes.

The 6-axis transformation that is realized in the NC comprises:

- three linear coordinates (e.g. x,y, z)
- three orientation coordinates φ, ϑ, ψ (z. B. phi, theta, psi)
- three linear axes (e.g. X, Y and Z)
- three rotary axes (e.g. A, B and C)

There are two 6-axis transformation versions (see chapter 11.21.2 "Variant 1 (with tensor orientation) " on page 344 and chapter 11.21.3 "Variant 2 (with linear motion of the rotary axes)" on page 345).

11.21.2 Variant 1 (with tensor orientation)

Function

Transformation name	Transformation type	Point of action	Coordinate system
6-axis transformation with tensor orientation	3333301	AT2	MCS, BCS

Tab. 11-87: Function - Variant 1

Output variables in the MCS: X,Y,Z,C,B,A

Input variables in BCS: x, y, z, φ, ϑ, ψ

Type 3333301 allows TCP programming via three linear coordinates and orientating the tool by programming Euler angles φ (phi), ϑ (theta) and ψ (psi) (see Euler angles chapter 10.2.2 "Euler Angles " on page 158).

The orientation motion is executed as a TCS rotation around a rotary axis with is fixed in space (see the following Figure).

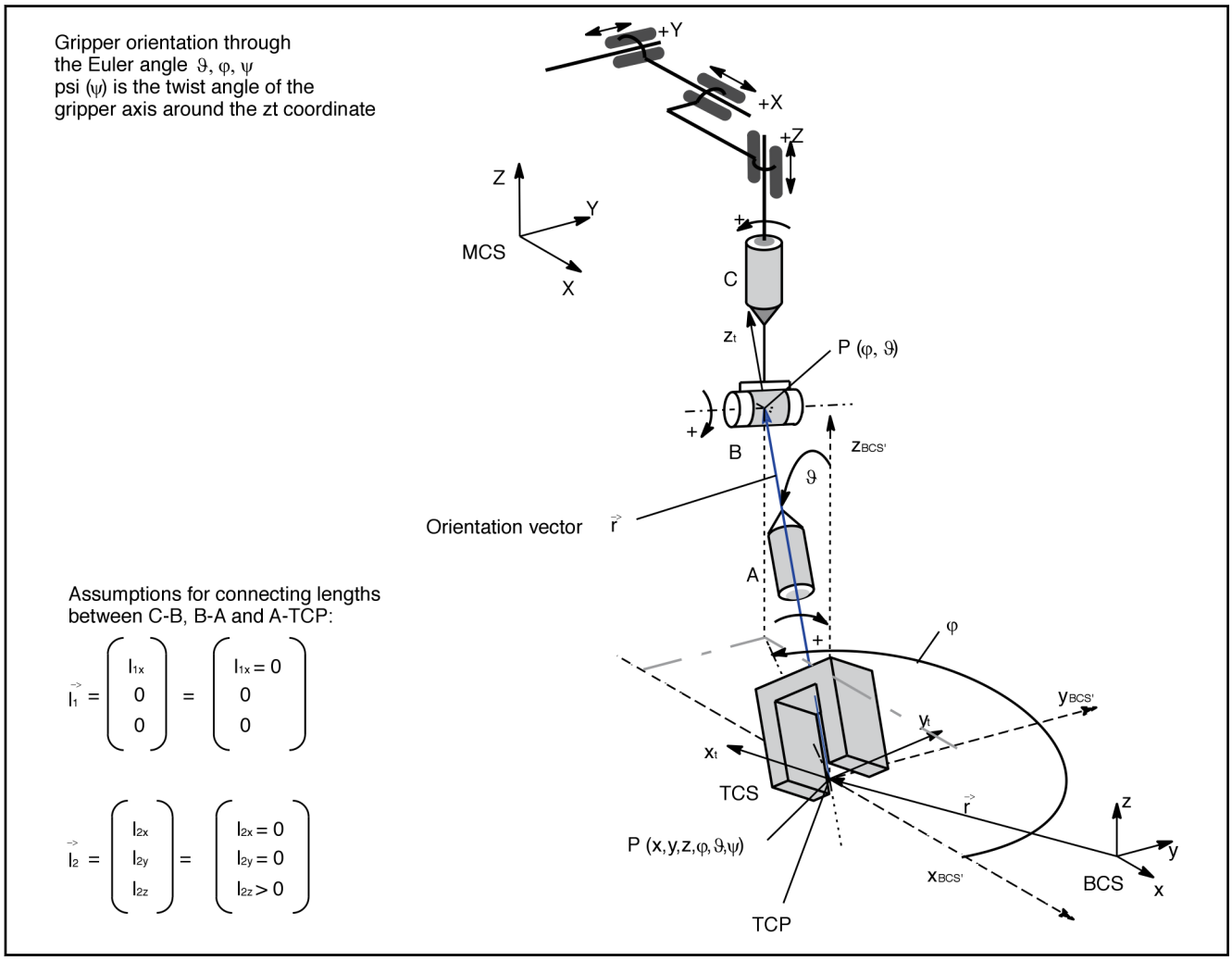


Fig. 11-66: Tensor orientation

11.21.3 Variant 2 (with linear motion of the rotary axes)

Function

Transformation name	Transformation type	Point of action	Coordinate system
6-axis transformation with linear motion of the rotary axes	3033001	AT2	MCS, BCS

Tab. 11-88: Function - Variant 2

Output variables in the MCS: X,Y,Z,C,B,A

Input variables in the BCS: x,y,z,C,B,A

Type 3033001 supports the TCP programming using three linear coordinates and the orientation of the tool by programming the three rotary axes.

The orientation motion is executed in a linear fashion in the rotary axis positions.

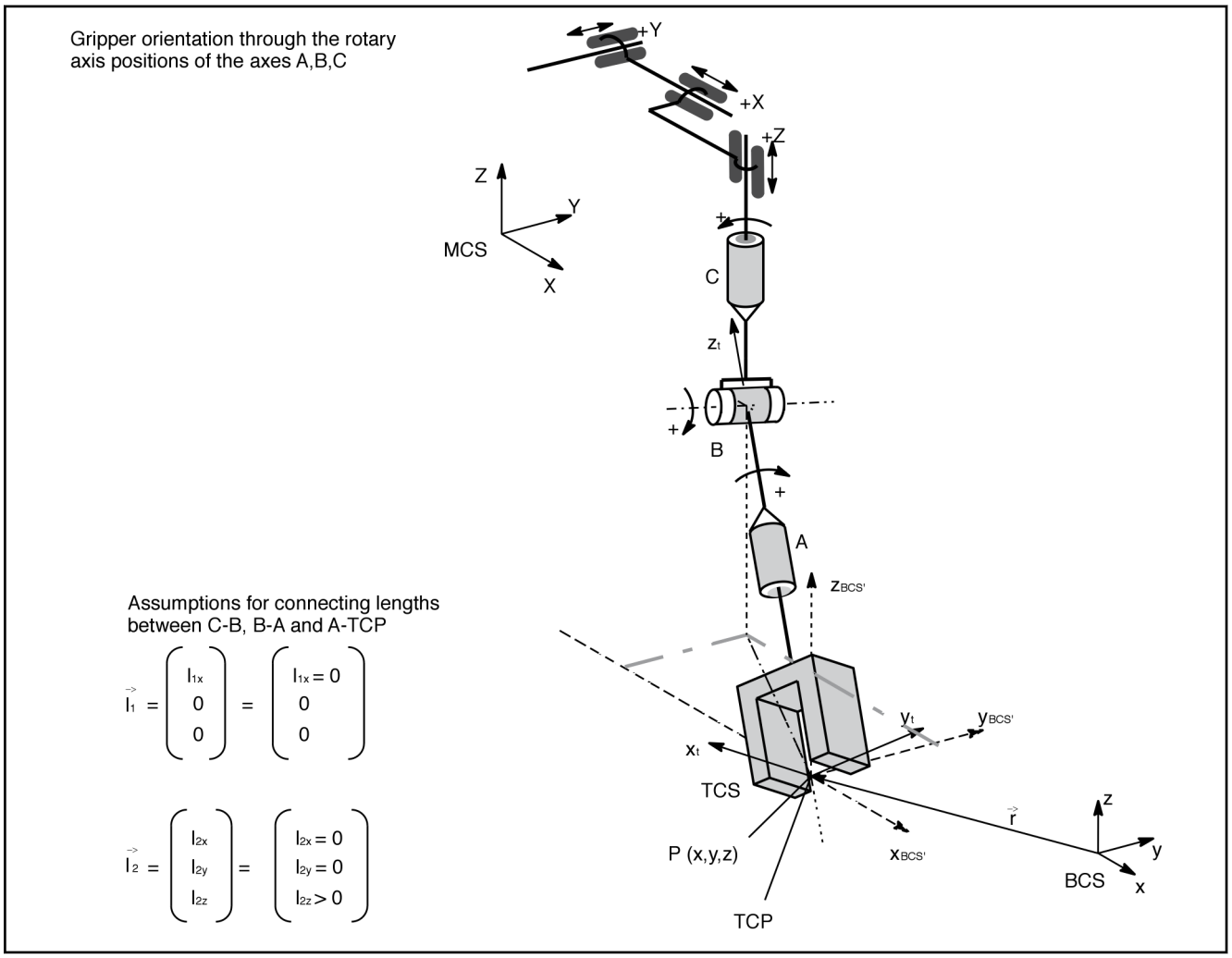


Fig. 11-67: Linear motion of the axes of rotation

11.21.4 Common characteristics of the variants 1, 2

Relevant NC functions

COORD(<i>)	i=1..20 is used to activate a 5-axis transformation which is defined in MP 1030 00110.
COORD(0,2)	The active transformation is deactivated.

Tab. 11-89: Relevant NC functions

Relevant machine parameters

1030 00110	Defines the transformation type
1030 00120	Defines the system axes/coordinates participating in the transformation.
1030 00125	Defines the axis classification of the transformation axes
1030 00130	Defines the axis positions of the reference position
1030 00140	Defines the length and angle parameters
1030 00150	Reference orientation of tool coordinate system TCS
1030 00160	Epsilon environments

Tab. 11-90: Relevant machine parameters

Type and position of the rotary axes

The rotary axes are arranged behind each other at the tool and stand vertically upon each other. In reference position, they rotate the tool by one each of the linear basic coordinates x, y, or z in positive sense of direction.

The positions of the rotary axes are named α_i, α_j, α_k. The indices i, j, and k (1, 2 or 3) indicate the respective type of axis.

The type is defined as follows:

Rotary axis type	Explanation	Axis classification in MP 1070 00125
1	The rotary axis rotates around x. Its position is identified by α_1.	100
2	The rotary axis rotates around y. Its position is identified by α_2.	200
3	The rotary axis rotates around z. Its position is identified by α_3.	300

Tab. 11-91: Rotary axis type

Sequence of the rotary axes regarding the tool:

The rotary axes of types 1, 2, and 3 can be arranged in **six** different sequences at the tool.

The following **subconfigurations** result:

Subconfiguration type	Position of the first rotary axis (farthest from the TCP)	Position of the second rotary axis	Position of the third rotation axis (close to the TCP)
123	α_1	α_2	α_3
132	α_1	α_3	α_2
213	α_2	α_1	α_3
231	α_2	α_3	α_1
312	α_3	α_1	α_2
321	α_3	α_2	α_1

Tab. 11-92: Subconfigurations

Example:

Subconfigurations

Type 132= the first rotary axis rotates around the x-coordinate of the BCS, the second around the z-coordinate and the third around the y-coordinate.

Sub configurations of the rotary axis arrangement:

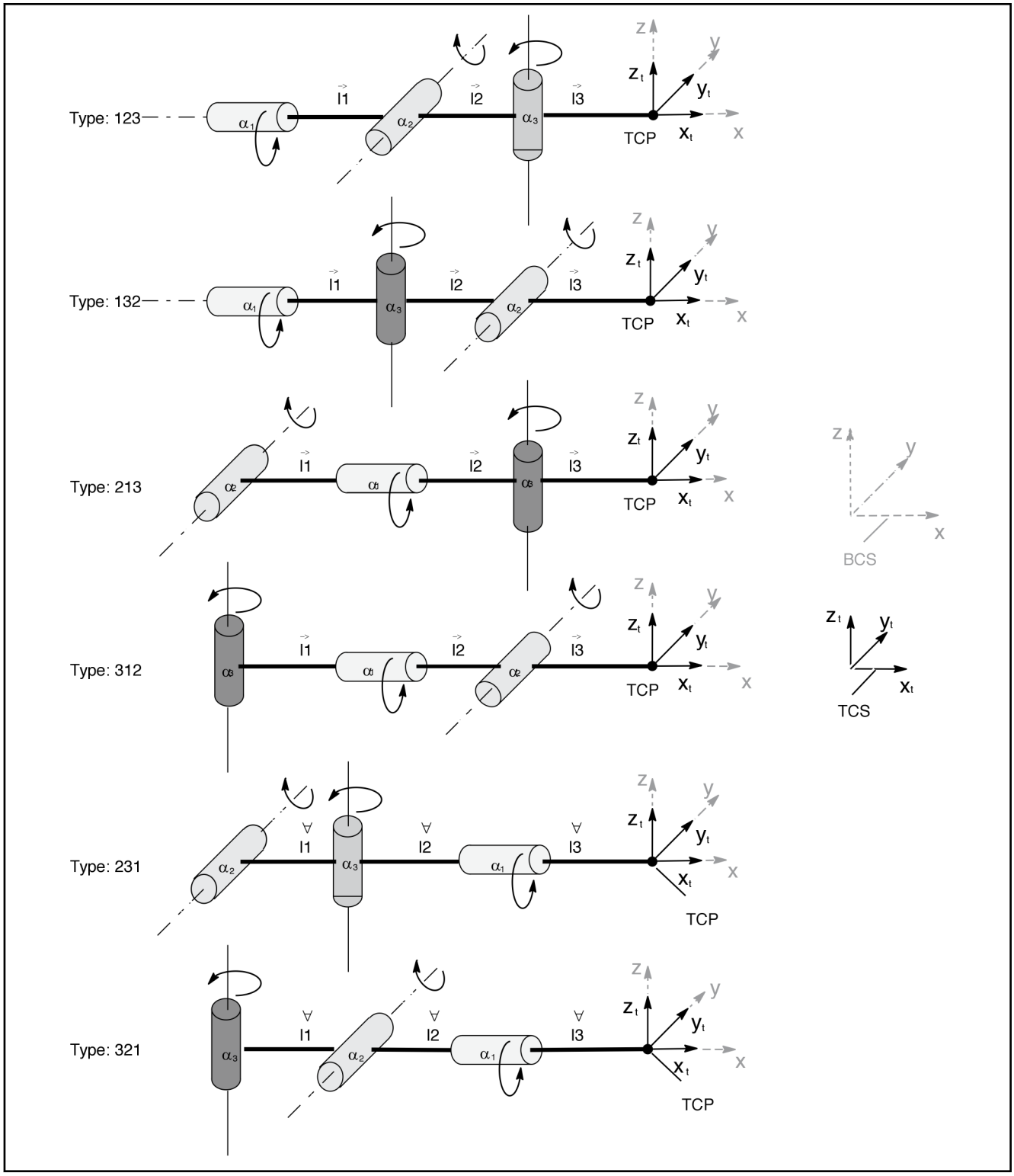


Fig. 11-68: Subconfigurations of the rotary axis arrangement

Reference position

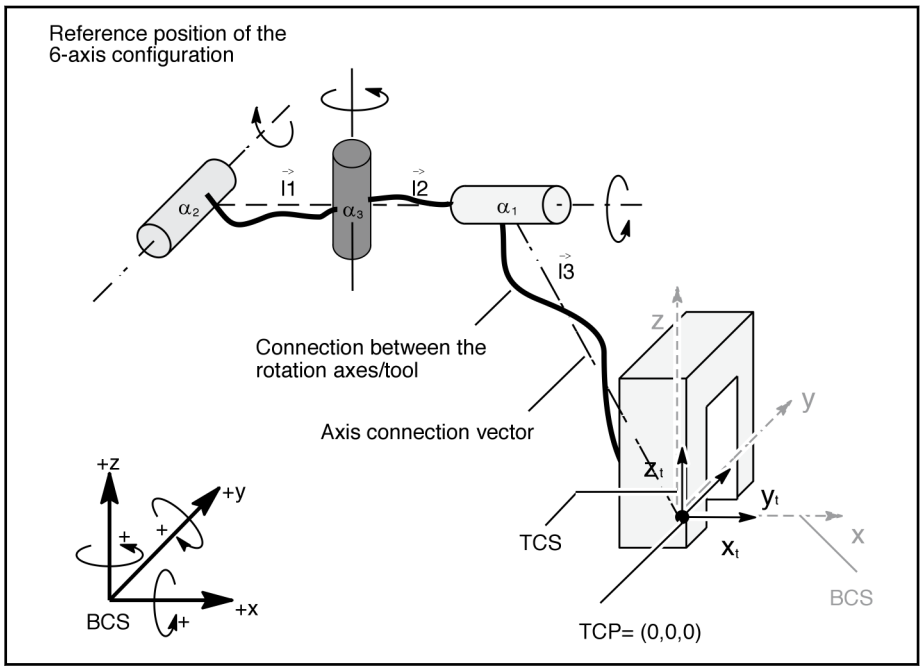


Fig. 11-69: Reference position of the 6-axis configuration

A significant factor for the configuration parameters of the axis kinematics is the **reference position**.

The reference position is characterized as follows:

- The TCP stands on the **spatial position (0,0,0)** in the BCS (zero point of the basic workpiece coordinate system).
- The coordinates (x_t, y_t, z_t) of the workpiece coordinate system (TCS) stand **parallel** or **antiparallel** to the coordinates of the basic workpiece coordinate system.
- The position taken by the rotary axes ensures that their rotary axes rotate in accordance with their respective type around x, y, and z of the BCS. They are arranged **behind each other** at the tool and stand **vertically**upon each other. The sense of rotation of the rotary axes is mathematically positive looking from a coordinate point in the direction of the coordinate origin, when the sense of rotation is anti-clockwise.
- The linear axes (X. Y, Z) form a right-handed axis coordinate system which lies parallel to the BCS.
- **Axis connection vectors**

  In the reference position, the geometry to the 6-axis kinematics is described by means of three axis connection vectors $\vec{l}_1$; $\vec{l}_2$ and $\vec{l}_3$. The vectors can have any x, y, and z components in respect of the BCS.

  The connection vector $\vec{l}_i$ (i=1,2,3) looks as follows:

$$\vec{l}_i = \begin{bmatrix} l_{ix} \\ l_{iy} \\ l_{iz} \end{bmatrix}$$

- $\vec{l}_1$ lies between the first and the second rotary axis. The first rotary axis is the rotary axis farthest away from the tool (TCP). It is followed by the second rotary axis connected to the third rotary axis via $\vec{l}_2$.

  $\vec{l}_3$ connects the third rotary axis to the tool (TCP).

  The l_{ix}, l_{iy} and l_{iz} components of the axis connection vectors are stored in MP 1030 00140.

  Position of the axis connection vectors and their x-, y-, and z-components:

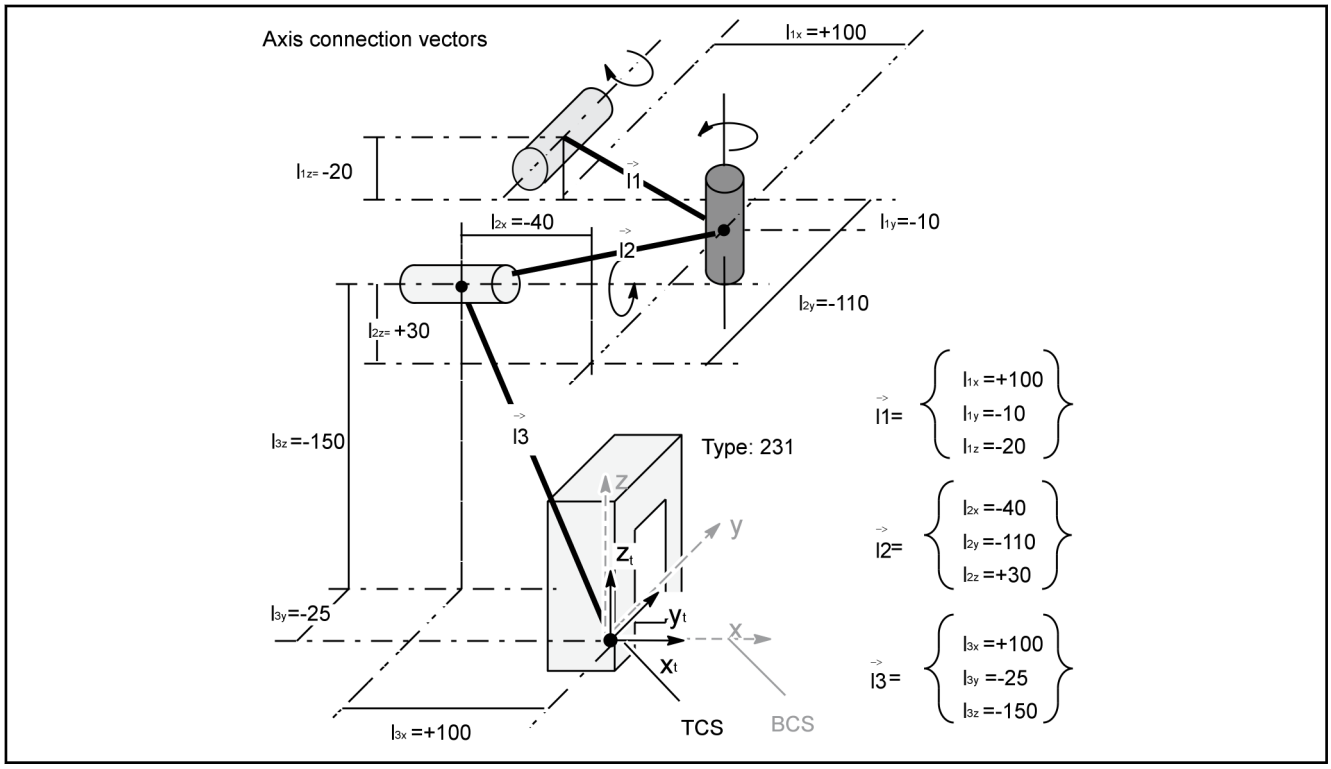


Fig. 11-70: Axis connection vectors

- Reference position of the tool coordinate system (TCS)

  The **orientation** of the TCS (x_t, y_t, z_t) in respect of the BCS (x, y, z) in reference position can be configured with MP 1030 00150.

  The following conditions are applicable for the reference positions of the TCS:

  – The main directions (x_t, y_t, z_t) of the TCS have to be parallel or anti-parallel to the main directions (x, y, z) of the BCS.

  – The main directions of the TCS and the BCS have to show a right-handed, right-angled coordinate system.

  – The presetting of MP 103000150 (reference position of the tool coordinate system) defines a TCS0, which corresponds to the orientation of the BCS in the reference position of the axis transformation:

    [1] 1

    [2] 2

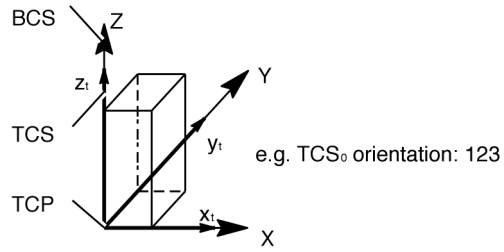
[3] 3

Deviating from this preset value, a position for the **TCS_1** can be defined in MP 1030 00150 which takes one of the 24 reference positions (orientations) in respect of the BCS.

For better **overview**, the 24 potential orientations of the TCS_1 are subdivided as follows:

The x_t-coordinate of the TCS_1 runs parallely or antiparallely to one of the three main directions x, y or z of the BCS. As a result, there are eight different TCS_1 orientations for **each** of the three main directions.

x_t is located along the **x-coordinate** of the BCS:

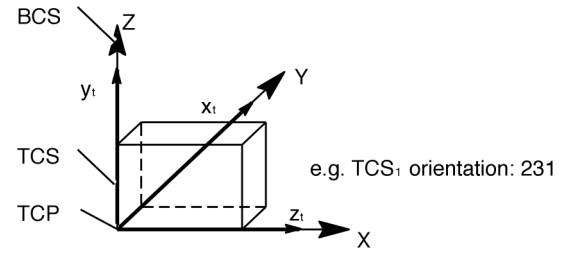


TCS main direction x_t --> x coordinate of the BCS

xt [1] is located along the BCS coordinate:	yt [2] is located along the BCS coordinate:	zt [3] is located along the BCS coordinate:	Reference position of the tool coordinate system (TCS) in the MP 1030 00150 [1]	[2]	[3]
x	y	z	**1**	**2**	**3**
x	z	-y	1	3	-2
x	-z	y	1	-3	2
x	-y	-z	1	-2	-3
-x	z	y	-1	3	2
-x	-y	z	-1	-2	3
-x	y	-z	-1	2	-3
-x	-z	-y	-1	-3	-2

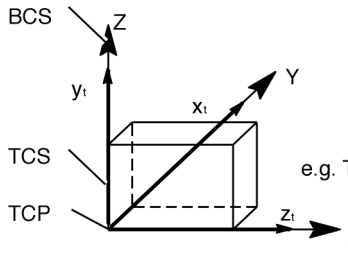
Tab. 11-93: Orientation of the TCS - x-coordinate

x_t is located along the **y-coordinate** of the BCS:



TCS main direction x_t --> y coordinate of the BCS

xt [1] is located along the BCS coordinate:	yt [2] is located along the BCS coordinate:	zt [3] is located along the BCS coordinate:	Reference position of the tool coordinate system (TCS) in the MP 1030 00150 [1]	[2]	[3]
y	z	x	**2**	**3**	**1**
y	x	-z	2	1	-3

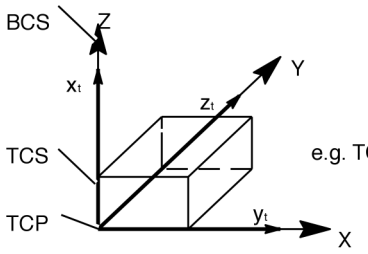


TCS main direction x_t --> y coordinate of the BCS

xt [1] is located along the BCS coordinate:	yt [2] is located along the BCS coordinate:	zt [3] is located along the BCS coordinate:	Reference position of the tool coordinate system (TCS) in the MP 1030 00150 [1]	[2]	[3]
y	-x	z	2	-1	3
y	-z	-x	2	-3	-1
-y	x	z	-2	1	3
-y	-z	x	-2	-3	1
-y	z	-x	-2	3	-1
-y	-x	-z	-2	-1	-3

Tab. 11-94: Orientation of the TCS - y-coordinate

x_t is located along the **z-coordinate** of the BCS:



TCS main direction x_t --> z coordinate of the BCS

xt [1] is located along the BCS coordinate:	yt [2] is located along the BCS coordinate:	zt [3] is located along the BCS coordinate:	Reference position of the tool coordinate system (TCS) in the MP 1030 00150 [1]	[2]	[3]
z	x	y	**3**	**1**	**2**
z	y	-x	3	2	-1
z	-y	x	3	-2	1
z	-x	-y	3	-1	-2
-z	y	x	-3	2	1
-z	-x	y	-3	-1	2
-z	x	-y	-3	1	-2
-z	-y	-x	-3	-2	-1

Tab. 11-95: Orientation of the TCS - z-coordinate

Special characteristics:

- **Feed:**

  With 6-axis transformation active, the programmed feed (F) now only relates to the programmable position coordinates, i.e., the path velocity of the tool center point (TCP) is programmed with the F-word.

This path velocity is not changed by additional orientation and pseudo coordinate motions. The orientation and pseudo coordinate motion is included synchronously, i.e. the end position is reached simultaneously for all coordinates.

However, the included motion of the orientation and pseudo coordinates may result in an additional limitation of path kinematics (maximum path velocity and acceleration) as the limits for all axes involved in the motion are monitored.

- The rotary axes A, B, and C can be endless or rotary axes.

TCS, TCP

TCS stands for Tool Coordinate System.

TCP_0/TCP_1 stands for the Tool Center Point **without/with** correction.

Tool correction

Tool correction with 6-axis transformation type 3333301:

The TCS_1 can be shifted and rotated by means of tool compensations. This results in a tool coordinates system TCS_c:

- The offset (correction vector) of the TCS_c in respect of TCS_1 is defined by $\vec{l}_t$ defined.
- The orientation of the TCS_c in the coordinate system TCS_1 is defined by rotary matrix $\overleftrightarrow{T}_t$ or by the three Euler angles φ, ϑ, ψ The correction vector and the Euler angles can be set by means of the CPL command "TC".

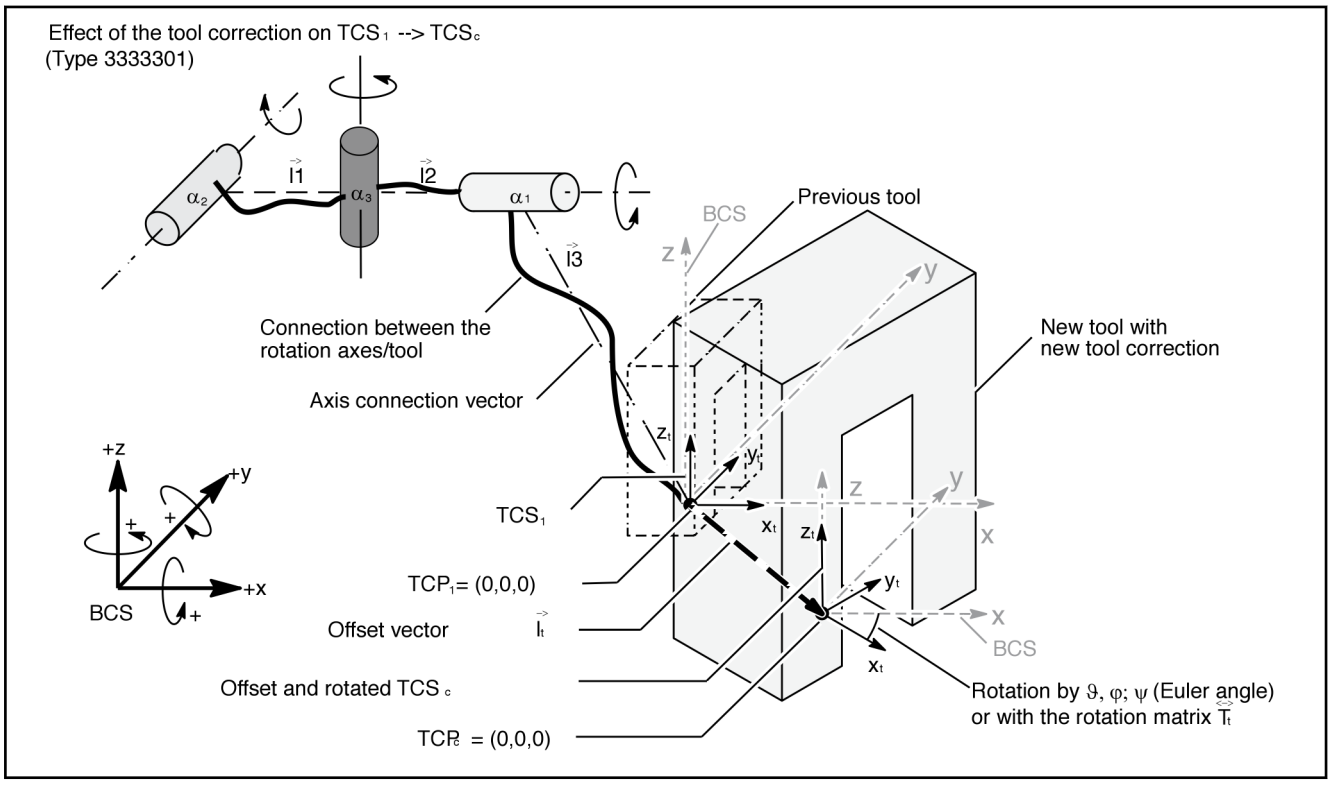


Fig. 11-71: Effect of the tool correction

The tool correction has the same effect as the replacement of the configuration vector $\vec{l_3}$ by $\vec{k}$ + $\vec{l_3}$.

Tool correction with 6-axis transformation type 3033001:

By tool corrections, the **TCS_1** is freely shifted in space (here, TCS_1always = TCS_0). This results in a tool coordinates system TCS_c:

The offset (correction vector) of the TCS_c in respect of TCS_0 or TCS_1 is defined by $\vec{k}$ defined.

☞ A rotation of the TCS as with the 3333301 type is **not** possible.

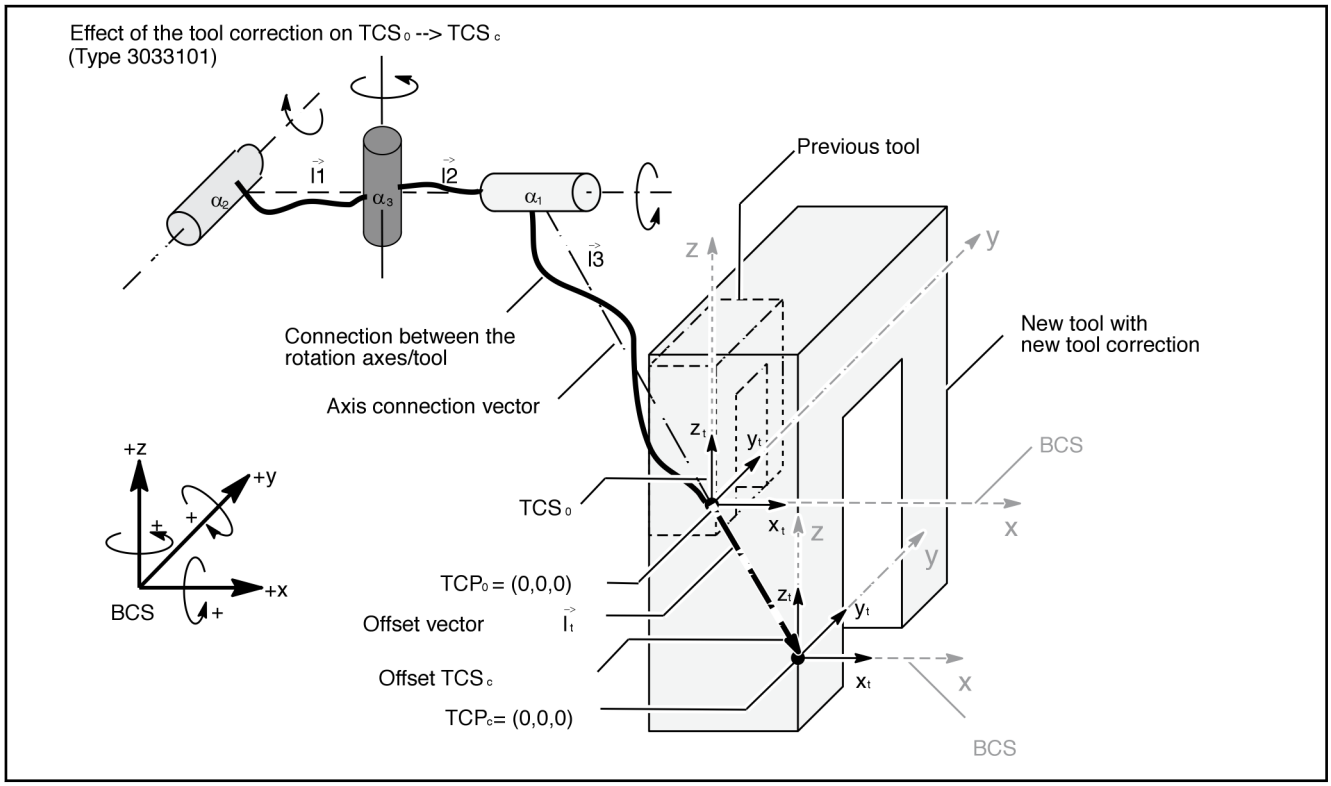


Fig. 11-72: Effect of the tool correction

The tool correction has the same effect as the replacement of the configuration vector $\vec{l_3}$ by $\vec{k}$ + $\vec{l_3}$.

11.21.5 Handling instruction: 6-axis transformation LLLRRR

Applying

The axis transformation can be applied via the relevant machine parameters. Therefore, the axis transformation function has to be activated via the parameter **tra** "Axis transformations" in the setup (SUP). The path of the machine parameter name is always "TRA/AxTrafo[1..20]/...".

Alternatively, the axis transformation can be applied via a user-defined system date of type SysAxTrafo_t.

The content of both variants is identical.

The axis positions of the reference position have to be determined.

The respectively equivalent machine parameter numbers, machine parameter names and SD elements are specified in the following table. To apply a transformation, one of the variants has to be assigned accordingly.

Transformation-relevant parameters

MP number	Description	
1030 00110	**Transformation type** MP name: TrafoType SD elem.: Type	
	3333301	LLLRRR with Tensor orientation
	3033001	LLLRRR with linear motion of the rotary axes
1030 00120	**System axes/coordinates of the transformation** MP name: FwdInCoordIndTrafo[1..8] SD elem.: AxisAssignment[1..8]	
	[1]	System axis number of X
	[2]	System axis number of Y
	[3]	System axis number of Z
	[4]	System axis number of R_1 [farthest from the tool (TCP)]
	[5]	System axis number of R_2
	[6]	System axis number of R_3 (at tool)
	[7..8]	Not relevant

<table>
<tr><th>MP number</th><th>Description</th></tr>
<tr><td>1030 00125</td><td>Axis classification of transformation axes
MP name: CoordClass[1..8]
MP name: CoordDir[1..8]
SD elem.: AxisClassification[1..8]</td></tr>
<tr><td></td><td>Defines the axes involved in transformation, including their direction of motion.
• MP no. or SD element:
Value ">0" means:
The axis rotates/travels positively as in the figures above.
Value "<0" means:
The axis rotates/travels negatively as in the figures above.
Possible values are:
– +/-1 = x-axis
– +/-2 = y-axis
– +/-3 = z-axis
– +/-100 = a-axis
– +/-200 = b-axis
– +/-300 = C-axis
• MP name:
– Possible values for CoordClass are
– X,Y,Z (linear)
– A,B,C (rotary)
– Possible values for CoordDir are:
– positive (regarding fig. "Reference position"
– negative (regarding fig. "Reference position"</td></tr>
<tr><td></td><td>If a parameter is not set, the following default settings apply:
<table><tr><td></td><td>MP no /
SD elem.:</td><td>CoordClass:</td><td>CoordDir:</td></tr><tr><td>[1]</td><td>1</td><td>X</td><td>Positive</td></tr><tr><td>[2]</td><td>2</td><td>Y</td><td>Positive</td></tr><tr><td>[3]</td><td>3</td><td>Z</td><td>Positive</td></tr><tr><td>[4]</td><td>200</td><td>B</td><td>Positive</td></tr><tr><td>[5]</td><td>100</td><td>A</td><td>Positive</td></tr><tr><td>[6]</td><td>300</td><td>C</td><td>Positive</td></tr><tr><td>[7..8]</td><td>Not relevant</td><td>0</td><td>Positive</td></tr></table></td></tr>
</table>

MP number	Description	
1030 00130	**Axis positions of the reference position** MP name: RefPosTrafo[1..8] SD elem.: AxZeroPos[1..8]	
	[1]	X^0
	[2]	Y^0
	[3]	Z^0
	[4]	$R_1{}^0$
	[5]	$R_2{}^0$
	[6]	$R_3{}^0$
	[7..8]	Not relevant
1030 00140	**Length and angle parameters** MP name: JointParTrafo[1..16] SD elem.: LenParam[1..16]	
	Defines the length vectors l_1, l_2, l_3 in [mm].	
	[1]	l_{1x}
	[2]	l_{1y}
	[3]	l_{1z}
	[4]	l_{2x}
	[5]	l_{2y}
	[6]	l_{2z}
	[7]	l_{3x}
	[8]	l_{3y}
	[9]	l_{3z}
	[10..16]	Not relevant
1030 00150	**Reference position of tool coordinate system** Defines the orientation of the TCS_1 in respect of the BCS in the reference position. MP name: CoordTcsXBcs, CoordTcsYBcs, CoordTcsZBcs MP name: DirTcsXBcs, DirTcsYBcs, DirTcsZBcs SD elem.: TCSOrientation[1..8]	
	[1]	Direction of x_t
	[2]	Direction of y_t
	[3]	Direction of z_t
	[4..8]	Not relevant
	Values: ±1 ,±2, ±3. The orientation of the TCS **must** result in a right-handed coordinate system.	
	The parameter 103000150 is not relevant for the type 3033001!	

MP number	Description
1030 00160	**Epsilon environments** MP name: EpsAxTrafo SD elem.: EpsilonRanges
	0.01

Tab. 11-96: Transformation-relevant parameters

Example:

6-axis transformation application

In the following example, the transformation types 3333301 and 3033001 are treated.

☞ All values to be entered relate to the reference position.

1. In MP 1030 00110 (no. 1), store the value 3333301 as a transformation type, and the value 3033001 in MP 1030 00110 (no. 2).
2. Enter the system axes numbers in MP 103000120 (no. 1 and no. 2).

   Ensure that the first rotary axis is farthest away from the tool (TCP), and that the third rotary axis has the shortest connection to the TCP.
3. The components of the vectors $\vec{l}_1$, $\vec{l}_2$ and $\vec{l}_3$ have to suitably determined. This can be done by measuring or directly by means of construction drawings. Length and offset values are stored in MP 1030 00140[1..9] for no. 1 as well as for no. 2.
4. Determination of axis zero position: Go to reference position. The axis positions in the reference position are stored in parameter 1030 00130[1..5] for no. 1 and for no. 2.
5. The reference position of the TCS is entered in parameter 1030 00150 for no. 1. For no. 2 (type = 3033001), no TCS definition is required.

   Ensure that the values result in a right-handed coordinate system. A left-handed coordinate system generates a runtime error.
6. Confirm parameter with system reset.
7. Using COORD(1) and COORD(2), check the 6-axis transformation in an NC program for functionality. If no runtime error is signaled, the parameters are consistent with each other.

Activating If the 6-axis transformation is configured via

- machine parameter, the transformation is enabled in the NC program with (i=1,..20):

  COORD(<i>)
- system date, the transformation is enabled in the NC program with (type "SysAxTrafo_t"):

  COORD(<SD.MtbAxTrafo>)

Example program:

The 6-axis transformation is defined in the fourth machine parameter block. No axis transformation is active at programming start:

Enable transformation

```
N100  G1 F1000 XA10 YA10 C30   ;Axes XA, YA and C are traversed
N110  COORD(4)                 ;Switching on the axis transformation
N130  x20 y30 phi0 psi10       ;the coordinates x,y,phi,psi are traversed
N140  x10 z10 theta45          ;the coordinates x,z,theta are traversed
        :
N999  COORD(0)                 ;Switching-off the axis transformation
                               ;Axes are again programmed from here
```

Deactivating **COORD(0)** or **COORD(0,2)**

switches off the axis transformation in the NC program.

11.22 3-axis cylinder jacket transformation

11.22.1 Description

General information

The cylinder jacket transformation is used for programming the machining processes on the jacket of a cylindrical workpiece.

There are 2 variants:

- The programmed contour relates to the developed cylinder jacket surface and
- The programmed contour relates to the plane projections on the cylinder jacket surface.

11.22.2 Variant 1 (unwinding)

Function

Transformation name	Transformation type	Point of action	Coordinate system
Cylinder jacket transformation (unwinding)	3021001 (RLL)	AT2	MCS, BCS

Tab. 11-97: Function - Variant 1 (unwinding)

Output variables in the MCS: X,Z,C

Input variables in the BCS: x, y, z

The cylinder jacket transformation 3021001 acts at point of action AT2 and transforms between the machine coordinate system MCS=(X,Z,C) and the basic coordinate system BCS = (x,y,z).

The machine configuration consists of two linear MCS coordinates X, Z which position the cutting tool, as well as a rotary axis C on which the cylinder to be machined is defined rotation-symmetrically.

The c-axis may be an endless or a rotary axis.

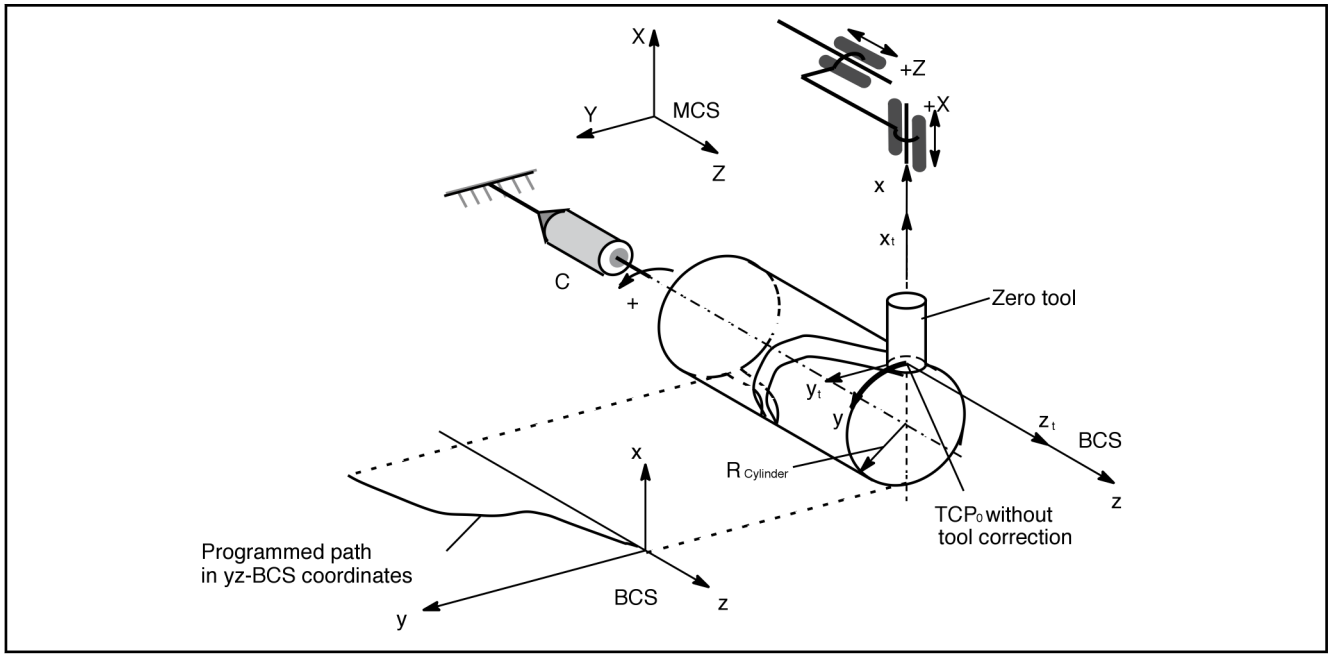


Fig. 11-73: Cylinder jacket transformation (unwinding)

The property of the transformation is defined in the contour description of the y-coordinate:

- Together with the z-coordinate, programming of the y-coordinate results in a contour which can be described as unwinding on a cylinder jacket surface.
- The y-coordinate runs on the jacket surface over the circumference of the cylinder.
- The y,z contour is programmed in relation to the developed cylinder jacket surface. The value range of the y-axis is not restricted:

Example:

Spiral slots

The angle position C = 20° and C = 380° describe different positions.

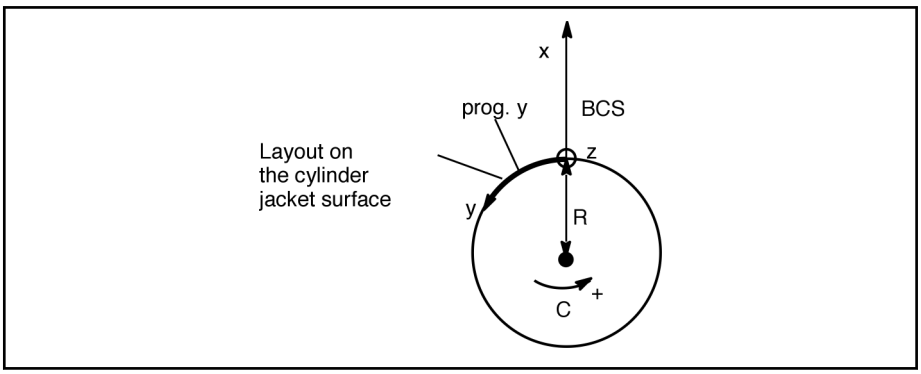


Fig. 11-74: Example - spiral grooves

11.22.3 Variant 2 (plane projection)

Function

Transformation name	Transformation type	Point of action	Coordinate system
Cylinder jacket transformation (plane projection)	3021002 (RLL)	AT2	MCS, BCS

Tab. 11-98: Function - Variant 2 (plane projection)

Output variables in the MCS: X,Z,C

Input variables in the BCS: x, y, z

The cylinder jacket transformation 3021002 acts at point of action AT2 and transforms between the machine coordinate system MCS=(X,Z,C) and the basic coordinate system BCS = (x,y,z).

The properties of these transformations are defined in the contour description of the y-coordinate:

- Contrary to the cylinder jacket transformation 3021001 with unwinding, programming of the y-coordinate results in a contour on a cylinder jacket surface implemented by **plane projection**. A contour is programmed in a plane which can be oriented in any fashion in space, e.g. by the "Inclined plane" function. The image (projection) of this contour on the cylinder surface creates the actual contour at the workpiece.
- The (y,z) contour is programmed in relation to any plane projection (e.g. pipe-in-pipe).

Example:

Projection on the basic coordinate system BCS

which means that no placement is active (BCS=WCS).

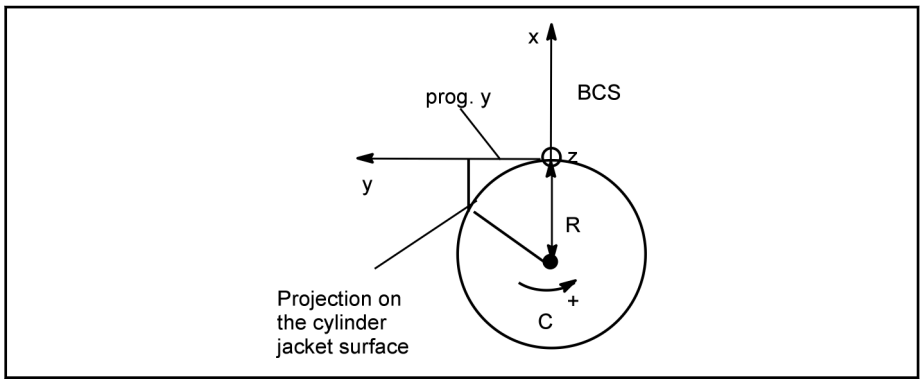


Fig. 11-75: Example projection on the basic coordinate system BCS

Backwards transformation is defined for |y| < R. In case of |y| ≥ R, a runtime error is generated. This range is irrelevant for practical applications.

Example:

Projection on the BCS

The following illustration shows the case of projection on the BCS when the Placement function (Inclined plane) is active (BCS is not identical with WCS).

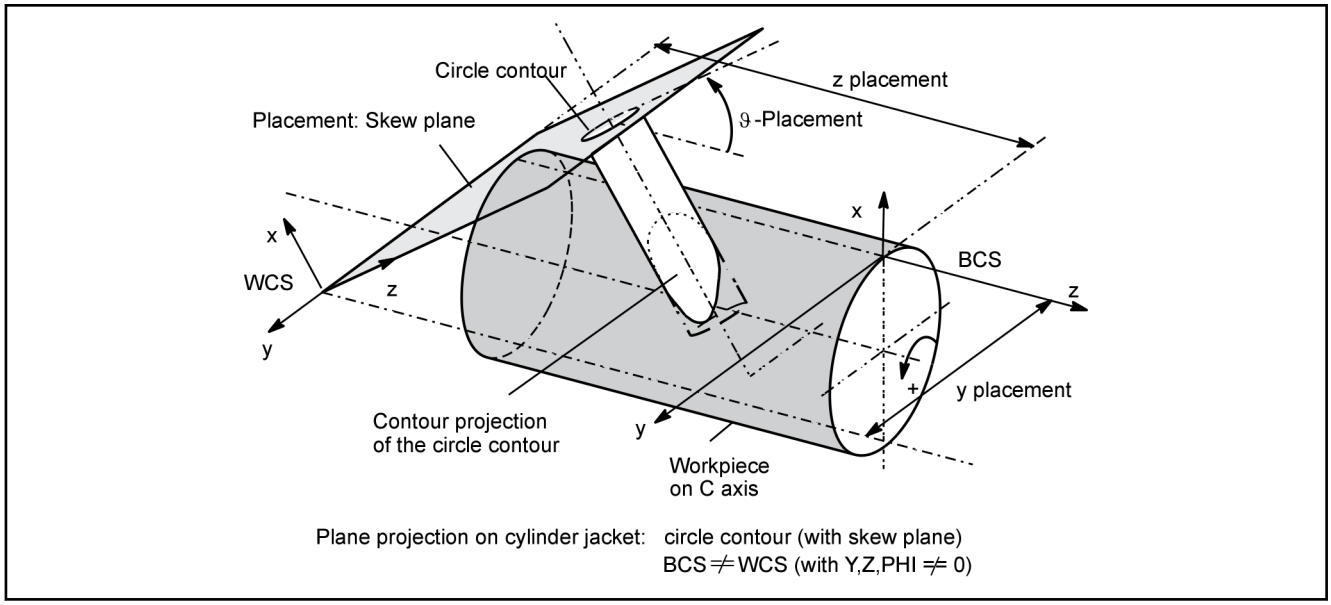


Fig. 11-76: Plane projection to cylinder jacket

11.22.4 Common characteristics of the variants 1, 2

Relevant NC functions

COORD (<I>) TRFOPT(<RADIUS>)	
	With I=1..20. Activation of the cylinder jacket transformation defined in MP 1030 00110 (axis transformation type). TRFOPT can be used to transmit the cylinder radius (RADIUS > 0). It overwrites the value of MP 1030 00140 [1].
Only for plane projection 3021002:	
G152.x	Activate Placement (Inclined plane), thus defining the projection plane. The placement has to be switched on **after** activation of the transformation. **Note**: In connection with the cylinder jacket transformation, the placement function "Inclined plane" G152.x does not show a "classical" placement behavior: G152.x generally defines the (Y,Z) projection plane with "X placement) = 0". X-values other than zero have no effect as the X-placement) component is required internally in the control to determine the respective point on the jacket surface.
G153	Deactivation of placement (inclined plane), i.e. BCS = WCS

Tab. 11-99: Relevant NC functions

Relevant machine parameters

The machine parameters are chiefly identical for both variants.

1030 00110	Axis transformation type
1030 00120	System axes of the transformation.

1030 00125	Axis classification of transformation axes
1030 00130	Axis positions of the reference position

Tab. 11-100: Relevant machine parameters

Reference position

In the reference position, the tool holder is situated in the rotary axis C. Then, the BCS zero point lies on the jacket surface at x = y = z = 0.

When the tool point (incl. length correction) stands on this point, this corresponds to the respective MCS-coordinate

$(X,Z,C) = (radius_{cylinder} + l_{tx} + X_0, Z_0, C_0)$.

That is to say that the reference position for the tool magazine point (TCP_0) is offset from the BCS zero point in negative x-direction by the radius R of the cylinder.

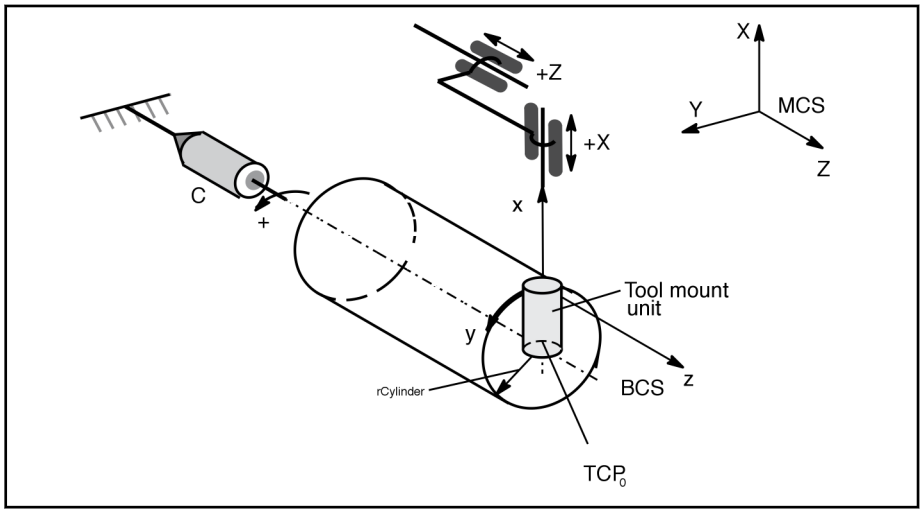


Fig. 11-77: Reference position

TCS, TCP

TCS stands for Tool Coordinate System.

TCP_0/TCP_1 stands for the Tool Center Point **without/with** correction.

Tool correction

When the tool magazine point TCP_0 is in the reference position (x = y = z = 0 in the BCS), the Tool Center Point (TCP) with active tool length correction is on position

$(x,y,z) = (-radius_{cylinder} - l_{tx}),0,0)$

related to the BCS zero point (see the above illustration).

With cylinder jacket transformation, the tool length correction has to only be effective in the direction of the x-coordinate. The tool length correction vector $\vec{l_t}$ points from the tool point of the TCS to the tool magazine of the TCS $_0$.

$$\vec{l_t} = \begin{bmatrix} L_{tx} \\ 0 \\ 0 \end{bmatrix}$$

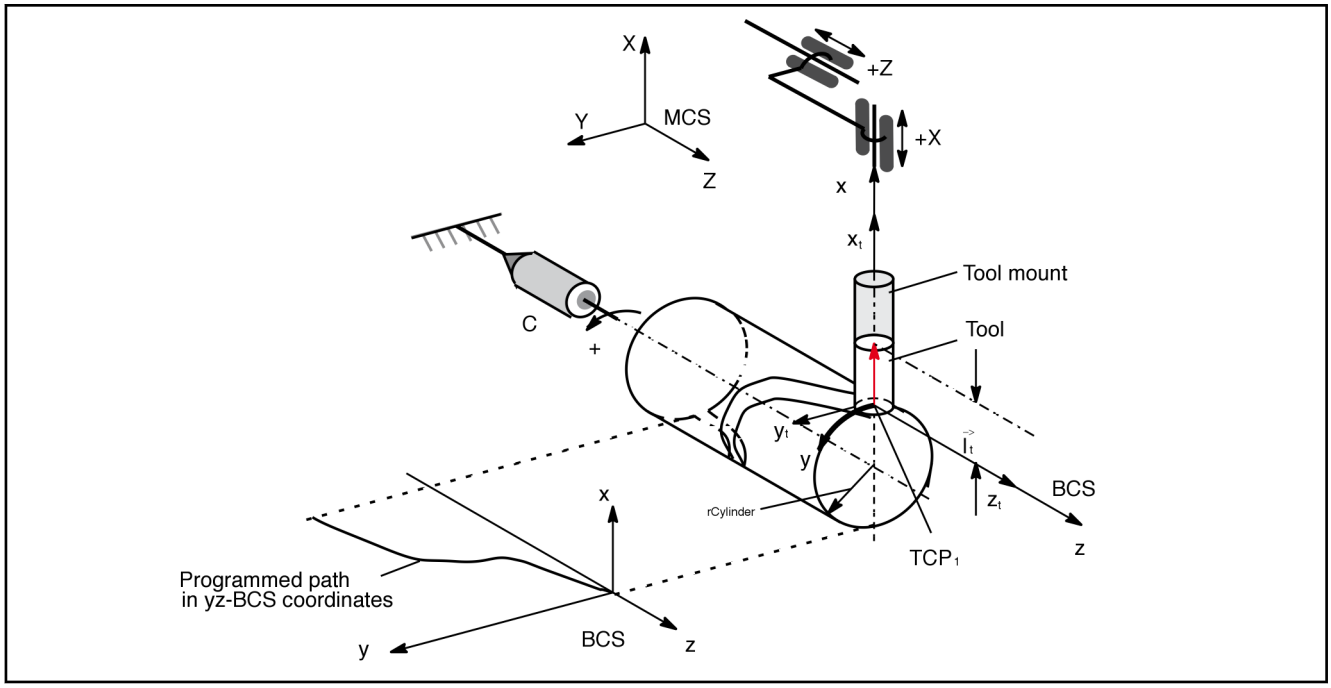


Fig. 11-78: Tool correction

Special features

The x-zero point is located on the cylinder jacket, i.e. the plunging depth is programmed with x < 0.

11.22.5 Handling instruction: 3-Axis Cylinder Jacket Transformation

Applying

The axis transformation can be applied via the relevant machine parameters. Therefore, the axis transformation function has to be activated via the parameter **tra** "Axis transformations" in the setup (SUP). The path of the machine parameter name is always "TRA/AxTrafo[1..20]/...".

Alternatively, the axis transformation can be applied via a user-defined system date of type SysAxTrafo_t.

The content of both variants is identical.

The axis positions of the reference position have to be determined.

The respectively equivalent machine parameter numbers, machine parameter names and SD elements are specified in the following table. To apply a transformation, one of the variants has to be assigned accordingly.

Transformation-relevant parameters

MP number	Description	
1030 00110	**Transformation type** MP name: TrafoType SD elem.: Type	
	3021001	Unwinding
	3021002	Project planning

<table>
<tr><th>MP number</th><th colspan="4">Description</th></tr>
<tr><td rowspan="5">1030 00120</td><td colspan="4">System axes/coordinates of the transformation
MP name: FwdInCoordIndTrafo[1..8]
SD elem.: AxisAssignment[1..8]</td></tr>
<tr><td>[1]</td><td colspan="3">System axis number of X</td></tr>
<tr><td>[2]</td><td colspan="3">System axis number of C</td></tr>
<tr><td>[3]</td><td colspan="3">System axis number of Z</td></tr>
<tr><td>[4..8]</td><td colspan="3">Not relevant</td></tr>
<tr><td rowspan="8">1030 00125</td><td colspan="4">Axis classification of transformation axes
MP name: CoordClass[1..8]
MP name: CoordDir[1..8]
SD elem.: AxisClassification[1..8]</td></tr>
<tr><td colspan="4">Defines the axes involved in transformation, including their direction of motion.
• MP no. or SD element:
Value ">0" means:
The axis rotates/travels positively as in the figures above.
Value "<0" means:
The axis rotates/travels negatively as in the figures above.
Possible values are:
– +/-1 = x-axis
– +/-300 = c-axis
• MP name:
– Possible values for CoordClass are
– X (linear)
– C (rotary)
– Possible values for CoordDir are:
– positive (regarding fig. "Reference position"
– negative (regarding fig. "Reference position"</td></tr>
<tr><td colspan="4">If a parameter is not set, the following default settings apply:</td></tr>
<tr><td></td><td>MP no /
SD elem.:</td><td>CoordClass:</td><td>CoordDir:</td></tr>
<tr><td>[1]</td><td>1</td><td>X</td><td>Positive</td></tr>
<tr><td>[2]</td><td>300</td><td>C</td><td>Positive</td></tr>
<tr><td>[3..8]</td><td>Not relevant</td><td>0</td><td>Positive</td></tr>
</table>

MP number	Description		
1030 00130	**Axis positions of the reference position** MP name: RefPosTrafo[1..8] SD elem.: AxZeroPos[1..8]		
	[1]	X^0	
	[2]	C^0	
	[3]	Z^0	
	[6..8]	Not relevant	
1030 00140	**Length and angle parameters** MP name: JointParTrafo[1..16] SD elem.: LenParam[1..16]		
	Defines the length vectors l_1, l_2 in [mm].		
	[1]	Radius Default value for the cylinder radius. When TRFOPT is programmed with activating the axis transformation, this overwrites the value.	
	[2]	yMod Only relevant for variant 1 when the C-axis is an endless modulo axis.	
		0	the switch-on value of the y-coordinate is located in the interval [-2 π *radius, 0]
		1	the switch-on value of the y-coordinate is located in the interval [- π *radius, π * radius] (recommended).
	[3..16]	Not relevant	

Tab. 11-101: Transformation-relevant parameters

Activating

If the cylinder jacket transformation is configured via

- machine parameter, the transformation is enabled in the NC program with (i=1,..20):

  COORD(<i>) TRFOPT(<cylinder radius>)

- system date, the transformation is enabled in the NC program with (type "SysAxTrafo_t"):

  COORD(<SD.MtbAxTrafo>) TRFOPT(<cylinder radius>)

Example program:

The cylinder jacket transformation is defined in the fourth machine parameter block. No axis transformation is active at programming start:

Enable transformation

```
N100  G1 F1000 XA10 ZA10 C30 ;the axes XA,ZA,C are traversed
N110  COORD(4) TRFOPT(100)   ;switching-on axis transformation with a radius of 100
N130  x20 y30                ;traversing the coordinates x,y
N140  x10 z5                 ;the coordinates x,z are traversed
        :
```

```
N999  COORD(0)              ;Switching-off the axis transformation
                            ;Axes are again programmed from here
```

☞ The parameter TRFOPT(<CylinderRadius>) has to be programmed with a cylinder radius value > 0.

Zero is the default value but causes a runtime error.

☞ **Project planning**

- **G152.x** Activate Placement (Inclined plane), thus defining the projection plane. The **placement has to be switched on** after activation of the transformation. Deactivate the placement (inclined plane) with G153.
- With cylinder jacket transformation with plane projection, please note that the transformation will only work for y values within the diameter range of the cylinder (that is cylinder radius < y < cylinder radius.

Deactivating **COORD(0)** or **COORD(0,2)**

switches off the axis transformation in the NC program.

11.23 4-Axis Cylinder Jacket Transformation

11.23.1 Description

Function

Transformation name	Transformation type	Point of action
4-axis cylinder jacket		
Unwinding	3131021	AT2
Project planning	3131022	AT2

The 4-axis cylinder jacket transformations are used for programming the machining processes on the jacket of a cylindrical workpiece. The transformations allow groove wall corrections (e.g. parallel groove walls). Refer to the following figure.

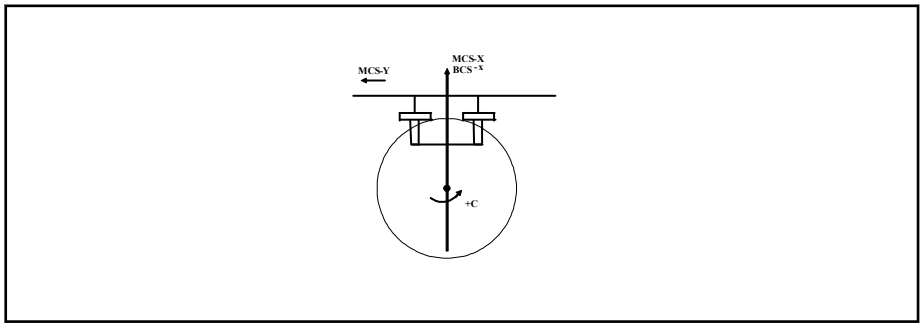


Fig. 11-79: Parallel groove walls

There are two variants with regard to the y-coordinate:

- The programmed contour relates to the developed cylinder jacket surface (standard)
- The programmed contour relates to the plane projections on the cylinder jacket surface.

The machine configuration consists of the three linear MCS coordinates X, Y and Z spanning a Cartesian coordinate system on the right. The tool is located at the x-axis. The cylinder to be machined is located rotationally symmetric on rotary axis c. The z-axis is parallel to the longitudinal direction of the cylinder (refer to fig. 11-80 "TCP in BCS zero position" on page 368 - fig. 11-83 "Position and orientation of the phi handling" on page 369).

The BCS zero point is located on the jacket surface (refer to fig. 11-80 "TCP in BCS zero position" on page 368). The (negative) x-coordinate describes the plunging depth.

The y-motion of rotary axis c is generated in both variants (handling and project planning (refer to fig. 11-81 "Orientation phi" on page 368). The linear y-axis creates the orientation phi at the TCP (refer to fig. 11-81 "Orientation phi" on page 368).

The two 4-axis cylinder jacket transformations are used at action point 2. They transform between the basic coordinate system (BCS) with the coordinates x, y, z and phi and the machine coordinate system (MCS) with the coordinates x, y, z and c.

- The x, y, z, phi BCS input variables are programmed
- The MCS output variables X, Y, Z, C are calculated

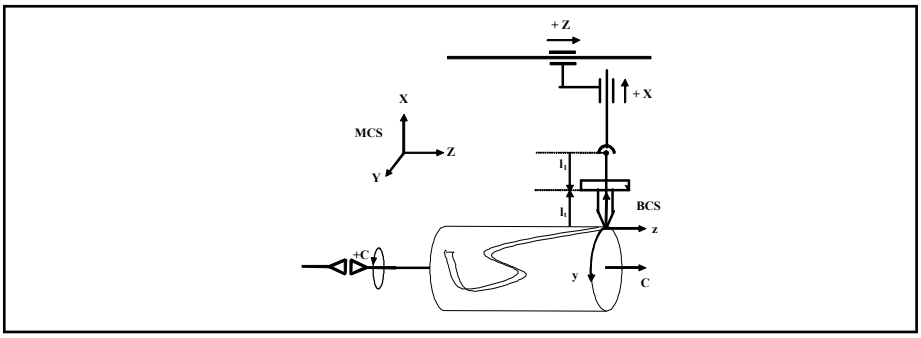


Fig. 11-80: TCP in BCS zero position

The tool correction vector and displays from TCP to tool acceptance.

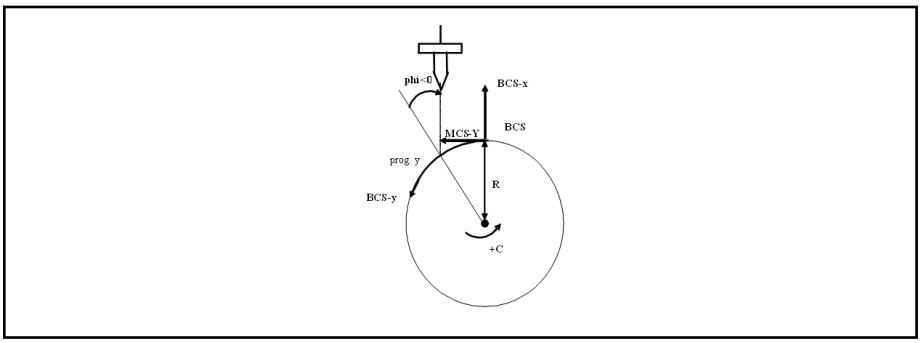


Fig. 11-81: Orientation phi

The reference position of the z-axis is located in the C-axis (refer to the following figure).

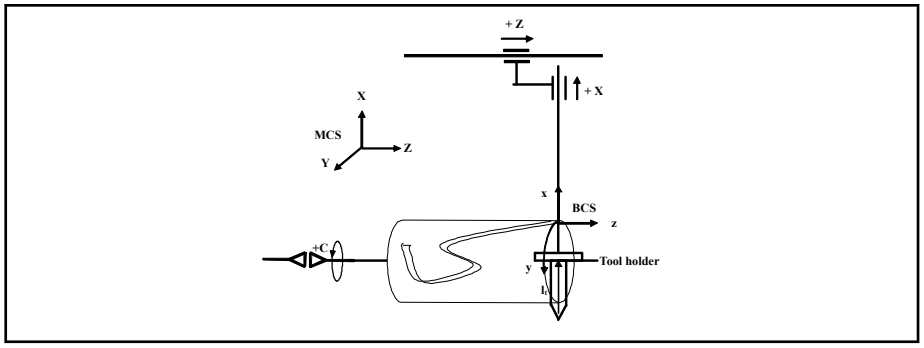


Fig. 11-82: Reference position of the z-axis

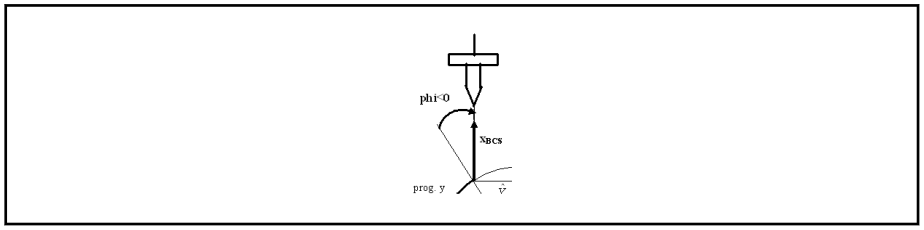


Fig. 11-83: Position and orientation of the phi handling

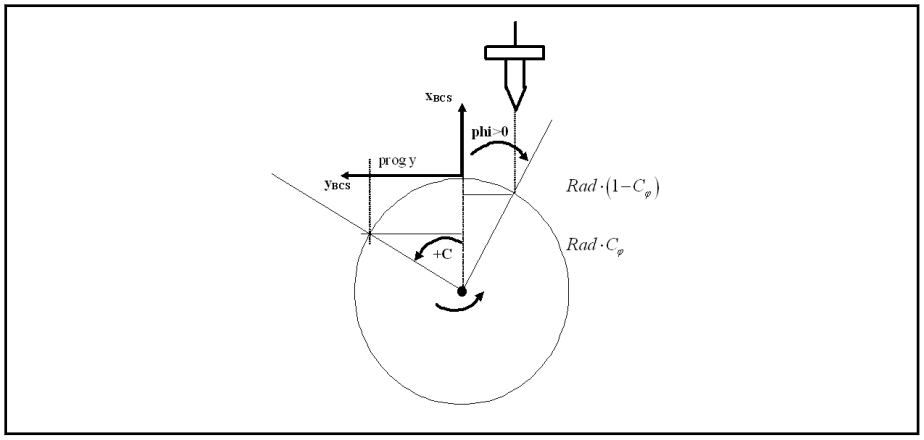


Fig. 11-84: Position and orientation of the phi project planning

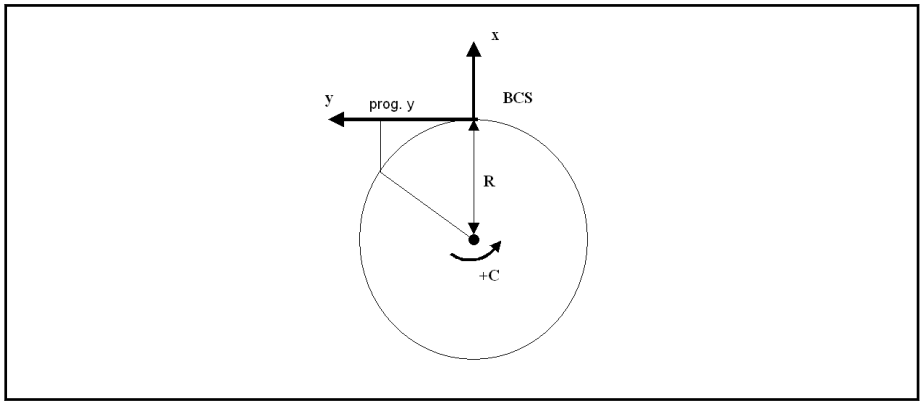


Fig. 11-85: Project planning

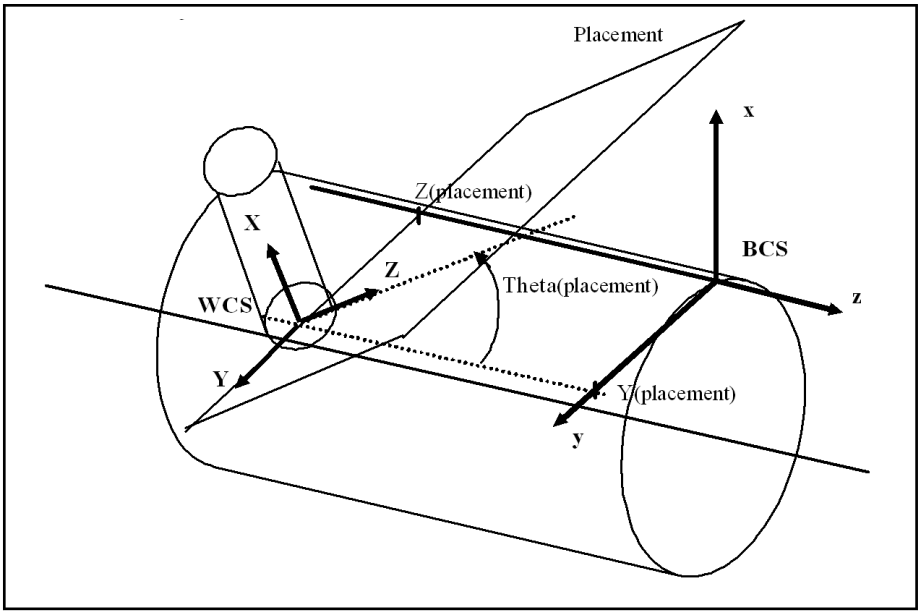


Fig. 11-86: Project planning

Reference position

In reference position, the tool holder (TCP_0) is located in rotary axis c (fig. 11-83 "Position and orientation of the phi handling" on page 369). The BCS zero point is on the jacket surface with (x,y,z)=(0,0,0) (fig. 11-80 "TCP in BCS zero position" on page 368).

If the tool tip (including the length correction) is positioned on that point, it corresponds to the respective MCS coordinates (X, C, Z, Y, B) = (RadiusCylinder + MTX + X_0, C_0, Z_0,Y_0,B_0). That means that the reference position for the tool holder is offset from the BCS zero point in negative x direction by the radius R of the cylinder (refer to fig. 11-80 "TCP in BCS zero position" on page 368, fig. 11-83 "Position and orientation of the phi handling" on page 369).

The tool orientation is aligned along x (phi=0).

Variants

There are two different variants of the 4-axis cylinder jacket transformation.

These differ in their y-coordinates:

1. The programmed contour relates to the developed cylinder jacket surface (standard)
2. the programmed contour refers to a plane projection on the cylinder jacket surface

The figures show the positive directions of rotation/traversing of the axes. Set them accordingly.

Orientation

Switching on the 4-axis cylinder jacket transformation enables a linear orientation motion. The angles phi are interpolated linearly from the starting to the end point.

Special features

- The C-axis can be defined as endless or rotary axis.

- **Tool correction**

  Both axis transformations support a tool correction. The tool correction vector l_t= (l_{tx},0,0) points from the corrected tool center point TCP_1 to the uncorrected TCP_0. Only the length correction L1 = l_{tx} can be compensated (thus, correction takes only place in direction of the x-coordinate.

- **TRFOPT (<Radius>)**

  When enabling the cylinder jacket transformation, the cylinder radius (value > 0) can be transferred using the optional transformation parameter TRFOPT. It overwrites the value of MP 1030 00140 [2].

Relevant NC functions

COORD(...)	The 4-axis cylinder jacket transformation is enabled. This can optionally be done using the machine parameter block (i=1..20) or via the MTB system data (of type SysAxTrafo_t).
COORD(0) - or - COORD(0,2)	The active transformation is deactivated.
G152.x	Activate Placement (Inclined plane), thus defining the projection plane. The placement has to be switched on after activation of the transformation. **Note**: In connection with the cylinder jacket transformation, the placement function Inclined plane G152.x does not show a classical placement behavior: G152.x generally defines the (Y, Z) projection plane with "X-placement = 0". X values other than zero have no effect as the x-placement component is required internally in the control to determine the respective point on the jacket surface.
G153	Switching off placement (Bcs=Wcs)

Tab. 11-102: Relevant NC functions

Relevant machine parameters

Up to 20 different axis transformations are declared with the following machine parameter block in the machine parameters.

1030 00110	Defines the transformation type.
1030 00120	Defines the system axes/coordinates participating in the transformation.
1030 00125	Defines the axis classification of the transformation axes.
1030 00130	Defines the axis positions of the reference position.
1030 00140	Defines the length and angle parameters.
1030 00160	Epsilon environments

Tab. 11-103: Relevant machine parameters

Relevant system data

The axis transformation can also be defined via system date instead of the machine parameters. Therefore, a system variable of type "SysAxTrafo_t"

has to be created (see "MTX Machine Parameters", chapter "System data"). The structure type "SysAxTrafo_t" is control-internally defined in the xml schema files "/feprom/schemas/sdaxtrf.xsd" and contains - analog to the relevant machine parameters - the following transformation-relevant structure elements.

Structural element	Description
Type	Defines the transformation type.
AxisAssignment	Defines the system axes/coordinates participating in the transformation.
AxisClassification	Defines the axis classification of the transformation axes.
AxZeroPos	Defines the axis positions of the reference position.
LenParam	Defines the length and angle parameters.
EpsilonRanges	Epsilon environments

Tab. 11-104: Relevant system data

11.23.2 Handling instruction: 4-axis cylinder jacket transformation

Applying

Applying

The axis transformation can be applied via the relevant machine parameters. Therefore, the axis transformation function has to be activated via the parameter **tra** "Axis transformations" in the setup (SUP). The path of the machine parameter name is always "TRA/AxTrafo[1..20]/...".

Alternatively, the axis transformation can be applied via a user-defined system date of type SysAxTrafo_t.

The content of both variants is identical.

The axis positions of the reference position have to be determined.

The respectively equivalent machine parameter numbers, machine parameter names and SD elements are specified in the following table. To apply a transformation, one of the variants has to be assigned accordingly.

Transformation-relevant parameters

MP number	Description	
1030 00110	**Transformation type** MP name: TrafoType SD elem.: Type	
	3131021	4-axis cylinder jacket handling
	3131022	4-axis cylinder jacket project planning

<table>
<tr><td>1030 00120</td><td colspan="4">System axes/coordinates of the transformation
MP name: FwdInCoordIndTrafo[1..8]
SD elem.: AxisAssignment[1..8]</td></tr>
<tr><td></td><td>[1]</td><td colspan="3">System axis number of X</td></tr>
<tr><td></td><td>[2]</td><td colspan="3">System axis number of C</td></tr>
<tr><td></td><td>[3]</td><td colspan="3">System axis number of Z</td></tr>
<tr><td></td><td>[4]</td><td colspan="3">System axis number of Y</td></tr>
<tr><td></td><td>[5..8]</td><td colspan="3">Not relevant</td></tr>
<tr><td>1030 00125</td><td colspan="4">Axis classification of transformation axes
MP name: CoordClass[1..8]
MP name: CoordDir[1..8]
SD elem.: AxisClassification[1..8]</td></tr>
<tr><td></td><td colspan="4">Defines the axes involved in transformation, including their direction of motion.
• MP no. or SD element:
Value ">0" means:
The axis rotates/travels positively as in the figures above.
Value "<0" means:
The axis rotates/travels negatively as in the figures above.
Possible values are:
– +/-1 = x-axis
– +/-2 = y-axis
– +/-3 = z-axis
– +/-100 = a-axis
– +/-200 = b-axis
– +/-300 = c-axis
• MP name:
– Possible values for CoordClass are:
– X, Y, Z (linear)
– A, B, C (rotary)
– Possible values for CoordDir are:
– positive (regarding fig. "Reference position"
– negative (regarding fig. "Reference position"</td></tr>
<tr><td></td><td colspan="4">If a parameter is not set, the following default settings apply:</td></tr>
<tr><td></td><td></td><td>MP no /
SD elem.:</td><td>CoordClass:</td><td>CoordDir:</td></tr>
<tr><td></td><td>[1]</td><td>1</td><td>X</td><td>Positive</td></tr>
<tr><td></td><td>[2]</td><td>300</td><td>C</td><td>Positive</td></tr>
<tr><td></td><td>[3]</td><td>3</td><td>Z</td><td>Positive</td></tr>
<tr><td></td><td>[4]</td><td>2</td><td>Y</td><td>Positive</td></tr>
<tr><td></td><td>[5..8]</td><td>Not relevant</td><td>0</td><td>Positive</td></tr>
</table>

1030 00130	**Axis positions of the reference position** MP name: RefPosTrafo[1..8] SD elem.: AxZeroPos[1..8]		
	[1]	X^0	
	[2]	C^0	
	[3]	Z^0	
	[4]	Y^0	
	[5..8]	Not relevant	
1030 00140	**Length and angle parameters** MP name: JointParTrafo[1..16] SD elem.: LenParam[1..16]		
	Defines the length vectors l_1, l_2 in [mm].		
	[1]	Radius	Default value for the cylinder radius. When TRFOPT is programmed with activating the axis transformation, this overwrites the value
	[2]	yMod	Only relevant for variant 1 (handling) when the C-axis is an endless modulo axis.
		0	the switch-on value of the y-coordinate is located in the interval [-2*Pi*Radius, 0]
		1	the switch-on value of the y-coordinate is located in the interval [-Pi*Radius, +Pi*Radius] (recommended)
	[3..16]	Not relevant	
1030 00160	**Epsilon environments** MP name: EpsAxTrafo SD elem.: EpsilonRanges		
	Not relevant		

Tab. 11-105: Transformation-relevant parameters

Activating

Activating *If the 4-axis transformation is configured via*

- machine parameter, the transformation is enabled in the NC program with (i=1,..20):

  COORD(<i>)

- system date, the transformation is enabled in the NC program with (type "SysAxTrafo_t"):

  COORD(<SD.MtbAxTrafo>)

Example program:

The 4-axis transformation is defined in the fourth machine parameter block. No axis transformation is active at programming start:

```
N01 G1 F1000 XA110 YA10 C30     ;the XA, YA, B, C are traversed
N02 COORD(4) TRFOPT(100)        ;Enable 4-axis cylinder transformation with the
                                 radius R=100 [mm]
```

```
N03 x-5 y30 z50 phi10          ;Approach (y, z) position with plunging depth
                               5 and orientation phi
N04 z10 phi20                  ;The coordinates z, phi are traversed

N99 COORD(0)                   ;Switch off 4-axis transformation
                               ;The axes are again programmed from here
```

Deactivating

Deactivating **COORD(0)** or **COORD(0,2)**
switches off the axis transformation in the NC program.

11.24 5-axis cylinder jacket transformation

11.24.1 Description

Function

Transformation name	Transformation type	Point of action
5-axis cylinder jacket		
Handling (Linear orientation)	3232121	AT2
Project planning (linear orientation)	3232122	AT2

The 5-axis cylinder jacket transformations are used for programming the machining processes on the jacket of a cylindrical workpiece. The transformations allow groove wall corrections (e.g. parallel groove walls). Refer to the following figure.

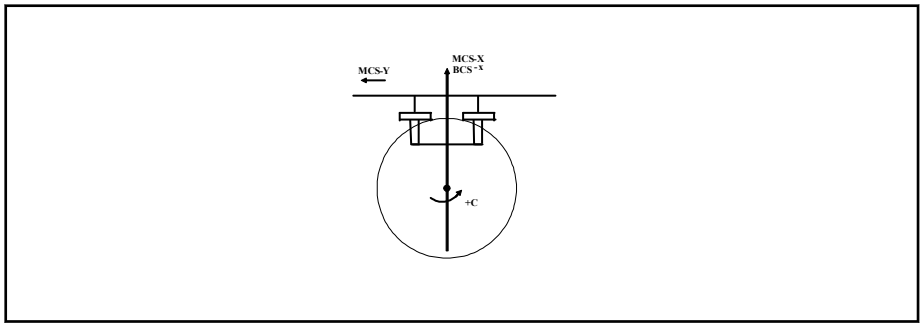


Fig. 11-87: Parallel groove walls

There are two variants with regard to the y-coordinate:

- The programmed contour relates to the developed cylinder jacket surface (standard)
- The programmed contour relates to the plane projections on the cylinder jacket surface.

The machine configuration consists of the three linear MCS coordinates X, Y and Z spanning a Cartesian coordinate system on the right. The tool is located at swiveling axis B. The cylinder to be machined is located rotationally symmetric on rotary axis c. The z-axis is parallel to the longitudinal direction of the cylinder (refer to fig. 11-88 "TCP in BCS zero position" on page 376 - fig. 11-91 "Reference position of the z-axis" on page 377).

The BCS zero point is located on the jacket surface (refer to fig. 11-88 "TCP in BCS zero position" on page 376). The (negative) x-coordinate describes the plunging depth.

The y-motion of rotary axis c is generated in both variants (handling and project planning (refer to fig. 11-89 "Orientation phi" on page 376). The linear y-axis creates the orientation at the TCP (refer to fig. 11-89 "Orientation phi" on page 376). The swiveling axis B creates the orientation theta (see fig. 11-90 "Orientation theta" on page 376).

The two 5-axis cylinder jacket transformations are used at action point 2. They transform between the basic coordinate system (BCS) with the coordinates x, y, z, phi, theta and the machine coordinate system (MCS) with the coordinates X, Y, Z, B, C.

- The x, y, z, phi, theta BCS input variables are programmed
- The MCS output variables X, Y, Z, B, C are calculated

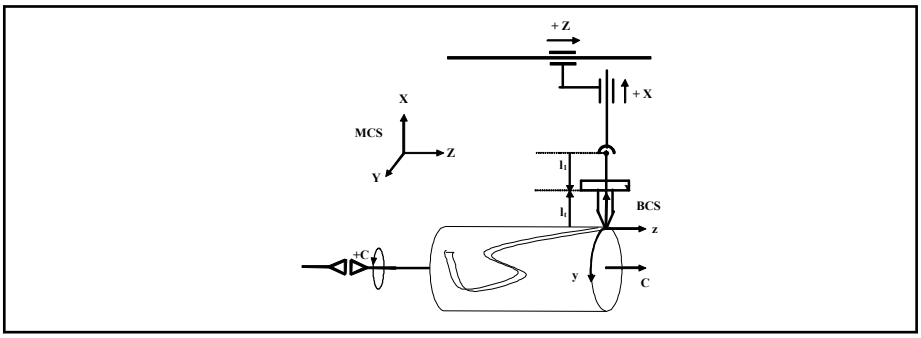


Fig. 11-88: TCP in BCS zero position

The vector $l_1 = (l_{1x},0,0)$ points from the b-axis to the tool holder. $l_t = (l_{tx},0,0)$ is the tool correction vector and points from the TCP to the tool holder.

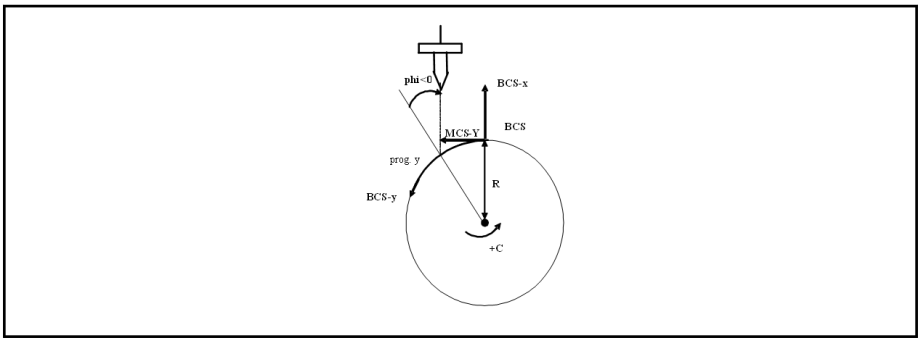


Fig. 11-89: Orientation phi

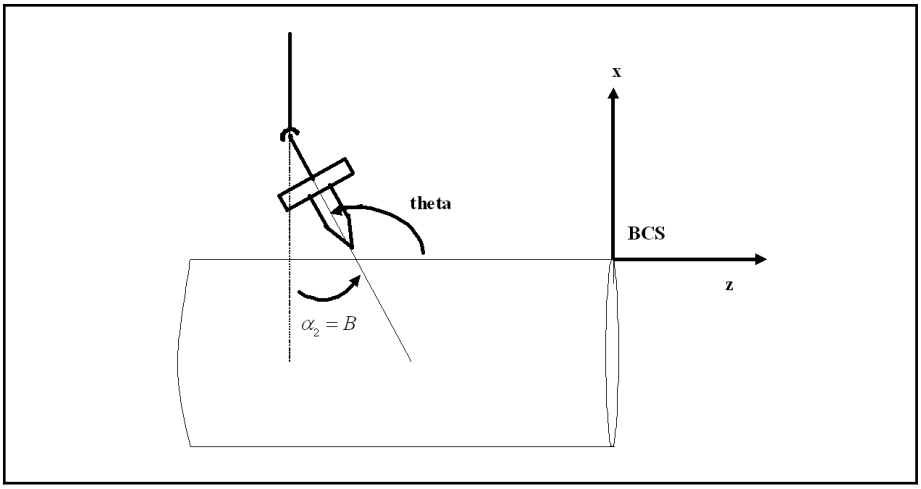


Fig. 11-90: Orientation theta

The reference position of the z-axis is located in the C-axis (refer to the following figure).

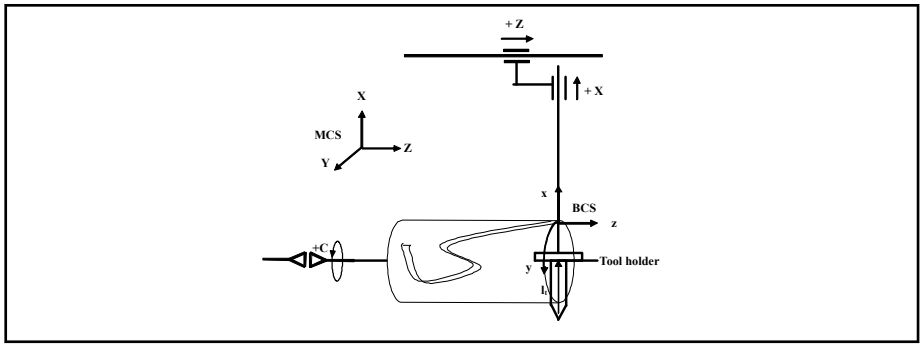


Fig. 11-91: Reference position of the z-axis

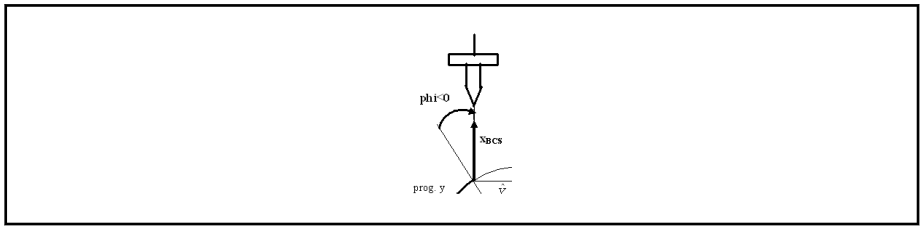


Fig. 11-92: Position and orientation of the phi handling

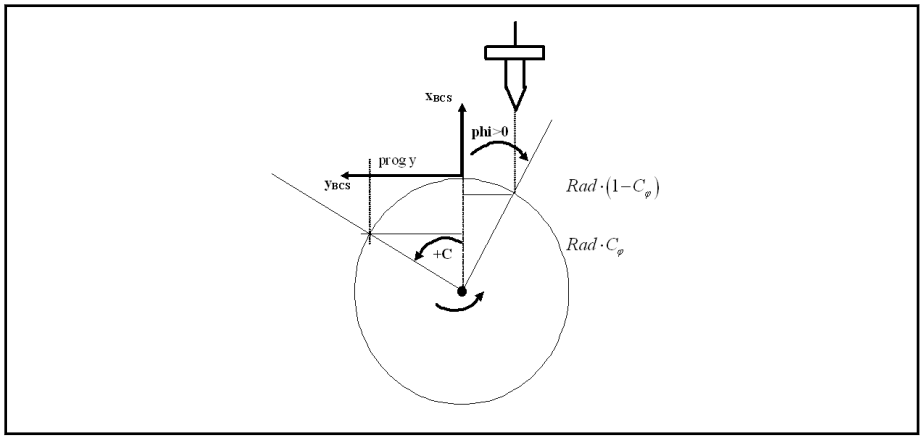


Fig. 11-93: Position and orientation of the phi project planning

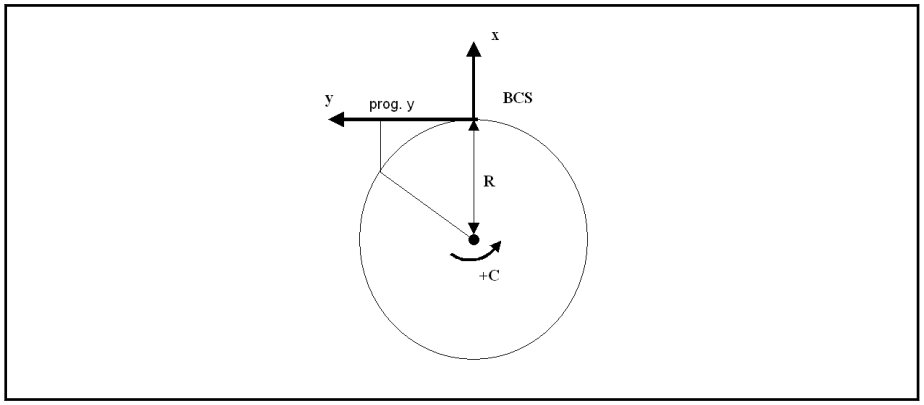


Fig. 11-94: Project planning

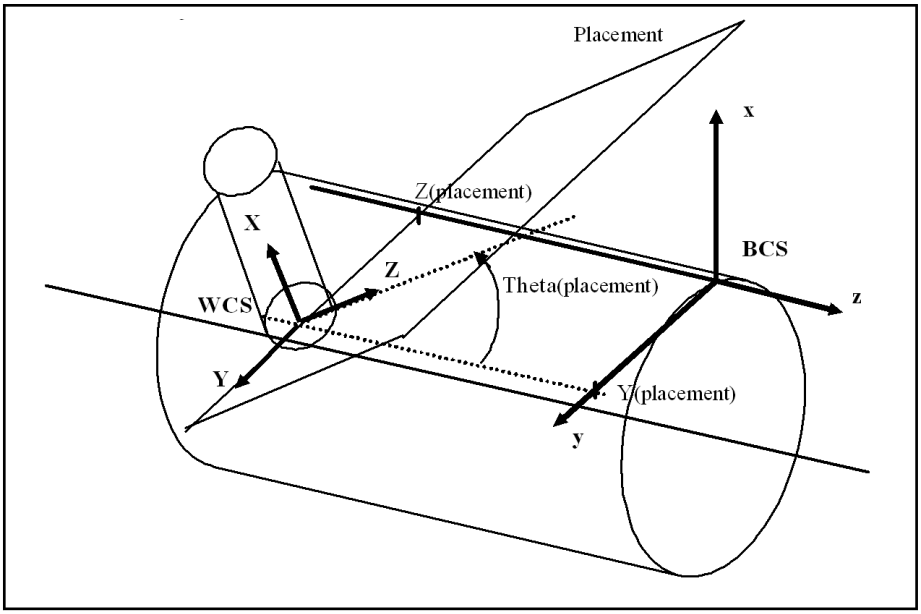


Fig. 11-95: Project planning

Reference position

The reference position points the b-axis vertically to the bottom with B=0. The tool holder (TCP_0) is located in rotary axis c (fig. 11-91 "Reference position of the z-axis" on page 377). The BCS zero point is on the jacket surface with (x, y, z)=(0,0,0) (fig. 11-88 "TCP in BCS zero position" on page 376).

If the tool tip (including the length correction) is positioned on that point, it corresponds to the respective MCS coordinates (X, C, Z, Y, B) = (RadiusCylinder + ltx + X_0, C_0, Z_0,Y_0,B_0). That means that the reference position for the tool holder is offset from the BCS zero point in negative x direction by the radius R of the cylinder (refer to fig. 11-88 "TCP in BCS zero position" on page 376, fig. 11-91 "Reference position of the z-axis" on page 377).

The tool orientation is aligned along x (phi=0, theta=0).

Variants

There are two different variants of the 5-axis cylinder jacket transformation. These differ in their y-coordinates:

1. The programmed contour relates to the developed cylinder jacket surface (standard)
2. the programmed contour refers to a plane projection on the cylinder jacket surface

Vector l_1= (l_{1x},0,0) defines in both cases the geometry of the 5-axis kinematics. The zero position point l_1 from the b-axis to the zero point of the BCS (refer to fig. 11-88 "TCP in BCS zero position" on page 376).

The figures show the positive directions of rotation/traversing of the axes. Set them accordingly.

Orientation

When switching on the 5-axis cylinder jacket transformation a linear orientation motion (type 3232121, 3232122) is activated.

Linear orientation

The angles phi and theta are interpolated linearly from the starting to the end point. The behavior of the "shortest way"-logic depends on the programming start and the orientation motion. In case of the phi and theta programming,

the "shortest way" logic is switched off. In case of the O(...) programming, the "shortest way" logic is switched off.

Phi and theta can assume values from the following ranges:

-indefinite<phi<definite ; -180≤theta<180



The linear orientation is not suitable for the "inclined plane". Programmed orientations always refer to the basic coordinate system BCS and not - as usual - to the workpiece coordinate system WCS.

Special features

- The traversing range of the b-axis is limited to +/-90 degrees.
- The C-axis can be defined as endless or rotary axis.
- **Tool correction**

  Both axis transformations support a tool correction. The tool correction vector $l_t = (l_{tx},0,0)$ points from the corrected tool center point TCP_1 to the uncorrected TCP_0. Only the length correction L1 = l_{tx} can be compensated (thus, correction takes only place in direction of the x-coordinate.
- **TRFOPT (<Radius>)**

  When enabling the cylinder jacket transformation, the cylinder radius (value > 0) can be transferred using the optional transformation parameter TRFOPT. It overwrites the value of MP 1030 00140 [2].

Relevant NC functions

COORD(...)	The 5-axis cylinder jacket transformation is enabled. This can optionally be done using the machine parameter block (i=1..20) or via the MTB system data (of type SysAxTrafo_t).
COORD(0) - or - COORD(0,2)	The active transformation is deactivated.
G152.x	Activate Placement (Inclined plane), thus defining the projection plane. The placement has to be switched on after activation of the transformation. **Note**: In connection with the cylinder jacket transformation, the placement function "Inclined plane" G152.x does not show a "classical" placement behavior: G152.x generally defines the (Y, Z) projection plane with "X-Placement = 0". X values other than zero have no effect as the x-placement component is required internally in the control to determine the respective point on the jacket surface.
G153	Switching off placement (Bcs=Wcs)

Tab. 11-106: Relevant NC functions

Relevant machine parameters

Up to 20 different axis transformations are declared with the following machine parameter block in the machine parameters.

1030 00110	Defines the transformation type.
1030 00120	Defines the system axes/coordinates participating in the transformation.
1030 00125	Defines the axis classification of the transformation axes.
1030 00130	Defines the axis positions of the reference position.
1030 00140	Defines the length and angle parameters.
1030 00160	Epsilon environments

Tab. 11-107: Relevant machine parameters

Relevant system data

The axis transformation can also be defined via system date instead of the machine parameters. Therefore, a system variable of type "SysAxTrafo_t" has to be created (see "MTX Machine Parameters", chapter "System data"). The structure type "SysAxTrafo_t" is control-internally defined in the xml schema files "/feprom/schemas/sdaxtrf.xsd" and contains - analog to the relevant machine parameters - the following transformation-relevant structure elements.

Structural element	Description
Type	Defines the transformation type.
AxisAssignment	Defines the system axes/coordinates participating in the transformation.
AxisClassification	Defines the axis classification of the transformation axes.
AxZeroPos	Defines the axis positions of the reference position.
LenParam	Defines the length and angle parameters.
EpsilonRanges	Epsilon environments

Tab. 11-108: Relevant system data

11.24.2 Handling instruction: 5-axis cylinder jacket transformation

Applying

Applying

The axis transformation can be applied via the relevant machine parameters. Therefore, the axis transformation function has to be activated via the parameter **tra** "Axis transformations" in the setup (SUP). The path of the machine parameter name is always "TRA/AxTrafo[1..20]/...".

Alternatively, the axis transformation can be applied via a user-defined system date of type SysAxTrafo_t.

The content of both variants is identical.

The axis positions of the reference position have to be determined.

The respectively equivalent machine parameter numbers, machine parameter names and SD elements are specified in the following table. To apply a transformation, one of the variants has to be assigned accordingly.

Transformation-relevant parameters

MP number	Description	
1030 00110	**Transformation type** MP name: TrafoType SD elem.: Type	
	3232121	5-axis cylinder jacket handling (linear orientation)
	3232122	5-axis cylinder jacket project planning (linear orientation)
1030 00120	**System axes/coordinates of the transformation** MP name: FwdInCoordIndTrafo[1..8] SD elem.: AxisAssignment[1..8]	
	[1]	System axis number of X
	[2]	System axis number of C
	[3]	System axis number of Z
	[4]	System axis number of Y
	[5]	System axis number of B
	[6..8]	Not relevant
1030 00125	**Axis classification of transformation axes** MP name: CoordClass[1..8] MP name: CoordDir[1..8] SD elem.: AxisClassification[1..8]	
	Defines the axes involved in transformation, including their direction of motion. • MP no. or SD element: Value ">0" means: The axis rotates/travels positively as in the figures above. Value "<0" means: The axis rotates/travels negatively as in the figures above. *Possible values are:* – +/-1 = x-axis – +/-2 = y-axis – +/-3 = z-axis – +/-100 = a-axis – +/-200 = b-axis – +/-300 = c-axis • MP name: – *Possible values for CoordClass are:* – X, Y, Z (linear) – A, B, C (rotary) – *Possible values for CoordDir are:* – positive (regarding fig. "Reference position" – negative (regarding fig. "Reference position"	

<table>
<tr><th>MP number</th><th colspan="4">Description</th></tr>
<tr><td></td><td colspan="4">If a parameter is not set, the following default settings apply:</td></tr>
<tr><td></td><td>MP no /
SD elem.:</td><td colspan="2">CoordClass:</td><td>CoordDir:</td></tr>
<tr><td></td><td>[1]</td><td>1</td><td colspan="2">X</td><td>Positive</td></tr>
<tr><td></td><td>[2]</td><td>300</td><td colspan="2">C</td><td>Positive</td></tr>
<tr><td></td><td>[3]</td><td>3</td><td colspan="2">Z</td><td>Positive</td></tr>
<tr><td></td><td>[4]</td><td>2</td><td colspan="2">Y</td><td>Positive</td></tr>
<tr><td></td><td>[5]</td><td>200</td><td colspan="2">B</td><td>Positive</td></tr>
<tr><td></td><td>[6..8]</td><td>Not relevant</td><td colspan="2">0</td><td>Positive</td></tr>
<tr><td>1030 00130</td><td colspan="4">Axis positions of the reference position
MP name: RefPosTrafo[1..8]
SD elem.: AxZeroPos[1..8]</td></tr>
<tr><td></td><td>[1]</td><td colspan="3">X^0</td></tr>
<tr><td></td><td>[2]</td><td colspan="3">C^0</td></tr>
<tr><td></td><td>[3]</td><td colspan="3">Z^0</td></tr>
<tr><td></td><td>[4]</td><td colspan="3">Y^0</td></tr>
<tr><td></td><td>[5]</td><td colspan="3">B^0</td></tr>
<tr><td></td><td>[6..8]</td><td colspan="3">Not relevant</td></tr>
<tr><td>1030 00140</td><td colspan="4">Length and angle parameters
MP name: JointParTrafo[1..16]
SD elem.: LenParam[1..16]</td></tr>
<tr><td></td><td colspan="4">Defines the length vectors l_1, l_2 in [mm].</td></tr>
<tr><td></td><td>[1]</td><td>Radius</td><td colspan="2">Default value for the cylinder radius. When TRFOPT is programmed with activating the axis transformation, this overwrites the value</td></tr>
<tr><td></td><td rowspan="3">[2]</td><td>yMod</td><td colspan="2">Only relevant for variant 1 (handling) when the C-axis is an endless modulo axis.</td></tr>
<tr><td></td><td>0</td><td colspan="2">the switch-on value of the y-coordinate is located in the interval [-2*Pi*Radius, 0]</td></tr>
<tr><td></td><td>1</td><td colspan="2">the switch-on value of the y-coordinate is located in the interval [-Pi*Radius, +Pi*Radius] (recommended)</td></tr>
<tr><td></td><td>[3]</td><td>l_{1x}</td><td colspan="2">Enter a value < 0</td></tr>
<tr><td></td><td>[4..16]</td><td>Not relevant</td><td colspan="2"></td></tr>
<tr><td>1030 00160</td><td colspan="4">Epsilon environments
MP name: EpsAxTrafo
SD elem.: EpsilonRanges</td></tr>
<tr><td></td><td colspan="4">Not relevant</td></tr>
</table>

Tab. 11-109: Transformation-relevant parameters

Activating

Deactivating *If the 5-axis transformation is configured via*

- machine parameter, the transformation is enabled in the NC program with (i=1,..20):

  COORD(<i>)

- system date, the transformation is enabled in the NC program with (type "SysAxTrafo_t"):

  COORD(<SD.MtbAxTrafo>)

Example program:

The 5-axis transformation is defined in the fourth machine parameter block. No axis transformation is active at programming start:

```
N01  G1  F1000  XA110  YA10  B20  ;the XA, YA, B, C are traversed
C30
N02 COORD(4) TRFOPT(100)          ;Enable 5-axis cylinder transformation with the
                                   radius R=100 [mm]
N03 x-5 y30 z50 phi10 theta70     ;Approach (y, z) position with plunging depth
                                   5 and orientation phi, theta
N04 z10 phi20 theta30             ;The coordinates z, phi, theta are traversed


N99 COORD(0)                      5-axis transformation
                                  ;The axes are again programmed from here
```

Deactivating

Deactivating **COORD(0)** or **COORD(0,2)**

switches off the axis transformation in the NC program.

11.25 Face transformation

11.25.1 Description

General information

The face transformation is used for the coordinates programming of a contour at the face of a rotating workpiece.

There are 2 variants:

- 2-axis transformation in (x,y)
- 3-axis transformation in (x, y, z)

In both variants, the main transformation characteristics, for example regarding transformation branch or singularities, are identical. In the 3-axis variant, only the z-coordinate (including z-tool length correction) is interpolated additionally.

11.25.2 Variant 1 (2-axis transformation)

Function

Transformation name	Transformation type	Point of action	Coordinate system
Face transformation, 2-axis transformation	2011001	AT2	MCS, BCS

Tab. 11-110: Function - Variant 1: 2-axis transformation

Output variables in the MCS: X, C

Input variables in the BCS: x, y

11.25.3 Variant 2 (3-axis transformation)

Function

Transformation name	Transformation type	Point of action	Coordinate system
Face transformation, 3-axis transformation	3021003	AT2	MCS, BCS

Tab. 11-111: Function - Variant 2: 3-axis transformation

Output variables in the MCS: X, Z, C

Input variables in the BCS: x, y, z

11.25.4 Common characteristics of the variants 1, 2

General information

Face transformation provides for coordinate programming (x,y) of a contour on the face of a workpiece.

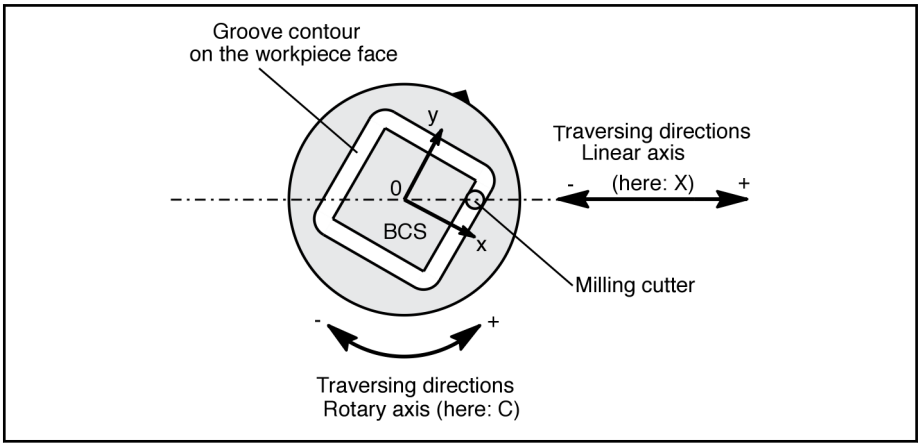


Fig. 11-96: Face transformation

To this end, the transformation establishes a relation between the coordinates x and y in the BCS and respective axis coordinates of the involved axes (here: X, C), in case of the following configuration:

- The workpiece is moved by a rotary or endless axis rotating around the workpiece feed coordinate.

  Example:

  Rotary or endless axis "C" as spindle/c-axis (before activating the face transformation, the spindle has to be switched to c-axis mode and confirmed in the channel).

- The tool (or the workpiece itself) is moved by a linear axis running horizontally parallel to the workpiece definition area. This may be an axis along a Cartesian main direction, or an axis parallel to the main direction.

  Example:

  Linear axis "X"
- The rotary/endless axis rotates around the workpiece feed axis of the BCS (here: z).
- The involved axes (here: X,C) have to be in the same channel when face transformation is activated.

Relevant NC functions

COORD(<i>)	i=1..20 is used to activate a face transformation which is defined in MP 103000110.
COORD(0,2) or COORD(0)	
	The active face transformation is deactivated.

Tab. 11-112: Relevant NC functions

Relevant machine parameters

Up to 20 different axis transformations are declared with the following machine parameter block in the machine parameters.

1030 00110	Defines the transformation type
1030 00120	Defines the system axes/coordinates participating in the transformation
1030 00125	Defines the axis classification of the transformation axes
1030 00130	Defines the axis positions of the reference position
1030 00140	Defines the length and angle parameters
1030 00160	Epsilon environments

Tab. 11-113: Relevant machine parameters

Relevant system data

The axis transformation can also be defined via system date instead of the machine parameters. Therefore, a system variable of type "SysAxTrafo_t" has to be created (see "MTX Machine Parameters", chapter "System data"). The structure type "SysAxTrafo_t" is control-internally defined in the xml schema file "/feprom/schemas/sdaxtrf.xsd" and contains - analog to the relevant machine parameters - the following transformation-relevant structure elements.

Structural element	Description
Type	Defines the transformation type
AxisAssignment	Defines the system axes/coordinates participating in the transformation
AxisClassification	Defines the axis classification of the transformation axes
AxZeroPos	Defines the axis positions of the reference position

LenParam	Defines the length and angle parameters
EpsilonRanges	Epsilon environments

Tab. 11-114: Relevant system data

Reference position

In the reference position, the rotary or endless axis (here: C) und die linear axis (here: X) is on position x = y = 0 of the BCS. This position is located on the rotary axis of the rotary / endless axis involved (here: C). At the same time, this is the position of the TCP (Tool Center Point) provided x = y = z = 0.

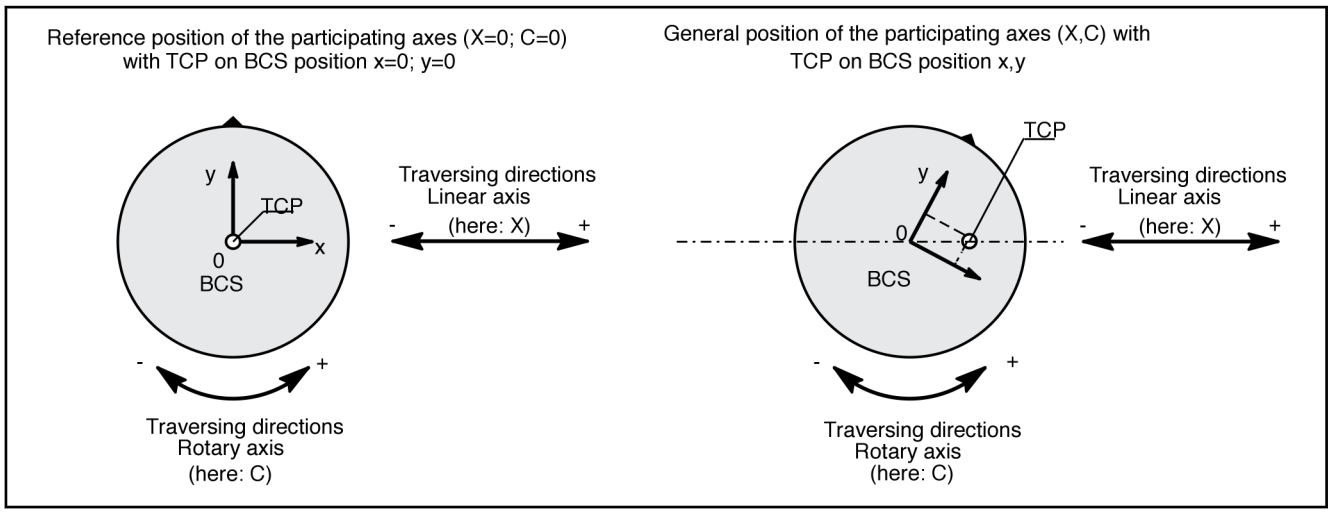


Fig. 11-97: Reference position and general position of the machine axes

The orientation of the BCS is fixedly connected with the face of the workpiece. For this reason, the BCS rotates along with the rotating workpiece.

The zero point of the BCS-coordinate axes x and y is situated on the rotary axis of the rotary/endless axes involved (here: C:) The rotary/endless axis rotates around the workpiece feed axis of the BSC (here: z).

TCS, TCP

TCS stands for Tool Coordinate System.

TCP_0/TCP_1 stands for the Tool Center Point **without/with** correction.

The TCP (Tool Center Point) is in reference position when the tool takes the position x = y = z = 0 in the BCS.

Tool correction

Tool length correction L_t:

The tool length correction becomes only effective in the TCS whereby the correction vector $\vec{l_t}$ points from the corrected tool center (TCPC) to the uncorrected TCP_0.

In case of the 2-axis variant, the tool length correction is realized in direction of the linear axes X, Z (corresponding to the L1 value), i.e. the correction vector has the shape

$$\vec{l_t} = \begin{pmatrix} L_1 \\ 0 \end{pmatrix}.$$

In case of the 3-axis variant, the tool length correction is realized in direction of the linear axes X, Z (corresponding to the L1- L3-value), i.e. the correction vector has the shape

$$l_t = \begin{pmatrix} L_1 \\ 0 \\ L_3 \end{pmatrix}$$

A correction value L2 ≠ 0 is irrelevant for practical applications. It corresponds to a tool which cannot machine the center. It would be impossible to achieve a circle with radius L2 about the BCS zero point.

Tool radius correction

With tool radius correction active, the tool engaging point moves on a straight line running parallel to the traveling direction of the linear axes; the extension of this straight line passing through the rotary axis of the rotary/endless axis involved.

The linear axis moves horizontally parallel to the workpiece definition area.

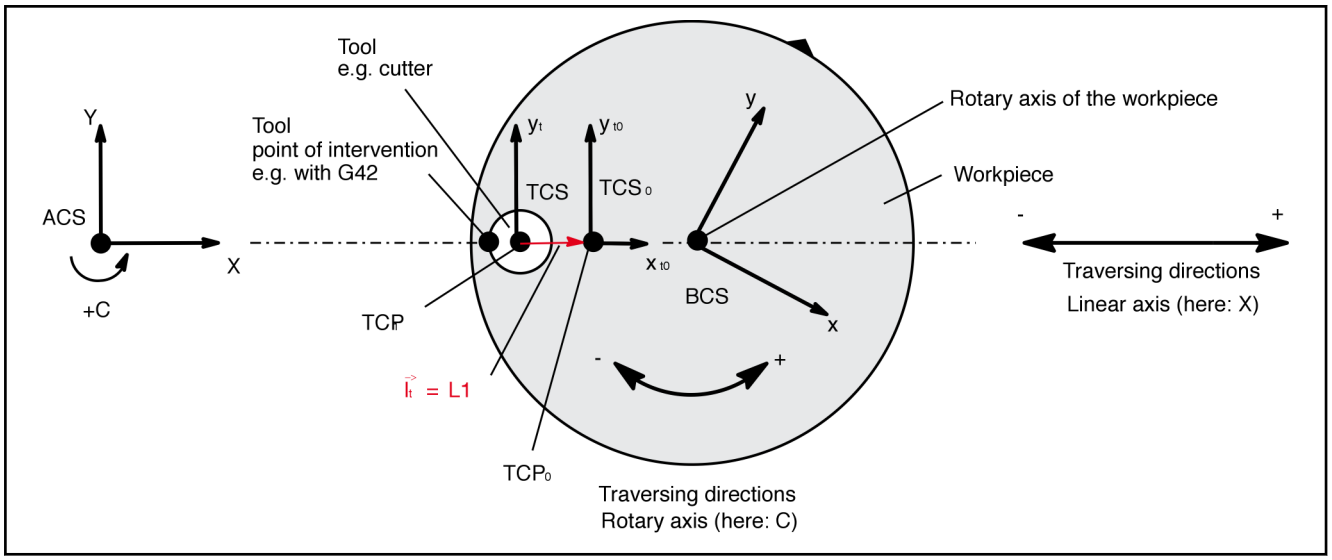


Fig. 11-98: Tool radius correction

Transformation branches and automatic branch change

With the exception of the BCS zero point, each TCP position can be mirrored by two different machine coordinate pairs of the axes involved (see the following illustration). For this reason, two "transformation branches" are implemented in the control for calculation of the transformation.

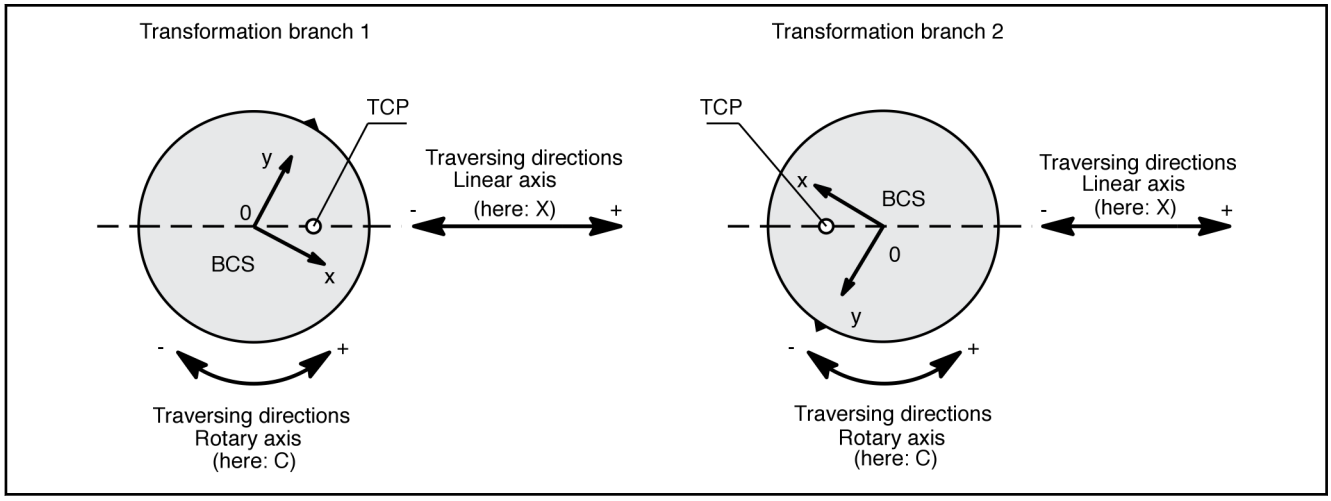


Fig. 11-99: Identical position of the TCP in the BCS; can be mirrored by 2 different pairs of machine coordinates

When the face transformation is switched on, the system decides which branch is used for the internal transformation calculations. Accordingly, an automatic transition between both branches is only possible when the zero point of the BCS coordinate axes x and y is passed.

The **automatic branch change** is always of advantage when the BCS zero point (x=0, y=0) is to be passed. If only one transformation branch were implemented in the control, the principle would generally necessitate the insertion of an intermediate block which rotates the rotary/endless axis by 180°. Thanks to the automatic branch change, no such 180° rotation is required.

However, in certain cases an automatic branch change may be undesirable. Automatic branch change must for example be prohibited if the traversing range of the linear axis is limited and is not sufficient to machine the complete end face.

In this case, the NC will automatically generated an intermediate block with rotation of the rotary/endless axis by 180°. In this way, the required traversing range of the linear axis can be reduced by half.

Through MP 1030 00140, configure whether automatic branch change is to be permitted.

Special features

Path motions by the rotary axis of the rotary/endless axis involved:

The rotary axis of the rotary/endless axis involved can be traveled

1. **Without change of direction** of the tool path (tangential) or
2. **With change of direction** of the tool path (non-tangential).

- About 1:

  With a permitted branch change, the control will change the branch automatically. Thus, a 180 degree rotation of the rotary/endless axis involved is not necessary.

  In case of a prohibited branch change, the NC will automatically split the branch into three block segments:

  First block segment: Path motion to the BCS coordinate 0,0.

  Second sub-block: 180° rotation of the rotary/endless axis involved.

  Third block segment: Remaining path motion from 0,0 to the end point.

- For 2.:

If the tool path changes direction at the BCS coordinate 0,0, this will result in an unsteadiness; the rotary/endless axis position would "jump" in this position.

The face transformation monitors the target path for this effect and will automatically insert an intermediate block if required in which the rotary/endless axis is rotated accordingly.

When a respective intermediate block is generated, the NC will also consider a branch change if permitted.

Example:

N05x100 y0

N10x0

N20x-100 y100

Between the blocks N10 and N20, the NC will rotate the rotary axis from position 0 degrees to 45° before continuing the TCP motion with N20. In the meantime, the transformation branch changes.

If automatic branch change were prohibited, the NC would rotate the rotary axis to position 225°. See the following figure.

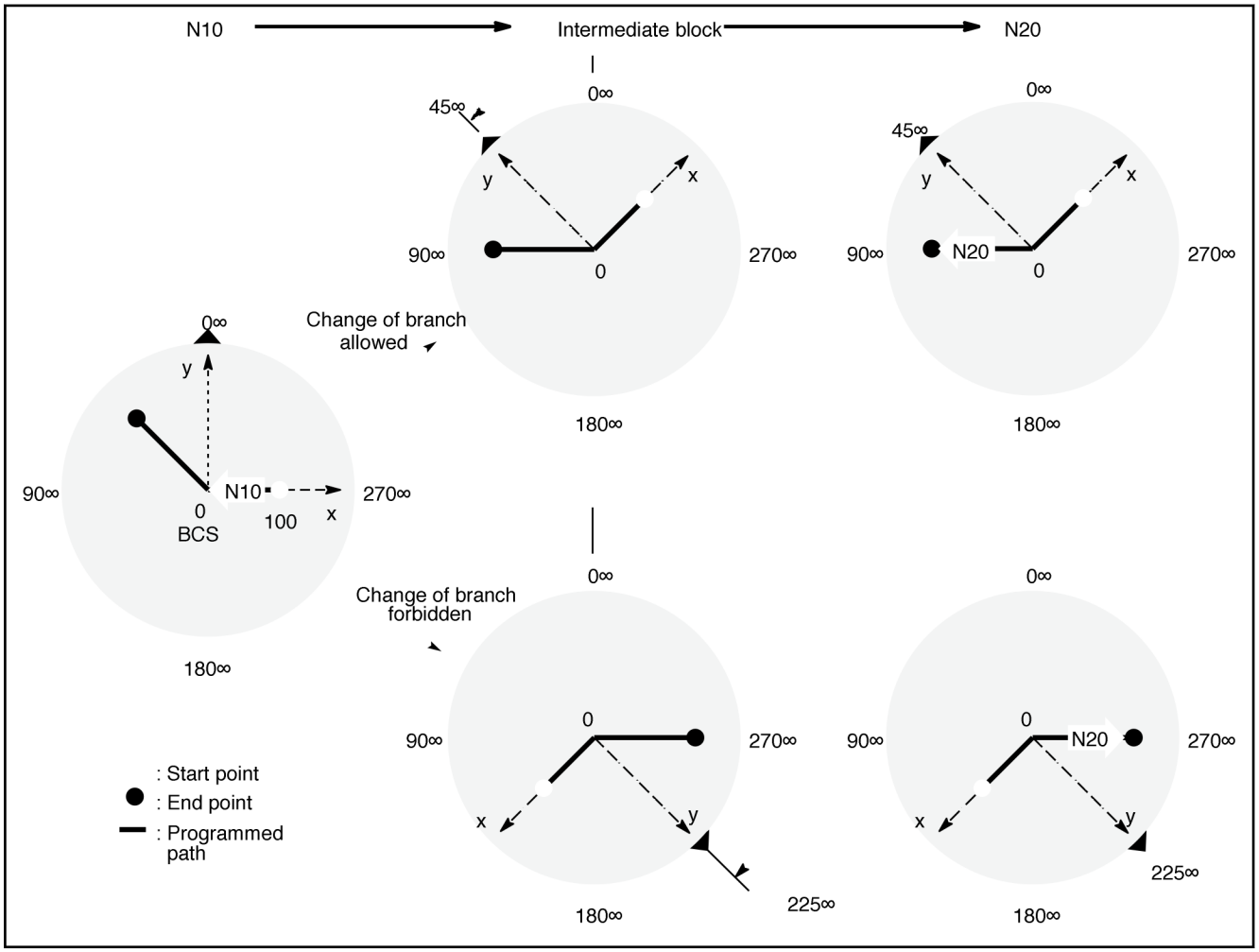


Fig. 11-100: Generation of an intermediate block on block transition in the BCS zero point

Path motions near the rotary axis of the rotary/endless axis involved:

For the approach of the tool path to the rotary axis of the involved rotary axis, different strategies are offered in dependence on the distance to the rotary axis:

- Distance ≤ 0.01 mm at the rotary axis:

  The NC imperatively leads tool paths through the BCS zero point (x=y=0).

  For passing paths, the following is applicable:

  – The block is split into two block segments. In one circle, two divided circles of the same radius as the original circle are generated if possible.

  – The first block segment ends and the second block segment starts in the zero point.

- Distance > 0.01 mm at the rotary axis

  The principle requires that the max. path velocity and path acceleration is reduced for these tool paths. The reduction is the greater the closer the path comes to the environment of the 0.01 mm limit.

  To prevent this reduction acting on the full path length (creeping motion over the full block), the path is split into sub blocks. Thus, only the block segments close to the zero point are reduced much.

11.25.5 Handling Instruction: 2/3-Axis Transformation Face

Applying: The axis transformation can be applied via the relevant machine parameters. Therefore, the axis transformation function has to be activated via the parameter **tra** "Axis transformations" in the setup (SUP). The path of the machine parameter name is always "TRA/AxTrafo[1..20]/...".

Alternatively, the axis transformation can be applied via a user-defined system date of type SysAxTrafo_t.

The content of both variant is identical.

The axis positions of the reference position have to be determined.

The respectively equivalent machine parameter numbers, machine parameter names and SD elements are specified in the following table. To apply a transformation, one of the variants has to be assigned accordingly

Transformation-relevant parameters

MP number	Description	
1030 00110	**Transformation type** MP name: TrafoType SD elem. : Type	
	2011001	2-axis variant
	3021003	3-axis variant

<table>
<tr><th>MP number</th><th colspan="4">Description</th></tr>
<tr><td rowspan="5">1030 00120</td><td colspan="4">System axes / coordinates of the transformation
MP name: FwdInCoordIndTrafo[1..8]
SD elem. : AxisAssignment[1..8]</td></tr>
<tr><td>[1]</td><td colspan="3">System axis number of X</td></tr>
<tr><td>[2]</td><td colspan="3">System axis number of C</td></tr>
<tr><td>[3]</td><td colspan="3">System axis number of Z (only 3-axis variant)</td></tr>
<tr><td>[4..8]</td><td colspan="3">Not relevant</td></tr>
<tr><td rowspan="11">1030 00125</td><td colspan="4">Axis classification of transformation axes
MP name: CoordClass[1..8]
MP name: CoordDir[1..8]
SD elem. : AxisClassification[1..8]</td></tr>
<tr><td colspan="4">Defines the axes involved in transformation, including their direction of motion.
• MP no. or SD element:
Value ">0" means:
The axis rotates/travels positively as in the figures above.
Value "<0" means:
The axis rotates/travels negatively as in the figures above.
Possible values are:
– +/-1 = x-axis
– +/-2 = y-axis
– +/-3 = z-axis
– +/-100 = a-axis
– +/-200 = b-axis
– +/-300 = c-axis
• MP name:
– Possible values for CoordClass are
– X,Y,Z (linear)
– A,B,C (rotary)
– Possible values for CoordDir are:
– positive (regarding Fig. "Reference position"
– negative (regarding Fig. "Reference position"</td></tr>
<tr><td colspan="4">If a parameter is not set, the following default settings apply:</td></tr>
<tr><td></td><td>MP no /
SD elem.:</td><td>CoordClass:</td><td>CoordDir:</td></tr>
<tr><td>[1]</td><td>1</td><td>X</td><td>Positive</td></tr>
<tr><td>[2]</td><td>300</td><td>C</td><td>Positive</td></tr>
<tr><td>[3]</td><td>3</td><td>Z</td><td>Positive</td></tr>
<tr><td>[4..8]</td><td>Not relevant</td><td>0</td><td>Positive</td></tr>
</table>

MP number	Description	
1030 00130	**Axis positions of the reference position** MP name: RefPosTrafo[1..8] SD elem. : AxZeroPos[1..8]	
	[1]	X^0
	[2]	C^0
	[3]	Z^0
	[4..8]	Not relevant
1030 00140	**Length and angle parameters** MP name: JointParTrafo[1..16] SD elem. : LenParam[1..16]	
	Here, specify whether linear axis X may cross the workpiece center point or not.	
	[1]	0 : Linear axis X may cross the center point. +1 : Linear axis X on positive positions restricted -1 : Linear axis X on negative positions restricted
	[2..16]	Not relevant
1030 00160	**Epsilon environments** MP name: EpsAxTrafo SD elem. : EpsilonRanges	
	0.01	

Tab. 11-115: Transformation-relevant parameters

Activating

If the face transformation is configured via

- Machine parameters. Then, the transformation is enabled in the NC program with (i=1,..20):

  COORD(<i>)

- System date. Then, the transformation is enabled in the NC program with (type "SysAxTrafo_t"):

  COORD(<SD.MtbAxTrafo>)

Example Program:

The face transformation is defined in the fourth machine parameter block. No axis transformation is active at programming start:

Enable transformation

```
N100  G1 F1000 XA10 ZA10 C30   ;the axes XA,ZA,C are traversed
N110  COORD(4)                 ;Switching on the axis transformation
N130  x20 y30                  ;the coordinates x,y,phi are traversed
N140  x10 z10                  ;the coordinates x,z,theta are traversed
        :
N999  COORD(0)                 ;Switching-off the axis transformation
                               ;Axes are again programmed from here
```

Deactivating

COORD(0) or **COORD(0,2)**

switches off the axis transformation in the NC program.

11.26 3-axis transformation LLR - Cartesian

11.26.1 Description

Function

Transformation name	Transformation type	Point of action
LLR - Cartesian 3-axis transformation	3021005	AT1
LLR - Cartesian 3-axis transformation	3021004	AT2

The kinematics of the 3-axis transformation LLR consists of two linear axes L_1, L_2 and one rotary axis R_1. The linear axes L_1 and L_2 are perpendicular to each other. The rotation axis R_1 rotates the tool around the L_2 axis using a cantilever and offsets it in the direction of space in which it could not have been offset by using L_1 and L_2. The three space coordinates of a tool reference point (spindle nose) are programmed. In case of an active tool correction G47 (XTR, YTR, ZTR) the coordinates of the tool center point (TCP) are calculated. As the tool orientation changes during the R_1 rotation, the axis transformation can only be used for tools where the orientation of the tool is not of vital importance.

The transformation can be used at point of action 1 and at point of action 2.

As axis transformation at point of action 1 (AT1), it acts between the machine coordinate system ACS = (L_1,L_2,R_1) and the basic coordinate system MCS = (X,Y,Z).

- The X, Y, Z, MCS input variables are programmed
- The ACS output coordinates L_1, L_2, L_1 are calculated

As axis transformation at point of action 2 (AT2), it acts between the machine coordinate system MCS = (L_1,L_2,R_1) and the basic coordinate system BCS = (x,y,z).

- The x, y, z, BCS input variables are programmed
- The MCS output coordinates L_1, L_2, L_1 are calculated

The point of action is specified in machine parameter 103000110 "Transformation type":

Point of action 1: MP 103000110 = 3021005

Point of action 2: MP 103000110 = 3021004

In machine parameter 103000120, the "system axes/system coordinates of the transformation" are specified:

System axis number of the L_1 axis: MP 103000120 (1)

System axis number of the L_2 axis: MP 103000120 (2)

System axis number of the R_1 axis: MP 103000120 (3)

The remaining inputs MP 103000120 (4..8) are not relevant.

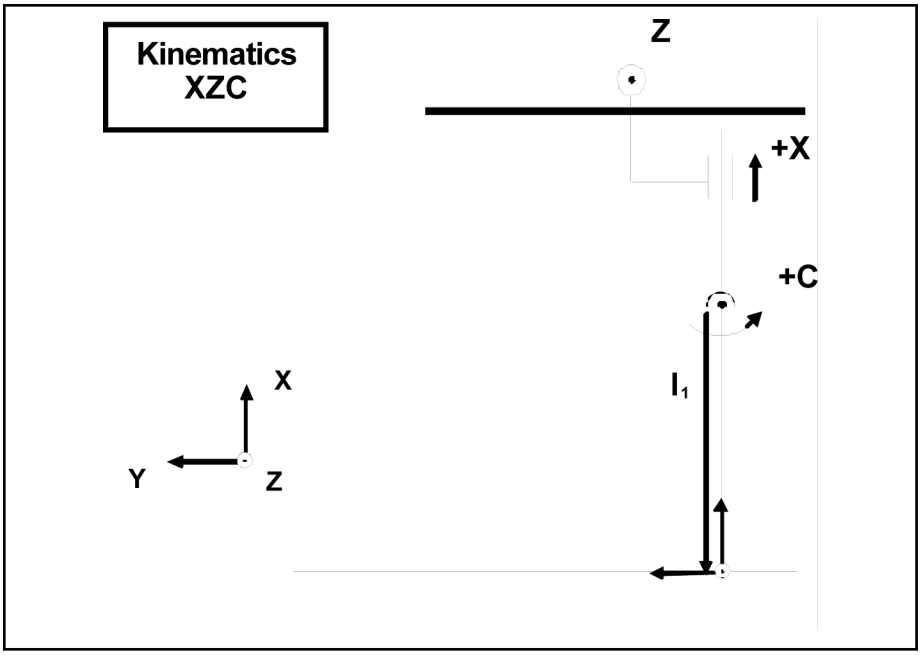


Fig. 11-101: XZC kinematics

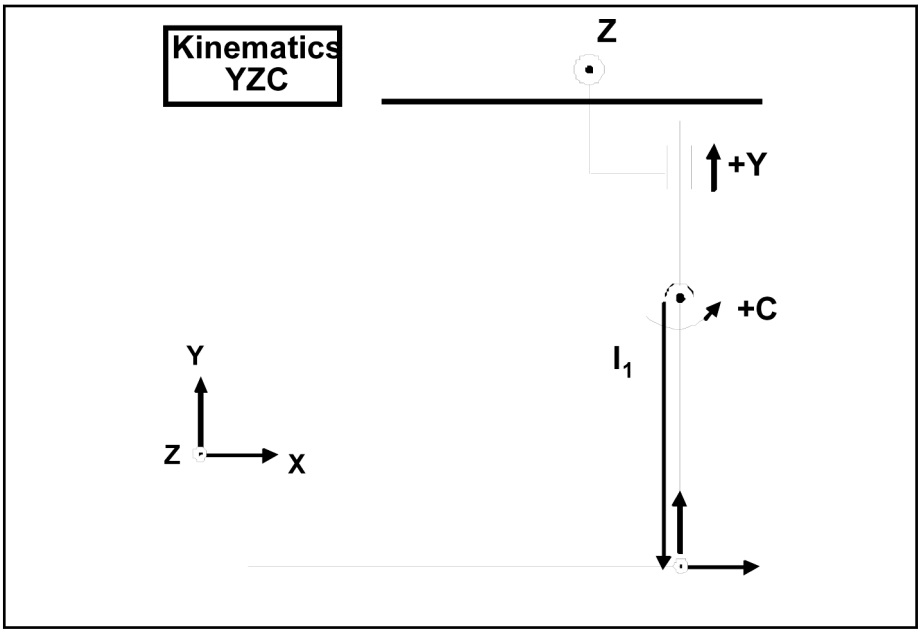


Fig. 11-102: YZC kinematics

Variants

Using machine parameter 103000125 "Axis classification of transformation axes", the meaning of each axis for AT is specified:

The axis with the meaning X for AT specifies the x/X direction of the input coordinate system.

The axis with the meaning Y for AT specifies the y/Y direction of the input coordinate system.

The axis with the meaning Z for AT specifies the z/Z direction of the input coordinate system.

The direction of the input coordinate system not specified in this way is perpendicular to the other two, resulting in a right-handed input coordinate system.

XZC kinematics:

For AT, the L_1 axis has the following meaning X: MP 103000125 (1) = 1

For AT, the L_2 axis has the following meaning Z: MP 103000125 (2) = 3

For AT, the R_1 axis has the following meaning C: MP 103000125 (3) = 300

For this kinematics, the L_1 axis determines the x/X direction and the L_2 axis determined the z/Z direction of the input coordinate system. The y/Y direction of the input coordinate system is perpendicular to the other two resulting in a right-handed coordinate system. fig. 11-101 "XZC kinematics" on page 394 shows this kinematics for the point of action 2, i.e. with the BCS as input coordinate system.

YZC kinematics:

For AT, the L_1 axis has the following meaning Y: MP 103000125 (1) = 2

For AT, the L_2 axis has the following meaning Z: MP 103000125 (2) = 3

For AT, the R_1 axis has the following meaning C: MP 103000125 (3) = 300

For this kinematics, the L_1 axis determines the y/Y direction and the L_2 axis determined the z/Z direction of the input coordinate system. The x/X direction of the input coordinate system is perpendicular to the other two resulting in a right-handed coordinate system. fig. 11-102 "YZC kinematics" on page 394 shows this kinematics for the point of action 2, i.e. with the BCS as input coordinate system.

YXA kinematics:

For AT, the L_1 axis has the following meaning Y: MP 103000125 (1) = 2

For AT, the L_2 axis has the following meaning X: MP 103000125 (2) = 1

For AT, the R_1 axis has the following meaning A: MP 103000125 (3) = 100

fig. 11-101 "XZC kinematics" on page 394 shows this kinematics if X is replaced by Y, Z by X, C by A, x by y, y by z, at the point of action 2

ZXA kinematics:

For AT, the L_1 axis has the following meaning Z: MP 103000125 (1) = 3

For AT, the L_2 axis has the following meaning X: MP 103000125 (2) = 1

For AT, the R_1 axis has the following meaning A: MP 103000125 (3) = 100

fig. 11-102 "YZC kinematics" on page 394 shows this kinematics if Y is replaced by Z, Z by X, C by A, y by z, x by y, at the point of action 2.

ZYB kinematics:

For AT, the L_1 axis has the following meaning Z: MP 103000125 (1) = 3

For AT, the L_2 axis has the following meaning Y: MP 103000125 (2) = 2

For AT, the R_1 axis has the following meaning B: MP 103000125 (3) = 200

fig. 11-101 "XZC kinematics" on page 394 shows this kinematics if X is replaced by Z, Z by Y, C by B, x by z, y by x, at the point of action 2.

XYB kinematics:

For AT, the L_1 axis has the following meaning X: MP 103000125 (1) = 1

For AT, the L_2 axis has the following meaning Y: MP 103000125 (2) = 2

For AT, the R_1 axis has the following meaning B: MP 103000125 (3) = 200

fig. 11-102 "YZC kinematics" on page 394 shows this kinematics if Y is replaced by X, Z by Y, C by B, y by x, x by z, at the point of action 2.

The remaining inputs MP 103000125 (4..8) are not relevant.

A negative sign is valid for each entry in the machine parameter 103000125. In this case, the relevant axis is taken into consideration in reversed direction for AT, e.g.:

YZC kinematics:

For AT, the $-L_1$ axis has the meaning Y: MP 103000125 (1) = - 2

For AT, the L_2 axis has the following meaning Z: MP 103000125 (2) = 3

For AT, the $-R_1$ axis has the meaning A: MP 103000125 (3) = - 300

Reference position

fig. 11-101 "XZC kinematics" on page 394 and fig. 11-102 "YZC kinematics" on page 394 show the axis transformations in reference position.

In this position, the tool reference point is positioned in the zero point of the input coordinate system and the output coordinate values are to be read out in this position. Enter these values as "Axis positions of the reference position" in the machine parameter 103000130 (1) for the L_1 axis, 103000130 (2) for the L_2 axis and 103000130 (3) for the R_1 axis.

The distance between the R_1 axis and the tool reference point for the kinematics XZC is specified by MP 103000140 (1), as the respective distance vector of the R_1 axis to the tool reference point in the reference position (fig. 11-101 "XZC kinematics" on page 394) is oriented parallely to the x/X direction of the input coordinate system. MP 103000140 (1) can be specified as either positive or negative value.

In a positive case, the distance vector in case of reference position points in the x/X direction of the input coordinate system, in a negative case, it points to the -x/X direction.

fig. 11-101 "XZC kinematics" on page 394 describes the case scenario MP 103000125 (1) > 0 and MP 103000140 (1) < 0.

Correspondingly, this distance for the kinematics XZC is specified by MP 103000140 (2) (fig. 11-102 "YZC kinematics" on page 394), for YXA by MP 103000140 (2), for ZXA by MP 103000140 (3), for ZYB by MP 103000140 (3) and for XYB by MP 103000140 (1).

All other entries in machine parameter 103000140 are not relevant.

The machine parameter 103000160 is also not relevant.

Special features

Ambiguous axis positions

If a position is specified in the input coordinate system, this position can be reached using two different axis positions in the output coordinate system. These two axis position are called transformation branches. The transformation branch is specified by the value of the R_1-axis.

Tool correction

Tool corrections are only supported in point of action 2, i.e. only for AT 3021004.

If the tool corrections are activated with G47(XTR,YTR,ZTR), they are applied to the tool coordinate system (TCS). In reference position, the TCS matches the BCS. However, it rotates with the previously mentioned distance vector of the R_1 axis to the tool reference point.

Relevant NC functions

COORD(...)	The 3-axis transformation LLR is activated. This can optionally be done using the machine parameter block (i=1..20) or via the MTB system data (of type SysAx-Trafo_t).
COORD(0,2) - or - COORD(0,1)	The active transformation is deactivated.

Tab. 11-116: Relevant NC functions

Relevant machine parameters

Up to 20 different axis transformations are declared with the following machine parameter block in the machine parameters.

1030 00110	Defines the transformation type.
1030 00120	Defines the system axes/coordinates participating in the transformation.
1030 00125	Defines the axis classification of the transformation axes.
1030 00130	Defines the axis positions of the reference position.
1030 00140	Defines the length and angle parameters.
1030 00160	Epsilon environments

Tab. 11-117: Relevant machine parameters

Relevant system data

The axis transformation can also be defined via system date instead of the machine parameters. Therefore, a system variable of type "SysAxTrafo_t" has to be created (see "MTX Machine Parameters", chapter "System data"). The structure type "SysAxTrafo_t" is control-internally defined in the xml schema files "/feprom/schemas/sdaxtrf.xsd" and contains - analog to the relevant machine parameters - the following transformation-relevant structure elements.

Structural element	Description
Type	Defines the transformation type.
AxisAssignment	Defines the system axes/coordinates participating in the transformation.
AxisClassification	Defines the axis classification of the transformation axes.
AxZeroPos	Defines the axis positions of the reference position.
LenParam	Defines the length and angle parameters.
EpsilonRanges	Epsilon environments

Tab. 11-118: Relevant system data

11.26.2 Handling instruction

Applying The axis transformation can be applied via the relevant machine parameters. Therefore, the axis transformation function has to be activated via the parameter **tra** "Axis transformations" in the setup (SUP). The path of the machine parameter name is always "TRA/AxTrafo[1..20]/...".

Alternatively, the axis transformation can be applied via a user-defined system date of type SysAxTrafo_t.

The content of both variants is identical.

The axis positions of the reference position have to be determined.

The respectively equivalent machine parameter numbers, machine parameter names and SD elements are specified in the following table. To apply a transformation, one of the variants has to be assigned accordingly.

Transformation-relevant parameters

MP number	Description	
1030 00110	**Transformation type** MP name: TrafoType SD elem.: Type	
	3021005	Point of action AT1
	3021004	Point of action AT2
1030 00120	**System axes/coordinates of the transformation** MP name: FwdInCoordIndTrafo[1..8] SD elem.: AxisAssignment[1..8]	
	[1]	System axis number of L_1
	[2]	System axis number of L_2
	[3]	System axis number of R_1
	[4..8]	Not relevant

<table>
<tr><th>MP number</th><th colspan="4">Description</th></tr>
<tr><td rowspan="11">1030 00125</td><td colspan="4">Axis classification of transformation axes
MP name: CoordClass[1..8]
MP name: CoordDir[1..8]
SD elem.: AxisClassification[1..8]</td></tr>
<tr><td colspan="4">Defines the axes involved in transformation, including their direction of motion.
• MP no. or SD element:
Value ">0" means:
The axis rotates/travels positively as in the figures above.
Value "<0" means:
The axis rotates/travels negatively as in the figures above.
Possible values are:
– +/-1 = x-axis
– +/-2 = y-axis
– +/-3 = z-axis
– +/-100 = a-axis
– +/-200 = b-axis
– +/-300 = c-axis
• MP name:
– Possible values for CoordClass are:
– X,Y,Z (linear)
– A,B,C (rotary)
– Possible values for CoordDir are:
– positive (regarding fig. "Reference position"
– negative (regarding fig. "Reference position"</td></tr>
<tr><td colspan="4">If a parameter is not set, the following default settings apply:</td></tr>
<tr><td></td><td>MP no /
SD elem.:</td><td>CoordClass:</td><td>CoordDir:</td></tr>
<tr><td>[1]</td><td>1</td><td>X</td><td>Positive</td></tr>
<tr><td>[2]</td><td>3</td><td>Z</td><td>Positive</td></tr>
<tr><td>[3]</td><td>300</td><td>C</td><td>Positive</td></tr>
<tr><td>[4..8]</td><td>Not relevant</td><td>0</td><td>Positive</td></tr>
<tr><td rowspan="5">1030 00130</td><td colspan="4">Axis positions of the reference position
MP name: RefPosTrafo[1..8]
SD elem.: AxZeroPos[1..8]</td></tr>
<tr><td>[1]</td><td colspan="3">L_1^0</td></tr>
<tr><td>[2]</td><td colspan="3">L_2^0</td></tr>
<tr><td>[3]</td><td colspan="3">R_1^0</td></tr>
<tr><td>[4..8]</td><td colspan="3">Not relevant</td></tr>
</table>

MP number	Description	
1030 00140	**Length and angle parameters** MP name: JointParTrafo[1..16] SD elem.: LenParam[1..16]	
	Defines the length vector l_1 in [mm].	
	[1]	Only relevant for the kinematics XZC and XYB
	[2]	Only relevant for the kinematics YZC and YXA
	[3]	Only relevant for the kinematics ZXA and ZYB
	[4..16]	Generally not relevant
1030 00160	**Epsilon environments** MP name: EpsAxTrafo SD elem.: EpsilonRanges	
	Not relevant	

Tab. 11-119: Transformation-relevant parameters

Activating

If the 3-axis transformation is configured via

- machine parameter, the transformation is enabled in the NC program with (i=1,..20):

  COORD(<i>)

- system date, the transformation is enabled in the NC program with (type "SysAxTrafo_t"):

  COORD(<SD.MtbAxTrafo>)

Example program:

The 3-axis transformation is defined in the fourth machine parameter block. No axis transformation is active at programming start:

Machine parameter setting for program in channel 1:

1003 00 001 (1..8) XS1 XS2 XS3 XS4 XS5 XS6 CS1 CS2

1003 00 002 (1..8) 1 1 1 1 1 1 1 1

1030 00 110 (4) 3021004

1030 00 120 (4) 5 3 8

1030 00 125 (4) 3 2 200

1030 00 130 (4) 0 0 0

1030 00 140 (4) 0 0 90

1030 00 160 (4) 0.1

7010 00 010 (1) X1 X2 X3 X4 X5 X6 C1 C2

7080 00 010 (1) x y z phi theta psi

Program:

```
010   A!  = 100 * SQRT(3)
020   A3!  = A! / 3
030   AM! = -A!           :  A3M! = -A3!
N040  Coord(0,2)
N050  Coord(0,1)
N060  G53 G48 ED0
N070  X1=0 X2=0 X3=0 X4=0 X5=0 X6=0 C1=0 C2=0 G1 F10000
080   DCT(1,5,0) = A3!
090   DCT(2,5,0) = -30
```

```
100   DCT(3,5,0) = -10
N110  Coord(4)
;                                        XS5 XS3 CS2
N120  G47(XTR,YTR,ZTR) ED5   ;   =>    0   0   0
N130  x[A3M!] y0 z10         ;   =>    0  -30   0
N140  x[A3!]                 ;   =>    0  -30   60
N150  G48 ED0
N160  Coord(0)
      M30
```

Deactivating **COORD(0)** or **COORD(0,2)**
switches off the axis transformation in the NC program.

11.27 Polar kinematics

11.27.1 Description

Function

Transformation name	Transformation type	Point of action
Polar kinematics		
2-axis variant	2011003	AT2
3-axis variant	3021006	AT2

The machine configuration of the polar kinematics consists of a rotary axis c on which a linear axis x is mounted. The linear axis rotates along by the rotary axis. The vertical axis z can optionally be available (cf. fig. 11-103 "2/3-axis polar kinematics (side view)" on page 401, fig. 11-104 "2/3-axis polar kinematics (top view)" on page 402). The transformation is used at point of action 2, i.e. it acts between the machine coordinate system MCS = (C,X (Z)) and the basic coordinate system BCS = (x,y (z)).

- Programmed are the BCS input values x, y (z)
- Calculates the MCS output values c, x (z)

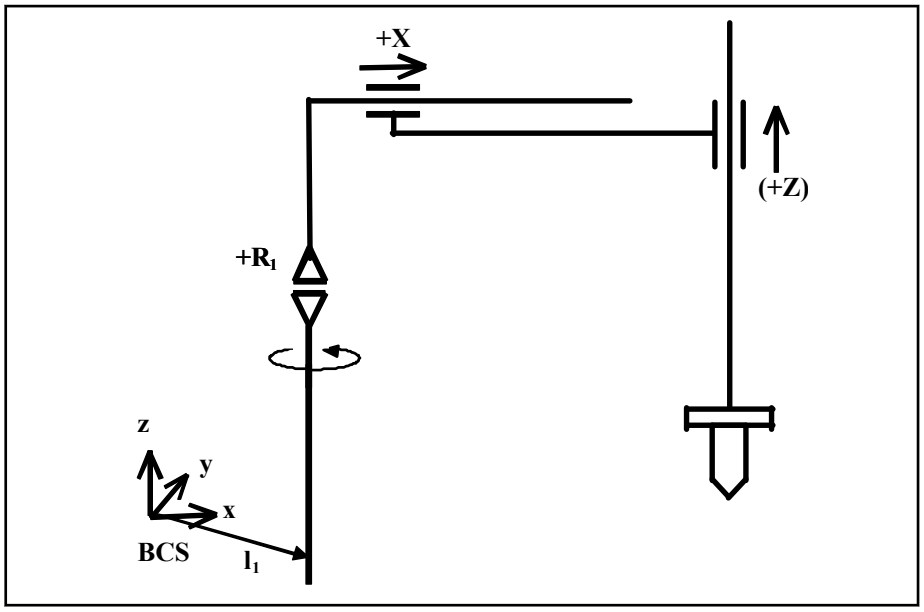


Fig. 11-103: 2/3-axis polar kinematics (side view)

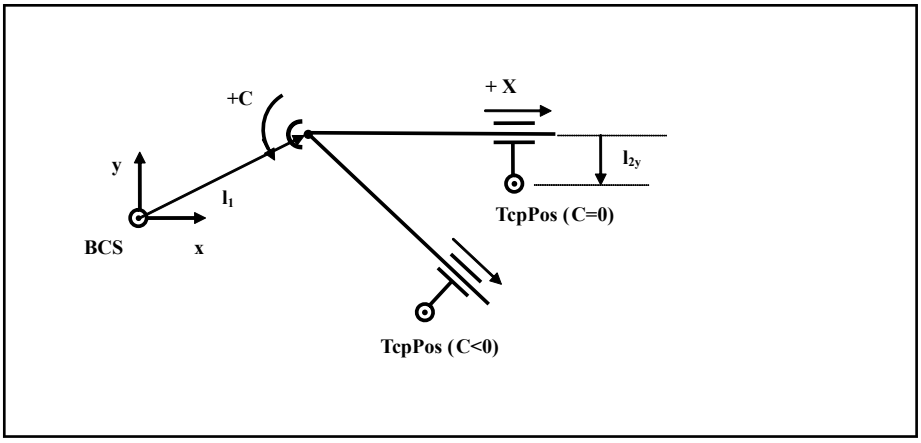


Fig. 11-104: 2/3-axis polar kinematics (top view)

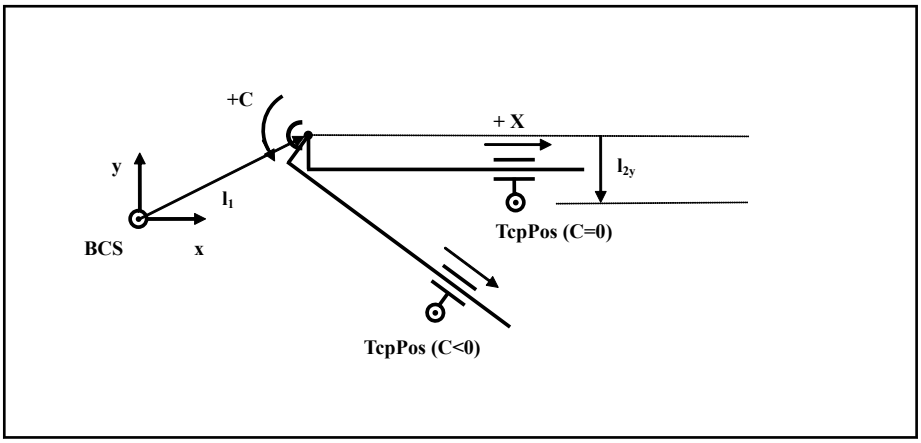


Fig. 11-105: 2/3-axis polar kinematics (top view)

The BCS zero point can be located at any position of the working face (refer to fig. 11-103 "2/3-axis polar kinematics (side view)" on page 401).

The vectors l_1 and l_2 define the geometry of the polar kinematics. l_1 points from the BCS zero point to the rotary axis c. l_2 describes the eccentricity of the tool center point to the x-axis.

Reference position

- In the reference position (C=0), the x-axis is aligned in parallel to the x-coordinate of the BCS.
- The zero point of the x-axis is located in rotary axis c.
- The tool points in negative z-direction if available (tool point vertically towards the bottom).
- The BCS is parallel to the MCS.

The BCS zero point is in rotary axis R_1.

Special features

Ambiguous axis positions

If a TCP position [x,y (z)] is specified, this position can be reached using two different axis positions. These are called transformation branches. The transformation branch is determined by the sign of the x-axis:

- Branch +1: X >= 0
- Branch -1: X < 0
- The BCS is parallel to the MCS.

Which of the two transformation branches is used depends on the specified position of the x-axis when switching on the axis transformation. Accordingly, a transition from branch +1 to -1 and vice versa is only possible when traveling through the singularity at X=0.

Tool correction

Axis transformation supports a tool correction in the tool coordinate system TCS. In the reference position, BCS corresponds to TCS_0 (spindle nose). The tool correction vector $_t$ points from the corrected tool center point TCP_1 to the uncorrected TCP_0. It comprises the length corrections L1, L2 and L3 from the D-correction table.

Relevant NC functions

COORD(...)	The axis transformation of the polar kinematics is enabled. This can optionally be done using the machine parameter block (i=1..20) or via the MTB system data (of type SysAxTrafo_t).
COORD(0) - or - COORD(0,2)	The active transformation is deactivated.

Tab. 11-120: Relevant NC functions

Relevant machine parameters

Up to 20 different axis transformations are declared with the following machine parameter block in the machine parameters.

1030 00110	Defines the transformation type.
1030 00120	Defines the system axes/coordinates participating in the transformation.
1030 00125	Defines the axis classification of the transformation axes.
1030 00130	Defines the axis positions of the reference position.
1030 00140	Defines the length and angle parameters.
1030 00160	Epsilon environments

Tab. 11-121: Relevant machine parameters

Relevant system data

The axis transformation can also be defined via system date instead of the machine parameters. Therefore, a system variable of type "SysAxTrafo_t" has to be created (see "MTX Machine Parameters", chapter "System data"). The structure type "SysAxTrafo_t" is control-internally defined in the xml schema files "/feprom/schemas/sdaxtrf.xsd" and contains - analog to the relevant machine parameters - the following transformation-relevant structure elements.

Structural element	Description
Type	Defines the transformation type.
AxisAssignment	Defines the system axes/coordinates participating in the transformation.
AxisClassification	Defines the axis classification of the transformation axes.
AxZeroPos	Defines the axis positions of the reference position.

LenParam	Defines the length and angle parameters.
EpsilonRanges	Epsilon environments

Tab. 11-122: Relevant system data

11.27.2 Handling instruction: Polar kinematics

Applying

Applying

The axis transformation can be applied via the relevant machine parameters. Therefore, the axis transformation function has to be activated via the parameter **tra** "Axis transformations" in the setup (SUP). The path of the machine parameter name is always "TRA/AxTrafo[1..20]/...".

Alternatively, the axis transformation can be applied via a user-defined system date of type SysAxTrafo_t.

The content of both variants is identical.

The axis positions of the reference position have to be determined.

The respectively equivalent machine parameter numbers, machine parameter names and SD elements are specified in the following table. To apply a transformation, one of the variants has to be assigned accordingly.

Transformation-relevant parameters

MP number	Description	
1030 00110	**Transformation type** MP name: TrafoType SD elem.: Type	
	2011003	2-axis variant
	3021006	3-axis variant
1030 00120	**System axes/coordinates of the transformation** MP name: FwdInCoordIndTrafo[1..8] SD elem.: AxisAssignment[1..8]	
	[1]	System axis number of C
	[2]	System axis number of X
	[3]	System axis number of z (if available)
	[4..8]	Not relevant

<table>
<tr><th>MP number</th><th colspan="4">Description</th></tr>
<tr><td rowspan="13">1030 00125</td><td colspan="4">Axis classification of transformation axes
MP name: CoordClass[1..8]
MP name: CoordDir[1..8]
SD elem.: AxisClassification[1..8]</td></tr>
<tr><td colspan="4">Defines the axes involved in transformation, including their direction of motion.
• MP no. or SD element:
Value ">0" means:
The axis rotates/travels positively as in the figures above.
Value "<0" means:
The axis rotates/travels negatively as in the figures above.
Possible values are:
– +/-1 = x-axis
– +/-2 = y-axis
– +/-3 = z-axis
– +/-100 = a-axis
– +/-200 = b-axis
– +/-300 = c-axis
• MP name:
– Possible values for CoordClass are:
– X, Y, Z (linear)
– A, B, C (rotary)
– Possible values for CoordDir are:
– positive (regarding fig. "Reference position"
– negative (regarding fig. "Reference position"</td></tr>
<tr><td colspan="4">If a parameter is not set, the following default settings apply:</td></tr>
<tr><td></td><td>MP no /
SD elem.:</td><td>CoordClass:</td><td>CoordDir:</td></tr>
<tr><td>[1]</td><td>300</td><td>C</td><td>Positive</td></tr>
<tr><td>[2]</td><td>1</td><td>X</td><td>Positive</td></tr>
<tr><td>[3]</td><td>3 (if available)</td><td>Z</td><td>Positive</td></tr>
<tr><td>[4..8]</td><td>Not relevant</td><td>0</td><td>Positive</td></tr>
<tr><td colspan="4">Axis positions of the reference position
MP name: RefPosTrafo[1..8]
SD elem.: AxZeroPos[1..8]</td></tr>
<tr><td>[1]</td><td colspan="3">C^0</td></tr>
<tr><td>[2]</td><td colspan="3">X^0</td></tr>
<tr><td>[3]</td><td colspan="3">Z^0 (if available)</td></tr>
<tr><td>[4..8]</td><td colspan="3">Not relevant</td></tr>
</table>

MP number	Description	
1030 00140	**Length and angle parameters** MP name: JointParTrafo[1..16] SD elem.: LenParam[1..16]	
	Defines the length vectors l_1, l_2 in [mm].	
	[1]	l_{1x}
	[2]	l_{1y}
	[3]	l_{1z}
	[4]	l_{2x}
	[5]	l_{2y}
	[6]	l_{2z}
	[7..16]	Not relevant
1030 00160	**Epsilon environments** MP name: EpsAxTrafo SD elem.: EpsilonRanges	
	0.01	

Tab. 11-123: Transformation-relevant parameters

Activating

Activating *If the 5-axis transformation is configured via*

- machine parameter, the transformation is enabled in the NC program with (i=1,..20):

  COORD(<i>)
- system date, the transformation is enabled in the NC program with (type "SysAxTrafo_t"):

  COORD(<SD.MtbAxTrafo>)

Example program:

The 2-axis polar kinematics is defined in the fourth machine parameter block. No axis transformation is active at programming start:

```
N01 G1 F1000 XA10 C30          ;the XA, C axes are traversed
N02 COORD(4)                   ;Activating 2-axis transformation
N03 x20 y30                    ;the x, y coordinates are traversed
N04 x10 y100                   ;the x, y coordinates are traversed

N99 COORD(0)                   ;Deactivating 2-axis transformation
                               ;The axes are again programmed from here
```

Deactivating

Deactivating **COORD(0)** or **COORD(0,2)**
switches off the axis transformation in the NC program.

11.28 Axis kinematic calibration

11.28.1 Description

Basics

The axis kinematic calibration function is used to optimize axis transformation configuration parameters such that a higher positioning accuracy is achieved. This calculation is executed automatically with the help of measuring cycles.

When axis transformation is active, several measuring points are approached in a measuring cycle and the actual values of the coordinates are measured by means of a suitable measuring system (e.g., touch probe). The measuring data, consisting of a number of actual and command axis/coordinate values, are either offset directly in a calibration file, or - depending on the measuring method in question - first processed in a measuring cycle file and then offset in a calibration file.

Then, the NC will calculate the optimum configuration parameters from the measurement data.

Optimizable configuration parameters:

Usually, axis kinematics is defined from the following configuration parameters:

MP 103000110:	Axis transformation type
MP 103000120:	System axes of the transformation
MP 103000130:	Axis positions of the reference position
MP 103000140:	Length and angle parameters
MP 103000150:	Reference position of tool coordinate system

Of the two faulty parameters MP 103000130 and MP 103000140, only the **length and angle parameters** stored in MP 103000140 are optimized with the axis kinematic calibration function.

Optimizing of MP 103000130 "Axis positions of the reference position" is not provided.

Selection of axis transformation:

Axis kinematic calibration can be used in all axis transformations, with AT1 as well as with AT2.

Transformation between the axis coordinate system ACS and the Cartesian machine coordinate system MCS is usually required when the ACS has no Cartesian design (typical kinematics for point of action AT1: rod kinematics as e.g. Bipod, SCARA, and non-orthogonal linear axes).

The Cartesian MCS generated with the axis transformation at AT1 is transformed to the basic coordinate system (BCS) on application of another transformation at AT2 (e.g. 5-axis transformation) on the basic coordinate system BCS.

For axis kinematic calibration, this means:

If the measuring cycle is executed at AT2 with active axis transformation, the latter is automatically considered the axis transformation to be calibrated.

If there is no axis transformation at AT2, the measurement data refers to the axis transformation at AT1.

11.28.2 Measuring methods

Function

The axis kinematic calibration measuring methods comprise two different methods to measure the tool center point (TCP):

- Direct measuring:

  At this measurement, the measuring result is independent of the configuration parameters of the current axis transformation (e.g. laser interferometry).

- Indirect measuring:

  The TCP (tool center point) is measured indirectly. At this measurement (e.g. measuring of a swage block with axis touch probe), the result depends on the parameters, as the machine or basic coordinates are calculated based on the current parameter block from the sampled axis positions via forward transformation.

Direct measuring: The preset coordinates command value (exactly known coordinate position in the WCS) $\vec{\rho}^{e}$ = ($p^{e}_{i,1}$, $p^{e}_{i,2}$,...., $p^{e}_{i,c}$) and the directly measured WCS coordinates actual value are assigned in the WCS $\vec{\rho}^{m}$ = ($p^{m}_{i,1}$, $p^{m}_{i,2}$,...., $p^{m}_{i,c}$).

The deviation | $\vec{\rho}^{e}$ - $\vec{\rho}^{m}$ | is generated by an inaccurate parameter block $\vec{p}$ = (P_1, P_2, ..., P_n) (= length and angle parameters, ...) in the current axis transformation.

☞ Other errors which the axis transformation cannot compensate for (lead screw error, curved axis guidance) are not considered here.

The coordinate point $\vec{\rho}^{e}$ is assigned to the axis point $\vec{q}^{e}_{i}$ in the ACS. Both are linked to each other via the forward transformation $\vec{\rho}^{e}$ = $\vec{f}$ ($\vec{q}^{e}_{i}$ $\vec{p}$) (see the following illustration).

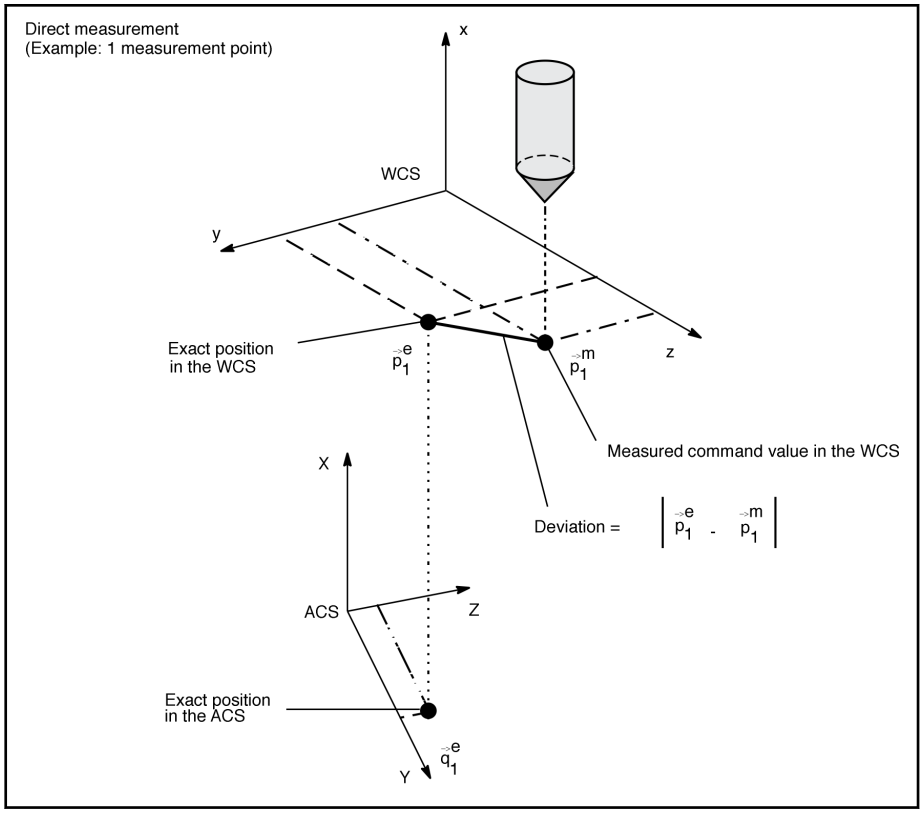


Fig. 11-106: Direct measuring

In the calculation (iterated linear adjustment calculation), is specified to ensure that the "quadratic deviation" (criterion function) between the actual and the command coordinate values is reduced to a **minimum**.

$$E(\vec{P}) = \sum_{i=1}^{k} \left| \overrightarrow{pi}^{m} - \overrightarrow{pi}^{e} \right|^2 = \sum_{i=1}^{k} \left| \overrightarrow{pi}^{m} - f(q, \vec{P}^{e}) \right|^2$$

"k" the number of the measuring points.

At iteration, the coordinate points $\overrightarrow{\rho i}^{m}$ and the axis points $\overrightarrow{qi}^{e}$ are constant.

The parameter block is varied $\vec{\rho}$.

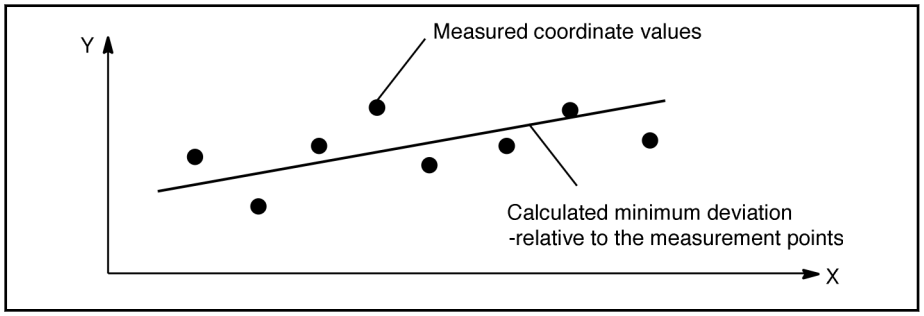


Fig. 11-107: Iteration

Indirect measurement (variant 1):

The indirect measuring method measures known WCS positions exactly $\overrightarrow{\rho}^{e}$, e.g. bores on a swage block (see chapter "Calibration cycle" on page 420).

At measuring the points, a touch probe provides the axis points first $\vec{qi}^{m}$ in the ACS that have to be converted to coordinate points via $\vec{\rho i}^{m} = \vec{f}(\vec{qi}^{m}, \vec{\rho})$. In this case, the quality function is

$$E(\vec{P}) = \sum_{i=1}^{k} \left| \vec{pi}^{m} - \vec{pi}^{e} \right|^2 = \sum_{i=1}^{k} \left| \vec{pi}^{m} - f(q, \vec{P}^{e}) \right|^2$$

Calculation of a bore position:

A bore i is measured by sampling three axis points $\vec{qi,1}^{m}$, $\vec{qi,2}^{m}$,..., $\vec{qi,3}^{m}$. To specify the center point $\vec{\rho i}^{m}$, the three sample coordinate points are determined first via $\vec{pji}^{m} = \vec{f}(\vec{qji}^{m}, \vec{\rho})$ From these, the center point of the circle $\vec{\rho i}^{m}$ can be calculated which in turn depends on the parameter block used $\vec{\rho}$.

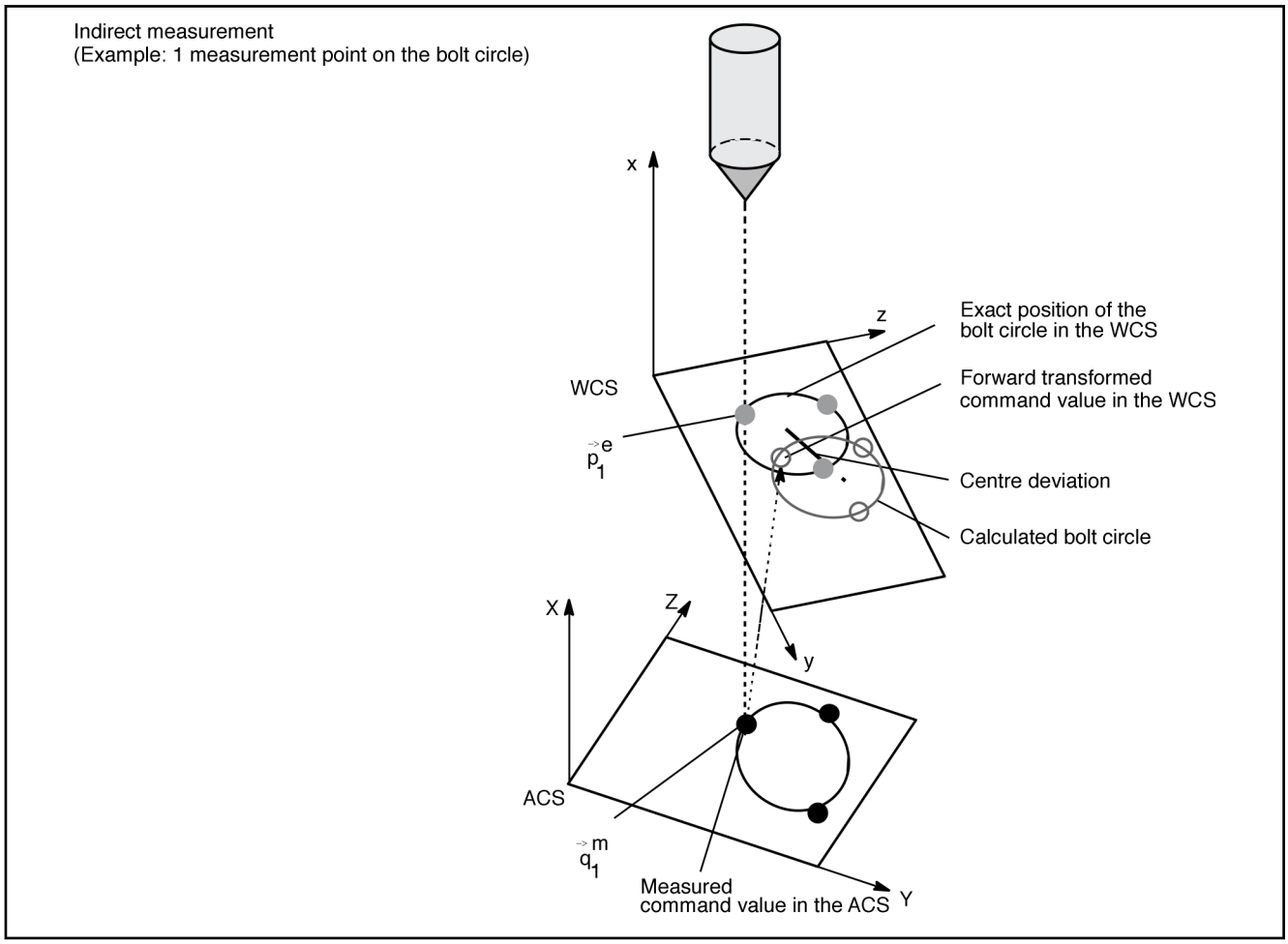


Fig. 11-108: Indirect measuring

Indirect measurement (variant 2):

The following variant 2 of the indirect measuring method described should be preferably used (compared to variant 1), since the convergence characteristics are better. Variant 2 is based on sampling the calibrated ball. The existence of this calibrated ball is absolutely mandatory. Additionally, all transformation-relevant axes have to be detected per touch probe. That means that all transformation-relevant axes are connected to the touch probe.

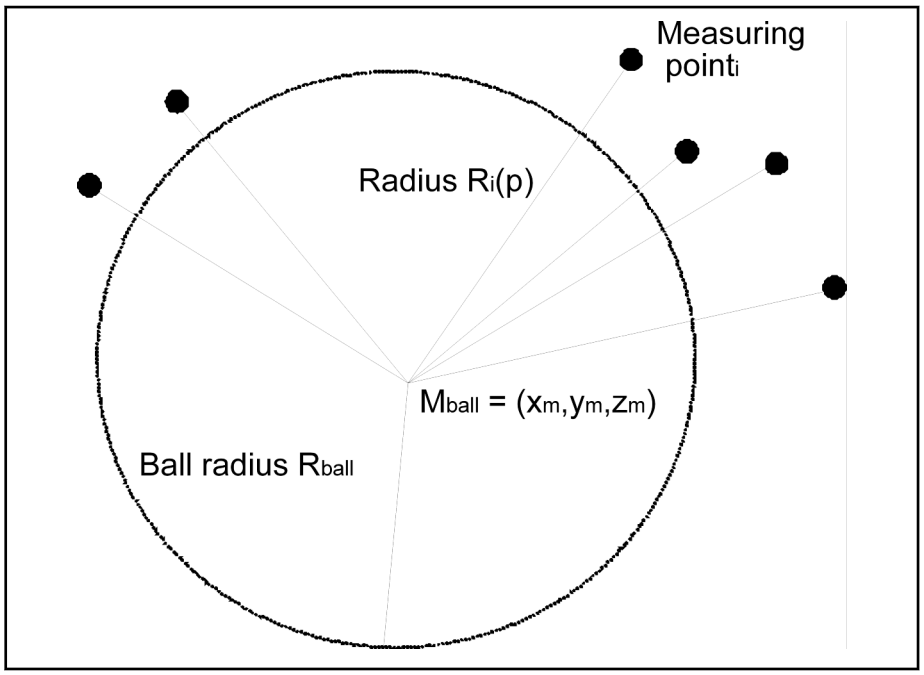


Fig. 11-109: Calibration: Indirect processing using a calibrated ball

Measuring positions on the calibrated ball are calculated. These are approached via touch probe. It has to be approached radially to eliminate the influence of the touch probe ball radius. It is then always effective in tool z-direction. When triggering the touch probe, the axis positions are stored $\vec{q}_i^{\,m}$.

This calculates the respective coordinate positions control-internally $\vec{p}_i^{\,m}$.

Relevant NC functions

The following NC functions permit parameter optimizing on the basis of a single measuring cycle for direct as well as indirect measuring methods:

ATCAL(..)	Calculating an optimum parameter block
ATGET(..)	Reading the currently valid parameters (applied to the system data from the machine parameter)
ATPUT(..)	Writing the optimized configuration parameter to the machine parameter 1030 00140
ATFWD(..)	Executing a forward transformation with specified parameters (Converting any axis point (in the ACS) with preset configuration parameters to a coordinate point (in the WCS).

Tab. 11-124: Relevant NC functions

NC commands of the function "Axis kinematic calibration"

A number of NC commands is provided which permit optimization of parameters based on the single measuring cycle both for the direct and the indirect

measuring methods. The structure and examples of calibration cycles are described in chapter "Calibration cycle" on page 420.

Calculating an optimum parameter block:

The optimization calculation of the configuration parameters is performed via the NC command ATCAL (**A**xis**T**ransformation**CAL**ibration)

```
ATCAL(<CalibFile>,<LenParamVar>,<LenParamMaskVar>
[,<ResultVar>][,n])
```

The parameters have the following meaning:

CalibFile

Name of the calibration file where the coordinate values $\vec{\rho}_i^{\,e}$, $\vec{\rho}_i^{\,m}$ and, if necessary, the axis values $\vec{q}_i^{\,e}$ or $\vec{q}_i^{\,m}$ are stored. The file name can be transferred with or without path specification. If a path is not specified, the subroutine search path set in the machine parameters applies. The structure of the calibration file is described in chapter "Calibration cycle" on page 420.

LenParamVar

Specifies the name of a CPL variable array into which the ATCAL logic stores the optimized length and angle parameters. The variable has to be of DOUBLE CPL type with a dimension of >= 16. The values have the same unit as the machine parameter 1030 00 40.

Examples:

(i)	As permanent variable (entry in wmhperm.dat)
	DEF DOUBLE @LENPARAM(16)
(ii)	As global variable in the calibration cycle
	DIM #LENPARAM!(16)
(iii)	As local variable in the calibration cycle
	DIM LENPARAM!(16)

LenParamMaskVar

Specifies the name of a CPL variable in a bit mask for the LENPARAM single parameters to be optimized is stored. The variable is of the CPL type INT.

Example:

```
LENPARMASK% = 1+2+4+16:          Optimize REM parameters 1, 2, 3, and 5
ATCAL("CalFile",LENPARAM!,LENPARMASK%)
```

ResultVar

This variable can be optionally specified. It is of the type DOUBLE with the dimension >= 4. After ATCAL has been called, this variable contains the values of the quality function and the maximum deviation between command and actual values. It is possible to use a local, global or permanent variable.

It applies:

ResultVar(1):

Value of the quality function $\sqrt{E(\vec{P}_0)}$ before optimization, i.e. with the non-optimized parameter block $\vec{\rho}_0$.

The unit is mm.

ResultVar(2):

Value of the quality function $\sqrt{E(\vec{P}_1)}$ after optimization, i.e. with the optimized parameter block $\vec{\rho}_1$.

The unit is mm.

ResultVar(3):

Value of the maximum deviation $\max_i |\overrightarrow{\rho_i}^m - \overrightarrow{\rho_i}^e|$ before optimization in mm.

ResultVar(4):

Value of the maximum deviation $\max_i |\overrightarrow{\rho_i}^m - \overrightarrow{\rho_i}^e|$ after optimization in mm.

n

The forth parameter can be used to limit the maximum number of iterations from the very beginning. At n = 1, this is a linear adjustment calculation. If "n" is not specified or if n = –1, the number of iterations is unlimited. In this case, the iteration is only completed if the deviation between two successive parameter blocks is practically equal to zero. An NC-internal barrier is used.

Example:

```
01 DIM LENPARAM!(16)
01 DIM RESULTS!(4)
01 LENPARMASK% = 2+4+32
N1 ATCAL("CalibData.txt",LENPARAM!,LENPARMASK%,
RESULTS!,-1)
```

Reading the current parameters from the NC:

The currently valid configuration parameters of the axis transformation are read from the machine parameters with ATGET.

```
ATGET(<McdParamVar>[,<AxTrafoNo>] [,<McdParamNo>])
```

McdParamVar

Cf. Section "LenParamVar". Specifies the name of a CPL variable array which - after been called - includes the currently valid values of the machine parameter "McdParamNo" of the axis transformation "AxTrafoNo". The variable has to be of the DOUBLE CPL type and has to have a dimension of >= 8 (McdParamNo = 1030 00130) or >= 16 (McdParamNo = 1030 00140). The values have the same unit as the corresponding machine parameter.

AxTrafoNo

Number of the axis transformation whose parameters are to be read. If AxTrafoNo is not specified, use is made of the active axis transformation. Here, the order of AT2 before AT1 is applicable.

The "ATGET" command causes a runtime error if AxTrafoNo is not present and no axis transformation (COORD(0,2) and COORD(0,1)) is active.

McdParamNo

Number of the machine parameter which is read. Permissible values for McdParamNo are 1030 00130 (axis positions of the reference position) and 1030 00140 (length and angle parameters). If McdParamNo is not programmed, 1030 00140 is used by default, i.e. the length and angle parameters are read.

Examples:

```
01 DIM LENPARAM!(16)                          ; The following values are read:
01 DIM ZEROPOS!(8)
N1 ATGET(LENPARAM!)                           ; MP 1030 00140 of the current axis transformation
N1 ATGET(LENPARAM!,3)                         ; MP 1030 00140 of axis transformation 3
N1 ATGET(LENPARAM!,2,103000140)               ; MP 1030 00140 of axis transformation 2
N1 ATGET(ZEROPOS!,,103000130)                 ; MP 1030 00130 of the current axis transformation
N1 ATGET(ZEROPOS!,3,103000130)                ; MP 1030 00130 of axis transformation 3
```

Writing the optimized parameters into the NC:

The optimized parameters are applied to the NC with ATPUT.

```
ATPUT(<McdParamVar>[,<AxTrafoNo>] [,<McdParamNo>])
```

McdParamVar

See "ATGET".

AxTrafoNo

Number of the axis transformation for which the parameters are to be transferred. The same applies as in "ATGET".

McdParamNo

See "ATGET".

The parameter block stored in the variables McdParamVar is applied to the appropriate machine parameters of axis transformation AxTrafoNo.

☞ The new parameter block becomes active only after a system reset and reprogramming of COORD(<AxTrafoNo>).

The ATPUT command requires a sufficiently high user level to write the machine parameter.

Executing a forward transformation with specified parameters:

With ATFWD (**A**xis**T**ransformation**F**or**W**ar**D**) any axis point $\vec{q}$ (ACS) with specified configuration parameters $\vec{\rho}$ can be converted to $\vec{\rho}$ (WCS) a coordinate point. That means $\vec{\rho} = \vec{f}_{Fwd}(\vec{q}, \vec{\rho})$.

The syntax is:

```
ATFWD(<CoordVar>,<AxisVar>[,<LenParamVar>])
```

CoordVar

Name of a CPL variable array of the type DOUBLE to which the WCS coordinate values $\vec{\rho}$ are returned. The dimension of the variables has to be at least as big as the number of the channel coordinates.

AxisVar

Name of a CPL variable array of the type DOUBLE type in which - at call - the ACS axis values $\vec{q}$ are stored. The dimension of the variables has to be at least as big as the number of the channel axes.

LenParamVar

Contains the specified configuration parameters $\vec{\rho}$ (AT2 before AT1). If LenParamVar is not programmed, the currently valid parameters are used.

If LenParamVar is programmed, the axis transformation has to be active. Otherwise, a runtime error is generated.

Executing a backward transformation with current machine parameters:

With ATFWD (**A**xis**T**ransformation**B**ack**W**ar**D**), any coordinate point $\vec{\rho}$ (WCS) with the currently valid configuration parameters $\vec{\rho}$ can be converted to an axis point $\vec{q}$ (ACS) a coordinate point. That means $\vec{\rho} = \vec{f}_{Bwd}(\vec{q}, \vec{\rho})$.

The syntax is:

`ATBWD(<AxisVar>,<CoordVar>)`

AxisVar

Name of a CPL variable array of the type DOUBLE to which the ACS axis values $\vec{q}$ are returned. The dimension of the variables has to be at least as big as the number of the channel axes.

CoordVar

Name of a CPL variable array of the type DOUBLE in which - at call - the WCS coordinate values $\vec{\rho}$ are stored. The dimension of the variables has to be at least as big as the number of the channel coordinates.

When ATBWD is called, the appropriate axis transformation has to be active.

Calibration file

The calibration file contains the data of one calibration cycle. It can be denominated by any name (e.g. "CalibDirect.cnc"). The file has to be stored in the RAM file system.

Structure of the calibration file:

Depending on the axis transformation used, the calibration file may contain the following lines:

#1 <FixCoordMask>	; e.g. "5" corresponds to the coordinates 1 and 3
#2 <DepCoordMask>	
#3 <AxisMask>;	;e.g. "21" corresponds to the axes 1, 3 and 5
#4 <MeasBallGeo>	;Geometry data of the calibrated ball
	;Calibrated ball radius, x-,y-,z-coordinates of the center point
#11 <FixCoordPos>	; Parameter-independent coordinate values of a
	;Measuring point
#12 <DepCoordPos>	; Parameter-dependent coordinate values of a
	;Measuring point
#22 <AxisPos>	; for #12, alternative axis position
#32 <EstAxisPos>	; Estimated axis position

Meaning of the lines:

#1 Defines which WCS-coordinates are stored in line #11.

The number of coordinates resulting from FixCoordMask has to correspond to the number of entries in #11.

#2 Defines which WCS-coordinates are stored in line #12.

The number of coordinates resulting from DepCoordMask have to correspond to the number of entries in #12.

#3 Defines which ACS-coordinates (channel axes 1 to 8) are stored in the lines #22 and #32.

#4 Contains the radius of the calibrated ball as first entry and the fix x-, y-, z-coordinate values as the entries 2, 3, 4 of the calibrated ball center point in the BCS.

The individual values are separated by spaces or tabs.

The indirect measuring method (variant 2) expects this entry. It is to be executed only once per calibration file.

#11

Contains the fixed coordinate values $\vec{\rho}_i^{\,m}$ At direct or $\vec{\rho}_i^{\,e}$ At indirect measuring method for a specified measuring point i.

The individual values are separated by spaces or tabs.

#12

Contains the parameter-dependent coordinate values $\vec{\rho}_i^{\,e}$ At direct or $\vec{\rho}_i^{\,m}$ At indirect measuring method for a specified measuring point i.

The individual values are separated by spaces or tabs.

#22

Contains axis values $\vec{q}_i^{\,e}$ or $\vec{q}_i^{\,m}$ which can be used instead of #12 if the axis transformation has an ambiguous backward branch (e.g. 5-axis transformation).

#12 und #22 **rule** each other **out**.

#32 Comprises estimated axis values in addition to #12. This line is only required when the axis transformation is ambiguous and the #22 values are unknown. #32 has to be contained in the calibration file, together with #12.

#22 and #32 **rule** each other **out**.

The comment character in the table is the semicolon.

The calibration file provides four definitions to structure the measuring points "i":

- For an unambiguous axis transformation, the measuring point is defined by:

  #11 <FixCoordPos>

  #12 <DepCoordPos> or #22 <AxisPos>
- For an ambiguous axis transformation, the measuring point is defined by:

  #11 <FixCoordPos>

  #22 <AxisPos>
- For an ambiguous axis transformation if <AxisPos> is **not** known exactly:

#11 <FixCoordPos>

#12 <DepCoordPos>

#32 <EstAxisPos>

- At indirect measuring method (variant 2) for an axis transformation, the measuring point is determined by:

  #22 <AxisPos>

Examples of calibration files:

Example 1:

Direct measuring method with 5-axis transformation.

Channel axes: X, Y, Z, U, V, W, B, C.

The 5-axis transformation 3032001 transforms between the axes X, Y, Z, B and C and the coordinates x, y, z, B and C. B and C are forwarded directly.

For the calibration, the points $\overrightarrow{\rho_i}^{e}$ (corresponding to x,y,z, b, c) are approached. The actual TCP position $\overrightarrow{\rho_i}^{m}$ (corresponds to x,y and z) is measured.

As the axis transformation is unambiguous, lines #11 and #12 can be used for the measuring points.

Accordingly, the calibration file comprises the following information:

#1	7					; $\overrightarrow{\rho_i}^{m}$ contains the coordinates ; 1, 2 and 3 (x, y, z) = 7 bit-coded
#2	199					; $\overrightarrow{\rho_i}^{e}$ -coordinates 1, 2, 3, 7, and 8 ; (X, Y, Z, B, C) = 199 bit-coded
#11	20.3	510.24	80.1			; $\overrightarrow{\rho_1}^{m}$
#12	20.4	510.14	80.1	20.0	181.5	; $\overrightarrow{\rho_1}^{e}$
#11	40.9	510.23	20.1			; $\overrightarrow{\rho_2}^{m}$
#12	50.0	510.13	19.0	40.0	165.0	; $\overrightarrow{\rho_2}^{e}$
...						
#11	100.32	23.2	220.1			; $\overrightarrow{\rho_k}^{m}$
#12	100.22	23.3	220.41	76.2	-36.1	; $\overrightarrow{\rho_k}^{e}$

Example 2:

Direct measuring method with Cardanic axis transformation (3232211).

Channel axes: X, Y, Z, B and C

The Cardanic axis transformation transforms between coordinates 1, 2, 3, 4, and 5 (coordinate names x, y, z, phi, theta) and axes X, Y, Z, B and C.

For the calibration, the points $\vec{\rho}_i^{\,e}$ (corresponds to x, y, z, ϑ, φ). The actual TCP-positions $\vec{\rho}_i^{\,m}$ are measured (corresponding to x,y,z).

The axis transformation is ambiguous. Thus, instead of the command coordinate points $\vec{\rho}_i^{\,e}$ (#12) shows the axis points $\vec{q}_i^{\,e}$ (#22) have to be used. The axis points are provided after the position $\vec{\rho}_i^{\,e}$ is approached with the CPL command SPOS.

Calibration file:

#1	7					; $\vec{\rho}_i^{\,m}$ contains the coordinates ; 1, 2 and 3 (x, y, z) = 7 bit-coded
#3	199					; $\vec{q}_i^{\,e}$ Channel axes 1, 2, 3, 7 and 8 ; (X, Y, Z, B, C) = 199 bit-coded
#11	20.3	510.24	80.1			; $\vec{\rho}_i^{\,m}$ Coordinates x, y, z
#22	100.54	28.3	470.22	0.0	181.5	; $\vec{q}_i^{\,e}$ Axis positions X,Y,Z,B,C
...						
..						
.						
#11	100.32	23.2	220.1			; $\vec{\rho}_k^{\,m}$
#22	200.23	49.3	522.81	76.2	-36.1	; $\vec{q}_i^{\,e}$

Example 3:

Indirect measuring method (variant 1) with Bipod axis transformation (3131001).

The axes YL, YR, Za, B, C are in the channel. Here, C is a spindle/c-axis.

The axis kinematics has the coordinates 1, 2, 3 and 6 (coordinate names Y, Y, Z, psi) and the axes YL, YR, Za, and C.

For the calibration, the parameter-dependent bore center points $\vec{\rho}_i^{\,m}$ (coordinates x and y) are measured via touch probe. The exact bore center points $\vec{\rho}_i^{\,e}$ are known.

Calibration file:

#1	3					; $\overrightarrow{\rho_i^e}$ Coordinates 1 and 2 (x, y) = 3 bit-coded
#2	3					; $\overrightarrow{\rho_i^m}$ Coordinates 1 and 2 (x, y) = 3 bit-coded
#11	20.3	510.2				; $\overrightarrow{\rho_i^e}$ exact position (x,y) of the first bore
#12	20.3	510.3				; $\overrightarrow{\rho_i^m}$ exact position (x,y) of the second bore
...						
..						
.						
#11	100.5	10.2				; $\overrightarrow{\rho_k^e}$
#12	100.6	10.1				; $\overrightarrow{\rho_k^m}$

Example 4:

Indirect measuring method (variant 2) with Cardanic axis transformation (3232211).

Channel axes: X, Y, Z, B and C

The Cardanic axis transformation transforms between the coordinates 1, 2, 3, 4, and 5 (coordinate names x, y, z, phi, theta) and the axes X, Y, Z, B and C.

The geometry data of the calibrated ball is known. Ball radius = 23 mm, ball center point coordinates (x,y,z) = (10.10, -21.85, 185.26).

For the calibration, the points $\overrightarrow{\rho_i^e}$ = (x, y, z, phi, theta) are approached on the calibrated ball. The axis positions are measured $\overrightarrow{q_i^m}$ (#22).

Calibration file:

#1	31					; $\overrightarrow{\rho_i^e}$ - Coordinates 1...5 (x, y, z, phi, theta)
#3	199					; $\overrightarrow{q_i^m}$ Channel axes 1, 2, 3, 7 and 8 ; (X, Y, Z, B, C) = 199 bit-coded
#4	23.0	10.10	-21.85	185.26		; Calibrated ball radius, x-, y-, z-coordinate of the calibrated ball center point.
#11	14.1	-6.0	213.6	0.0	10.0	; $\overrightarrow{\rho_i^e}$ = (x, y, z, phi, theta) exactly approached calibrated ball position

#22	-306.0	-176.8	-33.3	10.0	270.0	; $\vec{q}_i^{\,m}$ Measured axis positions X, Y ,Z, B, C
...						
..						
.						
#11	26.9	-11.5	205.7	330.0	50.0	; $\vec{\rho}_i^{\,e}$
#22	-302.6	-61.1	-91.5	50.0	288.0	; $\vec{q}_i^{\,m}$

Calibration cycle

The following processes describe the measures for execution of a calibration cycle with the direct and indirect measuring method.

Calibration cycle with the direct measuring method:

General configuration of a CPL calibration cycle in a direct measuring method:

1. Selection and activation of the axis transformation to be calibrated (e.g., COORD(3))
2. Open calibration file for writing
3. Write the coordinates and axes used to the file
   (#1, #2, #3)
4. Measuring points (approaching command coordinate values $\vec{\rho}_i^{\,e}$) and measuring coordinate positions $\vec{\rho}_i^{\,m}$.
   $\vec{\rho}_i^{\,m}$ to be entered in line #11 and $\vec{\rho}_i^{\,e}$ in line #12.
   If the transformation is ambiguous, use entry #22 instead of #12 $\vec{q}_i^{\,e}$. The approached axis position can be queried with the CPL command SPOS.
5. Close the measuring file.
6. Determine the current configuration parameter with (ATGET(LENPAR!)).
7. Store individual parameters to be optimized in LENPARMASK% as bit mask.
8. Start the calibration calculation with (ATCAL("CalibData.txt",LENPAR!,LENPARMASK%))
9. The calculated data is returned in the variable LENPAR!(1,..,16).
10. If applicable, perform a plausibility check of the parameters. Here, the parameters can be limited if necessary.
11. Writing new data (N.. ATPUT(LENPAR!) to the machine parameter 103000140.
12. End of calibration cycle
13. Execute system reset

After that, calibration is completed. New data is located in MP 103000140 of the axis transformation used.

Example:

Calibration cycle: Cardanic 5-axis transformation 3232211.

The axis configuration in the channel is {Xa, Ya, Za, U, V, W, B ,C}.

Due to the transformation, the axes {Xa, Ya, Za, B ,C} are transformed into the coordinates {x,y, z, phi, theta}.

The coordinate positions x, y, and z are measured at each measuring point.

For the parameters to be optimized, a permanent variable @LENPAR!(16) is created.

The parameters 1, 4, 5, 6, and 7 are optimized.

The cycle generates a calibration file "Calib.txt" configured as shown in the example 2 (see chapter "Calibration file" on page 415).

Calibration cycle program **CalibDirect.cnc** for calibration:

```
000 DIM AXPOS!(5)
N10 X0Y0Z0 B170 C-180 F10000 G1
;
; Open the calibration file and enter the lines #1,#3
010 OPENW(1,"CalibData.txt",512)
010 REWRITE(1)
010 PRN#(1,"#1 7") : REM coordinates 1,2,3
010 PRN#(1,"#3 199") : REM channel axes 1,2,3,7,8
N20 Coord(2)
020 FOR THETA=80.0 STEP -35.0 TO 45.0
020 FOR PHI=0.0 STEP 60.0 TO 300.0
; Approach measuring point
N30 x10 y10 z0 phi[PHI] theta[THETA]
N40 WAIT
; Read axis positions of the approached measuring point
040 AXPOS!(1) = SPOS(1)
040 AXPOS!(2) = SPOS(2)
040 AXPOS!(3) = SPOS(3)
040 AXPOS!(4) = SPOS(7)
040 AXPOS!(5) = SPOS(8)
;
; Measure TCP-position (direct measuring method)
; MeasurePoint.cnc returns the position in #MSDPOS!(1,2,3)
040 CALL MeasurePoint.cnc
;
; Write coordinate and axis positions to CalibData.txt
050 PRN#(1,"#11 ";)
050 PRN#(1,#MSDPOS!(1),#MSDPOS!(2),#MSDPOS!(3))
050 PRN#(1,"#22 ";)
```

```
050 PRN#(1,AXPOS!(1),AXPOS!(2),AXPOS!(3),AXPOS!(4),AXPOS!(5))
050 NEXT PHI
050 NEXT THETA
050 CLOSE(1)
;
;Perform a calibration calculation
060 LENPARMASK% = 1+8+16+32+64 : REM parameters 1,4,5,6,7
N60 ATGET(@LENPAR!) ;Read current parameters
N70 ATCAL("CalibData.txt",@LENPAR!,LENPARMASK%) ;Optimize parameters
N80 ATPUT(@LENPAR!) ;Write optimized parameters
M30
```

After completion of the program, a **system reset** has to be executed.

Calibration cycle with the indirect measuring method (variant 1):

Structure of the calibration cycle

At indirect measuring method, the calibration cycle is preceded by its own measuring cycle.

Such measuring cycle can for example be based on sampling bores on a swage block by a touch probe:

The exact positions $\vec{a}_i^{\,e}$ of the bores are known.

For each bore, three sample points are required to determine the position $\vec{a}_i^{\,m}$.

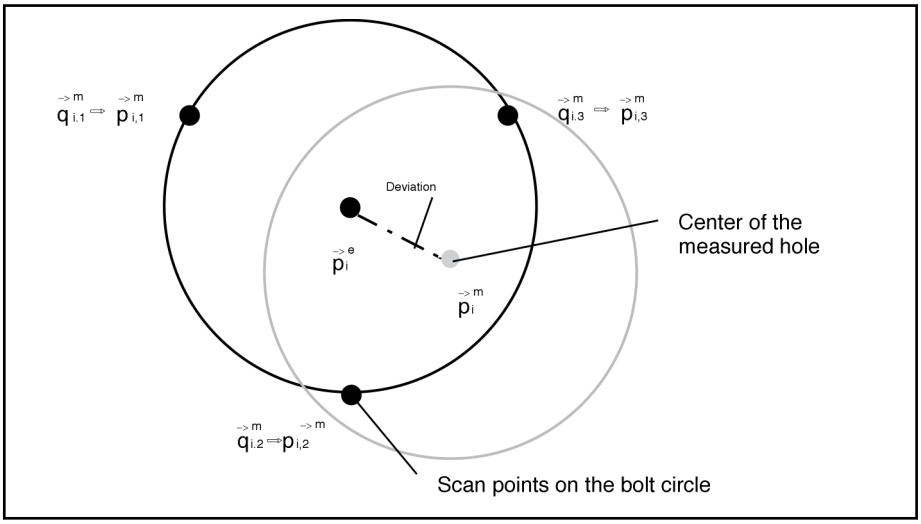


Fig. 11-110: Sample points at the hole circle

The results of the measurement are stored in a measuring file with the following structure (example):

Bore no.	Exact pos.	Axis point 1	Axis point 2	Axis point 3
1	$\vec{P}^{\,m}_{1}$	$\vec{q}^{\,m}_{1.1}$	$\vec{q}^{\,m}_{1.2}$	$\vec{q}^{\,m}_{1.3}$

2	$\vec{P}^{m}_{2}$	$\vec{q}^{m}_{2.1}$	$\vec{q}^{m}_{2.2}$	$\vec{q}^{m}_{2.3}$
...				
k	$\vec{P}^{m}_{k}$	$\vec{q}^{m}_{k.1}$	$\vec{q}^{m}_{k.2}$	$\vec{q}^{m}_{k.3}$

with

$\vec{\rho}^{e}$:	Exact center of the bore.
$\vec{q}^{m}_{i,j}$:	Axis points determined via touch probe.
$\vec{P}^{m}_{i,j}$:	Off $\vec{q}^{m}_{i,j}$ calculated values.
$\vec{P}^{m}_{i}$:	Off $\vec{P}^{m}_{i,j}$ calculated center point of the grey circle.

The measuring cycle has to be executed only once, as all values of the measuring file are independent of the configuration parameters of the axis transformation used.

The next following calibration cycle uses the measuring file as basis for the parameter optimization.

General structure of a CPL calibration cycle in an indirect measuring method:

1. Selection and activation of the axis transformation to be calibrated (e.g., COORD(3))
2. Calculate the current configuration parameter with (ATGET(LENPAR!))
3. Open measuring file for reading
4. Open calibration file for writing
5. Write the coordinates and axes used to the calibration file (#1, #2, #3)
6. Exact position $\vec{\rho}^{e}$ to be transferred from the measuring file to the calibration file (line #11).
7. The axis sample points $\vec{q}^{m}_{i,1}$, $\vec{q}^{m}_{i,2}$ and $\vec{q}^{m}_{i,3}$ in the coordinate points $\vec{P}^{m}_{i,1}$, $\vec{P}^{m}_{i,2}$ and $\vec{P}^{m}_{i,3}$ are to be converted using the NC command ATFWD.
8. Bore center $\vec{P}^{m}_{i}$ from the coordinate points $\vec{P}^{m}_{i,j}$ are to be calculated.
9. Write to the line #12 of the calibration file.

   If the axis transformation is ambiguous, it is additionally possible to write one of the axis sample points to line #32.
10. Continue at point 6 until the last line of the measuring file is reached.
11. Close the measuring and calibration file.
12. Store individual parameters to be optimized in LENPARMASK% as bit mask.

13. Starting of calibration calculation
    N...ATCAL("CalibData.txt", LENPAR!,LENPARMASK%,RESULTS!)
    The optimized parameters are returned in the variable LENPAR!(1,..,16)
14. At minor deviation of RESULTS!(1) from RESULTS!(2), the cycle is completed.
    Otherwise, continue with point 3.
15. If necessary, perform a plausibility check of the parameters.
    Here, the parameters can be limited if necessary.
16. Writing new data (N.. ATPUT(LENPAR!) in the machine parameters 103000140.
17. End of calibration cycle
18. Perform a system reset.

After that, calibration is completed. The new data is located in MP 103000140 of the axis transformation used.

Example:

Calibration cycle for Bipod transformation of the type 3131001.

The axis configuration in the channel is {YL, YR, Za, B, C}.

The transformation transforms between the axes {YL, YR, Za, C} and the co-ordinates {X,Y, Z, psi}. The coordinate positions (X,Y) are measured at each measuring point. For the parameters to be optimized, a permanent variable @LENPAR!(16) has to be created. The parameters 1 to 12 are optimized. The cycle generates a calibration file "Calib.txt" configured as shown in the example 3 (see "Examples of calibration files:" on page 417). It is assumed that a measuring cycle ran before which generated the measuring file "Holes.txt" with the structure explained in the section above.

Calibration cycle program **CalibIndirect.cnc** for calibration:

```
000 DIM AXPOS!(5) : DIM COPOS!(5)
000 DIM AP!(2,3) : DIM CP!(2,3)
000 DIM ECP!(2) : DIM MCP!(2)
000 DIM RESULTS!(4)

N10 COORD(1)
N20 ATGET(@LENPAR!)

020 REPEAT
;
; Open measuring file for reading
020 OPENR(2,"Holes.txt")
;
; Open the calibration file for writing and enter the lines #1,#3
020 OPENW(1,"CalibData.txt",512)
020 REWRITE(1)
020 PRN#(1,"#1 3") : REM coordinates 1,2
020 PRN#(1,"#2 3") : REM coordinates 1,2

020 WHILE NOT (EOF(2)) DO
```

```
; Read in the exact hole position from Holes.txt into ECP!(2)
; Read in three axis sample points from Holes.txt into AP!(2.3)
020 INP#(2,.....)
; Convert the axis sample points AP!(2,3) into coordinate points CP!(2,3)
020 FOR J% = 1 TO 3
020 AXPOS!(1) = AP!(1,J%) : AXPOS!(2) = AP!(2,J%)
020 FOR I%=3 TO 5 : AXPOS!(I%) = 0.0 : NEXT I%
N30 ATFWD(COPOS!,AXPOS!,@LENPAR!)
030 CP!(1,J%) = COPOS!(1) : CP!(2,J%) = COPOS!(2)
030 NEXT J%
; From CP!(2,3), calculate hole center point MCP!(2)
<Calculation algorithm returns MCP!>
; Write coordinate positions ECP and MCP to CalibData.txt
030 PRN#(1,"#11 ";) : PRN#(1,ECP!(1),ECP!(2))
030 PRN#(1,"#12 ";) : PRN#(1,MCP!(1),MCP!(2))
030 END
030 CLOSE(1) : CLOSE(2)

; Perform a calibration calculation
040 LENPARMASK% = 1+2+4+8+16+32+64+128+256+512+1024+2048 : REM P1 to P12
N40 ATCAL("CalibData.txt",@LENPAR!,RESULTS!)
; Check quality improvement
040 DIFF! = ABS(RESULTS!(1) - RESULTS!(2))
040 UNTIL (DIFF! < 0.1)
N50 ATPUT(@LENPAR!)
M30
```

After completion of the program, a **system reset** has to be executed.

11.28.3 Handling instruction

Applying

- All transformation-relevant axes have to be detected via touch probe. That means that all transformation-relevant axes are connected to the touch probe.
- For the machine to be calibrated, ensure that the reference points of all axes and the reference position in the selected axis transformation have been set correctly.
- The axis transformation relevant for calibration has to be activated
- If there are two axis transformations, i.e. one at point of action AT1 and one at AT2, **only** activate axis transformation at AT2.
- At the direct measuring method, different coordinate points have to be specified in the relevant machining space for a workpiece which require exact measuring with regard to the machine coordinates.
- At indirect measuring method (variant 1), for example a swage block has to be adjusted on the workpiece fixture to establish a relation to the WCS.

When the direct measuring method is used, program a calibration cycle program.

When the indirect measuring method (variant 1) is used, generate - in addition to the calibration cycle program - a measuring cycle program to record the measuring points e.g. from sampling the bores on a swage block.

The programs should be written so that they ensure that the measuring values are determined automatically, the resulting deviation is calculated and that the optimization of the length and angle parameters can be automatically applied to the machine parameters.

Activating Start the calibration cycle program and - at indirect measuring - the measuring cycle program. After completion of the programs, confirm all values by a system reset.

Deactivating Irrelevant.

12 Technologies

12.1 Tapping

12.1.1 Description

General information

Tapping can be executed in two different ways:

- With compensation chuck
- Without compensating chuck

Only tapping without compensating chuck is described here.

Tapping without compensating chuck is only possible with Sercos spindles.

Function

Function 1: Tapping without compensation chuck

Higher quality demands in the accuracy of a thread require an exact correlation between drill axis and spindle.

To create a thread without compensation chuck, the NC synchronizes the linear feed motion of the drill axis with the rotation of the spindle. **No displacement** results, since the feed position of the drill axis and the spindle rotation interpolate with each other; i.e. drill axis and spindle accelerate/decelerate **at the same time** in the same ratio.

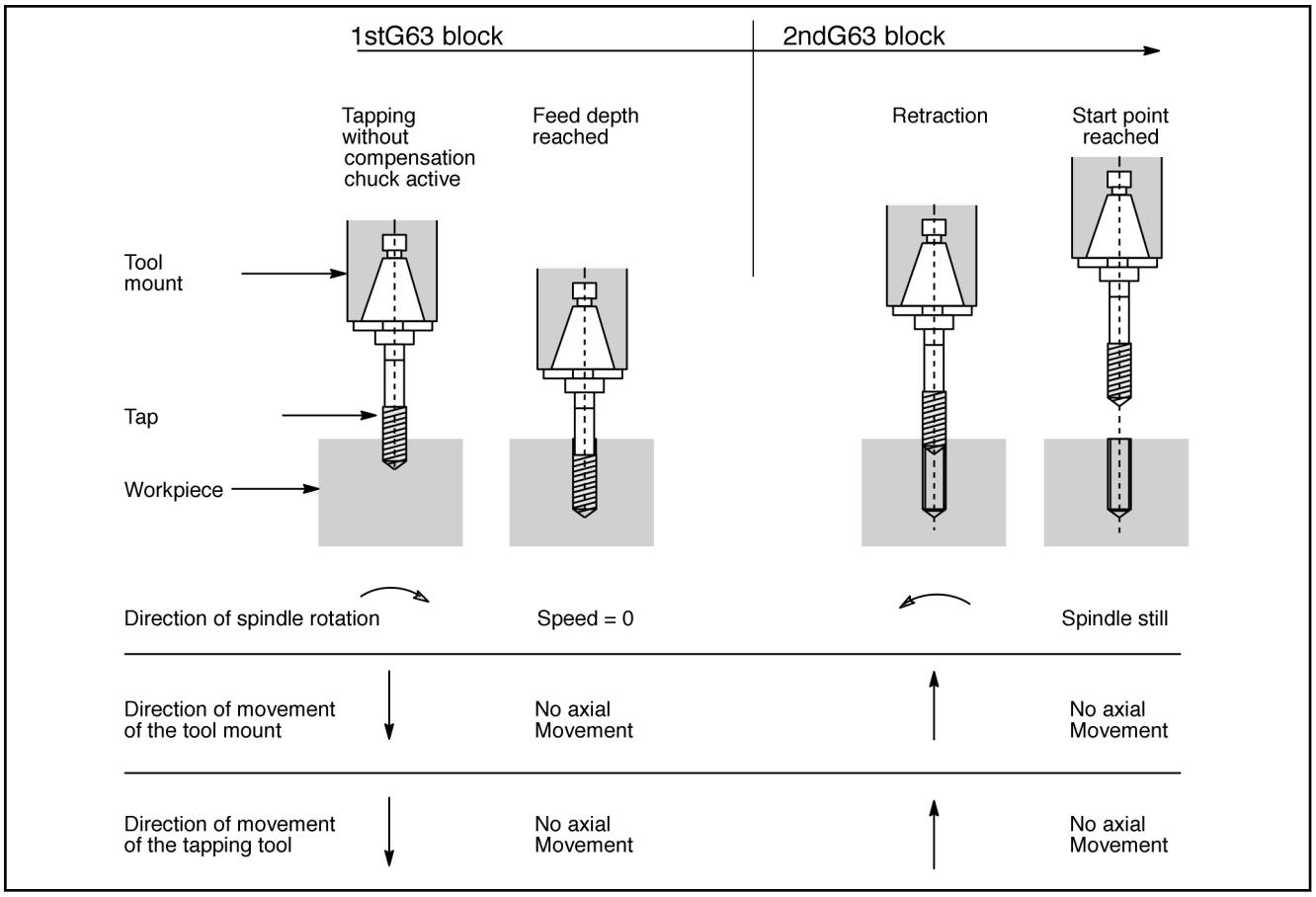


Fig. 12-1: Tapping without compensating chuck

"Tapping without compensating chuck " always consists of **two** successive blocks

- G63 for "drilling" and
- G63 or G63.2 for the "retract" from the drilling.

The thread pitch is specified as follows:

- As the ratio of the path feed of the drill axis to the spindle speed (F/S)..

  Programming:

  `G63(M...,S...) F..`

  - or -

- or by the direct specification of the thread pitch (H).

  Programming:

  `G63(M..., H...)`

Function 2: Retraction from tapping (without compensating chuck)

During tapping, the process can be interrupted. This can be caused as follows:

Cause of the interruption	Tapping data available	Continuation of tapping	Programming
Feed potentiometer = 0	Yes	Yes: Set feed potentiometer >0	-
Reset triggered	Yes	No: Automatic retraction of the screw tap	TappRet1 and TappRet2 with parameter F
Voltage failure (requires NC startup)	no	No: Manual retraction of the screw tap	TappRet1 and TappRet2 with the parameters S, F, M3/M4, drill axis and incremental retract path
Tool rupture	Yes	No: Automatic retraction of the screw tap after reset	TappRet1 and TappRet2 with F

Tab. 12-1: Causes of the interruption

The (automatic and manual) retraction is always initiated by the function "TappRet1" (switching the spindle to C-axis operation). The retract motion is started with "TappRet2".

are either provided automatically by the control (after reset) or are to be entered manually (after voltage failure and subsequent control startup):

- Starting position of the first tapping block (with G63)
- Thread pitch (F/S)
- Direction of rotation
- Involved spindle

TappRet1 and TappRet2 can be programmed via manual data input or via cycle.

To simplify the application of the functionality "Retract from tapping", it is recommended to write one subroutine (cycle) each for automatic and manual retract. With the machine parameters 3090 00001 and 309000002, these subroutines can be assigned to any available G-code:

Exemplary cycle for an **automatic** retraction:

AutoTR[feed]

```
N010
0020 DIM FCODE$(4)
0030 IF P1=NUL THEN
N040 (MSG, ** P1 NO FEED PROGRAMMED **)
N050 M0
0060 GOTO N40
0070 ELSE
0080 FCODE$=NCF("G94")
0090 FVORSCH%=SD (5,1,2)
N100 TappRet1
N110 G94 TappRet2 F[P1]
N120 [FCODE$]
0130 IF FCODE$="G94" THEN
N140 F[FFEED%]
0150 ENDIF
0160 ENDIF
M30
```

Exemplary cycle for an **manual** retract:

ManTR[drill axis number, path, master pitch, feed]

```
0010 DIM GABS_INK$(4)
0010 DIM GVORSCH$(4)
0020 IF P1=NUL THEN
N030 (MSG, ** P1 AXIS NOT PROGRAMMED **)
N040 M0
0050 GOTO N30
0060 ENDIF
0070 IF P2=NUL THEN
N080 (MSG, ** P2 PATH NOT PROGRAMMED **)
N090 M0
0100 GOTO N80
0110 ENDIF
0120 IF P3=NUL THEN
N130 (MSG, ** P3 THREAD PITCH NOT PROGRAMMED **)
N140 M0
0150 GOTO N130
0160 ENDIF
0170 IF P4=NUL THEN
N180 (MSG, ** P4 FEED NOT PROGRAMMED **)
N190 M0
0200 GOTO N180
0210 ENDIF
0220 BAXIS%=ROUND(P1)
```

```
0230 IF P3>0 THEN
0240 MCODE1$="M4"
0250 ELSE
0260 MCODE1$="M3"
0270 ENDIF
0280 STEIG=ABS(P3)
0290 GABS_INK$=NCF("G90")
0300 GVORSCH$ =NCF("G94")
0310 FVORSCH% =SD (5,1,2)
N320 TappRet1
N330 G91 G94 TappRet2 [AXP(BAXIS%,P2)] H[PITCH] F[P4]
M[MCODE1%]
N340 [GABS_INK$] [GFEED$]
0350 IF GVORSCH$="G94" THEN
N360F[FFEED%]
0370 ENDIF
M30
```

If desired, the cycles can be assigned to G-functions using the following settings:

Parameter: 3090 00001

0 6301

1 6302

2 ...

Parameter: 3090 00002

0 AutoTR

1 ManTR

2 ...

Example:

G6301[1000]	Automatic retract at a feed of F1000 mm/min
G6302[3,-100,0.5,500]	Manual retract with thread pitch 0.5, Retract path -100 mm, F500, Drill axis

Restrictions

- Drilling and retract always have to be programmed with an identical thread pitch (F/S or H).
- If a "pause" is required between "drilling" and "retracting", e.g. dwell time or auxiliary function, the retract has to be programmed with G63.2.
- After an interruption, retract motion with active placements or workpiece position correction (G152.x, G154.x..G159.x or BCR) can only be performed **manually**.

Relevant NC functions

G63	Tapping without compensating chuck (effective block-by-block) with specification of drill axis, infeed depth, feed, direction of spindle rotation, speed, thread pitch and starting angle.
G63.2	Continuous block of tapping without compensating chuck.

Tab. 12-2: Relevant NC functions

G84	Tapping
G89	Tapping with chip breaking

Tab. 12-3: Relevant NC cycles

Relevant Sercos parameters

S-0-1050.1.6	Configuration list AT There, depending on the encoder used, the Sercos parameter S-0-0051 (Actual position value 1 motor encoder) or S-0-0053 (Actual position value 2, external encoder), S-0-0040 (Actual torque value) and S-0-0144 (Signal status word) have to be entered.
S-0-1050.0.6	Configuration list MDT There, the Sercos parameters S-0-0047 (position command value) and S-0-0036 (speed command value) must be entered.
S-0-0026	Configuration list signal status word S-0-0331 (spindle stop) has to be entered. It is recommended to enter the Sercos parameter S-0-0403 (Actual status position values) into S-0-0026.
S-0-0032	Primary mode Has to be set to "Velocity control" mode.
S-0-0033 to S-0-0034	Secondary modes 1 to 2. Has to be set to "position control..." mode.
S-0-0044, S-0-0076 and S-0-0160	Scaling type "Rotary scaling in the modulo format" has to be set as scaling type.
S-0-0057	Positioning window
S-0-0261	Positioning window rough
S-0-0103	Modulo value 360 degrees is to be set as Modulo value.
If a drive-controlled operation mode change of an IndraDrive is enabled (Bit 8 in S-0-0034):	
P-0-0152	Synchronizing completed P-0-0152 also has to be contained in S-0-0026 (configuration list signal status word).

P-0-0142	Synchronizing acceleration Acceleration value to reduce the position difference for a drive-controlled change of operation mode.
P-0-0143	Synchronizing velocity Maximum velocity to reduce the position difference for a drive-controlled change of operation mode.

Tab. 12-4: Sercos parameters

Relevant IF signals

At the channel interface

The "G63 active" signal can be configured at the channel input interface under the active NC functions and is then active during tapping.

At the spindle interface

iSp_TurnCmd	"Turning command" during "C-axis switching active"
iSp_CAxSwitch	"C-axis switching" is active
iSp_CAxAct	"C-axis is active" during tapping

Tab. 12-5: Relevant interface signals at the spindle interface

at the axis interface

iAx_TrvCmd	"Travel command" for the C-axis and drill axis during tapping
iAx_InPos	"Axis in position" is checked at the beginning and at the end of each drilling motion.

Tab. 12-6: Relevant interface signals at the axis interface

Relevant machine parameters (MP)

100100001	Type of drive function (drive moves a spindle which can be switched temporarily to C-axis mode as well.)
100300004	Axis motion type: C-axis/spindle is a rotary axis with module calculation
104000001	Selection of spindle type (1 = Sercos spindle)
105000001	Specifies the type for each drive.
104000009	Drive parameter blocks per G-range (only for IndraDrive)
104000012	Max. spindle speed of the gear stage in rpm
104000051	Position interface: 1. Acceleration in rad/s2

Tab. 12-7: Relevant machine parameters

12.1.2 Handling instruction: Applying tapping without compensating chuck

Applying:

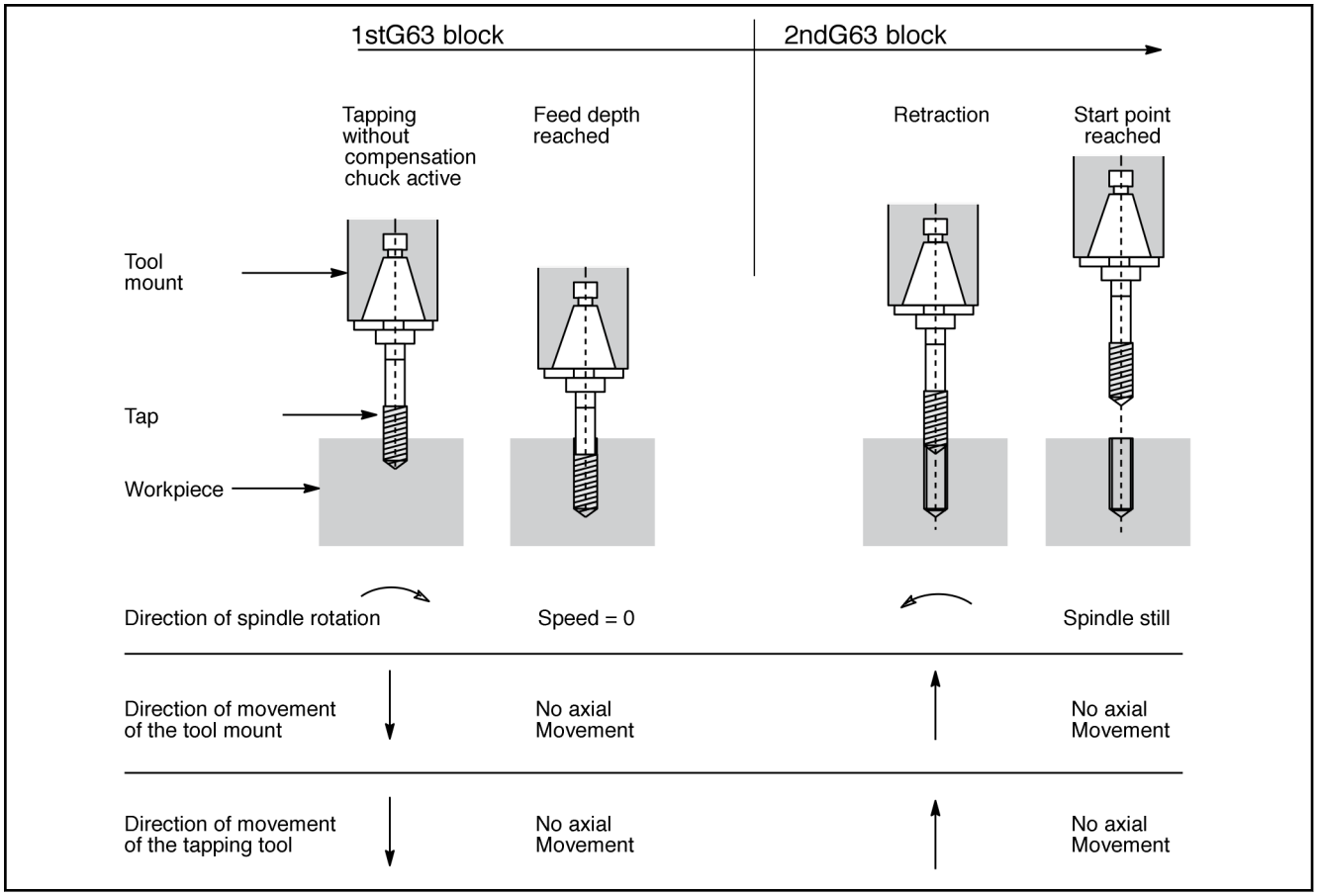


Fig. 12-2: Overview on tapping

IW Engineering/configuration: Creating spindle/C-axis

		Instruction
Instruction:		Create a c-axis

IW Engineering/configuration: Editing parameters

The spindle dynamics at tapping is determined by the following parameters:

- **MaxSpAccPosCtrl[1]** "Acceleration in the position controller range[1] of GearStep[1..4]" (1040 00051)
- **MaxSpSpeed** "Maximum speed of GearStep[1..4] of GearStep[1..4]" (1040 00012)
- **MaxSpJerkPosCtrl** "Maximum spindle jerk in position control mode of GearStep[1..4]" (1040 00055)

  Effective at active NC function **JerkControl**.

☞ If **MaxSpAccPosCtrl[1]** contains the value **0**, the following applies:

The spindle dynamics is determined by **MaxAxAcc** "Maximum axis acceleration" (1010 00001) and **MaxVel** "Maximum axis velocity" (1005 00002).

To further optimize the tapping, use an adjusted Sercos parameter block in the drive.

- **NofScsParSets** "Drive parameter blocks per gear level" (1040 00009[1])

- **PosCtrlSpSet** "Parameter set for G63 and position-controlled Spindle mode" (1040 00009 [2])

	Documentation	Instruction
Instruction:		Editing machine parameters
Documentation:	MTX Machine Parameters	Tapping

IW Engineering/IndraLogic: Linking status interface signals if necessary

- At the channel interface:
  - **iCh_ActFunc07** "Tapping active" shows whether tapping G63 is active
- At the spindle interface:
  - **iSp_TurnCmd** "Turning command" during "C-axis switching active"
  - **iSp_CAxSwitch** "C-axis switching" is active.
  - **iSp_CAxAct** "C-axis is active" during tapping.
- At the axis interface:

  The axis interface is important during tapping:
  - **iAx_TrvCmd** "Travel command" for the C-axis and drill axis during tapping.
  - **iAx_InPos**"Axis in position" is checked at the beginning and at the end of each drilling motion.

	Documentation	Instruction
Instruction:		Editing PLC Signals
Documentation:	MTX PLC Interface	Tapping

Activating: **IW Operation/NC programming: Using the Tapping Function**



For tapping, the following functions are available:

- G63 and G63.2 - Tapping without compensating chuck
- G84 – Cycle for tapping
- G89 - Cycle for tapping with chip breaking
- TappRet1/TappRet2 – Retract from tapping for tapping without compensation chuck

The cycles for tapping are described in the manual "Standard NC Cycles".

	Documentation	
Documentation:	MTX Programming Manual	Tapping

Disabling: After completion of the last G63 or G63.2 block, the spindle switches back from "C-axis mode" to "spindle mode".

This deactivated the tapping.

12.2 Thread cutting

12.2.1 Description

General information

There are multiple turning cycles to machine cut threads.

The following cycles are described in the "Standard NC Cycles" manual:

- **G165** Thread single cut
- **G166** Thread sequence
- **G167** Thread turning
- **G168** Face thread
- **G169** Tapered thread

These cycles internally use the thread cutting function "G33".

Function

The thread cutting function can cut several single and multiple turns of a thread for longitudinal, face and conical threads. The thread pitch can either be constant, decreasing or increasing.

All thread types can be used in a sequenced thread.

Thread cutting is a modal NC function. It belongs to the same NC function group as G0, G1, G2, G3, etc.

As it is the case with circular interpolation (G2, G3), the tapping function also depends on the active plane (G17 ... G20).

The feed rate of the cutting motion results from the current spindle speed and the programmed pitch portions (constant and variable). The feed potentiometer has no effect.

The general syntax is as follows:

G33 <EP> <FixedPitch> {<Var. Pitch>} {starting angle}	
with	
<EP>	Axis coordinates of main and secondary axis of the active plane. The active plane is defined by G17, G18, G19 or G20. Example: The active plane for G18 is usually defined by the axes Z (main axis) and X (secondary axis).
<fixed pitch>	Defines the path (in mm) traveled in the direction of the main or secondary axis per spindle revolution. This value is programmed by the interpolation parameter (I, J or K) valid in the corresponding active plane. For tapered threads, the thread pitch specified always has to refer to the main cutting direction. Example: For G18, the parameter K is assigned to the main axis, and I is assigned to the secondary axis. For a longitudinal thread (pitch in direction of main axis), the fixed thread pitch is programmed with the K-address.

<Var. Pitch>	Optional parameter with the address DF Defines the pitch increase/decrease per spindle revolution in mm. Programming: "DF<Value>" with <Value> in mm.
<Starting angle>	Optional parameter If the <Starting angle> is not programmed, 0 degree is assumed. The starting angle (offset) is required for multi-start threads. The interpolation parameter not assigned to the active plane is used as address. Example: The addresses I and K are assigned to the plane for G18. Therefore, the address of the starting angle is J.

Tab. 12-8: G33 syntax

Example of a longitudinal thread:

`G91 G18 G8 M3 S1000`	Activate relative dimension programming
	Activates the Z/X plane
`G0 X-10`	Infeed motion of the cutting tool (1)
`G33 Z-50 K2`	Thread cutting (2)
	End point: incremental by -50 mm in z-direction
	Fixed thread pitch: 2 mm/rev.
	Interpolation parameter: here K
`G0 X10`	Retract cutting tool (3)

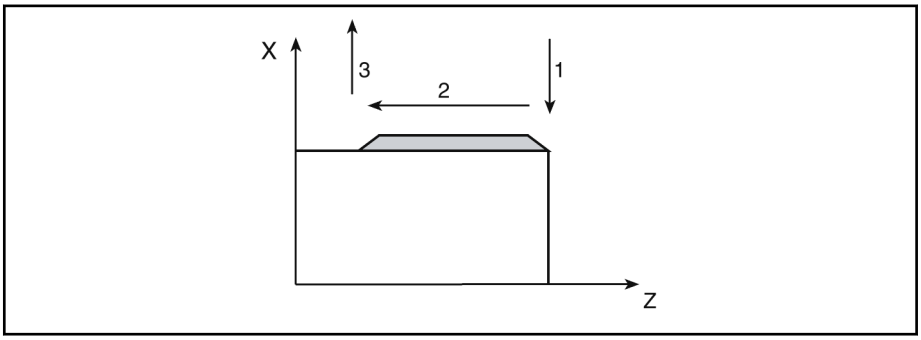


Fig. 12-3: Longitudinal thread

Dynamic behavior

At the beginning and at the end of a thread cutting manufacturing, the axes involved have to be accelerated or decelerated.

☞ Thus, always specify a sufficiently long entry path (to accelerate the cutting axes) and leaving path (to stop).

There are two options to determine the dynamics of the cutting motion:

- Start/end of the cutting motion with automatically calculated dynamics:

  Velocity jump, approach and deceleration acceleration are calculated by the NC.
- Start/end of the cutting motion with individually set dynamics:

The dynamic behavior with regard to the velocity jump, approach and deceleration can be set individually:

- Statically via machine parameter (see THRD/Ch[k]/Vel, THRD/Ch[k]/Acc and THRD/Ch[k]/Dec).
- Dynamically via part program with "ThreadSet(DYN)" (refer to the Programming Manual "Additional Functions for Thread Cutting ThreadSet, TST ").

The velocity jump is automatically limited by the NC.

The control uses the programmed starting angle to compute a starting angle offset taking into account the slope of the acceleration ramp. Thus, it can be ensured that the same thread is always cut irrespective of the acceleration.

The cutting axis (axes) is (are) decoupled from the spindle at the end of the thread and decelerated according to the active deceleration.

Fast retract

The "fast retract" can be used together with G33.

If retract data is configured and activated

- In the part program with "ThreadSet(RD(,),RON1))" (refer to the Programming Manual "Additional Functions for Thread Cutting ThreadSet, TST")
- In the machine parameters (see "THRD/Ch[k]/Retr/").

A positive edge at the channel interface signal qCh_Retract "fast retract" initiates the retract with the following sequence:

1. The cutting motion is superimposed by a motion oriented vertically to the main cutting direction.
2. If more than 90% of the retract distance are traveled, the cutting axis (axes) is (are) decoupled and decelerated.



- Retract motions are always carried out perpendicularly to the main cutting direction of the secondary cutting axis.
- A configured retract motion is automatically triggered by the NC when the events "Channel, system and spindle reset) occur!



If the retraction was initiated, this condition can only be canceled by a "control reset" or "leaving the contour".

An example for a retraction from a longitudinal thread with external machining:

`G18`	Activates Z/X plane (G18).
`TST(RON1)`	Activates fast retract (RON1).
`TST(RD(0,5))`	Retract motion (RD ...) by +5 mm in the secondary cutting direction (X in this case).
:	
:	
`G33 Z20 K1`	Thread cutting (G33)

Relevant NC functions

G33	Thread cutting with end point specification, pitch and starting angle.
MainSp Short form: MSP	Main spindle switching Thread cutting always uses an active main spindle
ThreadSet Short form: TST	Configuring a retract motion Configuring retract motion. Influencing dynamics at the beginning and at the end of the cutting motion.

Tab. 12-9: Relevant NC functions

G165	Thread single cut
G166	Thread sequence
G167	Thread turning
G168	Face thread
G169	Tapered thread

Tab. 12-10: Relevant NC cycles

Relevant Sercos parameters

S-0-1050.1.6	Configuration list AT The actual spindle position has to be transferred.
S-0-0043 and S-0-0055	Velocity polarity and position polarity of the spindle have to be equal.

Tab. 12-11: Sercos parameters

Relevant IF signals

qCh_CtrlReset	Reset If a fast retract is configured, it is triggered and the part program is subsequently completed.
qCh_Retract	Fast retract If a fast retract is configured, it can be triggered via this signal.
qCh_Override100	This signal is not evaluated for an active G33.
qCh_Override_xx	These signals are not evaluated for an active G33.
iCh_ActFunc06	The default assignment of this signal is "TST" (acc. to PLC/NcFuncBitIf/NcFunc[1..24]) ThreadSet(TCI1) can set this signal.
iCh_ActFunc08	The default assignment of this signal is "G33" (acc. to PLC/NcFuncBitIf/NcFunc[1..24]) This signal is set for an active G33.

Tab. 12-12: Relevant interface signals at the channel interface

qSp_Reset	Spindle reset If a fast retract is configured, it is triggered.
qSp_Override100	Default: This signal is not evaluated for an active G33. Option: By setting "/SysSpCmdData[ ]/ThreadOverride = 1", the signal analysis can be enabled.
qSp_Override_xx	Default: These signals are not evaluated for an active G33. Option: By setting "/SysSpCmdData[ ]/ThreadOverride = 1", the signal analysis can be enabled.

Tab. 12-13: Relevant interface signals at the spindle interface

Relevant machine parameters (MP)

1040 00001	Spindle type (SP/SpFunc/Sp[1..32]/Base/SpType) The thread cutting function (G33) requires a spindle with position feedback (SpType 1=Sercos, 4=external with encoder or 5=analog with encoder).
7020 00010	Main spindle (CHAN/Ch[k]/Couple/MainSp) The main spindle can be modified in the part program using "MainSp( )".
Of special importance are:	
7050 00610	Velocity jump (THRD/Ch[k]/Vel/VelJumpThrd) Specifies the maximum velocity jump at the beginning and at the end of the cutting motion.
7050 00615	Path acceleration (THRD/Ch[k]/Acc/AccThrd) Indicates the path acceleration at the beginning of the cutting motion. **Usually, the default value (0.000 m/s2) returns the best result.**
7050 00620	Path deceleration (THRD/Ch[k]/Dec/DecThrd) Indicates the path acceleration at the end of the cutting motion. **Usually, the default value (0.000 m/s2) returns the best result.**
7050 00640	Enable fast retract (THRD/Ch[k]/Retr/EnablRetrThrd) At runtime, this parameter can be overridden by "Thread-Set(RON)".

7050 00645	Axis number of the main axis (THRD/Ch[k]/Retr/DrIndRetrAx) and Axis number of the secondary axis (THRD/Ch[k]/Retr/DrIndSeconAx) Specifies the retract axes. These axes have to represent one plane in the channel. At runtime, this parameter and MP 705000650 can be overridden by “ThreadSet(RD(,))”.
7050 00650	Retract vector of the main axis (THRD/Ch[k]/Retr/RetrVectRetrAx) and Retract vector of the secondary axis (THRD/Ch[k]/Retr/RetrVectSeconAx) Specifies the retract distance including the sign for the axes specified in 705000645. At runtime, this parameter and MP 705000645 can be overridden by “ThreadSet(RD(,))”.

Tab. 12-14: Relevant machine parameters

12.2.2 Relevant system data

/SysSpCmdData[<Spindle>]/ThreadOverride	1: Spindle override is active during thread cutting

Tab. 12-15: Relevant system data



Please not that in case of an enabled spindle override, a fast change of the spindle override can sporadically result in a damage of the thread during machining.

12.2.3 Applying

No special steps are required for the thread cutting function.

The default values of the machine parameter "thread cutting (THRD)" (see THRD/Ch[k]/) do not have to be changed.

Prerequisite:

The thread cutting function (G33) requires a spindle with position feedback (SpType 1=Sercos, 4=external with encoder or 5=analog with encoder).

12.2.4 Activating

For thread cutting, the following functions are available:

- G33 "Thread cutting"
- G165 "Thread single cut"
- G166 "Thread sequence"
- G167 "Thread turning"
- G168 "Face thread"
- G169 "Tapered thread"

The thread cutting cycles (G165 ... G169) are described in the manual "Standard NC Cycles".

The normal procedure is:

1. Start main spindle

2. Approach starting position
3. Perform thread cutting G33 or a suitable drilling cycle.

If the "fast retract" function is to be used in case of emergency, configure and enable retract data using the "ThreadSet" function.

☞ The retract motion has to move away from the workpiece (note the sign!).

12.2.5 Deactivating

When selecting any other geometry function (G0, G1, G2, G3, G6), the thread cutting function G33 is deselected.

12.3 Circular rounding of corners

12.3.1 Description

Function

By the "Rounding (RND)" and "RoundEps (RNE)" functions, the control can insert transition arcs tangentially between linear or circular helical blocks in the active plane.

Thus, the programmed contour at block transition is slightly modified, but it is possible (with a sufficiently great radius) to reach continuous velocity and acceleration curves on the path.

The conditions under which the control inserts transition arcs can be set using the machine parameter "MinLen" and "MaxAng".

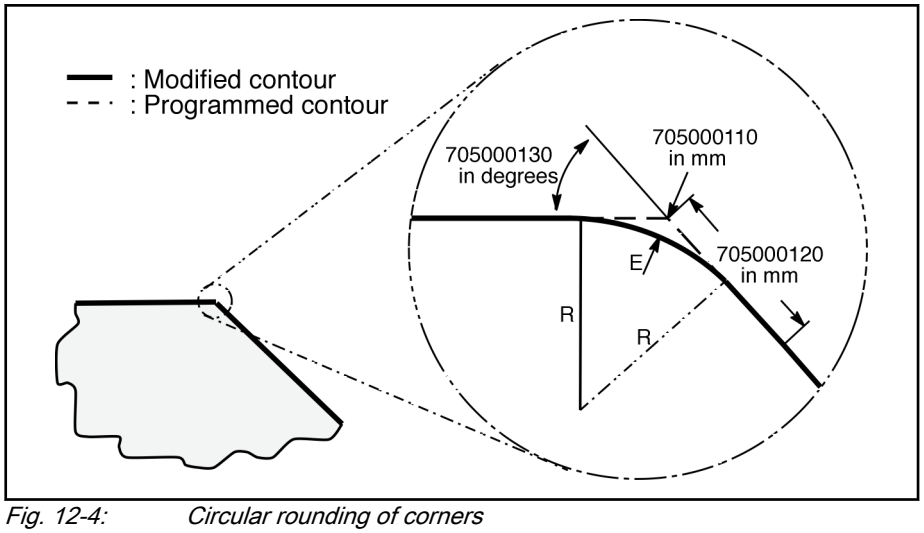


Fig. 12-4: Circular rounding of corners

Restrictions

- RoundEps (RNE) can only be used together with two linear blocks, while rounding (RND) additionally acts on circular blocks and helical blocks.
- When corner rounding is active, punching is not possible.
- Both neighboring blocks as well as RNE have to be located in the selected main plane to perform a rounding. With helical blocks, only the components of the circular plane are considered for rounding.

Relevant NC functions

Rounding (RND)	Activate "rounding corners" between linear or circular blocks. When rounding/RND is called, the desired radius of the transition arc is programmed in brackets after the function (e.g. Rounding(5)). It remains effective until overwritten by reprogramming.
RoundEps/RNE	Activate "rounding corners" between two linear blocks. The maximum permissible deviation between the modified and the programmed contour is programmed in round brackets after the function (e.g. RoundEps(2)). If the values preset from the machine parameters MaxDev are to apply, the keyword "DEF" within the round brackets is to be programmed.
Rounding(0) or RND(0) or RoundEps(0) or RNE(0)	
	Deactivate "rounding corners". Both functions are identical and deselect all variations of the chamfer and rounding programming.

Tab. 12-16: Relevant NC functions

☞ The NC functions "Rounding/RND" and "RoundEps/RNE" for a modal group and deselect each other together with the "ChLength/CHL" and "ChSection/CHS" functions.

Relevant machine parameters

NCP/CornRound/CH[x]/MaxDev (705000110)	
	Maximum allowed deviation from the programmed contour. On this basis, the control automatically computes the suitable radius of the transition arc.
NCP/CornRound/CH[x]/MinLen (705000120)	
	Minimum block length to start the corner rounding. If one of the neighboring traversing blocks is smaller, no transition arc is generated.
NCP/CornRound/CH[x]/MaxAng (705000130)	
	Maximum angle up to which block transitions are considered quasi-continuous. No transition arc is generated with smaller angles.

Tab. 12-17: Relevant machine parameters

12.3.2 Handling instruction: Applying and Activating/Deactivating Rounding of Corners Specifying the Deviation

This function inserts tangential transition arcs between two linear blocks in the main plane. On the one hand, this causes a minor modification of the programmed contour at these edges, but on the other hand, continuous velocity patterns are achieved during interpolation.

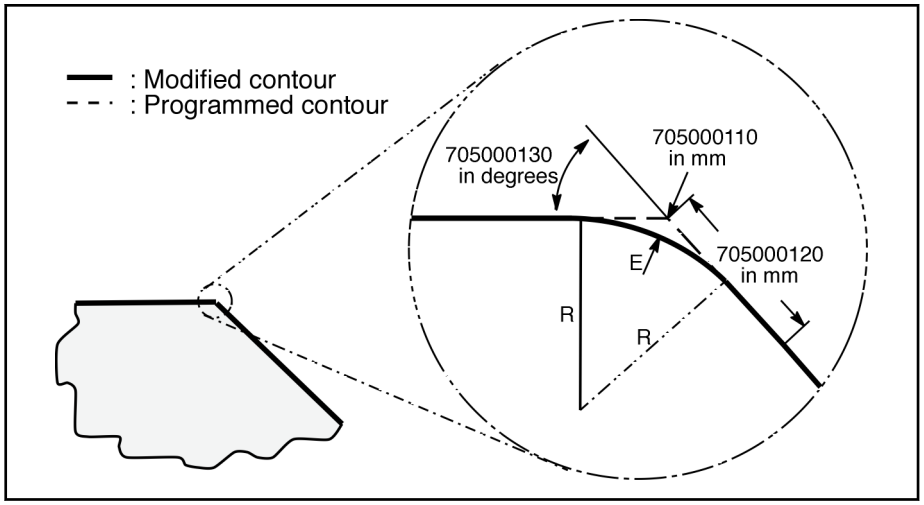


Fig. 12-5: Sketch on rounding of corners

Applying

IW Engineering/configuration: Editing parameters

- From the parameter **MaxDev** "Maximum permissible deviation from the contour" (7050 00110), the control automatically calculates the suitable radius of the transition arc.
- The parameter **MinLen** "Minimum block length at which rounding starts" (7050 00120) contains the minimum block length at which rounding starts. If one of the neighboring traversing blocks is smaller, no transition arc is generated.
- The parameter **MaxAng** "Angle up to which block transitions are considered cont." (7050 00130) contains the maximum angle up to which block transitions are considered quasi-continuous. No transition arc is generated with smaller angles.

	Documentation	Instruction
Instruction:		Editing machine parameters
Documentation:	MTX Machine Parameters	Rounding corners

IW Operation/machine: Triggering system reset

To apply the changed machine parameters, a system reset or a control start-up is required.

Activating

NOTICE **Workpiece or machine can be damaged!**

Do not reset the system during machining.

IW Operation/NC programming: Activating

- **RoundEps(<Deviation>)** or **RNE(<Deviation>)** "Rounding of corners specifying the deviation"

  <Deviation> Maximum permitted deviation (in mm) between the modified and the programmed contour. Number of decimals point are permitted. The control calculates a suitable tangential transition arc.

- or -

- **RoundEps(DEF)** Activate rounding of corners between two linear blocks with default deviation from the parameter **MaxDev** "Maximum permissible deviation from the contour" (7050 00110).

☞ The functions **ChLength(), ChSection(), RoundEps()** and **Rounding()** act modally and deselect each other.

Only contour transitions of blocks whose path is exclusively within the active working plane are taken into account.

	Documentation	
Documentation:	MTX Programming Manual	Rounding corners

Deactivating

IW Operation/NC programming: Deactivating

RoundEps(0) or **RNE(0)**

	Documentation	
Documentation:	MTX Programming Manual	Rounding corners

12.3.3 Handling instruction: Applying and Activating/Deactivating Rounding of Corners Specifying the Radius

This function inserts tangential transition arcs between two linear, circular or helical blocks in the main plane. On the one hand, this causes a minor modification of the programmed contour at these edges, but on the other hand, continuous velocity patterns are achieved during interpolation.

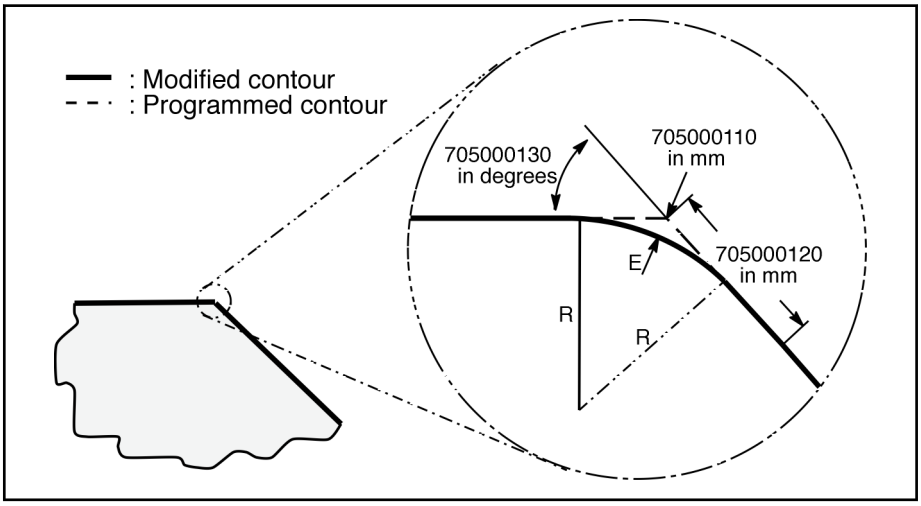


Fig. 12-6: Sketch on rounding of corners

Applying

IW Engineering/configuration: Editing parameters

- From the parameter **MaxDev** "Maximum permissible deviation from the contour" (7050 00110), the control automatically calculates the suitable radius of the transition arc.
- The parameter **MinLen** "Minimum block length at which rounding starts" (7050 00120) contains the minimum block length at which rounding starts. If one of the neighboring traversing blocks is smaller, no transition arc is generated.
- The parameter **MaxAng** "Angle up to which block transitions are considered cont." (7050 00130) contains the maximum angle up to which block transitions are considered quasi-continuous. No transition arc is generated with smaller angles.

	Documentation	Instruction
Instruction:		Editing machine parameters
Documentation:	MTX Machine Parameters	Rounding corners

IW Operation/machine: Triggering system reset

To apply the changed machine parameters, a system reset or a control start-up is required.

Activating

NOTICE **Workpiece or machine can be damaged!**

Do not reset the system during machining.

IW Operation/NC programming: Activating

Rounding(<Radius>) or **RND(<Radius>)** "Rounding of corners specifying the radius"

<Radius> Desired radius of the transition arc. Decimal digits are allowed.

☞ The functions **ChLength(), ChSection(), RoundEps()** and **Rounding()** act modally and deselect each other.

Only the components of the active working plane for rounding are taken into account. In case of spatial straight lines, this causes a change in direction in space. Similar applies to helical path segments.

	Documentation	
Documentation:	MTX Programming Manual	Activate rounding of corners

Deactivating

IW Operation/NC programming: Deactivating

Rounding(0) or **RND(0)**

	Documentation	
Documentation:	MTX Programming Manual	Rounding corners

12.4 Rounding corners with splines

12.4.1 Description

Function

The NC functions "rounding of corners with splines" round all contour transitions - the knee angles of which are greater than or equal to a configured minimum knee angle - with a spline. One modal NC function and one non-modal (local) NC function are available.

Properties:

- The modal NC function remains enabled until it is disabled.

  The local NC function is only enabled for the block transition in which it was programmed and disables the NC function for this transition if the function is also enabled.
- The transitions at the spline starting and end point are c^2-continuous, i.e. no axis acceleration jerks occur while traveling with constant path velocity.

- A maximum deviation "e" of the spline contour from the corner can be specified (see fig. 12-7 "Corner rounding when a maximum tolerance is specified " on page 446). The control now generates a symmetric spline which starts a distance "l" in front of the corner (on the pre-corner block) and ends after the same distance behind the corner (on the post-corner block). If one of the two corner contours is too short, the pre- and the post-corner distance is reduced to a common dimension l. l always remains smaller than or equal to the half of the shorter of the two corners contours, i.e. l ≤ ½min (S_1, S_2) with the path lengths S_1 and S_2 of the pre- and the post-corner block.

  e and l can be specified simultaneously (see above). The pre-corner and post-corner distances are limited to "L" in any case.
- Alternatively, pre- and post-corner distances l_1 or l_2 or a symmetrical pre-/post-corner L can be specified on the original contours instead of a tolerance e (refer to the figure "Corner rounding when a pre- and post-corner distance l_1 or l_2 is specified").
- It is rounded at each contour type.

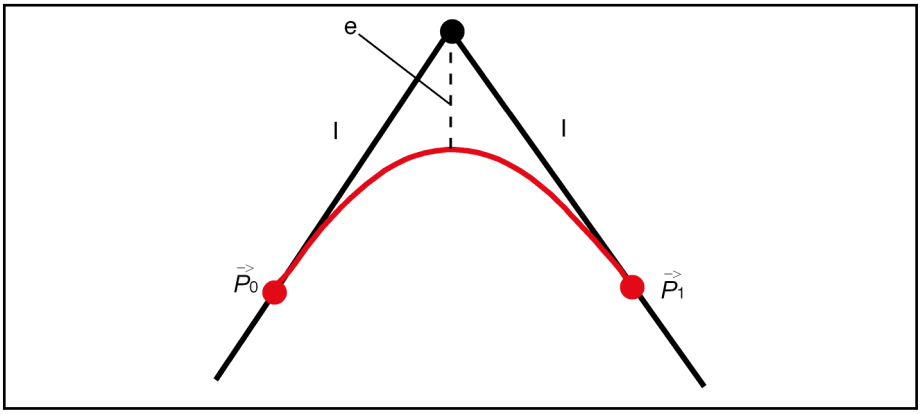


Fig. 12-7: Corner rounding when a maximum tolerance is specified

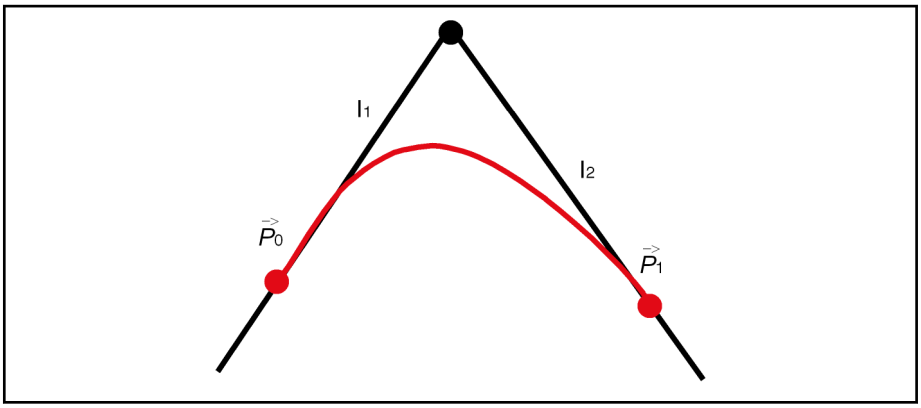


Fig. 12-8: Corner rounding when a pre- and post-corner distance l1 or l2 is specified

It is not rounded if one of the following conditions is met:

- The pre- or post-corner contour has a path length smaller than the value in the machine parameter MinLen. This is particularly applicable to non-traversing blocks.
- The motion in the two corner blocks belongs to various coordinate groups which calculate the feed.

  Example:

The pre-corner block comprises a motion in the coordinates x, y, z, and the post-corner block a pure orientation motion phi, theta.

- The knee angle is smaller than the value in the machine parameter MaxAng (quasi-tangential transition).
- There is direction reversal.

Special features:

With excessively large corner distances, "non-traversing blocks" are generated in the center of each of the original corner blocks (refer to the figure below).

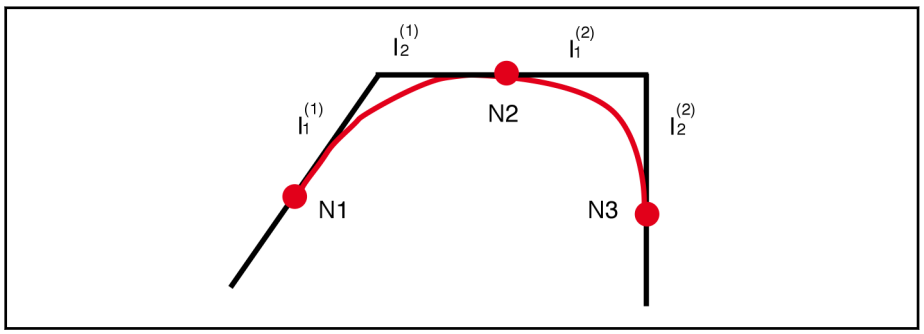


Fig. 12-9: Subsequent corners

$l_2^{(1)}$ and $l_1^{(2)}$ are distances shortened to the half of the N2 path length.

Restrictions

The simultaneous specification of corner distances l_1, l_2 and a corner tolerance e is not possible. Do not specify l and e simultaneously

Relevant modal NC function

SplineCornering(E<Distance>)

Rounding corners on (enabled); with a specified maximum permissible tolerance E.

SplineCornering(DEF)

Rounding corners on (enabled), with tolerance E from the machine parameter MaxDev.

SplineCornering(L1=<L1>,L2=<L2>)

Rounding corners on (enabled); with a specified pre-corner distance L_1 and a post-corner distance L_2.

SplineCornering(L<L>)

Corner rounding on; with a specified pre- and a post-corner distance $L_1 = L_2 = L$

SplineCornering(L<L>,E<distance>)

Rounding corners on; with specified maximum distance E allowed between the corner point and the spline and at the same time, a restriction on the pre-corner and post-corner distance to the value L.

SplineCornering(0)

Rounding corners off (disabled)

SCO

Short form

Relevant local (non-modal) NC function

SplineCorneringLocal(E<Distance>)

Rounding corners for the transition to the next NC block with a specified maximum permissible tolerance E.

SplineCorneringLocal(DEF)

Rounding corners on for the transition to the next NC block with tolerance E from the machine parameters MaxDev.

SplineCorneringLocal(L1=<L1>,L2=<L2>)

Rounding corners for the transition to the next NC block with a specified pre-corner distance L_1 and a post-corner distance L_2.

SplineCorneringLocal(L<L>)

Corner rounding on; with a specified pre- and a post-corner distance $L_1 = L_2 = L$

SplineCorneringLocal(L<L>,E<Distance>)

Rounding corners on; with specified maximum distance E allowed between the corner point and the spline and at the same time, a restriction on the pre-corner and post-corner distance to the value L.

SplineCorneringLocal(0)

No corner rounding for transition to next NC block.

SCOL

Short form

Interaction between modal and local NC function

If the local NC function is programmed in an NC block where the modal NC function is enabled, only the local function is applied for the transition to the next block. The modal function remains enabled but has no effect at this transition.

Relevant machine parameters

The same machine parameters are used as for circular rounding:

NCP/CornRound/CH[x]/MaxDev (705000110)	
	Maximum allowed deviation from the programmed contour. Is the switched-on state for the tolerance E.
NCP/CornRound/CH[x]/MinLen (705000120)	
	Minimum block length to start the corner rounding. If one of the neighboring traversing blocks is smaller, no rounding takes place.
NCP/CornRound/CH[x]/MaxAng (705000130)	
	Maximum angle up to which block transitions are considered quasi-continuous. With smaller knee angles, no rounding takes place.

Tab. 12-18: Relevant machine parameters

12.4.2 Handling instruction: Rounding corners with splines

Using the NC function in the NC program

1. Step **NC configurator: Editing machine parameters**

- Complete data in the machine parameter MaxDev "Maximum permissible deviation from the programmed contour". This is only required if SCO(DEF) is to be used.
- Enter data into the machine parameter MinLen "Minimum block length at which rounding starts" and MaxAng "Maximum angle up to which block transitions are considered quasi-continuous".

☞ These parameters are likewise applicable for circular corner rounding.

2. Step **NC program**

- In the NC program, switch on the function using SCO(E<Tolerance>), SCO(DEF) or SCO(L1=<L1>,L2=<L2>) and switch it off using SCO().

Example:

1- In the channel, the linear axes x, y and z and the rotary axis a are located.

NC blocks	Note
N0 FeedGrp(X,Y,Z,A)	Feed group is X,Y,Z,A
N1 X0 Y0	
N2 G91 SCO(E0.1)	Corner rounding on with a tolerance of 0.1 mm No rounding for N2/N3, as N2 has the path length 0
N3 X10	Rounding for N3/N4 with a tolerance of 0.1 mm
N4 Y10 SCOL(DEF)	Rounding for N4/N5 with a tolerance value from MP 705000110
N5 X20	Rounding for N5/N6 with a tolerance of 0.1 mm
N6 Y0 SCOL(0)	No rounding for N6/N7
N7 X30	Rounding for N7/N8 with a tolerance of 0.1 mm
N8 Y10 SCOL(L1=1,L2=2)	No rounding at N8/N9 due to tangential transition
N9 Y10	Rounding for N9/N10 with a tolerance of 0.1 mm Note: FeedGrp(X,Y,Z), no rounding would take place
N10 Y10 A10	No rounding with N10/N11 as N11 has path length = 0
N11 SCO()	Corner rounding off

Tab. 12-19: Example 1

Example:

2 - In the channel, there are the world coordinates X, Y, Z, phi and theta.

NC blocks	Note
N0 FeedGrp(x,y,z)	Feed group is x,y,z
N1 G90 x0 y0 z0 phi0 theta45	Initial state

NC blocks	Note
N2 SCO(DEF) x10	Corner rounding on with a tolerance value from MP705000110 Rounding for N2/N3
N3 y10 phi0 theta50	Rounding for N3/N4 Knee caused by xy motion
N4 x-10 phi0 theta70	No rounding for N4/N5 Different coordinate groups calculating the feed
N5 O(1,0,0)	Rounding for N5/N6 Knee by orientation motion
N6 O(0,1,0)	No rounding for N6/N7 N7 has the path length 0
N7 SCO()	Corner rounding off

Tab. 12-20: Example 2

Example:

3 - Quick bypassing of corners (blending)

NC blocks	Note
N0 G91	Incremental programming
N1 SCO(L1=5,L2=15) Y20	Start of spline at Y=15
N2 SCO(L1=15,L2=5) X100	End of spline at X=15 Start of spline at X=85
N3 SCO() Y-20	End of spline at Y=15

Tab. 12-21: Example 3

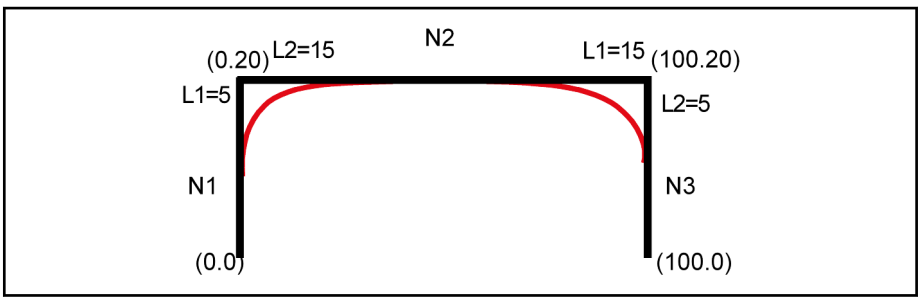


Fig. 12-10: Corner rounding for quick bypassing of corners (blending)

12.5 Punching and nibbling

12.5.1 Description

Function

Supplements the NC by various functions required for applications in punching or nibbling machines with hydraulic rams.

Features:

- Program-controlled switching on and off punching/nibbling via the NC functions "Nibble/NIB" and "Punch/PUN". The functions form a modal group, i.e. at one given time, only one of the functions can be active.

- Automatic splitting of a programmed traversing block into linear partial paths of identical length (block splitting). The end points of the partial paths are always on the programmed path.
- Automatic tangential approach of the punch or nibble tool to the programmed path at the block end by the “TangToolOri/TTO” (tangential tool orientation) function. See from chapter 12.6 "Tangential tool orientation " on page 455.
- External punching unit/punch control: Provision of a fast input and output signal for communication with a stroke-controlled punch control via the "interface for high-speed signal coupling" (high-speed I/O).
- Internal punching unit: Use of permanent CPL variables to communicate with the stroke-generating channel.
- The signal course between the NC and the punch control can be checked on the NC side by a stroke time monitoring function.
- NC-internal high-speed logic for exchange of information between the punch control, the NC, and the PLC.
- Stroke request by the NC or PLC. To communicate with the PLC, the NC provides different signals at the global interface. The PLC can for example use this function to trigger a stroke inhibit.
- Offset of the stroke release times by the NC (preliminary or subsequent stroke release). Time reference can either be defined as reaching the end point of a traversing motion by the interpolator or reaching the Inpos window of all axes.
- Skipping punching strokes by setting a control date. Punching strokes are skipped as long as the system date /SysPunch/Cmd/doSkipPunch has value 1: The punching unit does not contain any signals, the program is continued without punching stroke.

There are the following differences between punching and nibbling:

Function	For punching	For nibbling
Block splitting	Possible if required	Required
Stroke release at block end	Required only at the end of the programmed traversing block	Required at the end of each partial path
Stroke release at the start of a programmed traversing block	Not required	Only required if the previous block was no traversing block or neither punching nor nibbling was active in the previous block.

Tab. 12-22: Function in punching and in nibbling

When punching or nibbling is active, a stroke is generally released at the end of each block.

Exceptions:

- The current block does not comprise any axis coordinate in the active plane.

  Machining is resumed without stroke with the next block.
- The PLC suppresses stroke release.

  In this case, machining is resumed with the stroke only after release by the PLC.
- The function Skip stroke is active.

In this case, processing is continued without stroke and acknowledgement.

Restrictions

- Nibbling and punching are not suitable for machine tools with overrunning ram.
- With corner rounding active, nibbling and punching are not permissible.
- Nibbling or punching can only be switched on if MP 8001 00010 (function "Punching applied") has been assigned the value "1".
- Nibbling and punching can only be used in one single channel at a time.
- Precondition for punching/nibbling is a punch able velocity interpolation (G9 with SHAPE).
- External punching control: Interface for high-speed signal coupling required ("High-speed I/O").
- One high-speed digital input and one output is necessary to communicate with the punch control (configurable via MP 4075 00101 and MP 4075 00102).
- Internal punching unit (electric punching head): No digital input and digital output but connection via a permanent CPL variable is required:

  The permanent variables @HSIN and @HSOUT have to be created.

Relevant NC functions

Punch/PUN	Punching ON
Nibble/NIB	Nibbling on For nibbling, block splitting must be active ("LEN=" > 0).
PtInpos/PTI	Definition of axis-based stroke release times in respect of the time reference "Inpos window reached". Here, preliminary stroke release is not possible.
PtBlkEnd/PTE	Definition of axis-based stroke release times in respect of the time reference "interpolator end point reached". Both preliminary and subsequent stroke release are possible.
PtDefault/PTD	Set stroke release times and time reference of all axes to machine parameter values (MP 8001 00020 and MP 8001 00021).

Tab. 12-23: Relevant NC functions

Relevant IF signals

HSO-0	high-speed digital output (Can be set by means of MP 4075 00102). NC requests stroke release
HSI-0	high-speed digital input (Can be set by means of MP 4075 00101). Punch control acknowledges
iGen_StrokeIntend	""Stroke intended" (global interface) The NC informs the PLC that it wants to release a stroke.
iGen_NoStroke	"Stroke not running" (global interface) Image of HSI-0 for the PLC.

qGen_StrokeInhibit	"Stroke inhibit" (global interface) The PLC prevents setting of HSO-0.
qGen_StrokeReserv	"Stroke reservation" (global interface) The PLC reserves HSO-0 to execute its own stroke release.
qGen_StrokeRel	"Stroke ON" (global interface) The PLC commands the NC to release a stroke.

Tab. 12-24: Relevant IF signals

Relevant machine parameters (MP)

4075 00101	Assignment of digital inputs
4075 00102	Assignment of digital outputs
8001 00010	Punching function applied
8001 00020	Axis-specific stroke release times
8001 00021	Referencing stroke release times
8001 00030	Stroke check time

Tab. 12-25: Relevant machine parameters

Relevant system data

/SysPunch/Cmd/doSkipPunch	=1 strokes are skipped
/SysPunch/Cmd/ suppressSkipMsg	=1 no warning if the function "Skip strokes" is active

Tab. 12-26: Relevant system data

12.5.2 Handling instruction: Activate punching and nibbling

The punch or nibble package supplements the NC by various functions required for applications in punching or nibbling machines with externally controlled rams.

Applying

NOTICE **Workpiece or machine can be damaged!**

Do not reset the control during machining.

1. Set MP 8001 00010 (function "Punching applied") to the value "1".
2. Set the time reference (MP 8001 00021) and the stroke release times (MP 8001 00020) in correspondence with your application.
3. If desired, parameterize stroke monitoring (MP 8001 00030).
4. Via MP 4075 00101 and MP 4075 00102, reserve a high-speed digital input and output.
5. To specify the function state at startup and after reset, we recommend including the following settings as switched-on state in MP 7060 00010 (startup) and MP 7060 00020 (reset):

   PUN(0) or NIB(0) nibbling/punching OFF

   PTD stroke release times and time reference according to MP 8001 00020 and MP 8001 00021

   TTO(0) tangential tool orientation OFF.

6. Trigger control reset.

Activating

Prerequisite:

- Punching/nibbling is applied.
- A high speed digital input and output is reserved through MP 4075 00101 or MP 4075 00102.
- G9 with SHAPE is active.
- The necessary interface signals are evaluated or generated.

1. Optional for punching, mandatory for nibbling:
   Specify the length of the blocks segments via "LEN=".
2. Program punch(1) or PUN(1) to activate punching
   - or -
   Program nibble(1) or NIB(1) to activate nibbling.

Deactivating

Program Punch(0) or PUN(0), respectively Nibble(0) or NIB(0) (the two functions are equal and switch off nibbling and punching).

	Documentation	Instruction
Instruction:		Editing machine parameters
Documentation:	MTX Machine Parameters	Punching and nibbling

IW Operation/NC programming: Programming the LEN function

- Programming of **LEN=<Value>** is mandatory for nibbling and optional for punching. **LEN** splits the programmed traversing motions into subsegments according to the specified **<Value>**.
  - For linear blocks: **<Value> =**Partial path length
  - For circular blocks: **<Value> =**Arc length



The same programming unit as for the axis coordinates.

<Value> does not have to be an integral divisor of the programmed length of path. Within the NC, an effective LEN value smaller than/equal to the programmed LEN value is determined automatically so that the calculated subsegments show equal path lengths (The <value> is adjusted accordingly within the control.)

	Documentation	
Documentation:	MTX Machine Parameters	Function LEN

IW Operation/NC programming: Activate punching or nibbling

- Program **Punch(1)** or **PUN(1)** to activate punching.

- or -

- Program **Nibble(1)** or **NIB(1)** to activate nibbling.

IW Engineering/configuration: Editing parameters

- Allocate the "punching" function to one high-speed digital input - via parameter **DigIOAllocIn** "Digital input allocation" (4075 00101) - and to one digital output - via parameter **DigIOAllocOut** "Digital output allocation" (4075 00102).
- To specify the function state at startup and after reset, we recommend including the following settings as switched-on state in the parameter **NcResetState** "State after startup (switched-on state)" (7060 00010) and

in the parameter **ChResetState** "State after channel reset" (7060 00020).

- **PUN(0)** or **NIB(0)** nibbling/punching OFF
- **PTD** stroke release times and time reference according to **TimeOffs** "Stroke release time" (8001 00020) and **TimeRef** "Reference of the stroke release times" (8001 00021)
- **TTO(0)**tangential tool orientation OFF.

- The punching and nibbling function must be activated using the parameter **pun** "punching and nibbling"
- **EnablPun** "Activate punching and nibbling" (8001 00010)
- **StrokeMonTime** "Stroke monitoring time" (8001 00030)
- **TimeRef** "Reference of the stroke release times" (8001 00021)
- **TimeOffs** "Stroke release time" (8001 00020)

	Documentation	
Documentation:	MTX Machine Parameters	Activate punching and nibbling

12.6 Tangential tool orientation

12.6.1 Description

Function

When "punching" (Punch/PUN) or "nibbling" (Nibble/NIB) is active, this function provides a synchronous tool axis of the "endless" axis motion type (%MP 1003 00004) for each stroke tangentially to the programmed path in the active plane of a channel.

To this end, the tool axis moves synchronously with the plane-defining linear axes and reaches the required offset angle at the same time as the target position of these linear axes. Here, the term "target position" refers to the end position of a programmed block as well as to the end position of block segments which the NC can generate in connection with automatic block splitting.

If a stroke has to be released already at the start of the programmed block (as is sometimes required with nibbling), the NC automatically generates a separate block in which the tool axis is previously rotated to the required offset angle.

Features:

- Parameterizable setting angle (angle between the symmetry axis of the tool and the path tangent).
- The tool symmetry is taken into account. In this way, the control can independently optimize the setting rotation of the tool necessary for one stroke with contour knees ("shortest way" logic).
- Offset angle and tool symmetry can be parameterized in the program

Restrictions

- The "Tangential tool guidance" function (TangTool or. TTL; see chapter 10.4 "Tangential tool guidance" on page 180) must not be active simultaneously.
- The "tangential tool orientation" function can be programmed and activated at any time. However, the tool axis only performs approaching

motions if nibble/NIB ("nibbling ON") or punch/PUN ("punching ON") are active.

- TangToolOri/TTO has to be programmed in a separate block.
- The tool axis has to be defined through MP 7050 00210. Program-controlled changing of the tool axis is not possible.
- Only synchronous endless axes are permissible as tool axes.

Relevant NC functions

TangToolOri/TTO	Initialization and activation of the "tangential tool orientation". TangToolOri does not produce any traversing motion. The tool axis is only approached when the next punch or nibble traversing block is started, provided no stroke has to be released at block start (cf. chapter "Function" on page 455).
TangToolOri(0) **- or -** **TTO(0)**	Deactivation of the "tangential tool guidance"

Tab. 12-27: Relevant NC functions

Relevant machine parameters (MP)

The values of the machine parameters act as initialization values if TangToolOri is programmed without parameters. You can overwrite the initialization values in the TangToolOri block with the parameters given in brackets. See the "MTX Programming Instructions" for the syntax.

7050 00210	Rotary axis number
7050 00220	Tool symmetry (SYM)
7050 00240	Offset angle (ANG)

Tab. 12-28: Relevant machine parameters

12.6.2 Activation

Prerequisite:

- The axis of rotation already exists in the system as a synchronous endless axis and - in respect of velocity, dynamics, and positioning logic if applicable - has been correctly parameterized according to your requirements.
- MP 7050 00210 (number of rotary axis) has been parameterized according to your requirements.
- The necessary axis and channel-based interface signals are evaluated or generated.

1. If the "tangential tool guidance" function (TangTool) is active for the respective axis, switch it off (TangTool(0) or TTL(0)).
2. Activate nibble/NIB ("nibbling ON") or punch/PUN ("punching ON").
3. Activate "tangential tool guidance" through TangToolOri block.

   In the same block, set the initialization values of tool symmetry and offset angle in accordance with your application by means of the "SYM" or "ANG" parameters.

   See the "MTX Programming Instructions" for the syntax.

☞ "Tangential tool orientation" can also be parameterized and activated when punching and nibbling are not active.

There are only approaching motions of the tool axis if punching and nibbling are active!

12.6.3 Deactivating

Program TangToolOri(0) or. TTO(0) .

12.7 Laser machining

12.7.1 Path velocity-dependent laser power control

Function

The function controls the power of a laser in dependence on the current actual feed value (V_{path}). To this end, a corresponding voltage value is output at an analog output in accordance with the configuration in machine parameter AnaIOAllocOut[1..4].

Above a specified path velocity V_{path}, the laser power is restricted to an adjustable minimum. Below a specified path velocity V_{path}, the laser power is kept to an adjustable maximum.

Between V_{Min} and V_{Max}, the laser power is adjusted in dependence on the path velocity, as is shown in the diagram below:

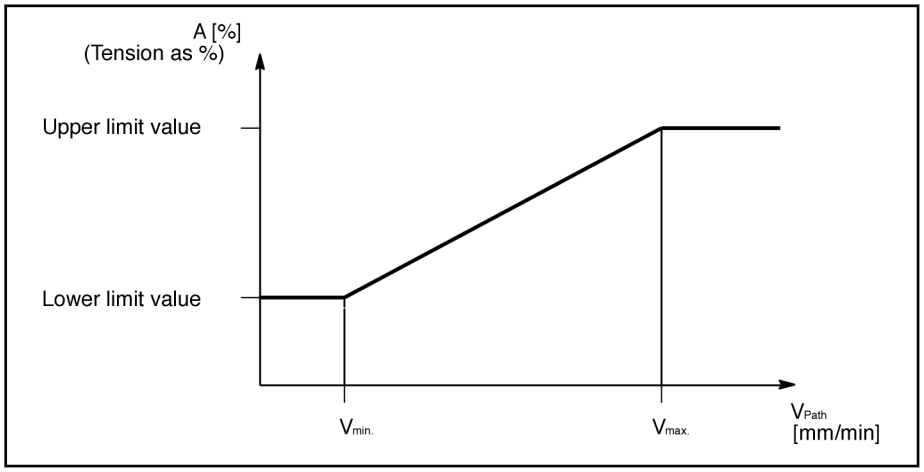


Fig. 12-11: Laser machining

The voltage output value is given in %.

100% voltage output value corresponds to 10 Volt at the analog output.

The effective path velocity V_{path} is derived from the velocities of the selected coordinates:

- By selecting the active level (APL) or the active space (ASP). In this, any active axis transformations or coordinates transformation (inclined plane) is taken into account as well.
- By direct selection of workpiece coordinates in the part program.

Restrictions

- The available analog outputs limit the number of channels that may use this function.

- When using the "inclined path" function, only "active room" with PL(ASP) can be used when selecting the coordinates for calculation the velocity V_{path}.
- In case of error (runtime error, status class1 error),of removal from "drive in operation" of one of the drives participating in the path (no enable signal, drive OFF), and of "Feed hold", no laser voltage is output.

Relevant NC functions

LFP	**Starts** path velocity-dependent laser power control. Individually programmed, the configuration data defined via machine parameters are applied. Optionally, additional parameters can be programmed within the parameter list of the function.	
LFConf/LFC	**Parameterizes** the active laser power control by means of the part program. Parameter for LFP and LFConf:	
	LL([<%Voltage>], [<VMin>])	
		Lower power limit: The voltage value specified here is output below the given path velocity.
	UL([<%Voltage>], [<VMax>])	
		Upper power limit: The voltage value specified here is output above the given path velocity.
	with	
	<%Voltage>	0% .. 100%: Corresponds to 0 .. 10 Volt
	<Vmin>	Lower benchmark of path velocity in mm/min or inch/min.
	<Vmax>	Upper benchmark of path velocity in mm/min or inch/min.
	PL(<level name>)	Coordinate selection for calculating the path velocity through a plane selection.
	with	
	<level name>	"APL": Current plane (G17, G18, G20) "ASP": Current space "MCD": Machine parameter values
	CD(<Coordinate1>, [<Coordinate2>], ... , [<Coordinate n>])	
		Coordinate selection for calculating the path velocity directly using the logic name.
	with	
	<Coordinates>	x = 1..n Logic names of the participating spatial coordinates or pseudo coordinates (axes)
LFP(0)	**Terminates** path velocity-dependent laser power control	

Tab. 12-29: Relevant NC functions

Relevant Sercos parameters

S-0-0040	Actual velocity value If S-0-0040 is not comprised in the drive telegram, the velocity is derived from the change in the actual axis position (S-0-0051 or S-0-0053).

Tab. 12-30: Relevant Sercos parameters

Relevant IF signals

At the **channel interface**	
qCh_FeedStop	Feed inhibit
qCh_FeedHold	Feed hold
at the **axis interface**	
qAx_DrvOn	Drive on
qAx_DrvLock	Feed inhibit

Tab. 12-31: Relevant interface signals at the channel/axis interface

Relevant machine parameters

PLC/AnaIO/FuncAllocOut/AnaIOAllocOut[1..4] (4075 00104)	
	Assignment of the analog outputs 701: Channel 1 .. 712: Channel 60
VDO/Ch[k]/Out/AnaOutUpVelLim (7050 00810)	
	Upper power limit value in %
VDO/Ch[k]/Out/AnaOutLowVelLim (7050 00810)	
	Lower power limit value in %
VDO/Ch[k]/Vel/UpVelLim (7050 00815)	
	Upper speed limit values in mm/min
VDO/Ch[k]/Vel/LowVelLim (7050 00815)	
	Lower speed limit values in mm/min
VDO/Ch[k]/CoordSel (7050 00820)	
	Selection of coordinates: • No coordinates • Active machining plane • Active working space

Tab. 12-32: Relevant machine parameters

12.7.2 Handling instruction: Path Velocity-Dependent Laser Power Control

Applying:

Prerequisite:

Analog outputs (e.g. through Profibus) have to exist.

The function controls the power of a laser in dependence on the current actual feed value (V_{path}). To this end, a corresponding voltage value is output at an analog output in accordance with the configuration in parameter **AnalOAllocOut** "Allocation of the analog outputs" (4075 00104).

IW Engineering/configuration: Editing parameters

- Allocate the parameter **AnalOAllocOut** "Allocation of the analog outputs" (4075 00104) a value of 0701...0712 (for channel 1..60).
- The velocity-proportional output function must be activated via the optional parameter group **VDO** "Velocity-proportional output" in Setup (SUP) in order to configure the following parameters for the utilization of the analog outputs:
  - **AnaOutUpVelLim** "Command value when reaching the upper velocity limit value" (7050 00810)
  - **AnaOutLowVelLim** "Command value when reaching the lower velocity limit value" (7050 00810)
  - **UpVelLim** "Upper velocity limit value" (7050 00815)
  - **LowVelLim** "Lower velocity limit value" (7050 00815)
  - **CoordSel** "Coordinate selection for calculating the path velocity" (7050 00820)

☞ The drive telegrams of the participating axes should contain the Sercos parameter **S-0-0040** "Actual velocity value".

If the Sercos parameter **S-0-0040** "Actual velocity value" is not comprised in the drive telegram, the velocity is derived from the change in the actual axis position (**S-0-0051/S-0-0053** "Actual position value encoder 1/2").

Example	Documentation	Instruction
Instruction:		Editing machine parameters
Documentation:	MTX Machine Parameters	Laser power control

Activating:

After control startup, laser power control is deactivated, and the settings of the relevant machine parameters are selected.

Specifications from the machine parameter NCResetState "On-state after startup" and the machine parameter ChResetState "Switched-on state after reset" overwrite this presetting.

IW Operation/NC programming: Activate the laser power control

Laser power control can be activated via the presettings made in the machine parameters; if necessary, they can additionally be influenced in the part program by means of parameters:

1. Activation of laser power control with **LFP{(1)}** (with the default parameters).

   - or -

2. Activation of laser power control with additional parameters **LFP(LL(..),UL(..),PL(..))**.

The active laser power control can only be set in the part program with **LFC(LL(..),UL(..),PL(..))**.

Activation/Deactivation of the path velocity-dependent laser power control with presetting. During operation, the parameters in the part program can be influenced. The function controls the power of a laser using an analog voltage signal (0...10 V), depending on the current path feed v_{path}.

Example:

Setting the machine parameters:

AnaIOAllocOut[3]	701	Channel 1 uses the 3rd analog output
AnaOutLowVelLim	20	Lower power limit value 20% (=2 V)
AnaOutUpVelLim	90	Upper power limit value 90% (= 9 V)
LowVelLim	500	Lower velocity limit value 500 mm/min
UpVelLim	2000	Upper velocity limit value 2000 mm/min
CoordSel	2	Actual path velocity is derived from the active plane.

Tab. 12-33: Example: Setting the machine parameters

Part program 1:

```
N010 G17 G8 G62 F2500
N020 G0 X0 Y0 Z0
N110 G1 X10 LFP(1)   Switch on laser
N120 Y10
N130 X0
N140 Y0 LFP(0)       Laser off
                     Attention:
                     The aces have not reached their end point.
N200 G54
N210 X10 LFP(1)      Switch on laser
N220 Y10
N230 X0
N240 Y0
N250 G4 F0.5 LFP(0)  Laser off after end of dwell time
N300 G55
N310 X10 LFP(1)      Switch on laser
N320 Y10
N330 X0
N340 Y0 G1(IPS) LFP(0)  Laser off after reaching the in-position window
M30
```

Explanation:

Output voltage is between 2 and 9 V.

If the path velocity in the active plane exceeds 2000 mm/min, 9 V are output.

If the path velocity falls below 500 mm/min, 2 V are output, e.g. at traveling to certain point, at corner (in dependence on the axis step change), at the end of motion.

Switching off the laser:

N140	When the laser is switched off, the axes have not yet reached the end point due to lag. For this reason, the machined part is faulty.
N250	Switching off the laser is delayed for a programmed period of time.
N340	The laser is switched off after the in-position window of all axes has been reached.

Part program 2:

N010 G17 G8 G62 F2500	
N020 G0 X0 Y0 Z0	
N030 LFConf(UL(, 1800)	Upper velocity limit 1800 mm/min
N040 LFConf(CD(X,Y)	Coordinate selection X and Y
N050 G1 X10 LFP(1)	Switch on laser
N060 Y10	
N070 X0	
N080 Y0 G1(IPS) LFP(0)	Laser off after reaching the in-position window.
N090 G54	
N100 LFP(PL(APL)	Select active plane and laser on.
N110 G1 X10	
N120 Y10	
N130 X0	
N140 Y0 G1(IPS) LFP(0)	Laser off after reaching the in-position window.
M30	All machine parameter data are effective once more.

Explanation:

Output voltage is between 2 and 9 V.

N030	Upper velocity limit 1800 mm/min
N040	If the path velocity in the selected coordinates (X and Y) with the active plane (N100) exceeds 1800 mm/min, 9 V are output.

If the velocity falls below 500 mm/min, 2 V are output.

Example	Documentation	Example
Example:		Example on laser machining
Documentation:	MTX Programming Manual	Laser power control

Disabling: **IW Operation/NC programming: Deactivate the laser power control**

1. **LFP(0)** switches off the power control at the end of an NC block.
2. After reset and M2/M30, power control is switched off, and the settings from the relevant machine parameters take effect once more.

	Documentation	
Documentation:	MTX Programming Manual	Laser power control

13 Axis configuration in the channel

13.1 Axis transfer

13.1.1 Description

Function

Within one channel, axes related in terms of machining are described as **axis group**. They form an interpolation group. For certain machining tasks, the **axis assignment** has to be changed, i.e. axes have to be removed from one axis group and integrated temporarily into the axis group of another channel. In this context, axis names within one axis group can be changed.

*The following **axis transfer functions** are provided:*

- Changing axes between axis groups, i.e.
  synchronous axis remains synchronous axis.
- removing axes from an axis group, i.e.
  synchronous axis becomes asynchronous axis.
- Integrating axes into an axis group, i.e.
  asynchronous axis becomes synchronous axis.
- Renaming of axes within an axis group.
- Changing the axis classification (operational importance).

Terms required:

Integrate axis:	Axis is **integrated** into the axis group.
Borrow axis:	Axis **changes**, i.e. the channel borrows an axis from an inactive channel.
Remove axis:	Axis is **enabled** from its old axis group.

☞ Adjust the machine parameter "Switched-on state after reset" (/CHAN/Ch[...]/Ini/ChReset) for the channels removing or accepting axes so that the reset is provided for a defined axis configuration. By entering the DefAxis or DAX command for the system reset, a coordinated removal of the axes from the corresponding channels and their subsequent acceptance by their default channels are provided.

Refer to chapter 3 "Drives (axes, spindles)" on page 13 for the following terms: drives, channel, axes, axis group, auxiliary axes, synchronous and asynchronous axes, system and channel axes, system and channel axis index and spindles.

Changing axes in between axis groups:

When changing axes between two channels, there are two different cases:

- **The channel "source axis group" is not active.**
  If the channel - the axis originates from - is not active, the axis can be integrated into a second axis group at any time, i.e. the second channel "borrows" the axis. If it is enabled there again, it is automatically integrated again into the source axis group.
  If a program or MDI is selected in the source axis group while an axis is borrowed, an error message is issued.

With this type of axis transfer, a user reset program in another channel can for example be started after a system reset which borrows all system axes, also from other channels, and moves them to an initial position.

- **The channel of the "source axis group" is active.**

  When changing an axis from an active channel (axis group 1), the axis is first removed and integrated into another channel (axis group 2) in a second step.

  To ensure that the axis can be accepted without any delay in block preparation, the axis previously has to be removed from axis group 1. This is the case if the respective NC block in axis group 1 has already been active upon transfer of the axis into axis group 2.

Example:

The x-axis of channel 1 is transferred to channel 2 (see the figure below).

In channel 1, block N1310 is already being prepared and block N1220 is active. Thus, enabling the x-axis is completed.

At this time, block N2110 is active in the second channel. Preparation is on block N2220 and wants to apply the system axis XP (formerly X of channel 1).

The XP-axis is now called ZA. As this name is already known in channel 2, the axis is integrated without any delay.

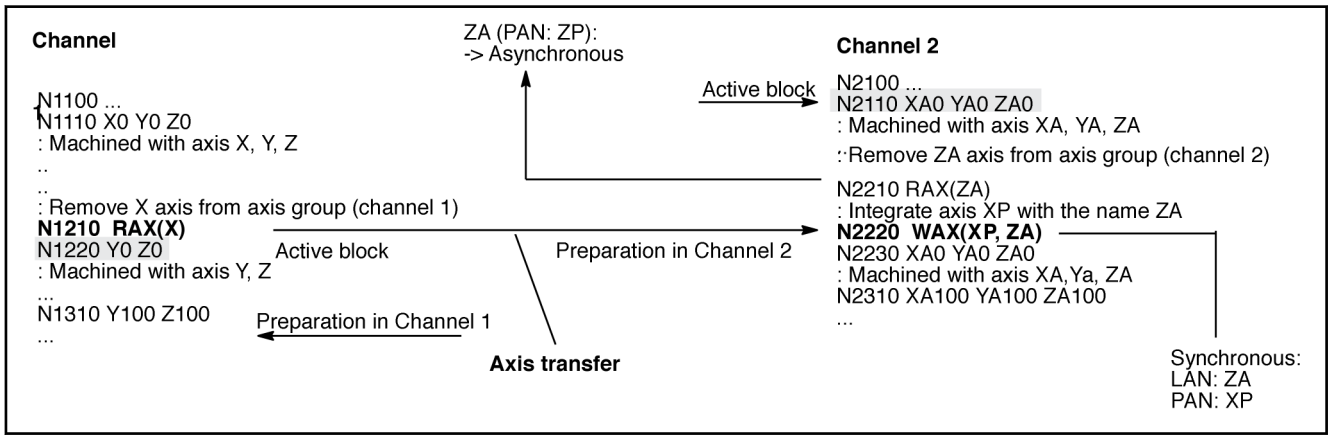


Fig. 13-1: Example - Changing axes between axis groups

Removing axes from an axis group:

Remove an axis from an axis group with "RemAxis" (RAX).

Thus, a synchronous channel axis is turned into an **asynchronous** axis.

During this action, block preparation is not stopped.

Example:

Channel axis Y is enabled. Afterwards, it can be programmed in all channels as asynchronous axis YP. The channel axes Z and B receive new channel axis numbers. In a channel axis display, the y-axis is no longer displayed. Programming channel axis Y results in an error message.

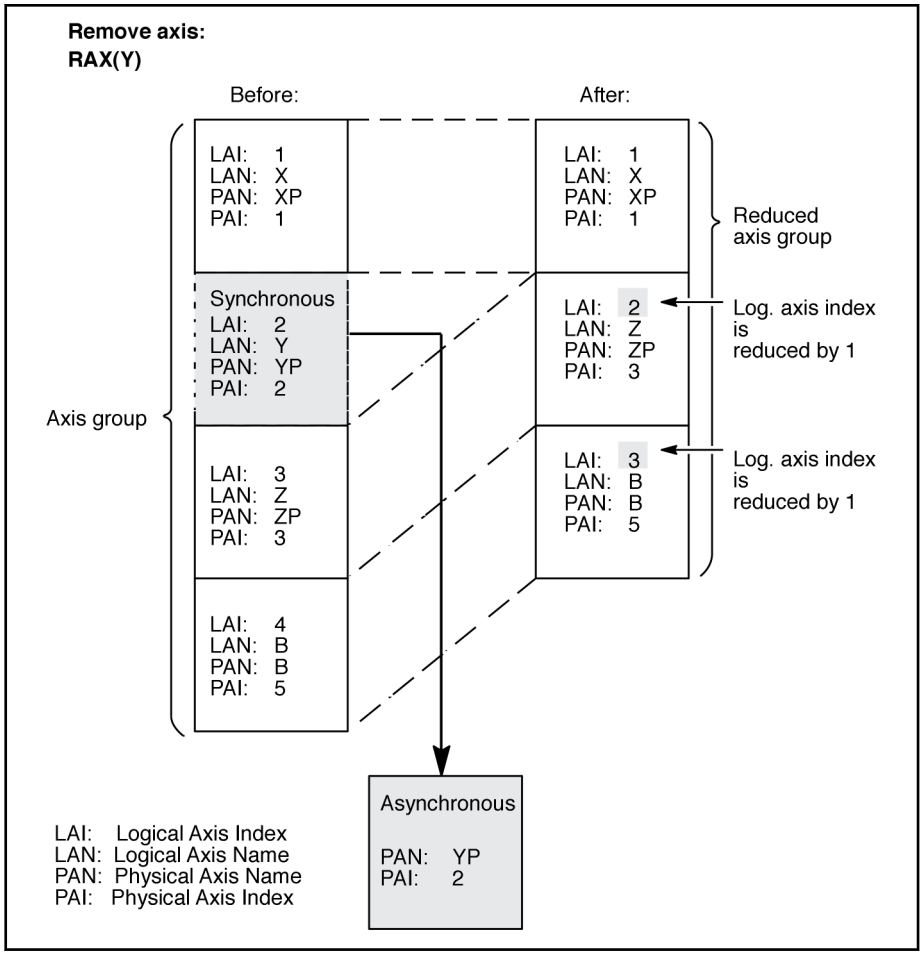


Fig. 13-2: Example - Remove axis

Applying axes to an axis group:

Use "GetAxis" (GAX) or "WaitAxis" (WAX) to apply an asynchronous axis to an axis group. This turns it into a synchronous channel axis.

- With "GetAxis" (GAX), the axis has to be static. Otherwise, an error message is issued and a block preparation is stopped.
- "WaitAxis" (WAX) implicitly waits for the axis to be at standstill.
- Optionally, a new channel axis name can be specified used to address the axis in the channel. This channel axis name has to be predefined in one of the two channel-specific machine parameters 701000010 "names of the channel axes" or 701000020 "optional axis name".

Example:

In the figure below, the system axis YP is integrated into an axis group. Subsequently, it can no longer be addressed in other channels as the asynchronous axis YP. In its own channel, it can either be programmed as synchronous axis with the channel or the system axis name. In a channel axis display, the y-axis is not displayed additionally.

The channel axes Z and B receive new channel axis indices.

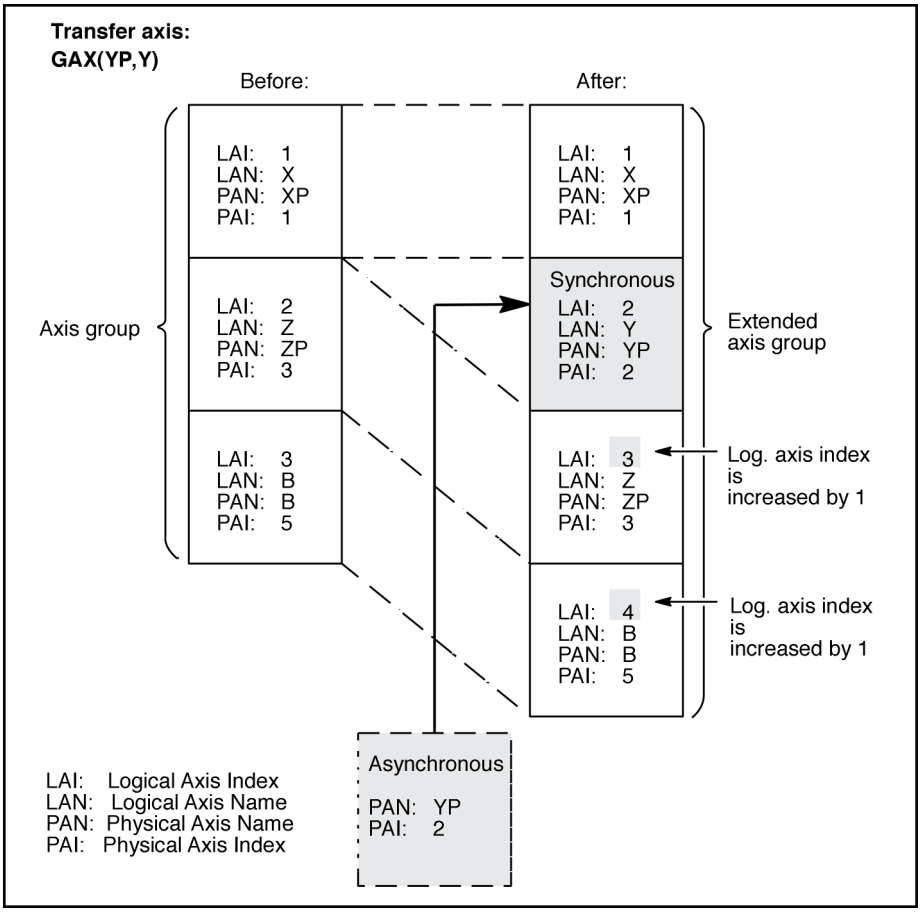


Fig. 13-3: Example - Applying axes to an axis group

An applied axis has no operational importance unless it is part of the default configuration of the channel. Otherwise, it receives its default classification from the machine parameters unless this has not already been specified in the channel.

Renaming axes within an axis group:

- When applying an axis into an axis group via GAX or WAX, a new channel axis name can optionally be specified.
- The channel axis name can be changed directly in "AssLogName" (ALN).

☞ The logic name has to be predefined in MP 7010 00010 "Name of the channel axes" or MP 7010 00020 "Optional axis name".

Assignment of a operational importance:

Via MP 7010 00030, a ***operational importance*** *(axis meaning) can be assigned to the channel axes of a channel:*

- Main, secondary, and normal axes of the workpiece coordinate system
- Assignment of the interpolation parameters (I, J, K)

Via axis transfer, the axis meaning can be influenced:

- If an axis is **removed** from an axis group located within the selected plane, the system implicitly switches to G16 "no plane". G16 can be used in the part program.
- If an axis is **applied** to an axis group, it only gets a operational importance if
  - it is assigned to this channel within the machine parameters.

– an axis meaning is assigned within the machine parameters and this axis meaning is currently not assigned to any other axis in the channel.

☞ By programming G17(...), G18(...) or G19(...), the operational significance of axes can be changed.

Example: G17(...) "Changing the axis classification"

`N100 G17()`	Default axis significance: X=1, Y=2, Z=3
`N200 RAX(Y)`	Y is removed from the axis group. It is implicitly switched to G16. Circular interpolation is no longer possible.
`N210 WAX(YA)`	The YA-axis is applied to the axis group. It does not get any operational significance.
`N220 G17(X,YA)`	YA is assigned axis significance 2, and the X/YA-plane is activated.
`N240 G2 X... YA...`	Circular interpolation is possible.

Axis display at axis transfer:

The axis display can change at axis transfer.

This is due to the following reasons (also refer to the examples in chapter 13.1 "Axis transfer " on page 465):

- The sequence of the current axis names and their axis assignments to the channel has changed.
- Axes were removed from the axis group or applied to the axis group.
- The names of the channel axes might have changed upon transfer

In MP 6005 00020, the sequence in the axis display is specified (1..16). Also without changing the sequence, the axis names in the display can jump due to the transfer.

The corresponding axis is only displayed if its is a synchronous channel axis or an asynchronous axis. If a negative value is entered into the parameter, it is always displayed.

Restrictions

- The functions GAX, WAX, RAX and DAX ("DefAxis") have to be programmed in the NC block before an axis.
- Plane selection G17(...), G18(...), G19(...):

  Operational significance
- Axis coupling "AxCouple" (AXC):

  Removing an axis coupled with AXC results in a runtime error.
- Tangential tool guidance "TangTool" (TTL):

  Removing a rotary axis from an axis group while TTL is active results in a runtime error.
- If necessary, ZO tables have to be adjusted.

Relevant NC functions

GAX(..)	Applying axis Error message if the axis is not enabled in its previous axis group.
WAX(..)	Applying axis with WAIT until axis is enabled.
RAX(..)	Removing axis from axis group.
DAX	Applying default axis settings from machine parameter.
ALN(..)	Assigning new logic name. The logic name has to be predefined in MP 7010 00010 "name of the channel axes" or MP 7010 00020 "optional axis name".
RLN(..)	Removing channel axis names in the calling axis group Deactivating plane selection. Circular interpolation is no longer possible. Main and secondary axes can be removed from the axis group.
G16	
G17(..),G18(..),G19(..)	Changing operational significance and selecting a plane.

Tab. 13-1: Relevant NC functions

Relevant CPL functions

SD(20..)	Provides the number of synchronous axes of the calling channel.
SD(21..)	Provides the number of synchronous axes of a channel.
SD(22..)	Provides the channel axis number of a system axis if the system axis is an axis of the calling channel.
SD(23..)	It provides the system axis number of a channel axis of the calling channel.
SD(24..)	Provides the system axis number of a channel axis.
SD(25..)	Provides the channel of a system axis at the active point in time.
MCODS(43, ...)	Axis - channel assignment (identifying active synchronous axes of a channel)
FXC(..):	Direct access to the axis zero point offset values
SCS, APOS and SPOS	Different commands for which the system axis index (1 .. 16) or the system axis name is programmed.
PCS; AXO;	
WCS; MCS; ACS;	
PPOS; PROBE;	
COF; DPC;	
SCL	Different commands for which the channel or the system axis index is programmed.

Tab. 13-2: Relevant CPL functions

Relevant IF signals

iAx_ChIndex_00 ... iAx_ChIndex_03	In these axis output signals, the channel number of the respective channel is coded. 0 = Axis is not assigned to any channel, e.g. if it is operated as asynchronous axis.

Tab. 13-3: Relevant IF signals

Relevant machine parameters (MP)

1003 00001	Names of the system axes. Defines the axis names for a maximum of 99 system axes (real axes at the machine).
1003 00002	Default axis assignment to the channels.
7010 00010	Names for channel axes. The coordinate axes of the machine, workpiece or program coordinate system are described as logic axes. They are limited to eight axes per channel.
7010 00020	Optional axis names. If unknown channel axis names are used when applying or renaming axes, preset them here.
6005 00022	Display priority for axes and spatial coordinates of the channel, which do not participate in any axis transformation or if no axis transformation is active. 6005 00022 has a higher priority than 6005 00020.
6005 00020	Display priority for asynchronous third-party axes via signs: **+**: Axis is only displayed if it has been assigned to the channel **-**: Axis is also displayed if it is assigned to a third-party channel.
6005 00021	Display priority for spatial coordinates of third-party channels or for spatial coordinates of the own channel if they are **not** to be displayed in front position.

Tab. 13-4: Relevant machine parameters

☞ Within a channel, the sequence, in which all spatial coordinates are displayed, is specified. Only the complete unit can be displayed in the display.

13.1.2 Handling instruction: Axis transfer

Applying: For certain machining tasks, the axis assignment has to be changed, i.e. axes have to be removed from one axis group and integrated temporarily into the axis group of another channel.

IW Engineering/configuration: Editing parameters

- Specifying system axis name
  SysDrName
  "Physical drive name " (1003 00001)
- Specifying channel axis name
  ChAxName

"Axis name" (7010 00010)

- Set additionally required (optional) channel axis names

  OpChAxName

  "Optional axis name" (7010 00020)

- Specifying system axis assignment to the channels

  DefaultCh

  "Channel assignment" (1003 00002)

- Determining display sequence for axes and spatial coordinates

  ModeAddAx

  "Displaying axis in additional window" (6005 00020)

  PrioAddAx

  "Sequence of the axis in the additional window" (6005 00020)

  ModeChPack

  "Displaying channel coordinate package in the additional window" (6005 00021)

  PrioChPack

  "Sequence of the channel coordinate package in the additional window" (6005 00021)

  ModeAxChWin

  "Displaying axis in the channel window" (6005 00022)

  PrioAxChWin

  "Sequence of the axis in the channel window" (6005 00021)

- To restore the default axis configuration at reset, the following parameter has to be configured:

  Enter the following keyword in the parameter **ChReset** "State after channel reset" (7060 00020):

  "#Reset:DAX" if the DAX function "Apply axis settings from MP" is only to be executed at channel reset

  or

  "#SysReset:DAX" if the DAX function "Apply axis settings from MP" function is to be executed

  at system reset.

☞ The DefAxis (DAX) function behind the **"#SysRes:DAX"** keyword activates the default axis configuration for all channels according to the parameter DefaultCh "Channel Assignment" (1003 00002).

	Documentation	Instruction
Instruction:		Editing machine parameters
Documentation:	MTX Machine Parameters	Axis transfer

IW Engineering/IndraLogic: Linking interface signals

Couple axis interface signals **iAx_ChIndex_00 ... iAx_ChIndex_03** in the PLC to provide information on axis assignment (synchronous/asynchronous channel assignment).

	Documentation	Instruction
Instruction:		Linking PLC signals
Documentation:	MTX PLC Interface	Axis assignment

IW Operation/program: Adjusting zero point offset

If necessary, enter the axes to be transferred to the respective zero point offset tables of the corresponding channel.

		Instruction
Instruction:		Zero offsets

IW Operation/diagnostics: Restart NC

To apply the set axis configuration, a control startup is required.

Activating:

Prerequisite:

From a programming point of view, the programs operating in different channels have to be synchronized in respect of axis transfer.

The axis to be transferred, for example, have to be enabled before it is integrated by another channel. If applicable, the program of a channel has to wait for a transfer state of the program in another channel.

For this purpose, the following functions are used depending on the use case:

- Requesting **GetAxis(...), GAX(...)** axis
- Requesting **WaitAxis(...), WAX(...)** axis and waiting if required
- Enabling **RemAxis(...), RAX(...)** axis

GetAxis, GAX

Transfers an asynchronous axis to the calling channel. This turns an asynchronous axis to a synchronous axis. The axis can be programmed in the current channel using its system or channel axis names.

The axis to be accepted has to be enabled by the system (that means that it may not belong to any other channel and may not participate in any axis motion).

IW Engineering/Programming:

GetAxis(<SAN> | <SAI>,{<KAN>}{,<SAN> | <SAI>,{<KAN>}}...)

Short form **GAX(...)**

<SAN> System axis name. Specifies the axis to be applied to the current channel.

<SAI> System axis index. Same effect as <SAN>.

<KAN> Channel axis name. If programmed, the axis to be accepted in the current channel receives the channel axis name <KAN>. <KAN> has to be defined in the parameter **ChAxName** “axis name” (7010 00010) or in the parameter **OptChAxName** "Optional axis name" (7010 00020)

If an axis to be applied is not yet enabled, an error message is output in contrast to the **WaitAxis(...)** function.

Axis positions in the same block always have to be programmed after **GetAxis(...)** and may only be programmed if axis transformation is not active.

Axes to be applied must not be participating in any active monitoring area.

Note that after accepting an axis, it may be necessary to span the WCS once again. This affects the following functions:

- Placements
- Active plane
- Tool corrections

	Documentation	
Documentation:	MTX Programming Manual	Apply axis (WAX)

WaitAxis, WAX

Transfers an asynchronous axis to the calling channel. This turns an asynchronous axis to a synchronous axis. The axis can be programmed in the current channel using its system or channel axis names.

If the axis to be accepted is not yet enabled, the system waits on enabling.

IW Engineering/Programming:

WaitAxis(<SAN> | <SAI>,{<KAN>}{,<SAN> | <SAI>,{<KAN>}}...)

Short form **WAX(...)**

<SAN> System axis name. Specifies the axis to be applied to the current channel.

<SAI> System axis index. Same effect as <SAN>.

<KAN> Channel axis name. If programmed, the axis to be accepted in the current channel receives the channel axis name <KAN>. <KAN> has to be defined in the parameter **ChAxName** “axis name” (7010 00010) or in the parameter **OptChAxName** "Optional axis name" (7010 00020)

If an axis to be transferred is not yet enabled, the block preparation waits for the axis standstill. The axis is subsequently transferred. As opposed to the function **GetAxis(...)**, no error message and program abort follows.

Axis positions in the same block always have to be programmed after **WaitAxis(...)** and may only be programmed if axis transformation is not active.

Axes to be applied must not be participating in any active monitoring area.

Note that after accepting an axis, it may be necessary to span the WCS once again. This affects the following functions:

- Placements
- Active plane
- Tool corrections

	Documentation	
Documentation:	MTX Programming Manual	Apply axis (WAX)

RemAxis, RAX

Removes a synchronous axis from the calling channel. This turns a synchronous axis into an asynchronous axis. The asynchronous axis can then be programmed in any channel using its axis name.

IW Engineering/programming: Removing axis from axis group

RemAxis(<SAN> | <SAI> | <KAN>{,<SAN> | <SAI> | <KAN>}...)

Short form **RAX(...)**

<SAN> System axis name. Specifies the axis to be removed from the current channel.

<SAI> System axis index. Same effect as <SAN>.

<KAN> Channel axis name. Same effect as <SAN>.

☞ Block preparation is not stopped when removing an axis.

Invalid axis names generate an error message.

If an axis to be removed is defined in the system but no longer present in the current channel, no error message is displayed.

Axis positions in the same block always have to be programmed after **RemAxis(...)**.

Axes linked in an active axis transformation cannot be removed from the channel.

Note that after the removal of an axis participating in the WCS spanning, it is not possible to retain the current workpiece coordinate system. This can affect the following functions:

- Placements
- Active plane
- Tool corrections

	Documentation	
Documentation:	MTX Programming Manual	Apply axis (WAX)

Example:

1 - Axis is temporarily given to a second channel

At a given point in time t1, machining is performed in the first channel with the channel axes X, Y and Z, and in the second channel with the axes U and B.

In the first channel, the ZO table `ZO1.zot` and in the second channel the table `ZO2.zot` with the columns U and B are effective.

In block N2220, the system waits until the YP-axis is enabled. This takes place at the active point in time of the block N1210. From then on, machining is continued in the second channel. YP is to be addressed as second channel axis; the corresponding column of the ZO table `ZO2.zot` takes effect.

At the point in time t2, axis transfer is completed. YP is now displayed in the axis display of the second channel as second channel axis.

In block N1310, the YP-axis is integrated again into the first channel. Through programming, it has to be ensured that the second channel has enabled the YP-axis at this point. Otherwise, a runtime error is issued.

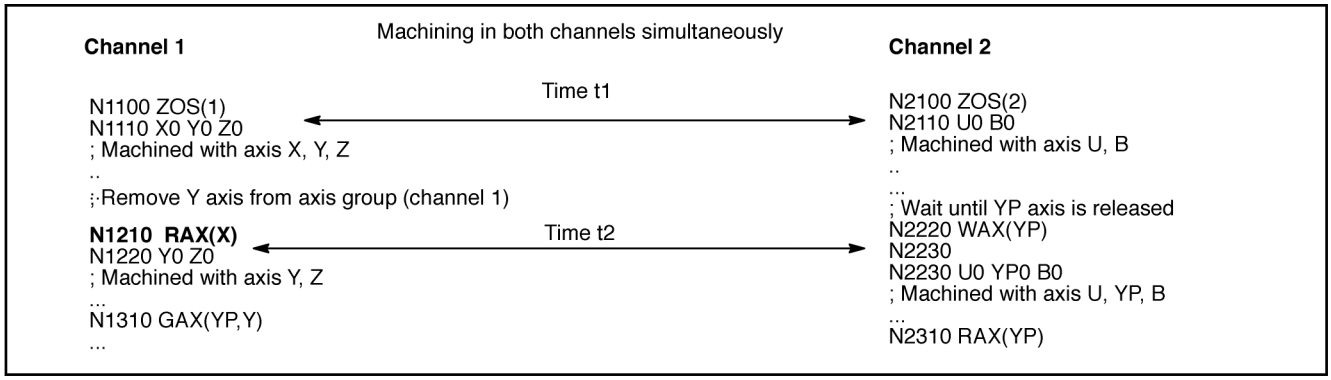


Fig. 13-4: Example 1 - The axis is temporarily borrowed to a second channel

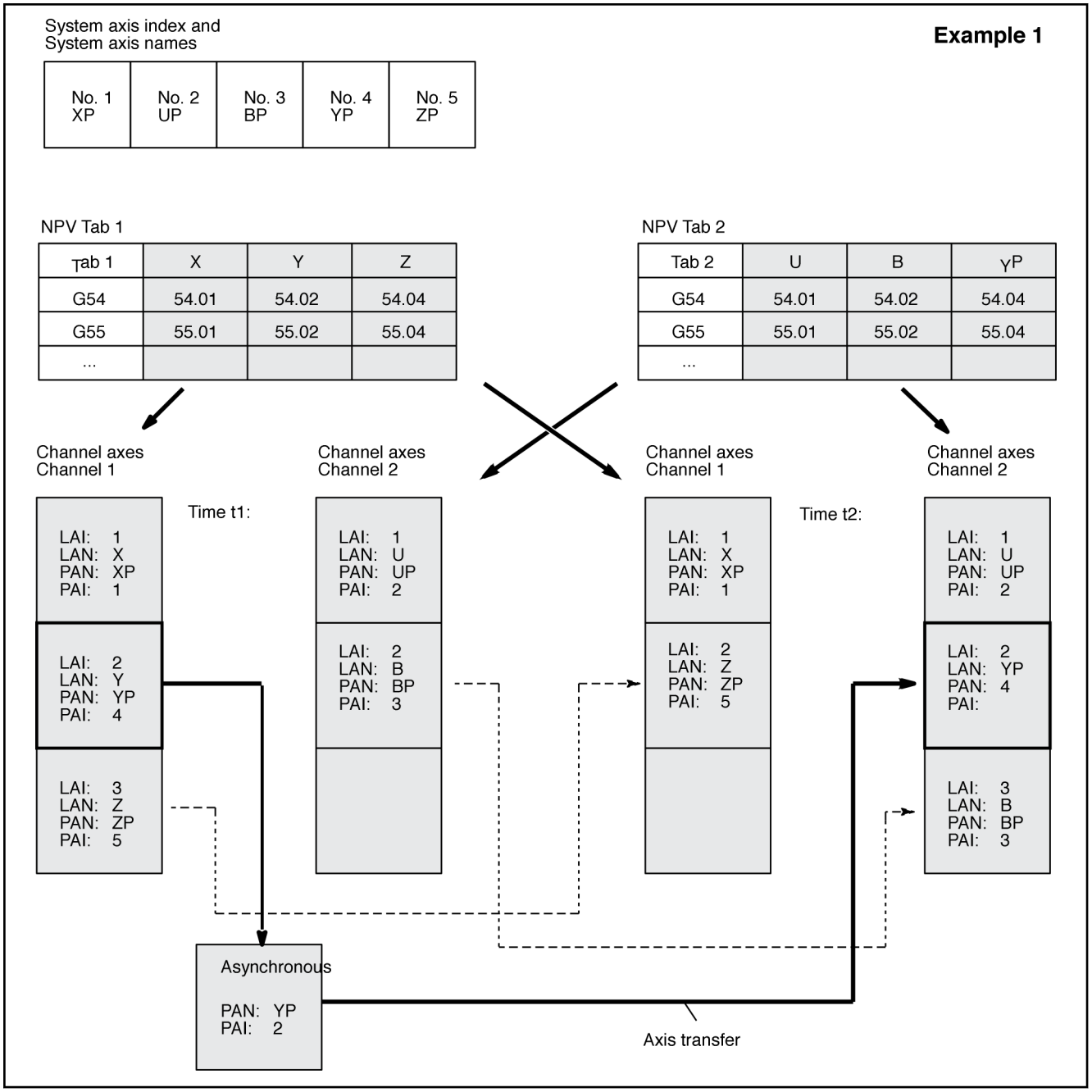


Fig. 13-5: Example 1

Example:

2 - Axis transfer with two real machining channels

At the given point in time t1, machining is performed in the first channel with the channel axes X, Y and Z, and in the second channel with the axes U, V and W. In the first channel, the ZO table `ZO1.zot` and in the second channel, the table `ZO2.zot` is effective.

In the blocks N1210 and N2210, the y-axis and the v-axes are enabled. Both axes participated in the plane selection. Thus, it was switched to G16 in both the channels.

In block N1220, the ZP-axis is provided with the new channel axis name Y.

In the blocks N1230 and N2220, the system waits until the requested axes are enabled.

In block N1240, the axis significance is redefined and the plane is set again. In the axis display of the first channel, X, Y and Z is displayed, Y being after the axis ZP and Z being after the axis VP.

The sequence of the axis display can be specified in the machine parameters.

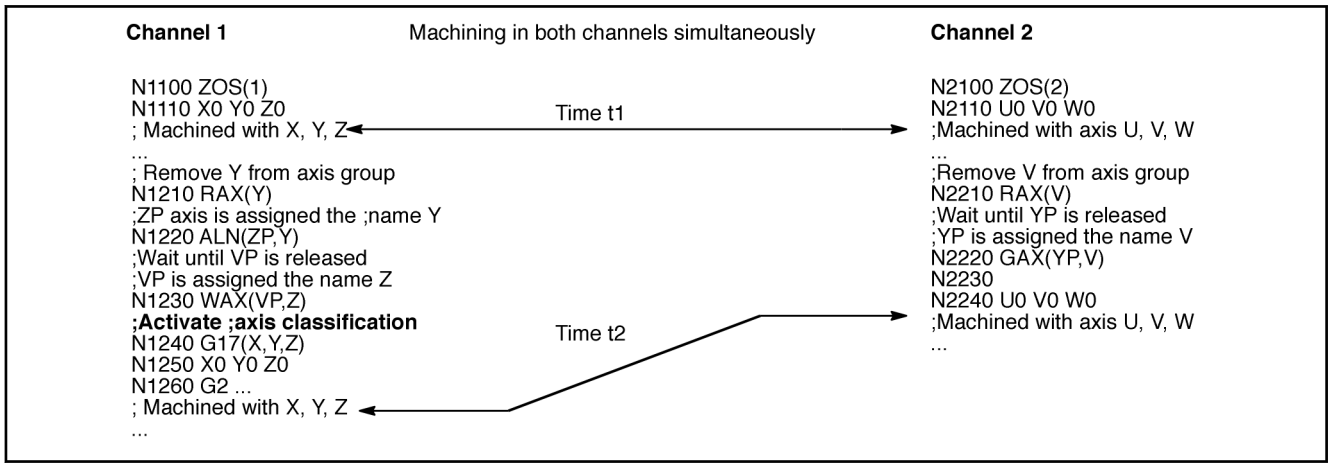


Fig. 13-6: Example 2 - Axis transfer with two real machining channels

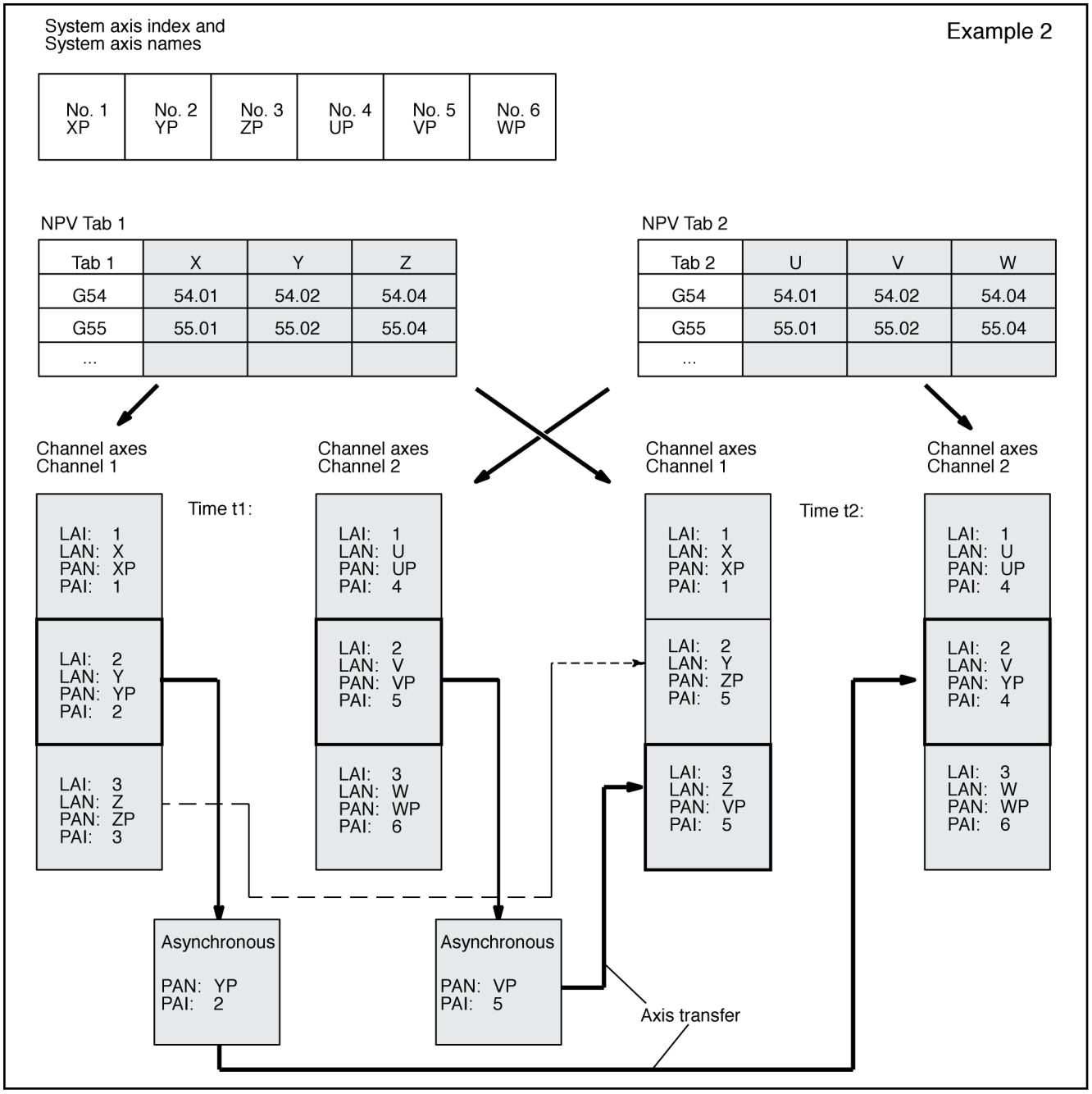


Fig. 13-7: Example 2

Example:

3 - Exchange of two axes of the same significance in one channel

At the point of time t1, machining is executed with the channel axes X, Y and Z. The system axis X1 is behind the x-axis.

In block N1210, the X1-axis is enabled and the X2-axis with the name X is integrated in block N1220.

From point of time t2, it can be machined with the X2-axis. For this axis, the ZO-values from the column X2 apply.

The axis display does not change.

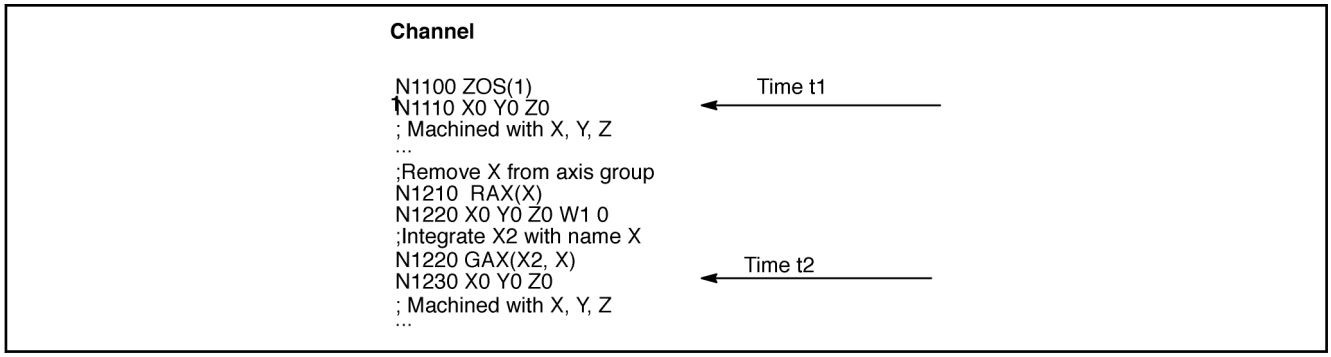


Fig. 13-8: Example 3 - Exchange of two axes of the same significance in one channel

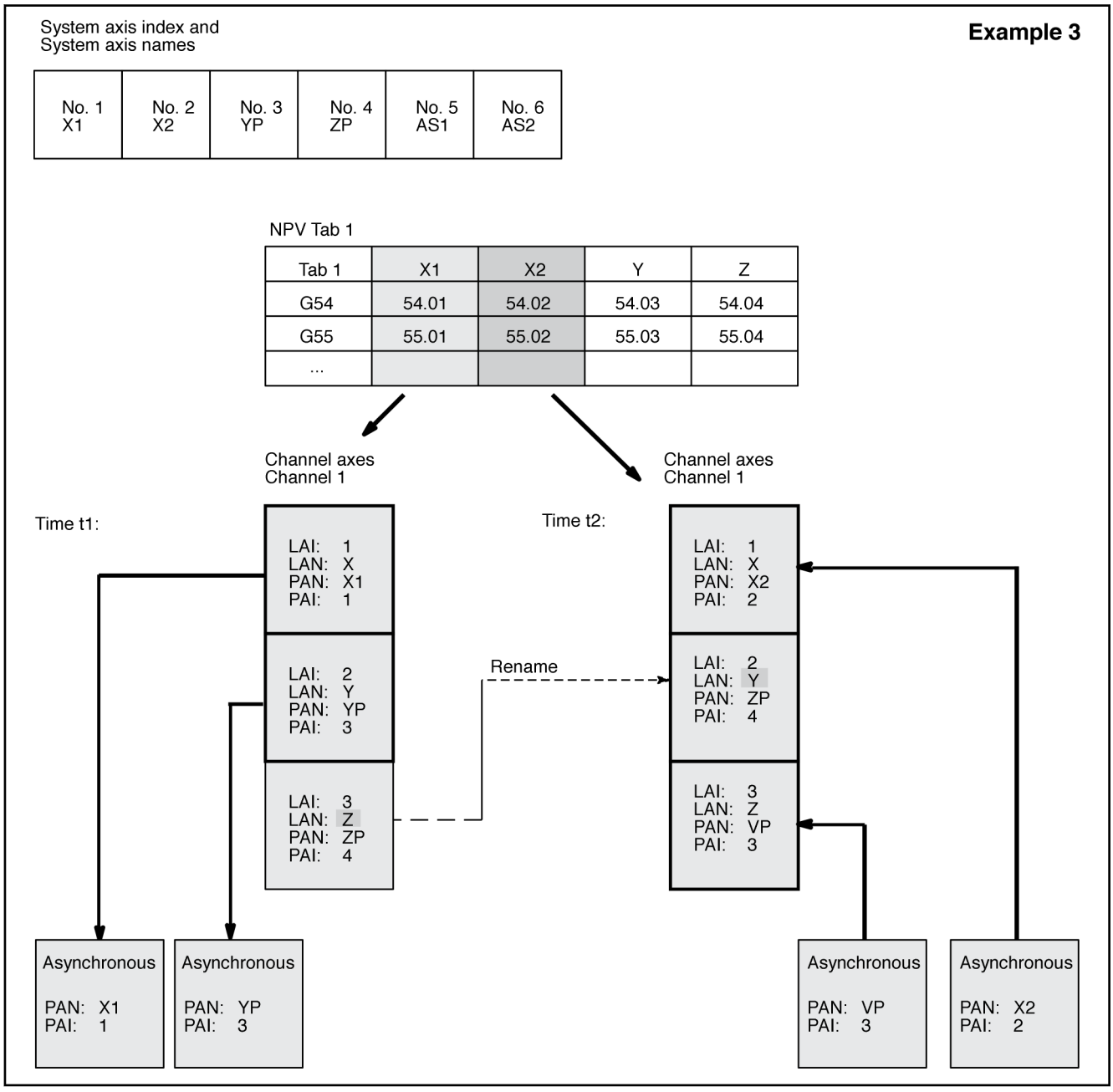


Fig. 13-9: Example 3

Example:

4 - More than eight synchronous axes in one channel

At the point of time t1, it is machined with the channel axes X, Y, Z, W1, W2, W3, W4 and W5.

In block N1210, the axes W2, W3, W4 and W5 are enabled and the system axes SY6, SY7, SY8 and SY9 with the names W2, W3, W4 and W5 are applied to the block N1230.

From now, the ZO values of the columns SY6 to SY9 become effective.

From the point of time t2, it can be machined with the SY6 to SY9 axes.

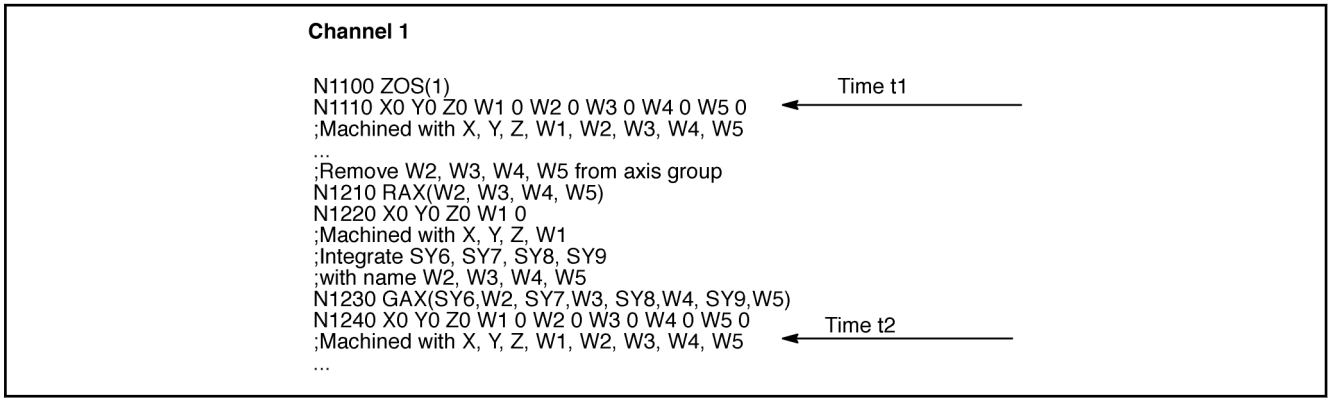


Fig. 13-10: Example 4 - More than eight synchronous axes in one channel

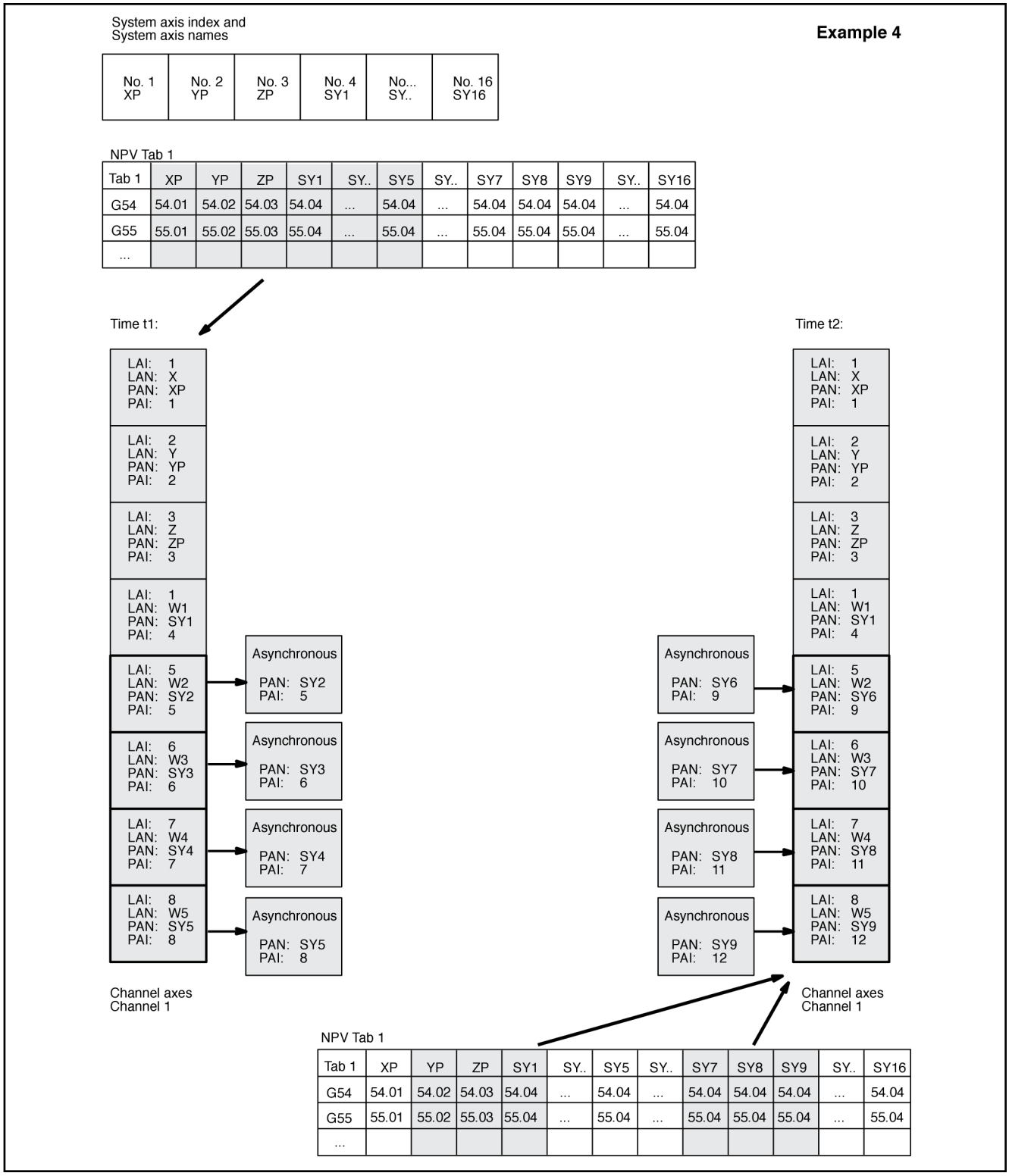


Fig. 13-11: Example 4

Disabling: **IW Engineering/Programming:**

DefAxis

Short form **DAX**

Different programming methods can be distinguished:

- Program **"DefAxis" (DAX)** directly into the part program or use the manual data input:

  If the axis has not yet been enabled, programming causes a runtime error.

  Remedy: Write DAX in the Init string if the function is to be executed only at reset (keyword #Reset:) or complete reset (keyword #SysRes:).

- With **Reset,** the current axis assignment is retained in the channel. However, the assigned axes are enabled, i.e. provided to the remaining system. The axes can be requested by other channels. In case of a program selection or specification via MDI, the axes have to have been enabled by the other channels. Otherwise, an error message is displayed.

  Switch back to the axis assignment of the machine parameter with **total reset** (system reset).

- **End of program (M30):**

  With NC functions, third-party axes can be enabled or the axis assignment can be applied from the machine parameters. In the latter case, error messages can occur if axes were assigned to other channels.

- **Channel reset:**

  Analog end of program (M30). Error messages can be issued if axes are assigned to other channels.

14 Couplings

14.1 Gantry axes

14.1.1 Description

Function

Gantry axes are rigidly coupled by mechanical devices and are traversed in parallel using separate drives. The command values of the slave are fixedly coupled to those of the master (command value coupling). Depending on the design of the Gantry group, an inclined position, i.e. an actual value offset among the drives can result.

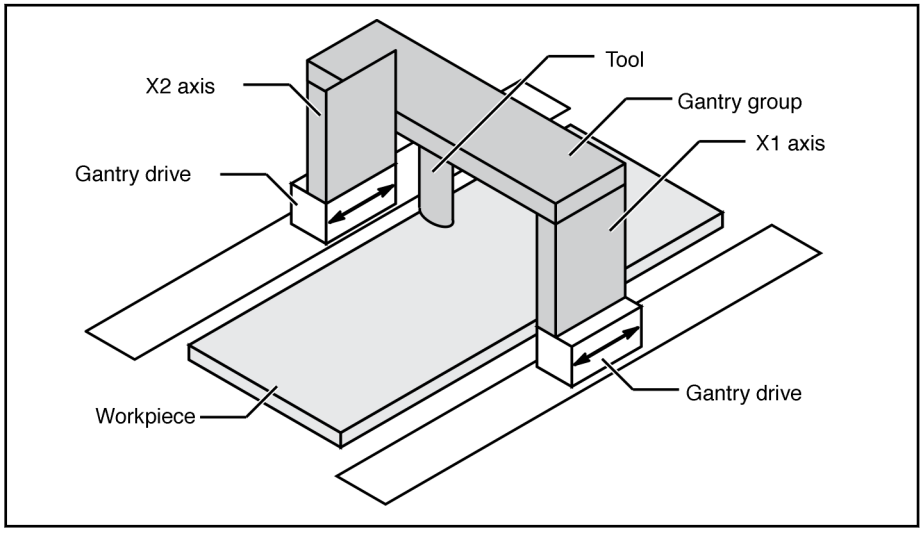


Fig. 14-1: Gantry axes

Requirements

Characteristics of the gantry group:

- Interface:

  The drives are digital drives according to Sercos.

- Drive data:

  The drive data of rigidly coupled axes have to be identical so that they act identically even in asynchronous mode (jogging, approaching reference point).

  All drive data on axis dynamics even have to be kept identical programming all gantry axes of a group. This data only must be changed when the axes are static, because switching is not detected and accordingly does not become active simultaneously (in the same Sercos cycle) in all drives.

- Command value coupling:

  Master and slave axis in a gantry group move together in one command value coupling.

- Measuring systems:

  All gantry axes require identical absolute or incremental (relative) measuring systems.

In one gantry group, the following axis type constellations are permitted:

Master:	linear	- Slave:	linear
Master:	Linear modulo	- Slave:	Linear modulo
Master:	endless	- Slave:	endless
Master:	round	- Slave:	round

For modulo axes, the modulo values of master and slave have to be identical.

The master axis has to be a synchronous axis. We recommend to set the slave axis as asynchronous axis!

Command value coupling

To ensure that master and slave move together, their command values are coupled (command $value_{slave}$ = command $value_{master}$).

If an actual value offset occurs, it can be applied as command value offset from the command values of the axes under certain conditions. In this case, the slave receives a command value that is different from the master (command value offset by the master command value).

When the drives are switched off, the command value coupling has no effect. After the activation of the drives, the command values (and thus also the command value offset) become immediately effective again. If axes drift off at the open controller, this can result in a drive error with excessive command value specification (F2037) or excessive deviation (F2028) if no other measures are taken as the slave wants to assume the command values of the master.

The control does not independently compensate for any offset of the actual position values of master and slave. A command value offset formed from the actual value offset can be active, but the control does not compensate it on its own.

Reference and measuring systems in the gantry group

During commissioning, **correctly** set references of all gantry axes are a prerequisite when determining the actual axis values.

The actual axis values are again imperative to determine the offset between the gantry axes. Thus, they are a specification for the command value offset.

The determination of the references depends on the measuring system.

Absolute and incremental (with or without distance-coding) measuring systems are used:

- For an absolute measuring system, each axis calculates its current axis position in standstill on its own. This axis position becomes the axis-related actual position value.

  Any inclined position of the group is not compensated!
- For a relative measuring system with distance-coded reference marks, the axis group moves between two reference marks. If a mark is detected, the drives automatically switch to the axis-related actual command value (each axis receives its own actual value). Any inclined position of the group is not compensated!
- For a relative measuring system without distance-coded reference marks, the axis group moves to the reference point and switches automatically at a defined position to the axis-related actual position value (each axis receives its own actual value). With gantry axes, "positioning" has to be applied (see Sercos ID S-0-0147). To evaluate the reference switch, all drives have to be connected to a reference switch!

  Any inclined position of the group is compensated when the reference point is approached!

☞ The references are determined according to the Rexroth IndraDrive drive documentation (firmware for drive controllers MPH-02, MPB-02, MPD-02).

Inclined positions

Development of inclined positions:

*An inclined position is a **actual value offset** if it is ensured that*

- the axes have been commissioned correctly,
- "Reference established" for all axes after startup
- no command value offset exists

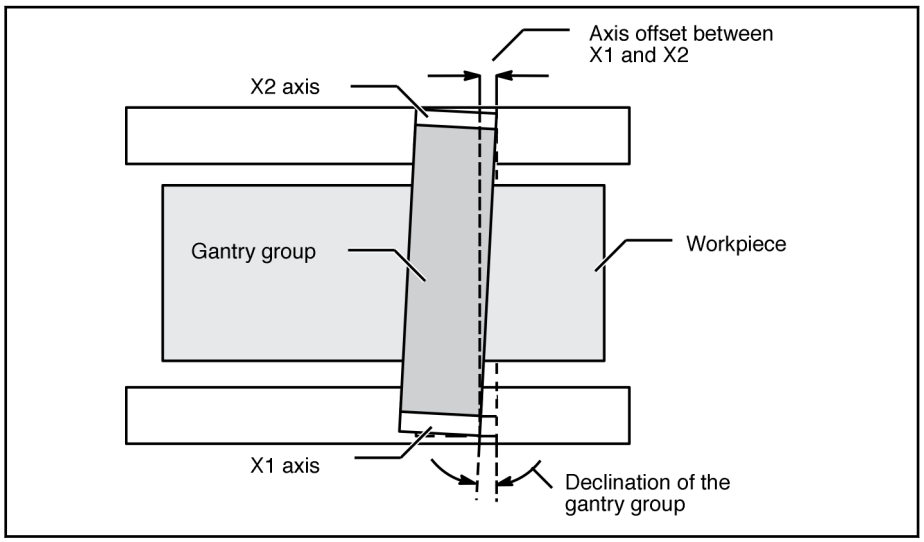


Fig. 14-2: Inclined position

***At runtime**, an inclined position arises if there is:*

- an unsymmetrical blockage
- or an unbalanced load distribution

The actual values of each individual axis follow the command values in different manners. This results in the actual value offset. But as the command values are identical for master and slave, **inclined position monitoring** is the only option for the control to take corrective measures.

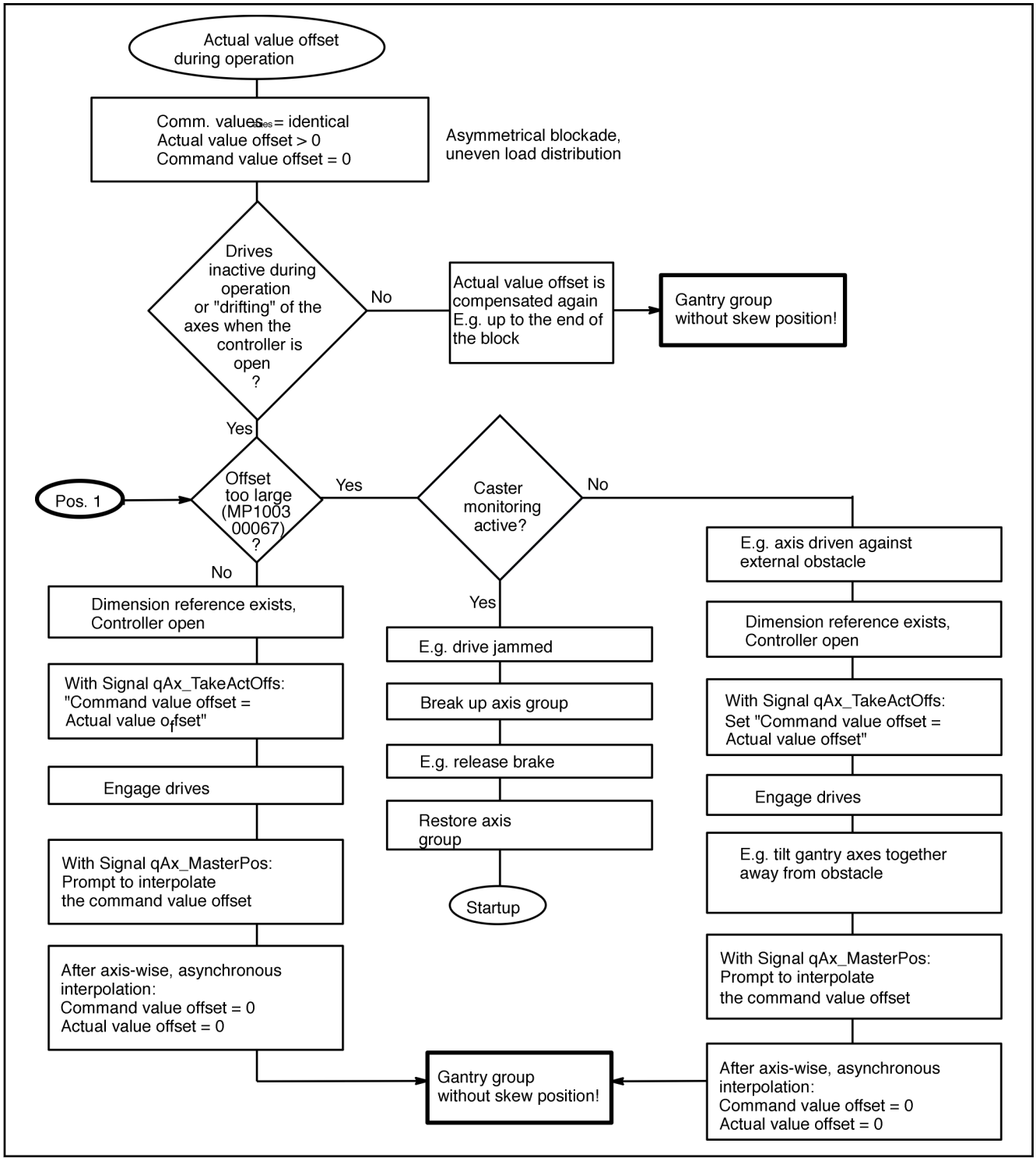


Fig. 14-3: Inclined position monitoring

*With **deactivated drives** or an **open controller**, an inclined position occurs due to:*

- mechanical shifting in a state without drive
- axis drifting at an open controller
- different braking response during deceleration

- an actual value offset which could not be compensated for during a path motion since the drives were switched off

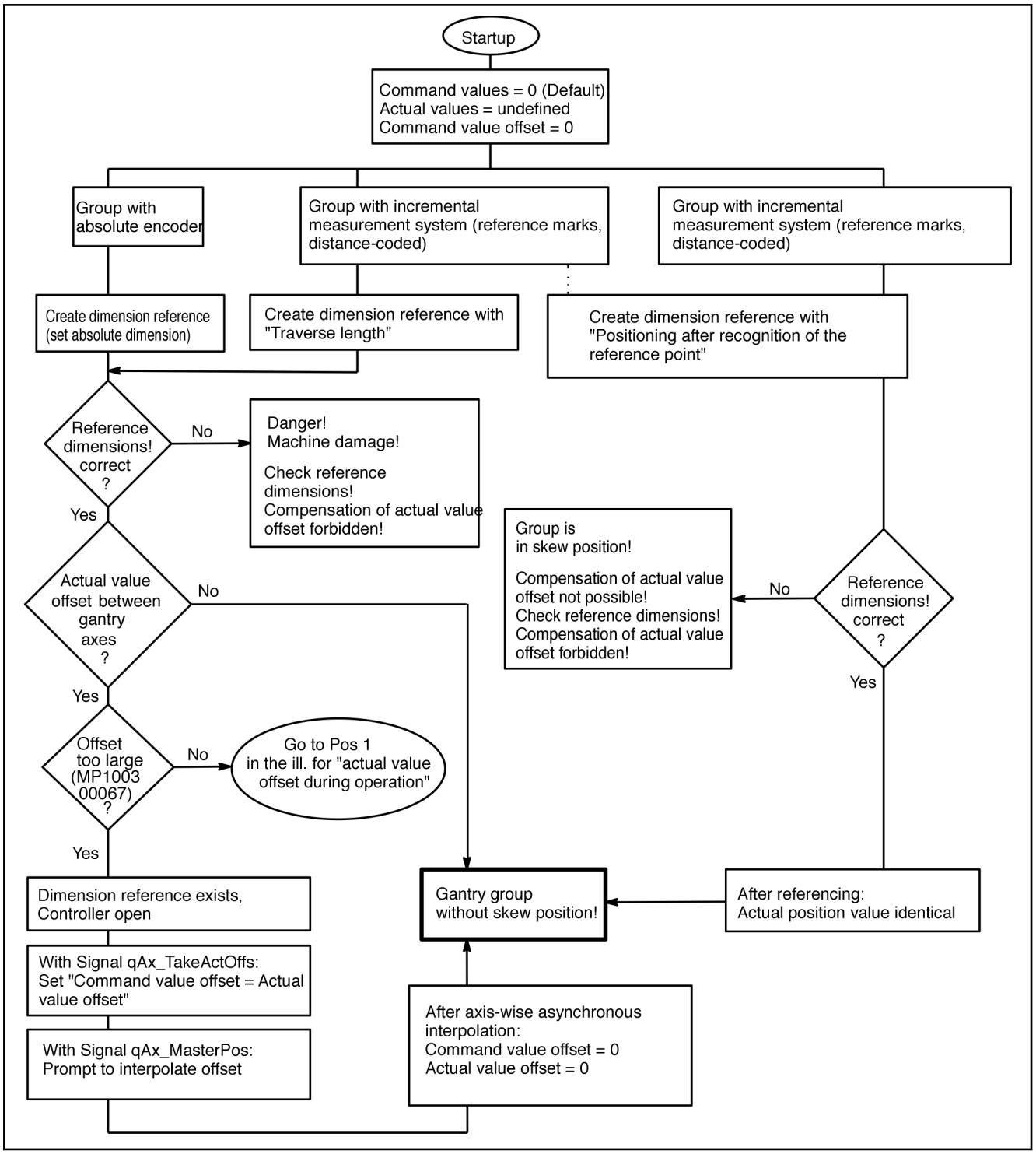


Fig. 14-4: Inclined position in case of activated drives or an open controller

Compensation of inclined positions:

- **Compensation during path motion**:

  If an inclined position results during the path motion, since the actual drive values follow their command values in different manners, the control autonomously compensates this actual value offset if the drives can follow their command values again (e.g. till block end).

☞ The drives cannot compensate the inclined position if either runtime or a Sercos error occurs or the drives are switched off before.

- **Compensation by compensation motion**:

  The function results in an **compensation of an inclined position** by an **interpolatory compensation motion** of the axes initiated by the control.

The command value offset is created from the offset of the actual values at certain defined points:

1. For drives with absolute encoders at control startup if the reference of the drives is established,
2. For drives with incremental encoders when establishing the reference or
3. For an existing reference by an interface signal (qAx_TakeActOffs) if the prerequisites are met.

If a command value offset exists, it can be compensated by interpolation.

An existing actual value offset is compensated by an asynchronous interpolatory compensation motion of an axis in the gantry group.

For the compensation motion, the following conditions have to be fulfilled:

- The axes are in reference (reference established).
- The axes are at rest. No traversing motion is active.
- The NC applied the inclined position as command value offset (iAx_CmdOffsExst = 1).
- The following error has to be in the permissible range (MP 100300060) or the axis interface signal qAx_LagErrOff "hide coupling error" has to be set.
- The command value offset has to be in the permissible range (MP 100300066, 1003 00067) (iAx_CmdOffsExceed = 0).
- If no "safe position" has been specified by the Safety engineering, no adjustment takes place.
- If no "safe velocity" has been specified by the Safety engineering, a safe velocity is used for adjustment.
- The command value offset is compensated with the compensation velocity set in "manual feed medium" (MP 1005 00004) and with the jog acceleration (MP 1010 00002). The channel override potentiometer is effective.

☞ When adding the drives again (without compensation of the actual value offset), the command values of the master become immediately effective for the slaves. In case of excessive offsets (outsides of the limits), Sercos errors may occur (F2028: excessive control deviation or F2037: excessive command value specification).

Signal sequence of the compensation motion:

- Master and slave drive have to be switched off.
- Command to apply the actual value offset as command value offset at an open controller with the axis interface signalqAx_TakeActOffs, edge 0 → 1"apply actual value offset".
- The NC creates the command value offset from the actual value offset:

  iAx_CmdOffsExst "Gantry command value offset active" is logic 1

- Add master drive and slave drive.

  The drives have to be active (iAx_DrvAct).
- Via the axis interface signal qAx_MasterPos, edge0->1 "gantry slave on master position", the compensation of the command value offset is triggered by the asynchronous interpolation.

  The motion is only executed if the drives comply with the command value specification (drive status: Bit3==TRUE).
- The control moves out the command value offset interpolatorily:

  iAx_TrvCmd "Travel command" is logic 1.

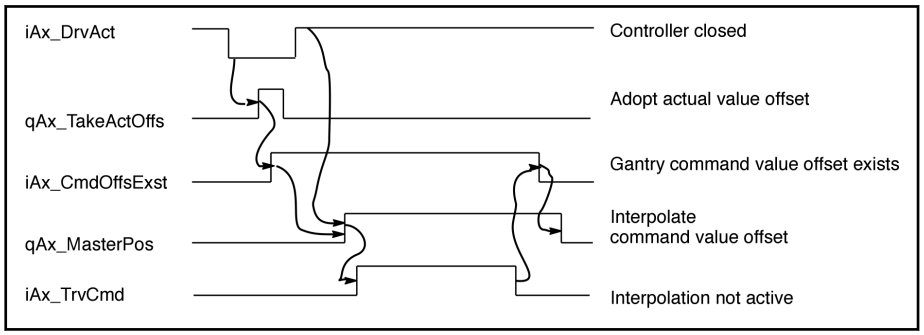


Fig. 14-5: Signal sequence of the compensation motion

Abort compensation:

The compensation motion is aborted by a falling edge "Gantry slave on master position" (qAx_MasterPos edge 1 → 0). If the compensation motion is aborted before the command value offset has been interpolated, a command value offset remains. The "gantry command value offset active" signal (iAx_CmdOffsExst) remains on logic 1.



A new commissioning is necessary after the drives, encoders or the drive software have been exchanged!

Restrictions

- An inclined position, which cannot be detected with actual position values, cannot be compensated.
- If the actual values of an axis in the gantry group do not reach the specified command values, an axis is blocked (static inclined position). The command value coupling prevents a compensation as the blockage is not detected.

  In the case, the block must be removed first, i.e. switch off the drives, release the blocked axis, and then execute "Offset compensation of all axes", or - when the axis group has been cancelled - "Referencing with offset compensation of all axes".
- If the inclined position is not compensated after startup, the axes traverses continuously with an offset.
- **No** "reference is established" for the drives. The command value offset cannot be determined (no data for the actual value offset). Thus, the inclined position is not defined. The control cannot compensate for the command value offset.
- "Reference established" for the drives, i.e. the reference measures are **correctly** applied:

  The command value offset of axes, which have drifted off while the control is inactive, can be determined and interpolatorily compensated by the control.

- "Reference established" for the drives, i.e. the reference measures are not **correctly** applied. This can be seen in the different actual position of the gantry axes. The command values of the master become effective immediately and without any offset to the slaves. This causes a Sercos error (excessive deviation or excessive command value specified).

  The control can compensate, **but is not allowed to compensate**, since the command value offset is random (inclined position based on incorrectly determined reference). The error was caused in the commissioning.

☞ An "invalid inclined position" of the gantry axes have to be mechanically cleared before control startup if gantry axes may not be traversed in this inclined position.

Relevant NC functions

All NC functions allowing for sensible programming of the gantry group are permitted.

Relevant Sercos parameters

S-0-0041	Reference velocity
S-0-0042	Reference acceleration
S-0-0115	Encoder type 2
S-0-0116/117	Resolution encoder 1/2
S-0-0052/54	Reference dimension 1/2
S-0-0147	Reference parameters
S-0-0150/151	"Reference dimension offset 1/2" in motor encoder/external encoder. If the reference dimension offset (Sercos parameter S-x-0150 in motor encoder, S-x-0151 for ext. encoder) of the master axis is changed, the reference dimension offset has to be adjusted for all slaves with the result that the difference of the reference dimension offsets (to the master!) remains constant.
S-0-0051/53	Actual position value of the encoder 1/2: Indicates the current position of the motor encoder/ext. encoder.
S-0-0177/178	"Absolute dimension offset" for motor encoders Distance from the machine zero point to the zero point of the motor encoder with absolute measurement.
S-0-0400	Switching state of the reference switch connected to the control device.
S-0-0403	Status of the actual position value
S-0-0189	Following error for the following error monitoring if in the cyclic telegram.
S-0-0084	Actual torque value for the torque monitoring.
S-0-1050.1.6	Inclusion of the actual torque value S-0-0084 and of the lag distance S-0-0189 into the cyclic telegram in the Sercos files (p2*.scs) for the respective gantry axes.

Tab. 14-1: Relevant Sercos parameters

☞ The IndraDrive drives with Sercos interface are suitable to commission the gantry axes. When drives (with Sercos interface) of other vendors are used, check whether the functionality is guaranteed and which parameter number is assigned to it!

Relevant IF signals

Channel interface signals	
qCh_OpModeSel_00 to qCh_OpModeSel_03	
	Select operation mode (Here: Approach reference point in case of incremental encoders with/without distance-coding measuring system)
qCh_OpModePlc	Operation mode specified by PLC (Here: Approach reference point in case of incremental encoders with/without distance-coding measuring system)
Axis interface signals	
iAx_DrvAct	Drive active (controller closed)
iAx_MasterIndex0	Index of master axis bit 0 ...
iAx_MasterIndex6	Index of master bit 6
qAx_TakeActOffs	Apply actual value offset For this slave, the actual value offset is applied as command value offset.
iAx_CmdOffsExst	Command value offset exists for this slave (logic1)
iAx_CmdOffsExceed	Command value offset to be compensated is exceeded for this slave (logic1).
qAx_MasterPos	Gantry on master position The selected slave receives the same command value as the master. The signal causes a change in the slave position if the command values of master and slave are identical. If the signal is reset before the new command value is active, a command value offset remains (only with interpolatory compensation).
iAx_TrvCmd	Travel command (interpolatory traversing motion is active)
iAx_CoupleLag	Lagging error Each axis of the gantry group receives this output signal. It is updated constantly, i.e. it will not remain static when an error has occurred and has been cancelled again! iAx_CoupleLag does not depend on qAx_LagErrOff

iAx_TrqExceed	Standstill error Each axis of the gantry group receives this output signal. It is updated constantly, i.e. it will not remain static when an error has occurred and has been cancelled again! In the gantry group, the iAx_TrqExceed signal is only relevant for the master axis. The signals for the slave axes are not evaluated.	
	Low:	The total torque of all axes is in the valid range.
	High:	The total torque of all axes has exceeded the permitted range.
	(iAx_TrqExceed does not depend on qAx_TrqErrOff!)	
qAx_LagErrOff	Hide coupling error Each axis of the gantry group receives this input signal, but it is only relevant for the master axis. If this signal is set for the master axis, the error analysis in the NC is suppressed if the lag is exceeded.	
	Low:	In case of error, error handling/error message by the NC.
	High:	In case of error, no error handling/error message by the NC.
qAx_TrqErrOff	Hide standstill error Each axis of the gantry group receives this input signal, but it is only relevant for the master axis. If this signal is set for the master axis, the error analysis in the NC is suppressed if the standstill torque is exceeded.	
	Low:	In case of error, error handling/error message by the NC.
	High:	In case of error, no error handling/error message by the NC.
qAx_DrvLock	Feed inhibit Before referencing, the PLC has to lock all gantry axes of a group.	
qAx_JogPlus qAx_JogMinus	Manual +/- In the "Approach to reference point" mode, the PLC sets the jog signals for all gantry axes.	

Tab. 14-2: Relevant IF signals

☞ *About the interface signals:*

- The PLC has to process the axis interface for the master axis as well as for the gantry slave axes (the interface of a gantry slave axis acts like the interface of a slave axis coupled to a G-function by programming).
- *For this reason, the following drive-based interface signals (NC inputs) always have to be kept identical within a coupling group:*
  - Drive on
  - Feed inhibit

  Has to be ensured by the PLC!

Relevant machine parameters (MP)

1003 00060	Max. following error for coupled axes *Defines for each **axis**:* • whether following error monitoring with active axis coupling is to be switched on or off. • The maximum allowed following error of a slave axis.
1003 00061	Index of the gantry master axis Defines whether the axis in a gantry group is a master or a slave axis.
1003 00062	Max. total standstill torque of a gantry group *Defines for each **master axis** of a gantry group* • whether the standstill torque monitoring of the group is to be enabled • The monitoring limit value
1003 00063	Synchronous run monitoring when referencing gantry The parameter defines whether monitoring takes place when referencing gantry axes if all axes in a monitoring window reached their reference point. The parameter has to be entered only at the master of the group.
1003 00064	Window synchronous run monitoring when "establishing gantry reference": The parameter indicates the size of the monitoring window in ms. In each case, the next multiple of the interpolator cycle becomes effective.
1003 00065	Reference switch monitoring during "Establish gantry reference" The parameter specifies whether it is monitored when "establishing the gantry reference" for gantry axes whether an axis is on the reference switch when "Establish reference" is started. The parameter has to be entered only at the master of the group.

1003 00066	Compensate gantry command value offset The parameter defines whether the command value offset between master axis and slave axis of a gantry group may be compensated by setting the corresponding interface signal and where the maximum distance to be compensated is specified. Applies for each slave axis.
1003 00067	Compensate gantry command value offset: Upper limit Defines the maximum distance up to which the command value offset between master axis and slave axis may be compensated by setting the corresponding interface signal. Applies for each slave axis.

Tab. 14-3: Relevant machine parameters

Special features

- Before control startup, the gantry axes have to be mechanically torque free (i.e. no terminals). Thus, the axes can be traversed in parallel after control and Sercos startup.
- After each exchange of motors, measuring systems and/or drive software, commission the gantry axes again.
- The vendor-specific Sercos parameters (P-x-xxxx) specified for commissioning are only relevant for IndraDrive drives. For other drives, use equivalent parameters. The drive vendor has to ensure the functionality.

Monitoring functions

Monitoring functions for the gantry group:

- **Commissioning monitoring:**

  Through a time window (MP), the referencing of all involved axes can be monitored. After the first axis has completed its referencing, the time window starts. In the preset time, all other axes have to have completed their "referencing". Otherwise, the referencing is aborted with an error.

  The relevant MP should be disabled after commissioning, since the compensation of an inclined position during referencing is suppressed by an error abort.

☞ For initial commissioning, the ident number "Status actual position values" (S-0-0403) should be comprised in the cyclic AT. This ensures that the reaction time of the control can be reduced to a minimum in case of error.

- **Monitoring the reference switch:**

  When "establish reference" is started, an axis must not be on the reference switch, as this causes a change in direction when "establishing a reference". The gantry axes might move in opposite directions.

  The monitoring function prevents the start of the "establish reference" with a runtime error if one of the gantry axes is on the reference switch (switching state "activated" of the reference switch in the parameter S-0-0400 in the cyclic AT).

- **Monitoring of following error**:

  At commissioning and at runtime, MP 100300060 (dimensional tolerance = max. following error for coupled axes) monitors the following error.

The control uses the lagging information by master and slave from the cyclic Sercos parameter S-0-0189. If S-0-0189 is not in the cyclic Sercos telegram, the following error is computed from the actual position values of the gantry axes. When the maximum permissible following error between master and slave is exceeded, the group is decelerated to standstill and a coupling error is output.

Error analysis:

If an invalid inclined position occurs, the interface signal iAx_CoupleLag "lag error" is set for the axis/axes. If it is fallen below the following error, the signal is cancelled again. Setting the axis signal qAx_LagErrOff for the master axis can suppress the error state in the NC. In this case, coupling lag monitoring has to be programmed in the PLC.

☞ The coupling is also not opened in case of error.

- **Torque/current monitoring:**

  Torque/current monitoring (standstill monitoring) only functions in **static state**, i.e. when all gantry axes are "in position" and no traveling command of the NC is activated.

  The active torques of the axes are determined from the Sercos telegram.

  The torque monitoring responds if the **total** of the torques of all drives of a gantry group exceeds the **maximum total standstill torque** specified in MP 003 00062. The total of the torques in "standstill" is an indication of the quality for the mechanical strain. The quantities have to be specified by the machine vendor. They are determined using test series.

☞ It is not reasonable to analyze the torque characteristics of the drives during the motion, as drive torques can vary to much.

Sercos setting:

To monitor the torque, the actual torque (S-0-0084) of the gantry drives has to be transmitted to the cyclic Sercos telegram (entry in the Sercos file "p2<xxx>.scs").

Error analysis:

If the torque/current monitoring function indicates an extraordinary inclined position, the **error cause** has to be eliminated first before the inclined position is cleared by referencing. The monitoring is triggered again if referencing is tried in an "incorrect" state.

*If the total torque is **exceeded**:*

- the iAx_TrqExceed axis signal is set for the master axis. If the actual torque falls below this total, it is reset.
- the NC issues an error message that can be deleted with reset.

  This error message can be suppressed by setting the axis signal qAx_TrqErrOff. The output signal iAx_TrqExceed does not depend on the qAx_TrqErrOff!

  Display of the gantry slave axis position:

A position for a gantry slave axis can only be displayed based on actual positions. Not all command values generated by the NC appear in the coordinate systems. Therefore, displaying slave positions is not reasonable. It should be hidden.

Recommendation:

A gantry axis should be specified an asynchronous axis.

14.1.2 Handling instruction: Applying gantry axis group

Applying: Gantry commissioning and "establishing reference".

Prerequisites:

For all axes of the gantry group, references are determined. The process for "Establishing reference" based on these references and performed during initial commissioning should be executed at low CO, low velocity, low acceleration or deceleration, as well as torque monitoring.

⚠ CAUTION **Risk of danger to machine or personal injury by incorrectly determined references.**

Perform the initial commissioning with reasonable axis parameters to avoid major damage!

The gantry group has to have been mechanically aligned in a torque-free state:

With **heavy portal gantry groups**, individual gantry axes have to be aligned in parallel to each other either mechanically or by jogging. It is required that the gantry coupling of the axes in MP 1003 00061 has to be cancelled (all parameter values =0) so that master and slave axes can be jogged **individually** after startup.

Select the following procedure:

1. Start NC and drives.
2. Set MP 1003 00061 so that no axis belongs in a gantry group.
3. Restart NC and drives.
4. Jog each axis to a position where the axes stand parallel to each other and where subsequent referencing can compensate for an inclined position.
5. In MP 1003 00061, define again to which group the gantry axes belong.

With **relatively inflexible gantry groups**, it can be assumed that no distortion forces are present.

Define gantry axes:

MP 1003 00061 index of the gantry master axis. Defines for **each** axis whether it belongs to a gantry group. Enter the axis index (= the axis number) of the master axis for all axes (incl. the master axis) belonging to a gantry group.

Establish reference (referencing):

Absolute and incremental (with or without distance-coding) measuring systems are used. References are determined according to the drive documentation (for IndraDrive).

☞ For the commissioning, the reference determination for the individual drives is described in detail in the documentation "Rexroth IndraDrive Firmware for Drive Controllers MPH-02, MPB-02, MPD-02".

☞ To determine the reference during the commissioning, the axes may not be in an inclined position.

Reference with absolute measuring system:

Commissioning :

If the axes are aligned, the reference of all gantry axes is set by setting identical actual position values ("set absolute dimension"). This is to be performed without drive enable. Since the command values of the control become effective at drive enable, a command value jump can result if the actual values differed before setting the "absolute dimension". This can only be avoided with a control startup.

Establish reference:

Upon control or Sercos, the reference is established, i.e. the actual position values of each axis are known **immediately**. For each drive, the "reference is established" if the reference dimension of the axes have been applied correctly (via "set absolute dimension" at the commissioning).

Compensate **"offset"**

An actual value offset that resulted at runtime or in switch-off state is only compensated by setting the interface signal **qAx_TakeActOffs** "Apply actual value offset". Set the **qAx_MasterPos** "Gantry slave on master position" interface signal to start the subsequent "interpolation" of the offset.

If the actual position values of the gantry axes differ in the gantry group after the commissioning after control startup (e.g. drifting at the open controller), the actual value offset corresponds to an inclined position between the gantry axes.

Apply the offset as command value offset using the signal qAx_TakeActOffs edge 0 → °1 "Apply actual value offset".

Use the signal qAx_MasterPos, edge 0->1 ""Gantry slave to master position" to have the control compensate for the actual value offset by asynchronous interpolation.

☞ The gantry group has to have been mechanically aligned in a torque-free state.

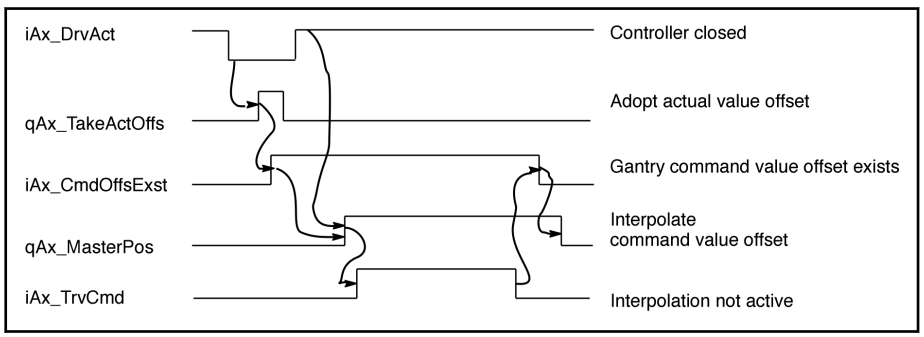


Fig. 14-6: Signal flow diagram

After an offset compensation, the actual position values of all gantry axes are identical again.

Restrictions:

- If no reference point is available, no "reference is established" for the drives. The command value offset is not defined and the control is not able to compensate the inclined position.
- If the "reference is established" for the drives, but the determined absolute dimension is incorrect, the command value offset is not defined. Any attempt to compensate the actual value offset might cause machine damage.

NOTICE **Potential machine damage due to an incorrect "setting of the absolute dimension"!**

Repeat commissioning!

Reference for incremental measuring systems:

Only drives with incremental measuring systems may be referenced.

Referencing may not started directly via a Sercos command (e.g. via DriveTop).

Referencing procedure:

- Before starting "establish reference", the PLC has to lock all gantry axes of a group via the **qAx_DrvLock** "Feed inhibit" interface signal. **(qAx_DrvLock = 1)**.
- Then, the PLC sets the "Approach to reference point" mode and sets the jog signals for all gantry axes at the interface.
- If there are travel commands present at the interface for all gantry axes, the PLC removes the "Feed inhibit" signal.
- The drives approach the reference points now.

  Attention:

  The signal sequence is obligatory.

Reference for a distance-coded incremental measuring system:

When distance-coded incremental measuring systems are used, the actual position values are **not** known after the control or Sercos startup. There is only a valid command value after the "reference has been established".

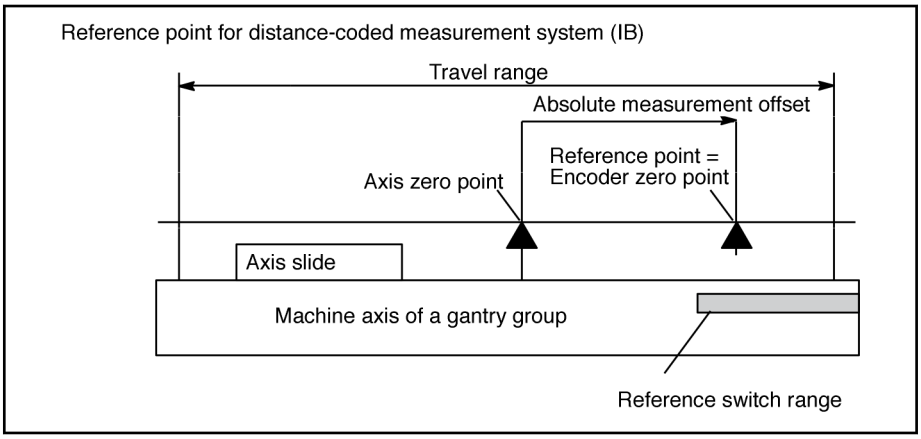


Fig. 14-7: Reference for a distance-coded incremental measuring system

Commissioning :

At commissioning, the parameter S-0-147 **has to** be set to **"Go distance"** for each axis (bit 7=0, bit 8=1) so that the drives can stand any position after the "reference has been established". The reference is established by traveling across two reference marks. After completing "Establish reference", the actual position value of the zero point of the encoder is equal to the **absolute dimension offset** specified in S-0-177 or S-0-178. The machine-specific zero point of the axis has to be adjusted using the specification in S-0-177 or S-0-178. If a reference is established later on by "Positioning after reference point detection", the reference dimension (S-0-52/S-0-54) has to be entered.

Establish reference:

After control or Sercos startup, the actual position values - and thus a potential actual value offset - is determined by "Establish reference" via "Go distance" or "Positioning after reference point detection".

☞ Before starting "Establish reference", the PLC has to lock all gantry axes of a group via the "Feed inhibit" interface signal.

To establish reference with **"Go distance"**, the axes travel a short distance over 2 distance-coded reference marks to determine the actual position value (for individual drives, "Reference established" after correct confirmation of the reference dimensions of the axes).

If the actual position values of the gantry group axes differ from each other after "Establish reference", the actual position offset corresponds to an inclined position between the gantry axes. The offset can now be applied as command value offset by the signal qAx_TakeActOffs edge 0 → 1 "Apply actual value offset". Use the signal qAx_MasterPos, edge 0 → 1 "Gantry slave to master position" to compensate the actual value offset by an asynchronous interpolation.

After the offset compensation, the actual position values of all gantry axes are identical.

If the reference is established by "Positioning after reference point detection", the reference dimension (S-0-52/S-0-54) has to be entered correctly. After a correct commissioning, any inclined position which had been present before "establishing the reference" is canceled with the "Establish reference" process.

Example:

Establish Reference in Gantry Axes with Distance-Coded Incremental Measuring System:

1. Establish reference of the gantry axes during commissioning :

Prerequisite:

- When setting the reference approach parameters of the gantry axes (S-x-0041:

  Velocity and S-x-0042 acceleration) have to be specified to ensure that the mechanical system is not damaged while "establishing the reference" in case of errors during the commissioning (e.g. incorrect offset ranges). The movement can for example be aborted at any time if necessary by suitable monitoring mechanisms (torque/current monitoring).
- Reference parameter: S-0-0147 bit 7 = 0 and bit 8 = 1.

This setting has to be kept until the correct reference displacement ranges have been entered into the Sercos files and have been applied from the drives by a Sercos or control startup.

- Before starting "establish reference", the PLC has to lock all gantry axes of a group via the **"Feed inhibit"** interface signal.

  Then, the PLC sets the "approach to reference point" mode and the jog signals at the interface for all gantry axes.

  If there are travel commands present at the interface for all gantry axes, the PLC removes the "Feed inhibit" signal. Subsequently, the drives approach the reference points.

☞ The signal sequence is obligatory.

The **reference offset** is determined for each of the master and slave(s). As the encoders are never exactly mounted (commissioning distance), this commissioning distance has to be considered as well for the reference offsets of the drives.

"Drive-controlled referencing":

- The "drive-controlled referencing" command is set and enabled after the master has assigned real-time bits to the required control and status signals via the service channel.
- The drives changes to the internal position control, deletes the actual positions values (S-0-0403) and ignores the specified cyclic command value specification.
- Considering the starting direction specified in the referencing parameter S-0-0147, the drive accelerates to the referencing velocity S-0-0041 with the referencing acceleration S-0-0042.
- The drive decelerates to standstill with the value from S-0-0042. At the same time, it calculates the actual position value and sets the actual position value status (S-0-0403).

  If necessary, the axis positions have to be measured with respect to the machine zero point.

2. Set Sercos parameter

- Enter the calculated **referencing dimension offsets** of the axes into the Sercos file **p3*.scs**.
- After "Establish referencing" has been completed during commissioning, the Sercos **referencing parameter** to "approach the reference point" has to be set again (S-0-0147 bit 7 = 1) so that the gantry group is aligned after referencing.
- The parameter P-0-0504 **cam position state** (reference point switch) is either saved via the command "save main memory" (S-0-0264) or it has to be set by the commissioning engineer.

☞ For the Sercos files **p3*.scs** of all gantry axes, the following applies:

Except for the reference offset S-x-0150, S-x-0151, S-0-0177, S-0-0178 (depending on the encoder type), all values have to be identical.

---

3. Apply parameters with a new startup.

☞ **If the reference dimension offsets of the axes have not been entered and applied correctly, and if the Sercos parameter reference point state (P-0-0504) is not yet stored in the drive.**

approaching the reference point is not allowed (S-0-0147 bit 7 = 0)!

NOTICE **Damage to the mechanical system of the gantry group!**

The referencing parameter of the gantry axes (S-X-0041: Velocity and S-X-0042 acceleration) have to be specified to ensure that the mechanical system is not damaged while "establishing the reference" in case of errors during the commissioning (e.g. incorrect offset ranges).

Ensure that a suitable monitoring function is present (e.g. torque/current monitoring) to be able to abort the motion at any time!

NOTICE **Potential machine damage due to an incorrect reference!**

Repeat commissioning!

Reference at distance-coded incremental measuring system:

When non distance-coded incremental measuring systems are used, the actual position values are not known after control or Sercos startup. There is only a valid command value after the "reference has been established".

☞ Before starting "Establish reference", the PLC has to lock all gantry axes of a group via the "Feed inhibit" interface signal.

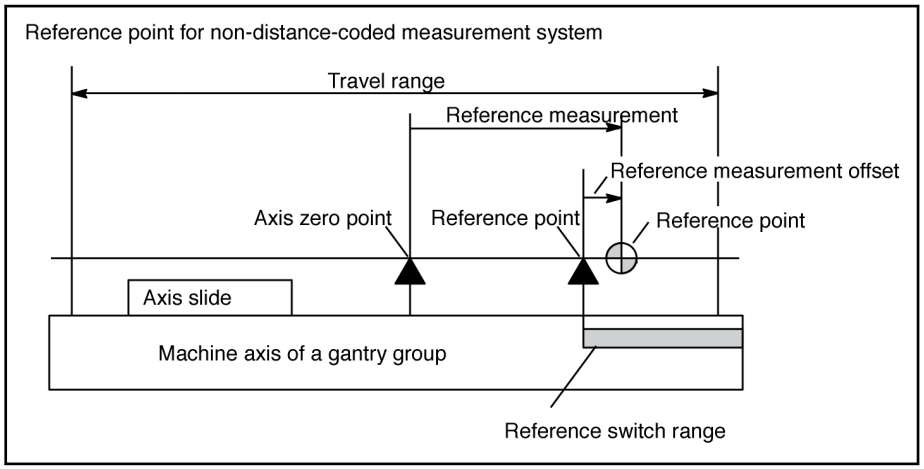


Fig. 14-8: Reference point for a non-distance-coded measuring system

Commissioning :

Two options to determine the correct reference dimensions:

- The measuring system is aligned:

  If the measuring system is **aligned** at each axis, the zero markers are identical for each axis. The reference mark offset and the reference di-

mension to the reference point can be freely selected, but have to be identical for all drives.

- The measuring system is not aligned:

  The following procedure is recommended (provided there are torque-free gantry axes):

  – If the measuring system at the axis is **not aligned**, the referencing offset has to be determined for all axes during commissioning with "search marker" P-0-14. The axes can be moved in jog mode for example until the zero marker of all gantry axes/master axis is detected. Enter the respective measure into the parameters (reference offset, see IndraDrive documentation).

  – For referencing, "positioning after the detection of the reference point" has to be set (S-0-147: Bit 7 0 = 1 Bit 8 = 0).

  – To detect the zero marker, each axis is moved in jog mode or by incremental programming until the zero marker (reference mark) of the **master axis** is detected by the reference switch (since zero markers can occur multiple time in the traversing range). The traversing distance has to be greater than the reference mark distance!

    This "marker position" A is read for each axis from the drive parameter (P-0-173). (also refer to the documentation "Rexroth IndraDrive Firmware for Drive Controllers MPH-02, MPB-02, MPD-02").

  – Now, the reference dimension is set for all axes to an identical value greater than the distance of one encoder revolution + referencing deceleration distance!

    Thus, at the end of "Establish reference" and after detecting the reference mark, the axes go to a reference position without reversing the direction.

  – The reference offset for each axis is determined by deducting the respective marker position from the reference dimension.

  – Store the data in the Sercos files and apply them at the next Sercos startup (or with the "DriveTop" tool).

Note on the **"grid jump"**:

If applicable, the **first** reference mark of all gantry axes determined during "Search marker" is not detected **first** during the subsequent referencing but only after another encoder revolution. Thus causes a mechanical distortion of the axes. Accordingly, the reference offset determined during commissioning is invalid.

☞ This error source cannot be excluded during the commissioning.

To avoid this and further major damage, activate the "commissioning monitoring" (see below). If it is activated, the system records which axis detected its reference mark first during referencing. The axis is indicated in case of error. This axis has to have the greatest reference offset. If this is not the case, all reference offset smaller than or equal to this value have to be increased by the reference mark distance (note: Encoder type S-0-0115 and encoder resolution S-0-0116, S-0-0117).

☞ If an encoder has several reference marks, the distance between two reference marks has to be constant.

Establish reference:

"Establish reference" after commissioning by approaching the reference point. "Positioning after reference point detection" has to be set. Furthermore, the reference dimension (S-0-52/S-0-54) has to be entered correctly. There is no inclined position in the reference point position.

☞ Before starting "Establish reference", the PLC has to lock all gantry axes of a group via the "Feed inhibit" interface signal.

Restrictions:

- While "establishing the reference" of the gantry axes, the zero marker must not be approached as the position of the zero marker depends on the installation.
- After commissioning, "positioning after reference point detection" has to be set to "establish the reference" and the reference dimension (S-0-52/ S-0-54) has to be entered correctly.
- After the reference point detection, stopping is **not** permitted.
- The reference offset (S-0-150 or S-0-151) of the axes has to be applied correctly. Otherwise, no unambiguous "establish reference" can be performed (no monitoring by the control). If an error occurred during commissioning (e.g. incorrect reference offset), the gantry axes have different actual position values after referencing, but this offset does **not** result in a valid command value offset.

☞ **The command values of the master become effective immediately and without any offset to the slaves. A Sercos error (excessive controller deviation or excessive command vale specification) can occur.**

- The reference point has to be behind the reference point (relevant zero marker) so that process does not start on the reference switch (S-0-0400, reference switch pressed).
- Note that the referencing direction for all gantry drives always has to be the same to ensure that the drives do not reverse their direction after the reference mark to approach the reference point has been detected. Therefore, evaluate the reference switch and a corresponding positioning is performed before the actual "establish reference" is executed.
- As IndraDrive Sercos parameters (P-x-xxxx) are used to commission the gantry axes, check other drives for the corresponding functionality and the parameter number used for activation!

Activating: **Prerequisite:**

All steps to commission the gantry drives have been executed.

Operation with gantry axes:

☞ Before starting "Establish reference", the PLC has to lock all gantry axes of a group via the "Feed inhibit" interface signal.

1. Start up the system.

After system startup, the gantry coupling is **active**.

2. Establish reference for the gantry axes
3. The compensation of an actual value offset of gantry axes at runtime or in switch-off is initiated when the qAx_TakeActOffs ("actual value offset is applied as command value offset for this slave") is set.

   The subsequent "interpolation" from the offset starts with the "qAx_MasterPos" ("gantry on master position") signal.

Disabling: The gantry group cannot be deactivated by programming.

Gantry axes are only switched off via MP 1003 00061 (index of the master gantry axis). Each axis in a gantry group gets the parameter "0". This resolves the group. After the next system startup, the gantry group does not exist anymore.

☞ With the function described above, the compensation of an inclined position of a gantry portal is not possible in decoupled state.

Optional safety measurement E-stop at gantry axes

The "E-stop" is a hardware coupling of the drive amplifiers belonging to one gantry group. The hardware coupling serves a quicker reaction to fatal errors of an axis (Y or Y2). To avoid damage to the mechanical system, both axes have to be decelerated as synchronously and fast as possible.

Drive amplifiers: Hardware wiring

Output X32:7 of the y-axis is wired to **input X32:6** of the Y2-axis. **Output X32:7** of the Y2-axis is wired to **input X32:6** of the y-axis.

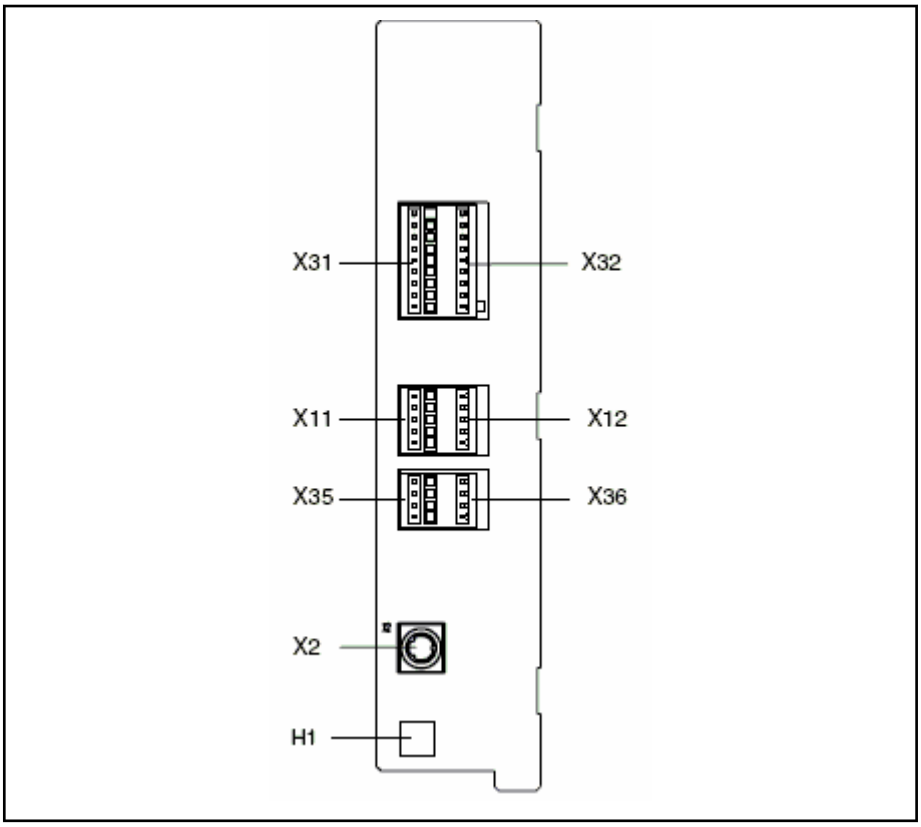


Fig. 14-9: Overview on connecting pins

IW Engineering / IW drive: Applying Sercos signals

E/A 7b has to be parameterized as output. The Sercos parameter **P-0-0115** "Device control: Status word", **Bit 0** is assigned to this output. **E/A 6b** is parameterized as input and this input is assigned the Sercos parameter **P-0-0223** "E-stop input", **bit 0**. Thus, the signal used for the BB relay is used to trigger the E-stop at the respective controller.

		Example
Example:		Digital inputs/outputs

IW Engineering / IW drive: Parameterize E-stop function

- Set the evaluation of the E-stop signal to active.
- Set the interpretation of the E-stop as warning.

		Example
Example:		E-stop signal

IW Engineering / IW drive: Parameterize error reaction

Set the error reaction to **"Immediate, optimum stopping"** and the optimum stopping to **"Velocity command value zeroing with ramp and filter"**. The deceleration **"Quick stop"** is set to the maximum axis acceleration value.

	Example	
Example:	Error reaction	Chapter heading

IW Engineering/configuration: Editing parameters

To edit the following parameter, activate the "electronic coupling" function via the parameter **coup** "Electronic coupling" in the setup (SUP):

- **MaxFollErr**
  "Maximum following error" (1003 00060)
- **MastInd**
  "Axis group index" (1003 00061)
- **MaxGrTroq**
  "Max. total of the standstill torque of the group" (1003 00062)
- **ENASynrMon**
  "Synchronous run monitoring on" (1003 00063)
- **SynchrMonWin**
  "Synchronous run monitoring window" (1003 00064)
- **ENRefSwiMon**
  "Reference switch monitoring" (1003 00065)
- **CmdOffsIpoMode**
  "Compensate gantry command value offset" (1003 00066)
- **MaxRedCmdOffs**
  "Upper limit for gantry command value offset" (1003 00067)

	Documentation	Instruction
Instruction:		Editing machine parameters
Documentation:	MTX Machine Parameters	Configure gantry coupling

IW Operation/program: Adapt Sercos files to phase 2

For each drive, the following Sercos parameters have to be set:

- S-0-0041 "Referencing velocity"
- S-0-0042 "Referencing acceleration"
- Set the encoder type in S-0-0115 "Position encoder type 2"
- S-0-0116/117 "Encoder 1/2 resolution"
- S-0-0052/54 "Reference dimension 1/2":

  Enter desired actual position value for a certain axis position (reference position) into this parameter.
- S-0-0147 "Referencing parameter"

  In this parameter, specify system, NC and drive installation information for the following commands.
  - S-0-0146, C4300 Command "NC-controlled referencing"
  - S-0-0148, C0600 Command "Drive-controlled referencing"
  - S-0-0171, C4400 "Calculate offset"
  - S-0-0172, C4500 "Offset to the reference system"

- S-0-0150/151 "Reference dimension offset 1/2"

  with motor encoder/external encoder.

  If the reference dimension offset (Sercos parameter S-x-0150 in motor encoder, S-x-0151 for ext. encoder) of the master axis is changed, the reference dimension offset has to be adjusted for all slaves with the result that the difference of the reference dimension offsets (to the master!) remains constant.

  $x_{Reference\ point} = x_{Reference\ point}$ + (S-0-0150 or S-0-0151)
- S-0-0051/53 "Actual position value of the encoder 1/2":

  Indicates the current position of the motor encoder/ext. encoder.
- S-0-0177/178 "Absolute offset" for the motor encoder.

  Distance from the machine zero point to the zero point of the motor encoder with absolute measurement.
- S-0-0400 "Reference switch".

  Switching state of the reference switch connected to the control device
- S-0-0403 "Status of the actual position value".

  In this parameter, the status messages of the actual position values of the connected encoders are displayed.
- S-0-0189 "Following error"

  For the following error monitoring if in the cyclic telegram. If an operation mode with drive-internal position control is active in the drive, the actual difference between the position command value (cf. P-0-0434 "Controller of the position command" and the actual position value

  (cf. S-0-0051/S-0-0053 "Actual position value of the encoder 1/2)" is displayed.
- S-0-0084 "Actual torque/force value"

  Enter the actual torque value for the torque monitoring into this parameter.

  Actual torque / force value= P-0-0043 "torque-forming current, actual value" * torque factor
- S-0-0016 "Configuration list of the drive telegram"

Inclusion of the actual torque value **S-0-0084** and of the following error **S-0-0189** in the cyclic telegram in the Sercos files (p2*.scs) for the respective gantry axes.

	Documentation	Instruction
Instruction:		Editing SCS parameters
Documentation:	Rexroth IndraDrive Sercos Parameter Description	Configure gantry coupling

IW Operation/IndraLogic: Set interface signals if necessary

- Channel interface:
  - **qCh_OpModeSel_00** to **qCh_OpModeSel_03** "Operation mode selection"

    (here: Approach reference point in case of incremental encoders with/without distance-coding measuring system)
  - **qCh_OpModePlc** "Operation mode specification by PLC"

    (here: Approach reference for an incremental encoder with/without distance-coding measuring system)
- Axis interface:
  - **qAx_TakeActOffs** "Apply actual value offset":

    For this slave, the actual value offset is applied as command value offset
  - **qAx_MasterPos** "Gantry on master position":

    The selected slave receives the same command value as the master. The signal does not cause any change in the slave position if the command values of master and slave are identical. If the signal is reset before the new command value is active, a command value offset remains (only with interpolatory compensation).
  - **qAx_LagErrOff** "Hide coupling error":

    Each axis of the gantry group receives this input signal, but it is only relevant for the master axis. If this signal is set for the master axis, the error analysis in the NC is suppressed if the standstill torque is exceeded.

    Low: In case of error, error handling/error message by the NC.

    High: In case of error, no error handling/error message by the NC.
  - **qAx_TrqErrOff** "Hide standstill error":

    Each axis of the gantry group receives this input signal, but it is only relevant for the master axis. If this signal is set for the master axis, the error analysis in the NC is suppressed if the standstill torque is exceeded.

    Low: In case of error, error handling/error message by the NC.

    High: In case of error, no error handling/error message by the NC.
  - **qAx_DrvLock** "Feed lock":

    Before referencing, the PLC has to lock all gantry axes of a group.
  - **qAx_JogPlus/qAx_JogMinus** "Manual +/-":

    In the "Approach to reference point" mode, the PLC sets the jog signals for all gantry axes.



The PLC has to process the axis interface for the master axis as well as for the gantry slave axes (the gantry slave interface axis acts like the slave axis interface coupled to a G-function by programming).

The following drive-based interface signals (NC inputs) have to remain identical in a coupling group (To be ensured by the PLC!):

- Drive enable
- Feed inhibit

	Documentation	Instruction
Instruction:		Editing PLC Signals
Documentation:	MTX PLC Interface	Set gantry interface signals

IW Operation/IndraLogic: Interface signals for checking

- Axis interface:
  - **iAx_DrvAct** "Drive active"(controller closed)
  - **iAx_MasterIndex0** "Master index bit 0" ... **iAx_MasterIndex6** "Master index bit 6"
  - **iAx_CmdOffsExst** "Command value offset exists" for this slave (logic1)
  - **iAx_CmdOffsExceed** "Command value offset to be compensated" exceeded for this slave (logic1)
  - **iAx_TrvCmd** "Travel command" (interpolatory traversing motion active)
  - **iAx_TrqExceed** "Standstill error":

    Each axis of the gantry group receives this output signal. It is updated constantly. That means that it does not remain static if an error has occurred and cancelled again!
  - **iAx_CoupleLag** "Tracking error": Each axis of the gantry group receives this output signal. It is updated constantly. That means that it does not remain static if an error has occurred and cancelled again! In the gantry group, the **iAx_CoupleLag** signal is only relevant for the master axis. The signals for the slave axes are not evaluated.

    Low: The total torque of all axes is in the valid range.

    High: The total torque of all axes has exceeded the valid range (**iAx_CoupleLag** is independent of **qAx_TrqErrOff**).

	Documentation	Instruction
Instruction:		Editing PLC Signals
Documentation:	MTX PLC Interface	Set gantry interface signals

14.1.3 Handling instruction: Activating Gantry Group

This handling instruction describes the activation of a gantry group.

Drive: Axis commissioning

All steps for the commissioning of the gantry drives have been executed.

All axes of the gantry group have to have identical parameters (amplification factors, acceleration/deceleration, velocity,...).

The vendor-specific Sercos parameters (P-x-xxxx) specified for commissioning are only relevant for IndraDrive drives. For other drives, use equivalent parameters. The drive vendor has to ensure the functionality.

After each exchange of motors, measuring systems and/or drive software, commission the gantry axes again.

IW Engineering/configuration: Activate coupling if necessary

- The electronic coupling function has to be activated using the parameter **coup** "electronic coupling" in the setup (SUP).
- In the parameter **MastInd** "axis group index" (1003 00061), the axis number of the master axis has to be deleted in all axes belonging to the gantry group and it has to be noted down for a possible re-activation.

	Documentation	Instruction
Instruction:		Editing machine parameters
Documentation:	MTX Machine Parameters	Deactivate the gantry coupling

Apply E-stop

The E-stop is a hardware coupling of the drive amplifiers belonging to a gantry group. The hardware coupling serves a quicker reaction to fatal errors.

		Instruction chapter 14.1.2 "Handling instruction: Applying gantry axis group" on page 498
Instruction:		Apply E-stop

Gantry group: Align

Jog each axis to a position where the axes stand parallel to each other and where subsequent referencing can compensate for an inclined position.

The gantry group has to have been mechanically aligned in a torque-free state.

With heavy portal gantry groups, individual gantry axes have to be aligned in parallel to each other either mechanically or by jogging.

IW Engineering/configuration: Activate coupling

- The "electronic coupling" function has to be activated using the parameter **coup** "electronic coupling" in the setup (SUP).
- In the parameter **MastInd** "axis group index" (1003 00061), the axis number of the master axis has to be assigned to all axes belonging to the gantry group.

Master axis is for example axis 5, thus the **MastInd** "Axis group index" (1003 00061) = 5. The axis group index of the master axis also has to be entered into the **MastInd** "Axis group index" (1003 00061) of all associated axes (slave axes).

The following interface signals always has to be kept identical within a coupling group:

- **qAx_DrvLock** "Feed lock"
- **qAx_DrvOn** "Drive Enable"

Before control startup, the gantry axes have to be mechanically torque free (i.e. no terminals). Thus, the axes can be traversed in parallel after control and Sercos startup.

	Documentation	Instruction
Instruction:		Editing machine parameters
Documentation:	MTX Machine Parameters	Activate the gantry coupling

Drive: Establish reference:

- Before starting "establish reference", the PLC has to lock all gantry axes of a group via the **qAx_DrvLock** "Feed inhibit" interface signal. **(qAx_DrvLock = 1)**.
- Upon control or Sercos, the reference is established, i.e. the actual position values of each axis are known immediately. For each drive, "referencing established" when the reference dimensions of the axes was confirmed correctly (via set absolute dimensions during commissioning). If the axes are aligned, the reference of all gantry axes is set by setting identical actual position values ("set absolute dimension").

The process for "Establish reference" which is based on these references and performed during initial commissioning should be executed at low CO, low velocity, low acceleration or deceleration, as well as torque monitoring.

IW Engineering/IndraLogic: Compensate for "inclined position"

An actual value offset that resulted at runtime or in switch-off state is only compensated by setting the interface signal **qAx_TakeActOffs** "Apply actual value offset". The subsequent "Interpolation" of the offset is started by setting the **qAx_MasterPos** "Gantry slave on master position" interface signal.

The gantry group has to have been mechanically aligned in a torque-free state.

With heavy portal gantry groups, individual gantry axes have to be aligned in parallel to each other either mechanically or by jogging.

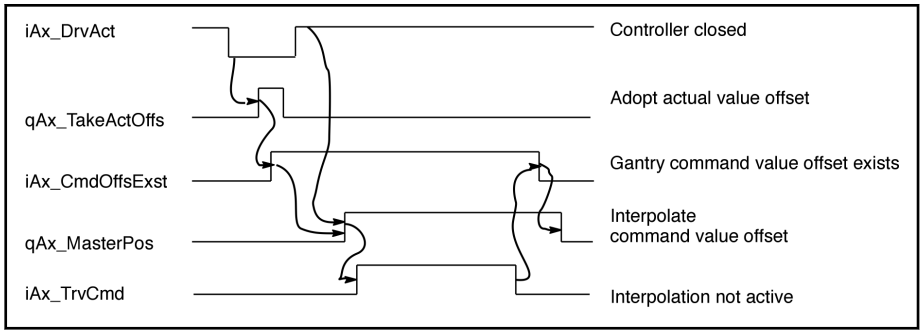


Fig. 14-10: Signal flow diagram

	Documentation	Instruction
Instruction:		Editing PLC Signals
Documentation:	MTX PLC Interface	Compensate for inclined position in gantry

IW Engineering/configuration: Parameterize monitoring

- The "electronic coupling" function has to be activated using the parameter **coup** "electronic coupling" in the setup (SUP).
- **MaxFollErr** "Maximum following distance" (1003 00060)
- **MastInd** "Axis group index" (1003 00061)
- **MaxGrTroq** "Max. standstill torque total of the group" (1003 00062)
- **ENASynrMon** "Synchronous run monitoring on" (1003 00063)
- **SynchrMonWin** "Synchronous run monitoring window" (1003 00064)
- **ENRefSwiMon** "Reference switch monitoring" (1003 00065)
- **CmdOffsIpoMode** "Compensate for the gantry command value offset" (1003 00066)
- **MaxRedCmdOffs** "Upper limit for gantry command value offset" (1003 00067)

	Documentation	Instruction
Instruction:		Editing machine parameters
Documentation:	MTX Machine Parameters	Program monitoring

14.1.4 Handling instruction: Establishing reference at absolute measuring system

This handling instruction describes the establishment of a reference with a absolute measuring system

Drive: Establish reference:

- After commissioning, the actual position values of each axis are immediately detected after control or Sercos startup. For each drive, "referencing established" when the reference dimensions of the axes was confirmed correctly (via set absolute dimensions during commissioning, without drive enable). ("Set absolute dimension").

Drive: Compensate for "inclined position"

If the actual position values of the gantry axes differ in the gantry group after Sercos startup (e.g. drifting with the controller open), the actual value offset corresponds to an inclined position between the gantry axes. The offset can now be applied as command value offset by setting the interface signal

qAx_TakeActOffs "Apply actual value offset". Set the interface signal **qAx_MasterPos** "Gantry slave on master position" to have the control compensate for the actual value offset by asynchronous interpolation. After an offset compensation, the actual position values of all gantry axes are identical again.

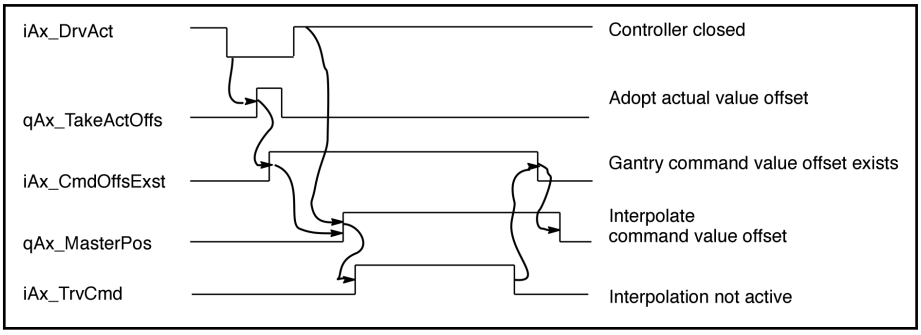


Fig. 14-11: Signal flow diagram

	Documentation	Instruction
Instruction:		Editing PLC Signals
Documentation:	MTX PLC Interface	Compensate for inclined position in gantry

Drive: Set simultaneous drive parameters

All axes of the gantry group have to have identical parameters (amplification factors, acceleration/deceleration, velocity,...).

IW Engineering/configuration: Activate coupling

- The "electronic coupling" function has to be activated using the parameter **coup** "electronic coupling" in the setup (SUP).
- In the parameter **MastInd** "Axis group index" (1003 00061), the axis number of the master axis has to be assigned to all axes belonging to the gantry group.

☞ Master axis is for example axis 5, thus the **MastInd** "Axis group index" (1003 00061) = 5. The axis group index of the master axis also has to be entered into the **MastInd** "Axis group index" (1003 00061) of all associated axes (slave axes).

The following interface signals always has to be kept identical within a coupling group:

- **qAx_DrvLock** "Feed lock"
- **qAx_DrvOn** "Drive Enable"

Before control startup, the gantry axes have to be mechanically torque free (i.e. no terminals). Thus, the axes can be traversed in parallel after control and Sercos startup.

After each exchange of motors, measuring systems and/or drive software, commission the gantry axes again.

The vendor-specific Sercos parameters (P-x-xxxx) specified for commissioning are only relevant for IndraDrive drives. For other drives, use equivalent parameters. The drive vendor has to ensure the functionality.

14.1.5 Handling Instruction: Applying Gantry Axis Group E-Stop

The "E-stop" is a hardware coupling of the drive amplifiers belonging to one gantry group. The hardware coupling serves a quicker reaction to fatal errors of an axis (Y or Y2). To avoid damage to the mechanical system, both axes have to be decelerated as synchronously and fast as possible.

Drive amplifiers: Hardware wiring

Output X32:7 of the y-axis is wired to **input X32:6** of the Y2-axis. **Output X32:7** of the Y2-axis is wired to **input X32:6** of the y-axis.

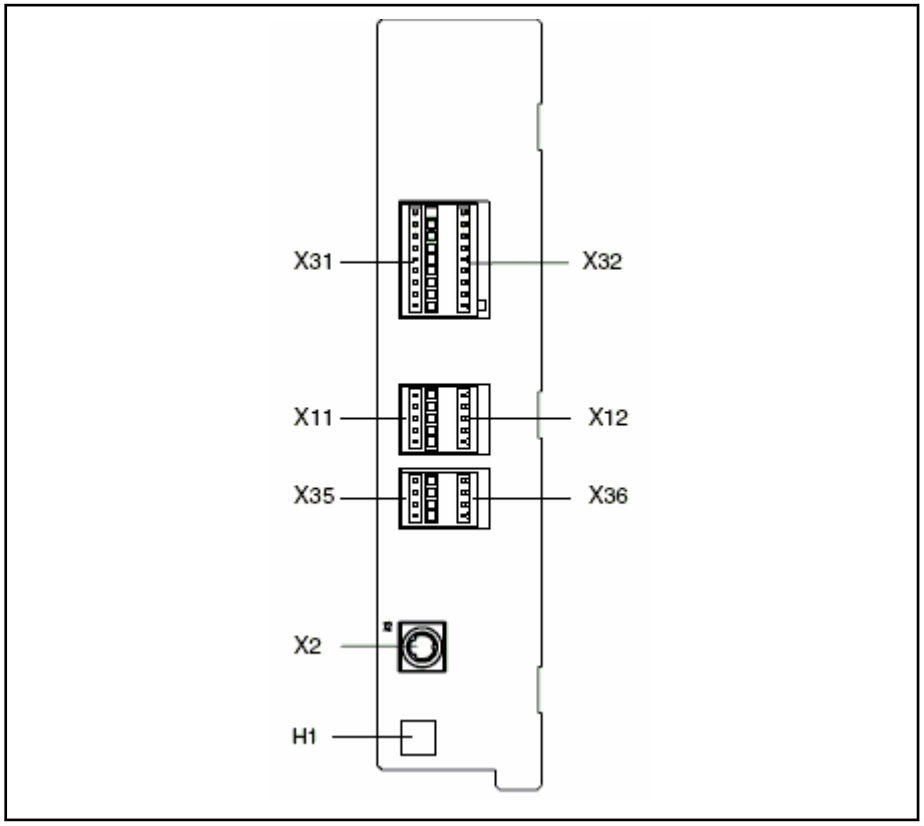


Fig. 14-12: Overview on connecting pins

IW-Engineering/IW-Drive: Applying Sercos signals

E/A 7b has to be parameterized as output. The Sercos parameter **P-0-0115** "Device control: Status word", **Bit 0** is assigned to this output. **E/A 6b** is parameterized as input and this input is assigned the Sercos parameter **P-0-0223** "E-stop input", **bit 0**. Thus, the signal used for the BB relay is used to trigger the E-stop at the respective controller.

		Example
Example:		Digital inputs/outputs

IW-Engineering/IW-Drive: Parameterize E-stop function

- Set the evaluation of the E-stop signal to active.
- Set the interpretation of the E-stop as warning.

		Example
Example:		E-stop signal

IW-Engineering/IW-Drive: Parameterize error reaction

Set the error reaction to **"Immediate, optimum stopping"** and the optimum stopping to **"Velocity command value zeroing with ramp and filter"**. The deceleration **"Quick stop"** is set to the maximum axis acceleration value.

	Example	
Example:	Error reaction	Chapter heading

14.2 Selective additive coordinate coupling

14.2.1 Description

Function

One or several workpiece coordinates of a channel (source) can additively influence workpiece coordinates of other channels (targets). If there is a traversing motion in the source channel, the path motions in the individual target channels can be superimposed by that of the source channel.

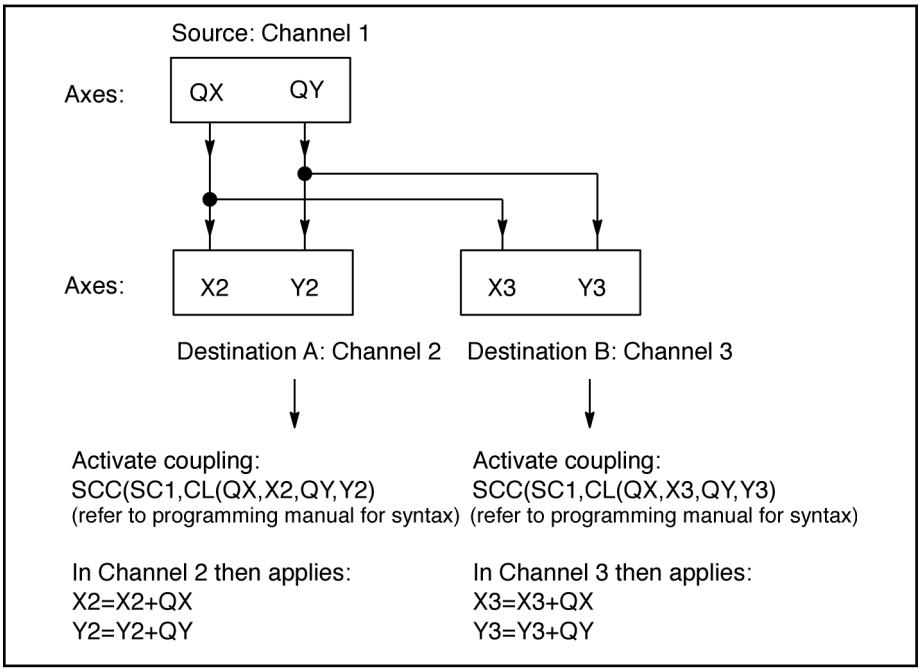


Fig. 14-13: Source channel

Restrictions and special features

- A source channel has to have a lower channel number than the target channel.
- The source channel as well as its coordinates can be assigned to several target channels at the same time.

  A target channel may only be coupled to one source channel at a time.
- When establishing and ending a coupling (SCC(..)), do not change the source coordinates.
- By superimposing two path motions, the maximum permitted path velocity, path acceleration and axis velocities in the target channel can be exceeded.
- For the superimposed motions, no limit switch is checked.

- When a coordinates transformation/axis transformation is switched on or off or over in the source or target channel, the coordinate coupling is completed and a runtime error is output in the target channel.

Example:

Selective additive coordinate coupling - restriction

A machine has two rectangularly arranged machining axes. Their motions are to be generated by superimposing two part programs (see below).

The machine is configured with two virtual (VX, VY) and two real axes (X, Y). The virtual axes are assigned to the 1. channel, the real axes to the 2. channel. Axis or coordinate transformations are not active.

Required machine parameters:

		Channel 2		Channel 1	
100100001	"Drive function type":	1,	1,	1,	1
100100010	"Virtual drive":	0,	0,	1,	1
100300001	"Physical axis name":	X,	Y,	VX,	VY
100300002	"Channel assignment":	2,	2,	1,	1

Required part programs:

The part program in channel 2 is to control the sequence.

Part program of channel 1 (Source channel) "TEST1":

```
10    VAP (-VX[@AXPOS], VY[@AY      ;Load axis position
      POS])
20    BITIF("ICH_CUSTOM1") = 1      ;Set CPL customer output 1
N20   WPV[@STARTALL=1]              ;Wait until the motion can start
N20   G1 G91 F1000 VY50
20    BITIF("ICH_CUSTOM1") = 0
M30
```

Part program of channel 2 (target channel):

```
10    @AXPOS=ACS( "X")              ;Save axis position
      @AYPOS=ACS( "Y")
      @STARTALL=0
21    WAIT
10    ERR=MCOPS(7,1,5)              ;Select automatic mode in channel 1
20    IF ERR <> 0 THEN
      <Error routine>
      ENDIF
30    WAIT
40    ERR=MCOPS(4,1, "TEST1",,,     ;Select program TEST1 in channel 1
      67)
50    IF ERR <> 0 THEN
      <Error routine>
      ENDIF
```

```
60    WAIT                          ;Wait for channel 1 to signal NC ready
      (BITIF( "ICH_NCREADY,1"))
70    ERR=MCOPS(6,1)                ;Start part program in channel 1
80    IF ERR <> 0 THEN
      <Error routine>
      ENDIF
90    WAIT                          ;Wait for channel 1 to set the CPL cus-
      (BITIF( "ICH_CUSTOM1",1))     tomer output 1
N10   SCC(SC1,CL(VX,X,VY,Y))        ;Couple coordinates VX and VY
N20   SPV[@STARTALL = 1]            ;Start motion
N30   G1 G91 F1000 X50
N40   SPV[@STARTALL = 0]            ;Reset variable
N50   SCC(0)                        ;Cancel coupling
M30
```

Relevant NC functions

SCC(..)	Establish/cancel coordinate coupling

Tab. 14-4: Relevant NC functions

Relevant machine parameters (MP)

100100001	Drive function type
100100010	Virtual drive
100300001	Physical axis name
100300002	Channel assignment
101000001	Maximum acceleration in feed mode
101000003	Maximum acceleration in rapid traverse mode
703000210 and 703000220	Path acceleration
706000020	Switched-on state after reset

Tab. 14-5: Relevant machine parameters

14.2.2 Handling instruction: Selective additive coordinate coupling

Applying

One or several workpiece coordinates of a channel (source) can influence workpiece coordinates of other channels (targets) in an additive manner. If there is a traversing motion in the source channel, the path motions in the individual target channels can be superimposed by that of the source channel.

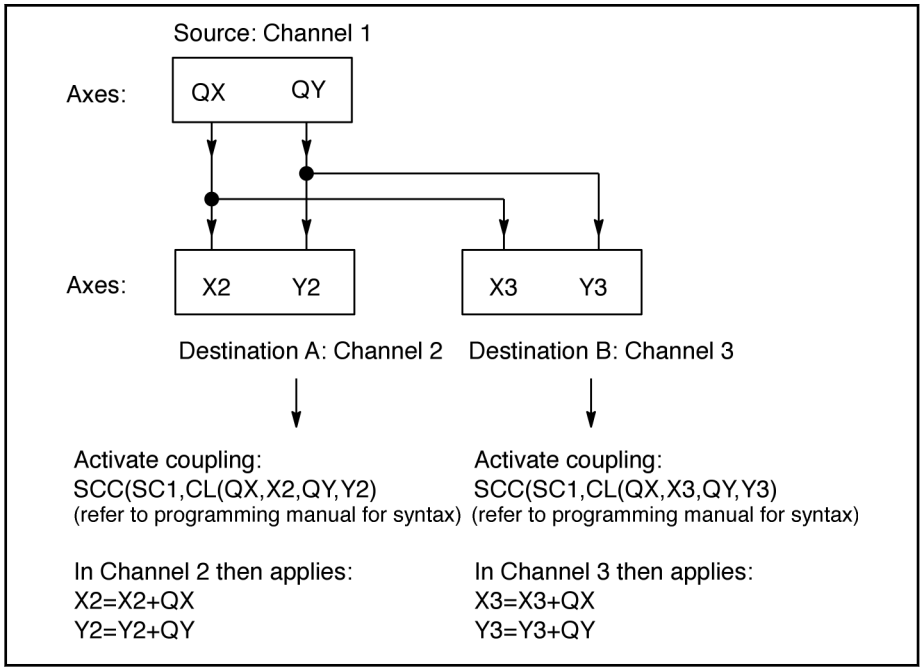


Fig. 14-14: Example overview: Selective additive coordinate coupling

IW Engineering/configuration: Editing parameters

The settings to be made in the following parameters depend on the respective use case:

- **EnablVirtMode** "Virtual drive" (1001 00010)
- **RefPos** "Reference point position" (1015 00110)
- **SysDrName** "Physical drive name" (1003 00001):

  Adjusting to the system axis names.
- **DefaultCh** "Channel Assignment" (1003 00002):

  Assign the axis to the desired channel.
- **FeedAxAcc** "Maximum acceleration in feed mode" (1010 00001):

  Determination of the maximum axis acceleration in feed mode.
- **RapidAxAcc** "Maximum acceleration in rapid traverse mode" (1010 00003):

  Determination of the maximum axis acceleration in rapid traverse mode.
- **MaxChVel** "Maximum path velocity" (7030 00120):
- **MaxChAcc** "Maximum path acceleration" (7030 00210):
- **MaxChDec** "Maximum path deceleration" (70330 00220):
- **ChResetState** "State after channel reset" (7060 00020): If the function is to be deactivated after **M2/M30**, **channel reset** or **System reset**, enter the **SCC()** function into the respective position in the parameter **ChResetState**.

	Documentation	Instruction
Instruction:		Editing machine parameters
Documentation:	MTX Machine Parameters	Coordinate coupling

Enabling/disabling

IW Operation/NC programming: Activate coordinate coupling

Program **SelCrdCouple(SC<Channel>,CL(<Q1>,<Z1>{{,<Qn>,<Zn>}...}))** or **SCC(SC<Channel>,CL(<Q1>,<Z1>{{,<Qn>,<Zn>}...}))**

☞ A source channel has to have a lower channel number than the target channel.

The source channel as well as its coordinates can be assigned to several target channels at the same time. A target channel may only be coupled to one source channel at a time.

When establishing and ending a coupling (SCC(..)), no movement in the source coordinates may be made.

By overlapping two path movements, exceeding the maximum permitted path velocity, path acceleration and axis acceleration in the target channel is possible.

For the superimposed motions, no limit switch is checked.

When a coordinates transformation/axis transformation is switched on or off or over in the source or target channel, the coordinate coupling is completed and a runtime error is output in the target channel.

IW Operation/NC programming: Deactivate coordinate coupling

Programming **SelCrdCouple()** or **SCC()**.

If the coupling is to be deactivated after **M2/M30**, **channel reset** or **System reset**, the **SCC()** function must be entered in the respective position in the parameter **ChResetState**.

	Documentation	Instruction
Documentation:	MTX Programming Manual	Activating/deactivating coordinate coupling

14.3 Selective additive coordinate coupling with tables

14.3.1 Description

Function

This function extends the options of SelCoordCouple/SCC.

Extensions are:

- Coupling two source coordinates to one target coordinate.
- Workpiece and machine coordinates can be used as source.
- The coupling relation can be adjusted using a factor and a table.
- The target coordinate can be provided with a scaling factor.

The workpiece position (WCS) of a channel can be additively corrected with this function by a coupling to the workpiece or machine position(s) of another channel. One or multiple coordinates can be coupled. During coupling, there is no traversing motion.

Note the following coupling rules:

- A channel can only couple to the coordinates of a channel with a lower channel number.
- When establishing a coupling, ensure that there is no motion in the source channel.

- For a maximum of 4 source coordinates, table couplings can be activated.
- A coordinate, which is already a target coordinate of an existing coordinate coupling, must not become the target coordinate of another coordinate coupling. The existing coordinate coupling has to be deactivated first.
- A coordinate, which is already the source coordinate of a coordinate coupling, can also become the source coordinate of another coordinate coupling.

If the coupling is active, note the following:

- The motion of coordinates and axes of the target channel result from the superposition of two motions. This has to be taken into consideration when selecting path velocity (F), path acceleration (PathAcc, MP 703000210 and 703000220) and axis acceleration (AxAcc, MP 101000001).
- When a coordinate transformation ("inclined plane") is switched on/off or changed in the target channel, the coordinate coupling is ended and a runtime error is output in the target channel.
- When a coordinate transformation ("inclined plane")is switched on/off or changed in the source channel, the coordinate coupling is ended and a runtime error is output in the target channel if a workpiece coordinate is used.

  When an axis transformation 1 is switched on or off or changed in the source or target channel, coupling of coordinates is terminated, and a runtime error is output in the target channel.
- When an axis transformation 2 is switched on or off or changed in the target channel, coupling of coordinates is terminated, and a runtime error is output in the target channel.
- When an axis transformation 2 in the source channel is switched on/off or changed, coordinate coupling is terminated, and a runtime error is output in the target channel, provided a workpiece coordinate is used.

When ending the coupling, ensure that the source coordinates do not move in the source channel.

Possible coupling relations when using a source coordinate:

$P_{De} = P_{De}^{0} + (P_{De}^{prog} – P_{De}^{0}) * \text{target factor} + (P_{So} – P_{So}^{0}) * \text{factor}$

$P_{De} = P_{De}^{0} + (P_{De}^{prog} – P_{De}^{0}) * \text{target factor} + (\text{Tab} (P_{So} – \text{TabOffs}) - \text{Tab} (P_{So}^{0} – \text{TabOffs})) * \text{factor}$

With:

P_{De}	= Current position of the target coordinate
P_{De}^{0}	= Position of the target coordinate when establishing the coupling
P_{De}^{prog}	= Current position of the target coordinate that results without coupling
Target factor	= scaling factor for the target coordinate (default = 1.0)
P_{So}	= Current position of the source coordinate
P_{So}^{0}	= Position of the source coordinate when establishing a coupling
Factor	= coupling factor (default = 1.0)

Tab = Tabulated function

TabOffs = offset for table access (default = 0.0)

When using two source coordinates, a second analog addend is added for the second source coordinate. That means:

... + ($P_{So2} – P_{So2}^0$) * Factor2

or:

... + (Tab (P_{So2} – TabOffs2) - Tab (P_{So2}^0 – TabOffs2)) * Factor2

Relevant NC functions (G-codes)

Establishing a coupling:

SelCrdCoupleTab(DE(<Name>,<TargetFact>**),SO**(<Channel>,<Name>,<Fact>,<Tab>,<Offs>,WCS|MCS)
{ **,SO2**(<Channel>,<Name>,<Fact>,<Tab>,<Offs>,WCS|MCS)})

The short form is: **SCCT**

Relevant NC Functions (G-codes) - Establishing a coupling

With the parameters:

DE (destination)

<Name> Name of the target coordinate

<Targetfact> Scaling factor of the target coordinate (1.0 unless programmed)

SO (source)

<Channel>	Number of the source channel
<Name>	Name of the source coordinate
<Fact>	Coupling factor (1.0 unless programmed)
<Tab>	Name of the coupling table
<Offs>	Offset value for table access (0.0 if not programmed)
WCS|MCS	Coordinate system of the source coordinate
	WCS = Workpiece coordinate
	MCS = Machine coordinate
	Default: WCS

SO2 (Source 2) analog to SO (source)

Examples: Establishing a coupling

SCCT(DE(YB), SO(1,YA))	The actual yb-position results as the sum of the yb-position specified by programming in the active channel and the ya-position.
SCCT(DE(YB,0), SO(1,YA))	The actual yb-position is specified by the ya-position. The yb-position specified by programming in the active channel has no effect.
SCCT(DE(YB), SO(1,YA, ,SCCTab.fct))	Couples the workpiece coordinate YA of channel 1 to the workpiece coordinate YB of the active channel via the coupling table SCCTab.fct.
SCCT(DE(YB), SO(1,YM,0.3, , ,MCS))	Through coupling factor 0.3, couples the machine coordinate YM of channel 1 to the workpiece coordinate YB of the active channel.

Canceling a coupling:

SelCrdCouple() or SelCrdCoupleTab() or SCC() or SCCT()

Creating a spline table file: A spline table is created irrespective of the creation of a coordinate coupling with

CoupleSplineTab(TAB(<Tab>,<1|0>**))**

.

The short form is: **CST**

With the parameters:

TAB <Tab>	Name of the coupling table

Example:

CST(TAB(curve.fct))	If necessary, it creates the spline table/ <Link directory>/curve.fct.s.

Switch-On/Off Behavior, Control Reset

Control startup: The function is inactive.

Reset Channel reset and system reset do not directly affect this function. If this function is to be deactivated during channel reset or system reset, enter SCCT() or SCC() to MP 706000020 "switched-on state after reset" at a suitable position (before #Reset:, after #Reset: or after #SysRes:) SCCT() or SCC().

M30 / M2: If the function is to be deactivated by M2/M30, SCC() or SCCT(), enter #Reset: in MP 706000020 "switched-on state after reset" before: SCC() or SCCT().

Example:

A machine with two machining axes whose motion is generated by superposition of two part programs. The two axes are arranged rectangularly. The machine is configured with two virtual and two real axes. The two virtual axes are assigned to the first channel, the two real axes to the second channel.

MP 100100001	"Drive function type"	1,	1,	1,	1
MP 100100010	"Virtual drive"	0,	0,	1,	1
MP 100300001	"Physical axis name"	X,	Y,	VX,	VY
MP 100300002	"Channel assignment"	2,	2,	1,	1

No axis transformation and no coordinate transformation is active. The part program in channel 2 controls the sequence.

Part program of channel 1 (source channel) "TEST1":

```
10 VAP VX[@MXPOS] VY[@MYPOS]              ;Load axis position
20 BITIF("ICH_CPL01") = 1                 ;Set CPL customer output1
N20 WPV[@STARTALL=1]                      ;Wait until the motion can start


N20 G1 G91 F1000 VY50


20 BITIF("ICH_CPL01") = 0


M30
```

Part program of channel 2:

```
10 @MXPOS=ACS("X")                          ;Save axis positions
11 @MYPOS=ACS("Y")

20 @STARTALL=0
21 WAIT

10 ERR=MCOPS(7,1,5)                         ;Select automatic mode in channel 1
20 IF ERR <> 0 THEN Error handling ENDIF
30 WAIT
40 ERR=MCOPS(4,1,"TEST1", , ,67)            ;Select program TEST1 in channel 1
50 IF ERR <> 0 THEN error handling ENDIF
60 WAIT(BITIF("ICH_NCREADY",1))             ;Wait for channel 1 to signal NC ready
70 ERR=MCOPS(6,1)                           ;Start part program in channel 1
80 IF ERR <> 0 THEN Error handling ENDIF
90 WAIT(BITIF("ICH_CPL01",1)                ;Wait for channel1 to set the CPL customer output1

N10 SCCT(DE(X),SO(1,VX))                    ;Couple coordinates VX
N11 SCCT(DE(Y),SO(1,VY))                    ;Couple coordinates VY

N20 SPV[@STARTALL = 1]                      ;Start motion
N30 G1 G91 F1000 X50
N40 SPV[@STARTALL = 0]                      ;Reset variable
N50 SCCT()                                  ;Cancel coupling

M30
```

Compatibility

The "SelCrdCoupleTab" function supplements the current function "SelCrdCouple".

14.3.2 Activation

The function is activated by programming the NC function "SelCrdCoupleTab (...)".

14.3.3 Deactivation

The function "SelCrdCoupleTab() / SCCT()" terminates the coupling.

14.4 Coupling tables

14.4.1 Description

Function

The coupling tables are used for the coordinate coupling (SCCT), the axis coupling (AXC) and the system axis coupling.

A coupling table contains the coupling relation between two positions. The coupling function is defined in a table using data point pairs (pmi, fi) (i=1,...,n).

From the pairs of interpolation points, the NC interpolator calculates the function values between the data points.

The following approximations can be used to calculate the positions between the data points:

- Linear - As distance between two data points
- Cubic spline - As a spline curve between two data points considering the previous and subsequent data points.

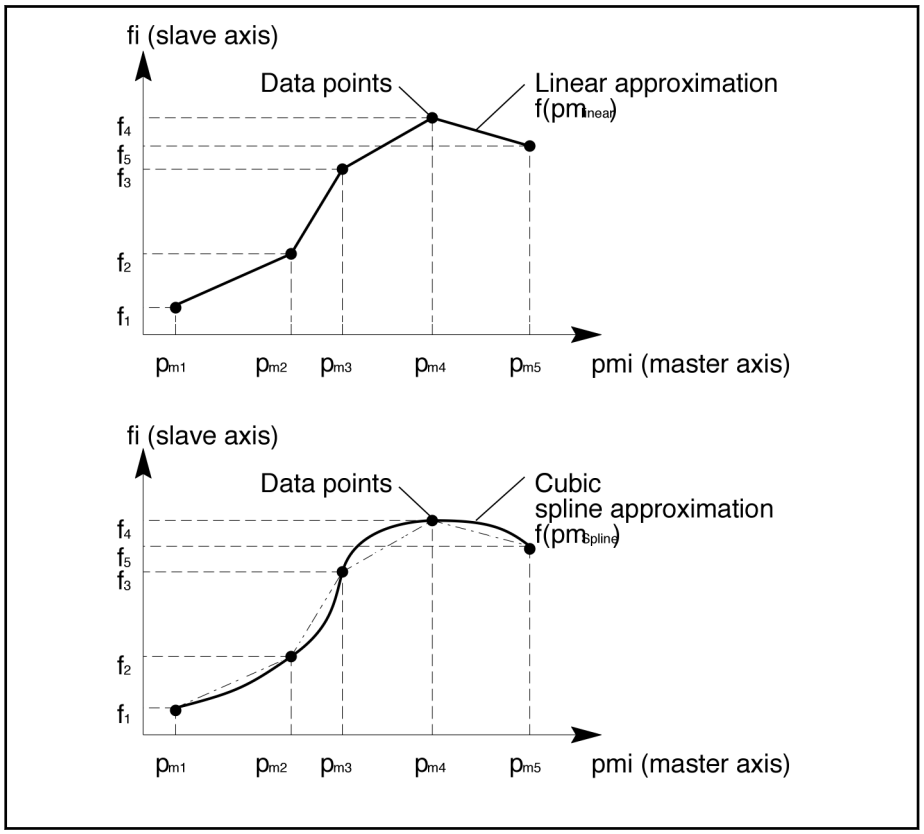


Fig. 14-15: Linear approximation - Cubic spline approximation

The spline approximation is preferably used for a curved characteristic between the data points if there is no data on the exact characteristic. The cubic spline approximation allows a curved characteristic with smooth transitions at the data points.

Coupling Table Syntax

The coupling tables are structured as follows:

#1	<Interpolation type of the interpolation points> ; 1 linear, 3 cubic spline
#11	<Unit pmi-values> ; -3 mm, -2 cm, -1 dm, 0 m, 1 inch, 2 degrees, 3 rad
#12	<Unit of the fi values ; Same as with #11

```
#20     <Periodic>
            ; 0 non-periodic, 1 periodic
        <Master axis velocity limitation>
#32         ;1 complete limitation
            ;0 incomplete limitation
        <pm1> <f1>
#100        ; 1. interpolation point pair
#100    <pm2> <f2>
            ; 2. interpolation point pair
.
        <pmn> <fn>
#100        ; n. Data point pair
```

#1 Defines the method of interpolation between the data points.

The types available are linear interpolation (value = 1) or cubic spline interpolation (value = 3).

The default value is 1, i.e. linear interpolation.

The value 1 must not be used for the channel coupling (AxCouple, AXC).

#11 Defines the unit of the pm values.

When specifying length units (values -3 through +1), the table is only to be used for linear master axes. When specifying angular units (values 2 and 3), the table is only to be used for rotary master axes.

The default value is -3 (mm).

#12 Defines the unit of the F-values (units as with #11).

In tables to be used for linear or rotary slave axes, no other values than -3 through +1 or 2 and 3 may be specified.

The default value is -3 (mm).

#20 Defines whether the coupling function is periodic or non-periodic.

If the coupling function is to be periodic (value = 1), the last p_m value defines the period. If the master axis position exceeds the period, the function value $f(p_m)$ is determined by counting "pm" back to the periodic interval using the modulo calculation.

For a non-periodic coupling function, the modulo calculation is disabled.

The default value is 0, i.e. non-periodic.

Note the following rules:

- #20 = 0, non-periodic:

  The limit switch range of the master axis is restricted to the periodic interval [pm1, pmn].

  Modulo axes (linear modulo axes or endless axes) may not be used as master axes.

- #20 = 1, periodic:

The pm values must begin with 0, e.g. p_{m1} = 0. The last p_m value defines the period.

The F-values of the first and the last pairs of interpolation points must be identical, i.e. $f(p_{m1}) = f(p_{mn})$.

If the master axis is a modulo axis, the AxModVal modmod cycle has to be 0, with AxModVal being the axis-specific modulo value (drive parameter).

#32 This parameter is not supported anymore!

#100 defines a pair of interpolation points.

Any number of data points can be entered in the table.

The pm values have to be entered in ascending order.

The comment character in the table is the semicolon.

Approximating f(pm) between the data points

The function $f(p_m)$ is always approximated between two data points $p_{m,i}$ and $p_{m,i+1}$ by a polynomial:

$$fi(p_m) = \sum_{j=0}^{3} c_{ij} \cdot (p_m - p_{m,i})^j \qquad p_{m,i} \le p_m < p_{m,i}; i = 1,\ldots,n$$

For a linear approximation, the coefficients c_{i2} and c_{i3} are equal to zero. For a cubical spline approximation, all coefficients of zero differ. The 4n coefficients c_{ij} (i=1,...,n; j=0,..3) are calculated from the data point pairs and stored in their own spline table (see next section). The NC interpolator accesses an image of this spline table and calculates from it the function values in accordance with the above formula.

Creating the corresponding spline table

The spline table is created in the block preparation while interpreting the coupling syntax or if "**CoupleSplineTab(**<TabName>{<1|0>}**)**". The spline table is structured as follows:

#-3	<Size of the coupling table>
#-2	<Path and name of the coupling table>
#-1	<Timestamp of the coupling table>
#1	<Interpolation type of the interpolation points> ; As the coupling table
#10	<Number of splines> ; Number of splines
#20	<Period> ; Unit increments
#30	<Master type> ; Permitted master (axis) types
#31	<Slave type> ; Permitted slave (axis) types
#200	<pm1> ; 1. master axis position

```
#201    <c10> <c11> <c12> <c13>
            ; Coefficients for 1st spline
#200    <pm2>
            ; 2. master axis position
#201    <c20> <c21> <c22> <c23>
            ; Coefficients for 2nd spline
. . . . .
#200    <pmn>
            ; n. Master axis position
#201    <cn0> <cn1> <cn2> <cn3>
            ;Coefficients n Spline
```

The spline tables are stored as read-only format in the directory of the link tables. They are named as the coupling tables, but with the extension ".s".

Example:

Coupling table "curve.fct", spline table "curve.fct.s"

In the following cases, a new spline table is created:

- The spline table does not exist in the link directory:
- One of the entries #-1, #-2 or #-3 does not correspond with the corresponding attribute of the pertinent coupling table.
- The generation has been explicitly programmed with CoupleSplineTab(<Name>, 1).

The entries #-3, #-2 and #-1 have the following meaning:

- **#-3 <Size of the coupling table>**

  The coupling table size is stored in bytes.
- **#-2 <Name of the coupling table>**

  The full coupling table name with its path is stored.

  Example: #-2 /mnt/CoCo/nocken.fct
- **#-1<Timestamp of the coupling table>**

  The time stamp of the coupling table is stored. Whether the corresponding spline table has to be created again depends on this data. A new spline table is created again if the coupling table is moved to another directory, its time stamp or size are changed.

14.5 System axis coupling

14.5.1 Description

With the system axis coupling, electronic gears can be set up across the channels. For each system axis, up to five master axis dependencies are provided. Consider the interpolation value of the slave axis additionally.

Restrictions and special features:

- The slave axis cannot be activated at the fixed stop to move against fixed stop.
- The slave axis cannot be referenced at the same time.
- The slave axis velocity or the slave axis acceleration can be exceeded due to the overlapping of several master axis motions.

- For the superimposed slave axis motion, the limit switch is not checked.
- The velocity of rotary axes and spindles is limited to half the modulo interval (mostly 180°) per Sercos cycle (e.g. Sercos cycle 2 ms corresponds to 15,000 rpm max.). To increase the limit speed for asynchronous axes, refer to the documentation “Functional Description - Basics, chapter 4, “Drives (axes, spindles).

In the following example, an electronic gear with the three master axes C, Z and A and the slave axis B is to be set up. The master axes are to be coupled in a certain speed ratio:

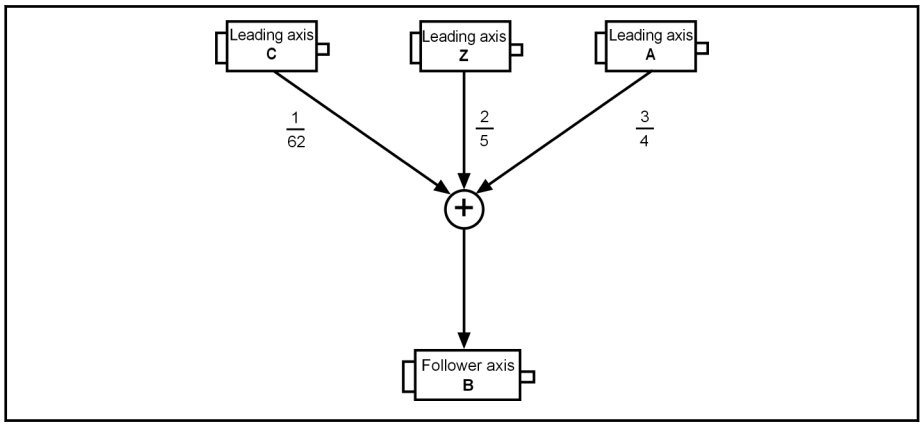


Fig. 14-16: Coupling example

This coupling group is defined via a system data structure. The user can thus specify the control data via the PLC program, via the CNC part program or via the user interface. The part program below defines the example coupling shown above:

Program:

```
100 C%=3 : Z%=4 : A%=5 : B%=1 : REM System axis indices
101 SD.SysAxCoupleCmd[B%].OnOff = 1
102 SD.SysAxCoupleCmd[B%].Master[1].AxIndex = C%
103 SD.SysAxCoupleCmd[B%].Master[1].N_Input = 62
104 SD.SysAxCoupleCmd[B%].Master[1].N_Output = 1
105 SD.SysAxCoupleCmd[B%].Master[1].Enable = TRUE
106 SD.SysAxCoupleCmd[B%].Master[2].AxIndex = Z%
107 SD.SysAxCoupleCmd[B%].Master[2].N_Input = 5
108 SD.SysAxCoupleCmd[B%].Master[2].N_Output = 2
109 SD.SysAxCoupleCmd[B%].Master[2].Enable = TRUE
110 SD.SysAxCoupleCmd[B%].Master[3].AxIndex = A%
111 SD.SysAxCoupleCmd[B%].Master[3].N_Input = 4
112 SD.SysAxCoupleCmd[B%].Master[3].N_Output = 3
113 SD.SysAxCoupleCmd[B%].Master[3].Enable = TRUE
114 SD.SysAxCoupleCtr.Validate=1
```

The coupling information is only given for slave axis B. Axis references are always made via the respective system axis numbers.

Linear axes, rotary axes and position-controlled spindles are valid as slave and master axes with command value coupling. Spindles in speed control can only be used as master axes with actual position coupling. For the special features of the spindle coupling, refer to chapter 14.5.5 "System axis coupling with spindles" on page 564.

A slave axis can be used as master axis for an additional gear. Thus, cascade groups with a depth of up to four master axes levels can be defined.

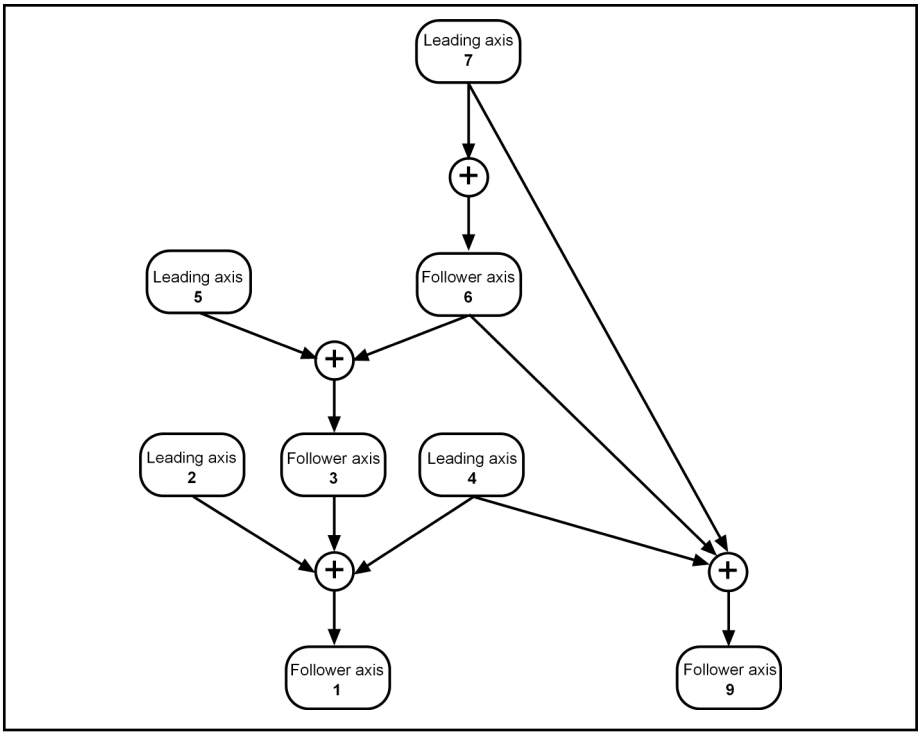


Fig. 14-17: Cascaded coupling with three master axis levels

The system axis coupling relation as mentioned in the example above is specified for the axes 1, 3, 6 and 9 in the CPL as follows:

Program:

```
1 SD.SysAxCoupleCmd[1].Master[1].AxIndex  = 2
1 SD.SysAxCoupleCmd[1].Master[2].AxIndex  = 3
1 SD.SysAxCoupleCmd[1].Master[3].AxIndex  = 4
1 SD.SysAxCoupleCmd[3].Master[1].AxIndex  = 5
1 SD.SysAxCoupleCmd[3].Master[2].AxIndex  = 6
1 SD.SysAxCoupleCmd[6].Master[1].AxIndex  = 7
1 SD.SysAxCoupleCmd[9].Master[1].AxIndex  = 4
1 SD.SysAxCoupleCmd[9].Master[2].AxIndex  = 6
1 SD.SysAxCoupleCmd[9].Master[3].AxIndex  = 7
```

Apart from the coupling relation, the system axis coupling is provided with control elements:

SAC control elements

OnOff	- Switching the coupling on/off
IpoEnable	- Consider the interpolation value of the slave axis
Master[m].Enable	- Consider reference value[m]
AccCouple	- Maximum acceleration
VelCouple	- Maximum slave axis velocity

The system axis coupling does not depend on the operation mode of the channels assigned to the axes, i.e. during jogging, MDI or automatic mode, slave axes remain coupled the same way.

Offsets of the slave axis compared to the reference value can additionally be specified in jog mode or via the part program, since the coupled slave axis can still be programmed.

Feed hold or override 0% cannot stop a slave axis moved by master axes.

Switching off all reference values decelerates the slave axis to zero velocity.

Generally, the system axis coupling can also be combined with other coupling variants. The following overview shows that the coordinate coupling (SCC/SCCT) and the channel axis coupling are connected in front of the system axis coupling while the gantry coupling is connected behind the system axis coupling.

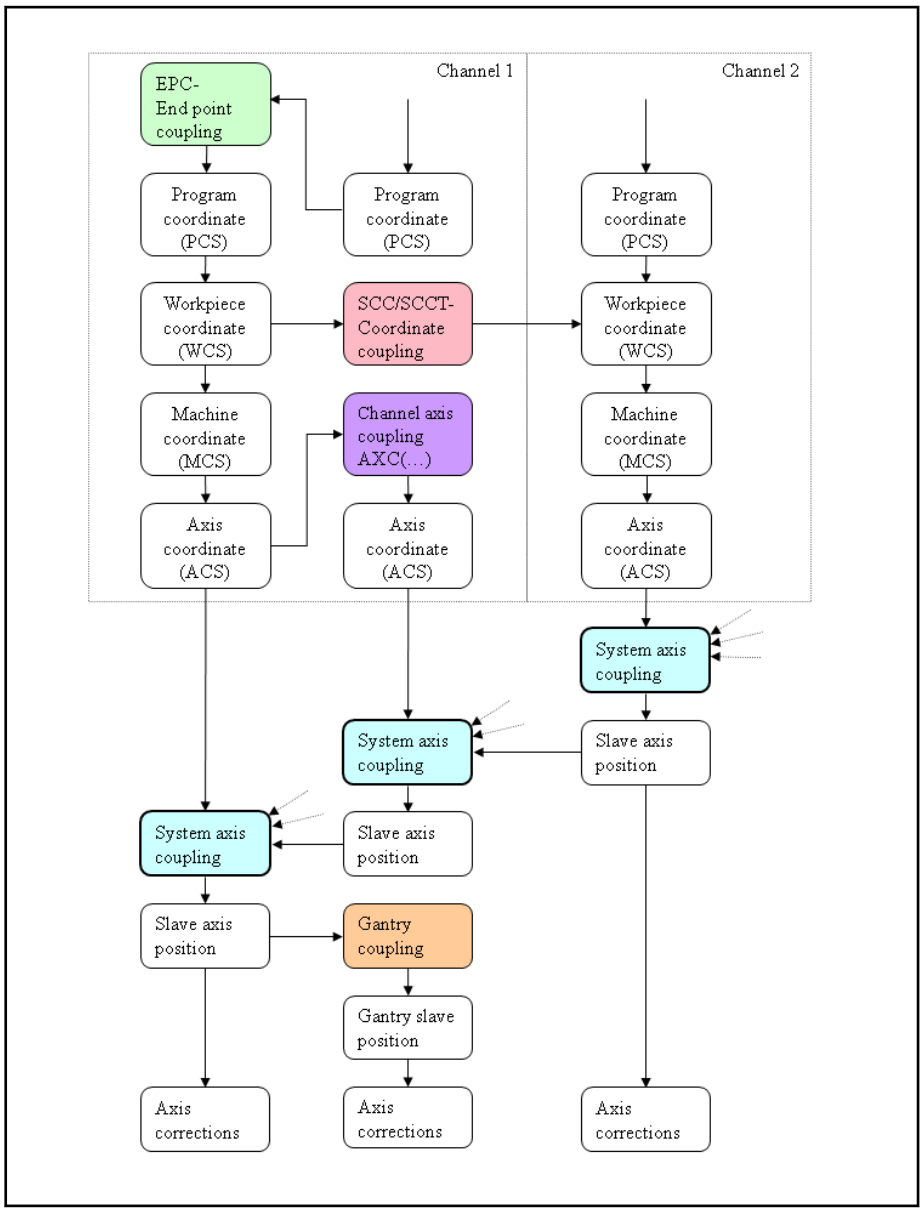


Fig. 14-18: Overview on the MTX coupling variants

Activating and deactivating the system axis coupling activates an interpolator that approximates the slave axis to the active reference value (velocity and position in the coupling relation of the master axes).

☞ Activating and deactivating or changing the coupling in case of differences in velocity and position between reference value and the slave axis position leads to compensation motion of the slave axis.

Activate and deactivate the coupling outside dangerous areas.

14.5.2 Applying the system axis coupling

System Data Overview: System axis coupling is controlled via the system data structure

SysAxCoupleCmd

For each system, the structure determines the dependence on the master axes with the relevant coupling relations. The structure reports the activation and deactivation of the coupling or the individual reference values to the interpolator. The structure remains also after switching off so that the coupling data can be specified again after the next control startup.

With the help of the control structure

SysAxCoupleCtr

the current control data of all system axes is validated in SysAxCoupleCmd. The system rejects incorrect data by issuing an error code and the relevant element index. If all elements are valid, the control data for the axis coupling are copied and enabled. After a successful activation, the control data is also displayed in the status data.

This status information as well as the coupling status are updated upon each change and are summarized in the structure

SysAxCoupleSta

The status data is deleted when switching on/off the control and is available again when activating the coupling.

Control of the systemaxiscoupling: The smallest coupling unit - the system axis coupling element (SAC) seen from the slave axis - is in the system data structure **SysAxCoupleCmd[i]** and controls the coupling and decoupling of the system axis with the same index:

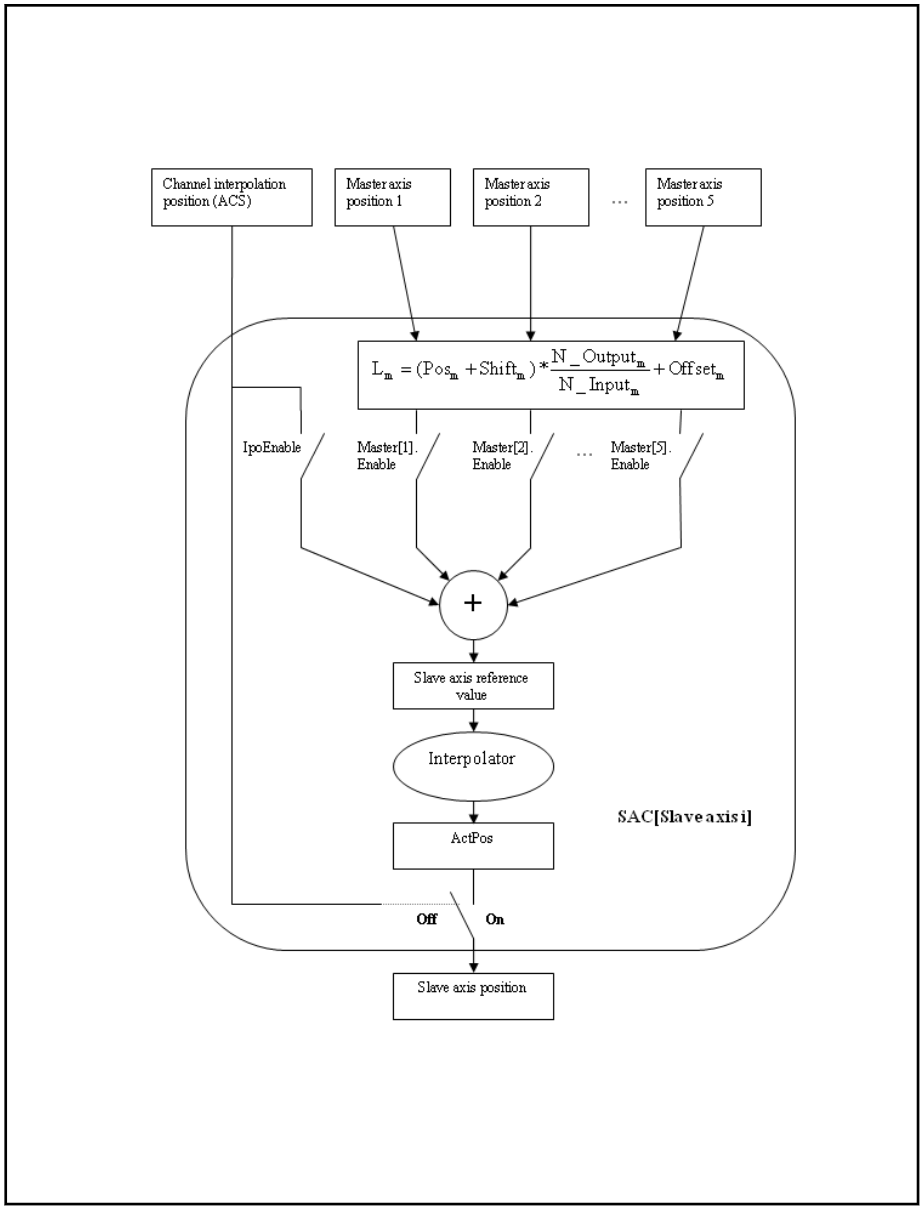


Fig. 14-19: Simplified diagram of the system axis coupling

After the system startup, all SACs are inactive. The **OnOff** switch activates the system axis coupling. Master axis values and interpolation values of the slave axes can be enabled and disabled separately (**Master[m].Enable, IpoEnable**). The SAC interpolator ensures that the slave axis is approximated to the slave axis reference position and the slave axis reference velocity. The system axis coupling as well as the activation/deactivation of reference values can be performed at standstill as well as flying in case of constantly changing reference values.

For master value coupling, the SAC interpolator takes into account either the velocity **VelCouple** and the acceleration **AccCouple** from the system data (**Rigidity=0**) or the maximum axis velocity and acceleration from the machine parameters (**Rigidity=1**).

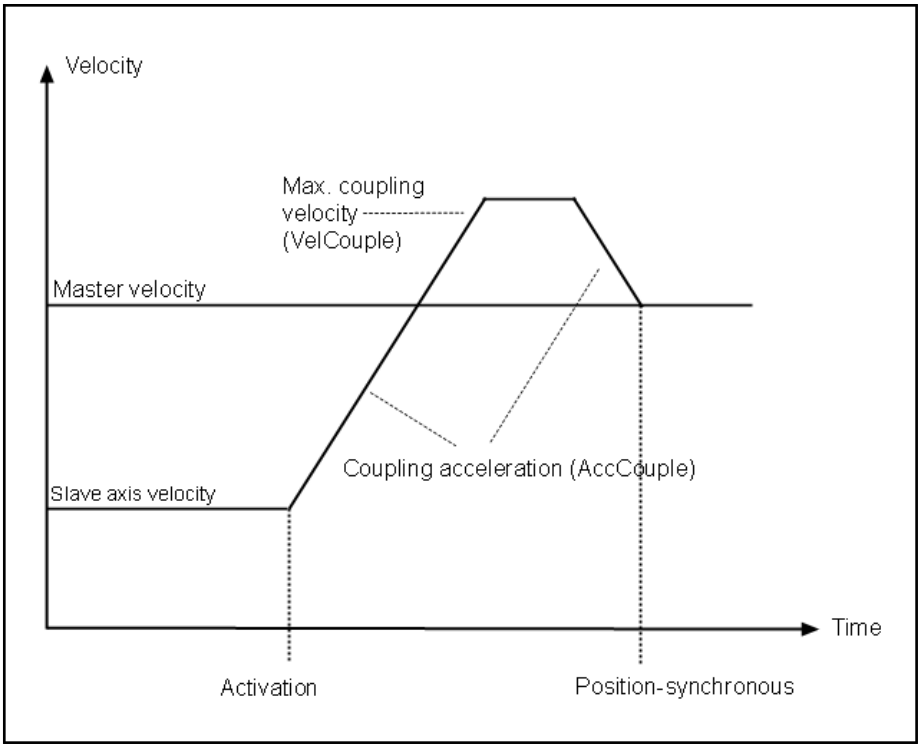


Fig. 14-20: Velocity profile of the coupling

The coupling is established as soon as the slave axis reaches the reference velocity and the reference position in the calculation cycle of the interpolator. In the coupled state, the maximum velocity and acceleration are always set to the machine parameters. The coupling can be interrupted if the reference value changes are too high. If a re-coupling is permitted (**RecoupleEnable**), the SAC interpolator will try to grasp the master value. Otherwise, an error is generated which can lead to an exception handling and finally to the coupling abort.

Upon deactivation of all master axis values (Master[m].Enable=FALSE) und the interpolation value (IpoEnable=FALSE) in the active SAC element, the slave axis decelerates with the delay permitted for the axis (Rigidity=1) or with AccCouple (Rigidity=0) until standstill.

With the deactivation of the (OnOff=0) system axis coupling, the SAC interpolator moves the slave axis to the interpolation position again. Once velocity and position correspond in the calculation cycle, the coupling is deactivated and the interpolation position becomes the direct axis specification again.

☞ The current slave axis position can be visualized in the MTX standard user interface as "$CommandPosition" (see documentation "MTX Standard NC Operation", chapter "Position Display").

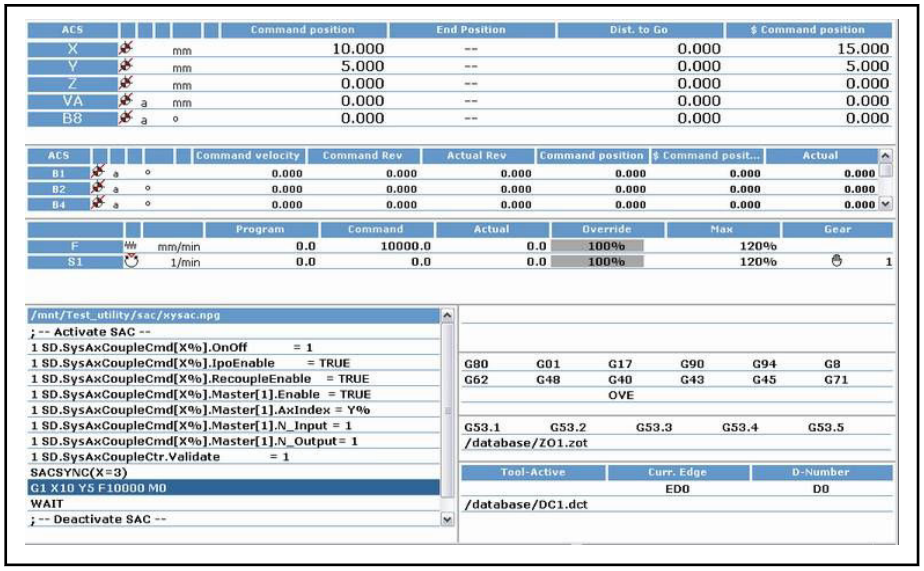


Fig. 14-21: Displaying the slave axis position with $CommandPosition

The system axis coupling shows the current coupling state in **SysAxCoupleSta[i].State**. Changes in the state can be started due to new system data specifications or by the SAC interpolator.

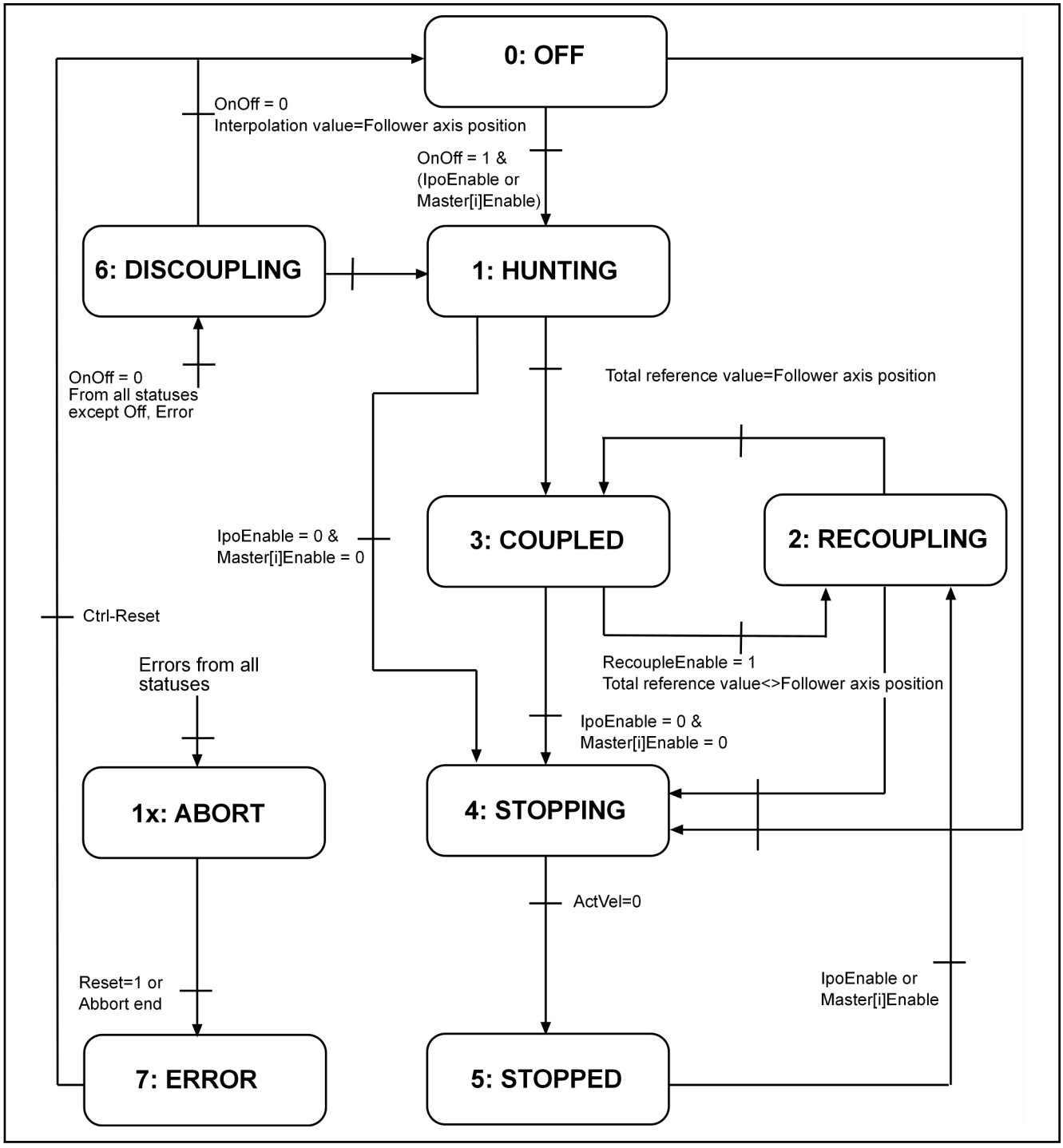


Fig. 14-22: State transitions of the system axis coupling

States of the system axis coupling:

0	OFF	-	The system axis coupling is inactive.
1	HUNTING	-	The SAC interpolator tries to reach the reference value for the first time.
2	RECOU-PLING	-	The SAC interpolator tries to re-establish the coupling status

3	COUPLED	-	Slave axis value and reference value correspond in the calculation cycle in position and velocity.
4	STOP-PING	-	All master values have been deactivated, the SAC interpolator stops the slave axis.
5	STOPPED	-	The slave axis has no reference value and stops.
6	DISCOU-PLING	-	The system axis coupling is to be deactivated, the SAC interpolator leads the axis back to the interpolation value.
7	ERROR	-	An exception occurred and the coupling was aborted.
1x	ABORT	-	An exception handling was triggered. The exception handling uses the SAC interpolator so that ABORT superimposes with the states 1-7 (e.g. 14 ABORT-STOPPING).
2x	PENDING	-	At least one conditional switching (SwitchCond > 0) is active and pending. The current system coupling state is superimposed until switching (e.g. 23 PENDING-COUPLED).

Tab. 14-6: States of the system axis coupling

Coupling ratios and determination of reference values:

The master axes can be coupled in the ratio "output revolutions (N_Output) / input revolutions (N_Input) plus a master axis offset using the master axis shift. The sum of all evaluated master axes with the axis value of the slave axis results in the total reference value:

$$\begin{aligned}\mathrm{SlaveAxis}_{\mathrm{ReferenceValue}} = \mathrm{SlaveAxis}_{\mathrm{InterpolatorValue}} \\ +(\mathrm{MasterAxisCommand\,position}_1+\mathrm{Shift}_1)\cdot\frac{\mathrm{N_Output}_1}{\mathrm{N_Input}_1}+\mathrm{Offset}_1 \\ +(\mathrm{MasterAxisCommandPosition}_2+\mathrm{Shift}_2)\cdot\frac{\mathrm{N_Output}_2}{\mathrm{N_Input}_2}+\mathrm{Offset}_2 \\ \ldots \\ +(\mathrm{MasterAxisCommandPosition}_5+\mathrm{Shift}_5)\cdot\frac{\mathrm{N_Output}_5}{\mathrm{N_Input}_5}+\mathrm{Offset}_5;\end{aligned}$$

Fig. 14-23: Reference value calculation

The coupling factors can also be specified as real numbers to consider shifts of linear master axes for example.



To couple two infinitely rotating rotary axes, always try to specify the ratio factors N_Output and N_Input, as integers. Otherwise, any representation error of a factor (14 digits) can accumulate to a measurable deviation after each master axis revolution.

During gearwheel machining such as hobbing cutting or gear shaping, the engagement area of the tool repeats for each tooth space. Thus, a faster coupling to any tooth space saves synchronization time. The user informs on the splitting of the gear wheel at the slave axis with the system data "Gear-Teeth". The control then couples to the adjacent partial area.

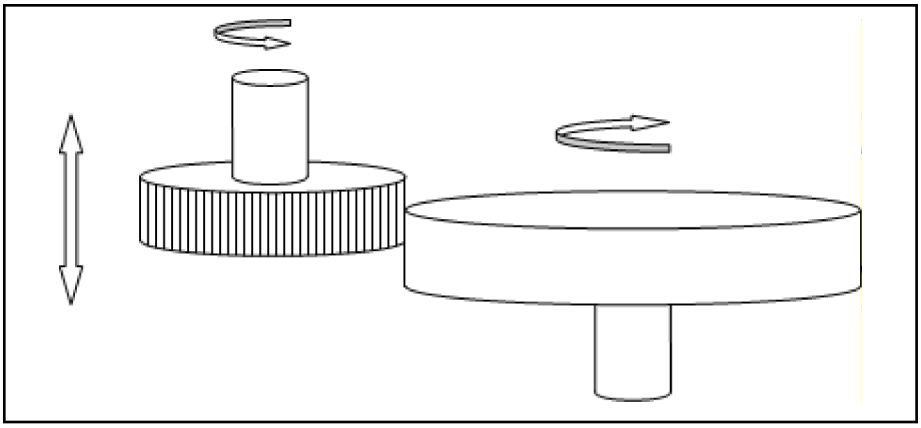


Fig. 14-24: Gear shaping

The gear blank is driven by the slave axis. The gear wheel is machined with 62 teeth. The tool has 42 teeth. Specify the gear ratio as N_Input = 62 and N_Output = -42. The splitting of the gear wheel is specified with GearTeeth = 62. The maximum position offset of the slave axis to the reference value is thus only half the partial area: 360°/(2*62) = 2.9032° instead of 180° with absolute coupling.

GearTeeth=0 forces an absolute coupling which stops when adding zero input revolutions of the master axis.

Master coupling with tables

Non-linear coupling relations between master axis and slave axis can be shown using the table coupling.

Specify a separate coupling table for every master axis (Master[m]) using the system axis coupling.

Write this table to the **SysAxCoupleCmd[i].Master[m].Tab**.

In case of an active table coupling, the total reference value of a slave axis results from:

$$SlaveAxis_{ReferenceValue} = SlaveAxis_{InterpolationValue}$$
$$+TableValue_1\left(MasterAxisCommandPosition_1 + Shift_1\right)\cdot\frac{N_Output_1}{N_Input_1} + Offset_1$$
$$+TableValue_2\left(MasterAxisCommandPosition_2 + Shift_2\right)\cdot\frac{N_Output_2}{N_Input_2} + Offset_2$$
$$\ldots$$
$$+TableValue_5\left(MasterAxisCommandPosition_5 + Shift_5\right)\cdot\frac{N_Output_5}{N_Input_5} + Offset_5$$

Fig. 14-25: Reference value calculation with tables

For a phase shift to the master axis position, use SysAxCoupleCmd[i].Master[m].Shift. This allows for example the use of a sine table to create the cosine function by a shift of 90°.

If this specification of the master table is missing, the following applies:

Table value$_i$= Master axis command position$_i$+Shift$_i$.

Structure and mode of action of coupling tables are described in detail in chapter 14.4 "Coupling tables" on page 524.

To use a table in the system axis coupling, it is to be loaded in the system table memory **SysAxCoupleTab** using the NC command **CoupleSplineTab(TAB(<table>),LOAD)**.

The loaded tables can be addressed via table name without path specification or via place number in the system table memory SysAxCoupleTab:

```
;Referencing the first table with the same name in SysAxCoupleTab
CST(TAB(/mnt/tables/stroke.fct),LOAD)
200 SysAxCoupleCmd[1].Master[1].Tab = "stroke.fct"
...
;Referencing table per place
CST(TAB(/mnt/tables/stroke.fct),LOAD,[ PLACE% ])
400 SysAxCoupleCmd[1].Master[2].Tab = $STR(PLACE%)
```

Example:

Replacing the mechanical cam coupling XB → YB with the system axis coupling:

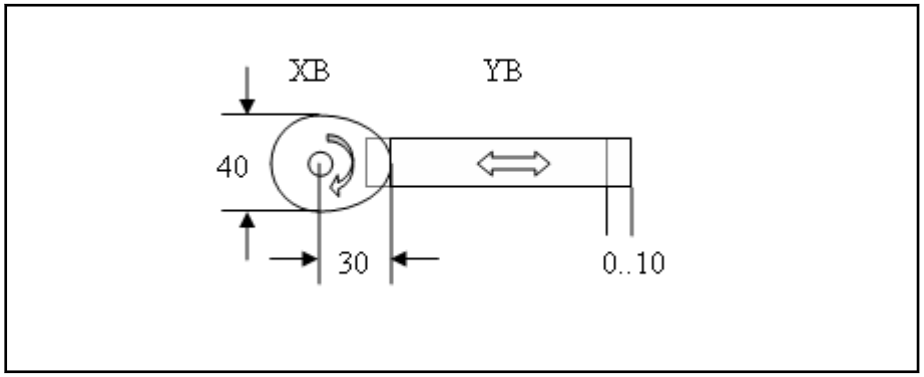


Fig. 14-26: Cam coupling

The XB angle is entered in the "nocken.fct" table file in a distance of 5° cam radii. For angles between the data points, the cam radius is to be cubically interpolated. An offset of -20.0 moves the YB-position to the defined interval [0..10] mm.

Table file "nocken.fct":

```
#1 3 ;Cubic interpolation
#20 1 ;Periodic table
#100 0.0 30.0 ;interpolation point
#100 5.0 29.73 ;interpolation point
#100 10.0 28.99;interpolation point
....
#100 360.0 30.0;interpolation point
```

Program:

```
;-------------------------------------------------------------
; Displacing the YB axis by a rotatory cam motion ; of the XB axis
;-------------------------------------------------------------
; System axis indices of the XB and YB axis: 1 XB%=4 : YB%=5 ...
;-------------------------------------------------------------
; 1. Table file loaded in the system table memory (cache):
;-------------------------------------------------------------
; CoupleSplineTab(TAB(nocken.fct),LOAD) ...
;-------------------------------------------------------------
; 2. Activating cam table in the coupling relation
;-------------------------------------------------------------
1 SD.SysAxCoupleCmd[YB%].Master[1].Enable   = TRUE
1 SD.SysAxCoupleCmd[YB%].Master[1].AxIndex  = XB%
1 SD.SysAxCoupleCmd[YB%].Master[1].Tab      = "nocken.fct"
1 SD.SysAxCoupleCmd[YB%].Master[1].Offset
                  = -20.0 1 SD.SysAxCoupleCmd[YB%].Master[1].N_Input
                  = 1.0 1 SD.SysAxCoupleCmd[YB%].Master[1].N_Output
                  = 1.0 1 SD.SysAxCoupleCtr.Validate=1 ...
```

```
;-------------------------------------------------------------
; 3. Synchronous motion of the YB axis with XB rotation
;-------------------------------------------------------------
; XB=S(100) ...
;-------------------------------------------------------------
; 4. Removing table from the coupling relation
;-------------------------------------------------------------
1 SD.SysAxCoupleCmd[YB%].Master[1].Enable   = FALSE
1 SD.SysAxCoupleCmd[YB%].Master[1].Tab      = ""
1 SD.SysAxCoupleCtr.Validate=1 ...
;-------------------------------------------------------------
; 5. Removing table from system table cache
;-------------------------------------------------------------
;CoupleSplineTab(TAB(nocken.fct),UNLOAD) ...
```

Creating and loading a spline table file:

Up to 100 different tables can be used at the same time in the system axis coupling. When the control is switched on, there are no tables in the memory anymore. The above specified conditions apply for the creation and use of the table

Refer to chapter 14.4 "Coupling tables" on page 524.

In the **SysAxCoupleTab[1..100]** system data, the storage location, time stamp, name of the loaded source file, table state and the size of the allocated table memory are displayed for each loaded table.

States of the coupling tables:

- **0 - Table memory not assigned:**

  The table memory is still empty and can be assigned with

  CoupleSplineTab(TAB(<tab>),LOAD).

- **1 - Table is loaded:**

  The loading for the table is not yet completed. Therefore, the table cannot be used.

- **2 - Table is loaded:**

  The table can now be used in the system axis coupling. That means that it is entered for at least one master in the

  SysAxCoupleCmd[axis].Master[m].Tab and activated using

  SysAxCoupleCtr.Valid=1.

  In this state, remove tables that are not used anymore with CoupleSplineTab(TAB(<tab>),UNLOAD).

- **3 - Table is used:**

  The table is used in the system axis coupling and cannot be removed from the table memory. Further master axis couplings can also use this table.

  If the table was deactivated in all the couplings, the table management applies the "2-table loaded" state again.

- **4 - Table is removed:**

  This state is assumed for a short time when removing the table with

  CoupleSplineTab(TAB(<tab>),UNLOAD)

System data definition of the table memory SysAxCoupleTab[i]:

Element	Type	Value range	Description
.FileName	String	1..100	File name with path specification of the source file
.State	Short	0..4	0 - Not assigned 1 - Table is loaded 2 - Table is loaded 3 - Table is used 4 - Table is removed
.FileTime	String	dd.mm.yy hh:mm:ss	Time stamp of the loaded source table file
.CacheSize	Int	-	Size of the assigned table memory

Tab. 14-7: Coupling table memory

The NC command is used

CoupleSplineTab(TAB(<Tab>,<1|0>**),LOAD**{,<Place>}**)** to prepare a table and load it to the table memory.

If no memory is available,

CoupleSplineTab(TAB(<Tab>,<1|0>**),UNLOAD**{,<Place>}**)**

can be used to remove the table that is not required anymore from the table.

The short form is: CST

With the parameters:

TAB <Tab>	Name of the coupling table
<1|0>	0 - Only prepare and load the table if the table file is newer than the table in the memory
	1 - Always prepare and load the table
LOAD	Load the table to system table memory
UNLOAD	Remove the table from the system table memory
Position	Loading and enabling the memory entry of the system table SysAxCoupleTab[Place]

Examples:

CST(TAB(curve.fct),LOAD)	If required, it creates the spline table /<Linkdirectory>/curve.fct.s. and loads the table to the system memory.
CST(,UNLOAD,5)	Remove table SysAxCoupleTab[5] from the system table memory

Formula couplings

Motion laws and geometry compensations that can be described using mathematic formulas can be directly specified with the formula processing of the system axis coupling in the system date **SysAxCoupleCmd[axis].Formula**.

Link individual reference values with mathematical operations and functions using the formula. An individual reference value is always defined by the system date SysAxCoupleCmd[axis].Master[m] and contains the scaling, shift and table analysis of the master axis position. The result of the formula is the slave axis reference value.

Example:

Compensation motion by system axis formula coupling

A tool has to follow the non-centered motion of a turned part with the radius "R" and the eccentricity "r" with regard to the workpiece axis **C** with the linear feed function **X**. The center of rotation of the C-axis is the coordinate origin of the x-axis.

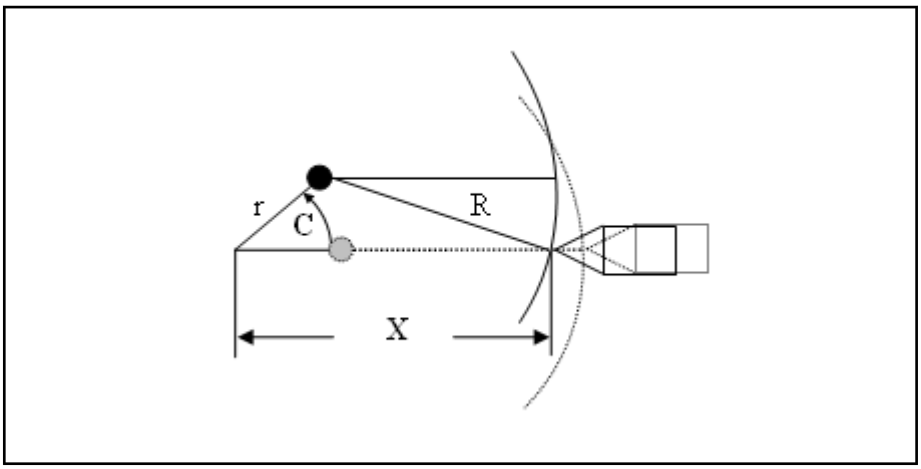


Fig. 14-27: Compensation of a eccentrically rotating turned part

The coupling law to deflect the x-axis as function of the C-axis is as follows:

$$X = r\cos(C) + R\sqrt{1-\left(\frac{r}{R}\right)^2 \sin^2(C)}$$

Fig. 14-28: Slider crank formula

The electronic gear can be directly defined using the system axis formula coupling:

Program:

```
; System axis indices for the master axis (C) and the slave axis (X)
10 X%=1 : C%=7

; master value definition (L1)
20 SD.SysAxCoupleCmd[X%].Master[1].AxIndex  = C%
30 SD.SysAxCoupleCmd[X%].Master[1].N_Input  = 1.0
40 SD.SysAxCoupleCmd[X%].Master[1].N_Output = 1.0

; modulo calculation in the C-axis to
; prevent overflow of the master value:
50 SD.SysAxCoupleCmd[X%].Master[1].Mode     = 1

; constant parameter: r=1.5, R=10
60 SD.SysAxCoupleCmd[X%].Formula="1.5*COS(L1)+10*SQRT(1-0.0225*SQR(SIN(L1)))"

; Aktivierung
70 SD.SysAxCoupleCtr.Validate = 1
```

In each interpolation cycle, a slave axis position x is determined using the formula from a time-discrete angle of the C-axis:

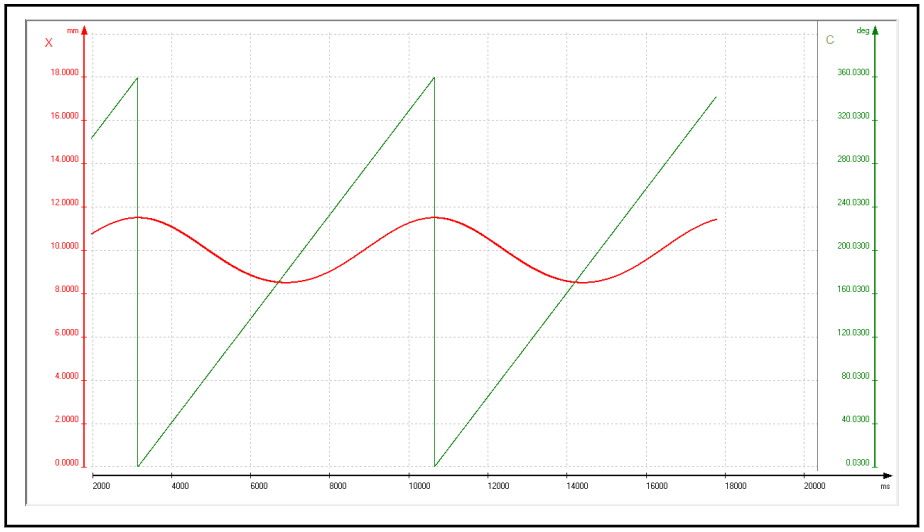


Fig. 14-29: Oscilloscope recording for the formula coupling

By replacing R with the slave axis $_{\text{interpolation value}}$ "AC", the programmed x-infeed of the tool can also be considered:

Program:

```
60 SD.SysAxCoupleCmd[X%].Formula="1.5*COS(L1)+AC*SQRT(1-SQR(1.5/AC*SIN(L1)))"
```

The standard reference value calculation uses the sum of the active individual reference values and generates the total reference value for the slave axis.

The formula processing substitutes the summing by the slave axis-specific calculation rule of the user:

$$SlaveAxis_{ReferenceValue} = Formula(AC, L1, .., L5)$$

with the operands

$$AC\,[mm/°] = SlaveAxis_{InterpolationValue}$$

$$L1[mm/°] = TableValue_1\left(MasterAxisCommandPosition_1 + Shift_1\right) \cdot \frac{N_Output_1}{N_Input_1} + Offset_1$$

...

$$L5[mm/°] = TableValue_5\left(MasterAxisCommandPosition_5 + Shift_5\right) \cdot \frac{N_Output_5}{N_Input_5} + Offset_5$$

Fig. 14-30:

Restrictions:

- If the formula processing is used very often for short NC cycle times, the interpolator can be overloaded.
- In formulas of the system axis coupling, only interpolation-synchronously changed permanent variables and system data are permitted. Variables of an interpolation-synchronous FA task and system data updated by the interpolator meet this condition. The formula string can be composed with the CPL string processing.
- The formula operations are executed in floating comma arithmetics with double accuracy. The maximum value range of all functions and operations is limited to +/-9.9999999999E99. When exceeding the maximum value range, the respective limit value is used and an overflow error is diagnosed. The smallest value permitted unequal to zero is +/- 1.0E-99. If it is fallen below this value range, the result is set to zero. This causes an error diagnostics for multiplications.

- Positions and reference values are calculated in mm (linear axes) or ° (rotary axes). Thus, velocities are given in mm/IPO cycle or °/IPO cycle. To standardize constant velocities and accelerations in the IPO cycle time, the functions FEED, SPEED and ACC are available.
- The trigonometric functions process angular units.
- Endlessly rotating master axes coupled to a linear axis using a formula can cause an overflow of the reference value if it is not included in the master axis modulo interval using SysAxCoupleCmd[axis].Master[m].Mode=1.
- Discontinuous or strongly changing function profiles of the formula cause the coupling to tear off when exceeding the maximum slave axis velocity of the maximum slave axis acceleration.
- The command position of the slave axis SPOS from the previous interpolation cycle can be used in formulas. Using SPOS can cause unwanted oscillations during the coupling.

Definition and activation of the formula processing

The formula is defined in the system date SysAxCoupleCmd[axis].Formula of the system axis coupling. The formula is stored as string for the respective slave axis.

Examples of formula definitions are:

Program:

```
;Default master value calculation
100 SD.SysAxCoupleCmd[X%].Formula="AC+L1+L2+L3+L4+L5"

;involute: x= r*(cos(phi)+phi*sin(phi)),
;           y= r*(sin(phi)-phi*cos(phi))
;           (for phi in [rad])
;           With r=10.0; phi=L1 [°], is the result:
200 SD.SysAxCoupleCmd[X%].Formula="10.0*(COS(L1)+L1/180*PI*SIN(L1))"
210 SD.SysAxCoupleCmd[Y%].Formula="10.0*(SIN(L1)-L1/180*PI*COS(L1))"

;Polynomial 4th degree: f(x) = ax^4 + bx^3 + cx^2 + dx + e
;           acc. to the Horner scheme with
;           x=L1, a=3.02E-12, b= -1.012E-10, c=4.25E-8, d=-1.33E-6 e=0.7E-3
300 SD.SysAxCoupleCmd[X%].Formula=
  "(((3.02E-12*L1-1.012E-10)*L1+4.25E-8)*L1-1.33E-6)*L1+0.7E-3"

;conditioned expression
400 SD.SysAxCoupleCmd[C%].Formula="ABS(L1)>0?ATAN(L2/L1):90.0"

;disable formula (add master values)
500 SD.SysAxCoupleCmd[C%].Formula=""
```

The formula is checked when the data is transferred (SysAxCoupleCtr.Validate=1) and prepared for the interpolator. Transfer errors are stored in SysAxCoupleCtr.CfgErrNum:

- 29 (syntax error)

  The formula syntax was not followed. Operands or operators are unknown or were programmed in an incorrect sequence. The abort position for syntactic formula errors is stored in the system date SysAxCoupleCtr.CfgErrMaster.
- 30 (runtime exceeded)

  The runtime measurement of all programmed formulas of the system axis coupling caused an invalid system load. Extend either the NC cycle time or use a hardware with higher performance.
- 31 (error during formula processing)

The nesting depth (e.g. parenthesis level) can be too big. The abort position is stored in the system data "SysAxCoupleCtr.CfgErrMaster".

The following CPL example shows a syntax error representation:

```
; Invalid character (#) in the formula definition:
100 SD.SysAxCoupleCmd[Y%].Formula="10.0#*(SIN(L1)-L1/180*PI*COS(L1))"
110 LP VALIDATE
...
; transfer program
LPS VALIDATE
210 SD.SysAxCoupleCtr.Validate = 1
220 ERRNO%  = SD.SysAxCoupleCtr.CfgErrNum
230 AXIS%   = SD.SysAxCoupleCtr.CfgErrAx
240 MASTER% = SD.SysAxCoupleCtr.CfgErrMaster
250 CASE ERRNO% OF
...
300 LABEL 29: ERRSTR$ = "Formula Syntax error"
310 LABEL 30: ERRSTR$ = "Formula runtime exhausted"
320 LABEL 31: ERRSTR$ = "Formula compile error"
...
410 IF MASTER% <> 0 THEN
420   IF (ERRNO% = 29) OR (ERRNO% = 31) THEN
430     FORMEL$=SD.SysAxCoupleCmd[AXIS%].Formula
440     ERRSTR$="SAC("+STR$("##",AXIS%)+") "+MID$(FORMEL$,1,MASTER%)+"^
                                  "+ERRSTR$
...
510 IF ERRNO% <> THEN
520   SETERR(ERRSTR$)
...
```

The transfer program VALIDATE generates the part program error "SAC(5) 10.0#^ Formula syntax error".

Basic value range of the formula processing

The value range for all mathematical functions and operations is limited to the intervals [-9.9999999999E99 .. -1.0E-99], [1.0E-99 .. 9.9999999999E99] and 0. This range is called **basic value range** [BW].

Diagnostics and runtime errors of the formula processing

An enabled formula does not provide a usable result in certain error situations due to the input data and the arithmetic operations. In this case, the coupling can be completed with a defined abort handling. The system axis coupling indicates formula processing errors in the system date **SysAxCoupleSta[axis].FormulaDiag**:

0 - No error

1 - Zero division

A zero division occurred. The formula processing continues with the operation result "zero".

2 - Overflow error

The valid basic value range [BW] for mathematical operations and functions was exceeded. The formula processing continues as operation result with the exceeded limit value (+/-9.9999999999E99).

3 - Definition range violation

The mathematical operation is not defined for the argument.

Example: "…ASIN(3.0)…".

The formula processing continues with the function-specific substitute result.

4 - Underflow error

The result of a multiplication or exponentiation is unequal to zero and lower than the valid basic value range with regard to the amount:

0 < | Expression1 * Expression2 | < 1.0E-99

The formula processing continues with the operation result "zero".

5..9 - Formula interpretation error

The specified formula cannot be processed completely. The result of the formula is set to zero. These errors indicate an internal problem. Report it to the customer service.

The formula errors 1, 2, 3, 4 and 5-9 can be enabled individually for the abort handling using SysAxCoupleCmd[axis].AbortEnable Bit16-Bit20.

The operator position in the formula string where the error occurred is stored in the system date **SysAxCoupleSta[axis].FormulaDiagPos**.

In case of runtime formula errors, the error code 7 – FORMULA_ERROR is stored additionally in the system date SysAxCoupleSta[axis].Error.

To check intermediate results in a formula, the WATCH function is provided in the formula string.

Program:

```
1 SD.SysAxCoupleCmd[X%].Formula="10.0*(COS(L1)+WATCH(L1*SIN(L1)))"
```

The function stores the result of the transferred expression in the system date SysAxCoupleSta[axis].FormulaWatch and returns the result for a further calculation. In the example, the result of the expression "L1*SIN(L1))“ is stored in FormulaWatch.

Operands of the formula processing

In formulas of the system axis coupling, only numeric constants, real-time variables and operands and system constants provided by the system axis coupling may be used:

- Reference values (AC,L1,..L5)
- Master axis velocities (V1,…,V5)
- Latest command position of the slave axis (SPOS)
- System constants (PI,TI)
- Numeric constants (+/-nnn.nnnE+/-nn)
- Real-time variables (@<var>, SD.<var>)

The interpolation value of the slave axis AC and the reference values of the master connections SysAxCoupleCmd[axis].Master[m] are specified in [mm] or [°]. The unscaled reference velocities Vm of the master axes refer to the position difference per interpolation cycle.

The following table shows the possible formula operands:

Operand	Description	Unit
AC	Interpolation value of the slave axis (axis coordinate); at a locked interpolation value (SysAxCmd[axis].IpoEnable = FALSE) returns AC=0	[mm,°]
L1 ... L5	Reference value of the master axis: $L_m = Tab_m\left(Pos_m + Shift_m\right) \cdot \frac{N_Output_m}{N_Input_m} + Offset_m$ Master axes (SysAxCmd[axis].Master[m].Enable = FALSE) returns Lm=0	[mm,°]
PI	Ludolph's number π	-
SPOS	$_{\text{Command position}}$ of the slave axis of the previous Ipo cycle	[mm,°]
TI	NC cycle time	[s]
V1 ... V5	Velocity of the master axis m: V_m = Command position of the master $axis_{m\ \text{Tipo}}$ - Command position of the master $axis_{m\ \text{Tipo - 1}}$	$\left[\frac{mm,^\circ}{T_{Ipo}}\right]$
<Constant>	Integer or double precise Real constant in the basic value range Examples: -1, 5, 1.4142135623731, -1.0E-50	-
<Realtime variable>	Valid realtime variables: • Permanent variables (@<name>) of type – INT – REAL – DOUBLE • System data (SD.<name>) of type – Boolean_t – Byte_t, UnsignedByte_t – Short_t, UnsignedShort_t – Int_t, UnsignedInt_t – Float_t – Double_t Field variables have to be indexed using constants Examples: @10, @Offset, @PermArray(1,3) SD.PlcFA.Offset, SD.SysAxCoupleSta[2].ActPos	-

Tab. 14-8: Formula operands

☞ Real-time variables used in the formula coupling can only be written synchronously to the interpolation, e.g. by an FA task (see documentation "MTX Functional Description - Special Functions", chapter "Fast actions")! In case of non-compliance, invalid variable values can be read sporadically!

Exclusively use variables that are changed interpolation-synchronously.

Formula operations

Formulas of the system axis coupling permit:

- Fundamental operations ("+", "-", "*", "/", MOD)
- Mathematical functions (such as SIN, SQRT)
- Comparisons ("=", "<>", "<", ">", "<=", ">=")
- Logic operators (NOT, AND, OR)
- Conditional expressions (condition ? Expression1 : Expression2)

Consecutive operations are subject to priority rules (such as multiplication and division first, then addition and subtraction) if not overruled by their setting in parenthesis:

Operator	Rang	Linking at same rank	Example	Design
(Expression)	1	-	(1.0 - L1) * 3	(1.0 - L1) * 3
"-" sign	2	Left	- - -3	-(-(-3))
"*"," /", MOD	3	left	1.0 * 20 MOD 2	(1.0 * 20) MOD 2
"+", "-"	4	Left	L2 + L3 - L5	(L2 + L3) - L5
Comparison ("=", "<>", "<", ">", "<=", ">="	5	Not permitted	1 < 2 < 3	Syntax error
NOT	6	right	2 OR NOT NOT 1	2 OR NOT(NOT 1)
AND	7	Left	0 AND 1 AND 2	(0 AND 1) AND 2
OR	8	Left	0 OR 1 OR 2	(0 OR 1) OR 2
"?", ":"	9	Right	1?2:3?4:5	1?2:(3?4:5)

Tab. 14-9: Ranking of the formula operators

Mathematical calculations

In the formula expression, operands are linked by mathematical operators and functions considering the priority rules. The operands Op as well as Op1 and Op2 in the following tables can represent either simple operands (constants, reference values) or composed expressions. The basic value range [BW] is obligatory for all calculations. MAX names the amount maximum (9.9999999999E99) and MIN the amount minimum (1.0E-99) in the basic value range [BW]. For some operations, the integer range from -2147483647 (MININT).. 2147483647 (MAXINT) applies.

Operators for the basic arithmetics:

Syntax	Description	Definition range	Result value range	Diagnostics: Substitute result
Op1 + Op2 Op1 + Op2	Addition Subtraction	Op1 and Op2 in [BW]	[BW]	Overflow: +/- MAX
Op1 + Op2	Multiplication	Op1 and Op2 in [BW]	[BW]	Overflow: +/- MAX Underflow: 0
Op1 / Op2	Division	Op1 in [BW], Op2 in [BW] without zero	[BW]	Overflow: +/- MAX Zero division: 0
Op1 MOD Op2	Modulo	Op1 in [BW], Op2 in [BW] without zero	[BW]	Zero division: 0

Tab. 14-10: Basic formula arithmetics

Amount, integer and signum function:

Syntax	Description	Definition range	Result value range	Diagnostics: Substitute result
ABS(Op)	| Op |	Op in [BW]	[BW]	-
INT(Op)	Integer (Op)	Op in [MININT.. MAXINT]	[MININT.. MAXINT]	Overflow Op < MININT: MININT Op > MAXINT: MAXINT
SGN(Op)	Signum Op < 0 : -1 Op = 0 : 0 Op > 0 : 1	Op in [BW]	[-1,0,1]	-

Tab. 14-11: Formula amount and integer arithmetics

To standardize feeds, speeds and accelerations in the interpolation cycle time, the functions FEED, SPEED and ACC are available. Length specifications in inch can be converted into mm with "INCH".

Standardization functions:

Syntax	Description	Definition range	Result value range	Diagnostics: Substitute result
ACC(Op)	Converts acceleration from m/s^2 to mm/IPO cycle^2	Op in [BW]	[BW]	-
FEED(Op)	Converts feed from mm/min to mm/IPO cycle	Op in [BW]	[BW]	Overflow: +/- MAX
INCH(Op)	Converts inch into mm	Op in [BW]	[BW]	Overflow: +/- MAX
SPEED(Op)	Converts speed from rpm to °/IPO cycle	Op in [BW]	[BW]	Overflow: +/- MAX

Tab. 14-12: Formula standardization functions

The trigonometric functions SIN, COS and TAN process the angular unit in degrees. ASIN, ACOS and ATAN correspondingly provide the angle in degree.

Trigonometric functions

Syntax	Description	Definition range	Result value range	Diagnostics: Substitute result
ACOS(Op)	Arcus cosine	Op in [-1..+1]	[0..180]	Undefined Op < -1 : 0 Op > 1 : 180
ASIN(Op)	Arcus sine	Op in [-1..+1]	[-90..90]	Undefined Op < -1 : -90 Op > 1 : 90
ATAN(Op)	Arcus tangent	Op in [BW]	-90]..[90	-
COS(Op)	Cosine	Op in [BW]	[-1..1]	-

Syntax	Description	Definition range	Result value range	Diagnostics: Substitute result
SIN(Op)	Sine	Op in [BW]	[-1..1]	-
TAN(Op)	Tangent	Op Period map in -90]..[90	[BW]	Undefined Op <= 90 : TAN(-MAXPHI) Op >= 90 : TAN(MAXPHI) with MAXPHI= 90 - 1.0E-14

Tab. 14-13: Trigonometric functions

Trigonometric hyperbolic and inverse hyperbolic functions

Syntax	Description	Definition range	Result value range	Diagnostics: Substitute result
ACOSH(Op)	Area hyperbolic cosine	Op in [1..Max]	[BW]	Undefined Op < 1 : 0
ASINH(Op)	Area hyperbolic sine	Op in [BW]	[BW]	-
ATANH(Op)	Area hyperbolic tangent	Op in -1]..[1	[BW]	Undefined Op <= -1 : ATANH(-MAXP) Op >= 1 : ATANH(MAXP) with MAXP= 1.0 - 1.0E-16
COSH(Op)	Cosine hyperbolic	Op in [-230..230]	[1..MAX]	Overflow: COSH(230)
SINH(Op)	Sine hyperbolic	Op in [-230..230]	[BW]	Overflow Op < -230 : SINH(-230) Op >230 : SINH(230)
TANH(Op)	Tangent hyperbolic	Op in [BW]	-1]..[1	-

Tab. 14-14: Hyperbolic functions

Functions to determine the polar coordinates

Use ATAN2 and DIST to convert from Cartesian coordinates into polar coordinates:

Syntax	Description	Definition range	Result value range	Diagnostics: Substitute result
ATAN2(Op1, Op2)	Coordinate angle atan2(y,x)	Op1 and Op2 in [BW]	[-180..+180]	-
DIST(Op1, Op2)	Hypotenuse $\sqrt{}$ ($Op1^2$ + $Op2^2$)	Op1 and Op2 in [BW]	[BW]	-

Tab. 14-15: Polar coordinate determination

Power functions, root functions and logarithms:

Syntax	Description	Definition range	Result value range	Diagnostics: Substitute result
CBRT(Op)	Cubic root $\sqrt[3]{}$ Op	Op in [BW]	[BW]	-
CUB(Op)	Op^3	Op in [BW]	[BW]	Overflow: +/- MAX Underflow: 0
EXP(Op)	e^{Op}	Op in [-MAX.. 230]	[0..Max]	Overflow: EXP(230)
EXP(Op1, Op2)	$Op1^{Op2}$	1. Op1<0, Op2 of N 2. Op1 or Op2 <> 0 3. |ln(|Op1|)*Op2| <230	[BW]	Undefined 0^0: 0 $-z^{n.nnn}$: 0 (complex) Overflow: +/- MAX
LN(Op)	Logarithm to the basis e	Op in [MIN..MAX]	[-227.956.. 230.259]	Undefined Op < MIN : LN(MIN)
SQR(Op)	Op^2	Op in [BW]	[BW]	Overflow: MAX Underflow: 0
SQRT(Op)	$\sqrt{}$Op	Op in [0..MAX]	[0..Max]	Undefined Op < 0 : 0

Tab. 14-16: Arithmetic functions

Conditional expressions

Conditional expressions ensure definition ranges of mathematical function or perform range-dependent couplings. The condition can include comparisons and logic operations.

Example:

The endless axis C [0°..360°] is to deflect the linear axis X over the range [0..100mm] in the interval 0°..180° with a sine^2 function:

Program:

```
1 SD.SysAxCoupleCmd[X%].Master[1].AxIndex  = C%
...
1 SD.SysAxCoupleCmd[X%].Formula="L1>=0 AND L1<=180?SQR(SIN(L1))*100:0.0"
...
```

The reference value 0.0 is provided outside the range:

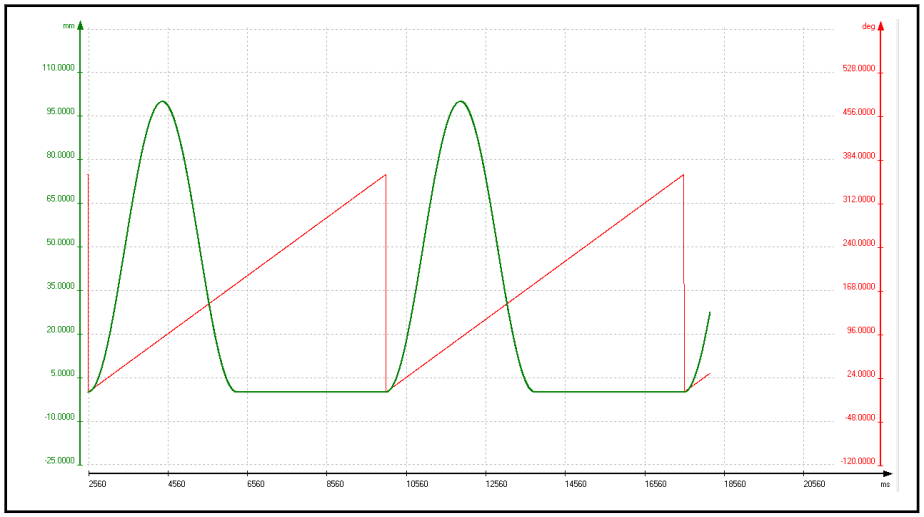


Fig. 14-31: Conditional formula coupling

The following tables show the possible comparative relations and logic operations:

Relations:

Syntax	Description	Definition range	Result value range	Diagnostics: Substitute result
Op1 = Op2	verifies equality |Op2-Op1| < 1E-13 : 1 |Op2-Op1| >= 1E-13 : 0	Op1 and Op2 in [BW]	0.1	-
Op1 <> Op2	verifies inequality |Op2-Op1| >= 1E-13 : 1 |Op2-Op1| < 1E-13 : 0	Op1 and Op2 in [BW]	0.1	-
Op1 < Op2 Op1 <= Op2 Op1 >= Op2 Op1 > Op2	Op1 <Op2 Op1 <=Op2 Op1 >= Op2 Op1 >Op2	Op1 and Op2 in [BW]	0.1	-

Tab. 14-17: Formula relations

Logic operators

Syntax	Description	Definition range	Result value range	Diagnostics: Substitute result
EMERGENCY OP	Negates expression	OP in [MININT.. MAXINT]	0.1	Overflow: 0
Op1 AND Op2	Logic AND: |Op1| >= 1 and |Op2| >= 1 : 1 Otherwise: 0	Op1 and Op2 [MININT.. MAXINT]	0.1	Overflow MAXINT limits Op1 and Op2
Op1 OR Op2	Logic OR: |Op1| >= 1 or |Op2| >= 1 : 1 Otherwise: 0	Op1 and Op2 [MININT.. MAXINT]	0.1	Overflow MAXINT limits Op1 and Op2

Tab. 14-18: Logic operators

Logic expressions especially require the condition of the "?" - ":" operation.

Conditional expressions

Syntax	Description	Definition range	Result value range	Diagnostics: Substitute result
Op1?Op2:Op3	Op1<>0: Op2 otherwise: Op3	Op1,Op2 and Op3 in [BW]	[BW]	-
MIN(Op1,Op2)	Op1<Op2: Op1 Otherwise: Op2	Op1 and Op2 in [BW]	[BW]	-
MAX(Op1,Op2)	Op1>Op2: Op1 Otherwise: Op2	Op1 and Op2 in [BW]	[BW]	-

Tab. 14-19: Conditional expressions

Complex interpolation groups can be applied using the formula coupling. Note that the slave axes cannot limit the master axes. Overloads with regard to velocity and acceleration always cause a decoupling of the slave axis and thus a violation of the formula specification. Especially conditional expressions can cause discontinuities of the reference value. The control tries to reach the reference value again after the decoupling (RECOUPLING) if no error handling (AbortEnable) was enabled.

Formula interpolation groups of the system axis coupling have no path control. Path velocities and path accelerations always result implicitly by the master axis specifications and the slave axes moved along due to the coupling rule.

Synchronization mode for slave axis:

The slave axis can be coupled to the specified reference value in several ways.

Especially for modulo axes there are two methods to reach the synchronous position:

- By exceeding the slave axis velocity compared to the master velocity, the advanced synchronous position is caught up.
- By reducing the slave axis velocity, the slave axis waits for the next lagging synchronous position.

The time required for the slave axis to catch up depends additionally on the maximum velocity and acceleration of the slave axis. Thus, the shortest path is not always the fastest one.

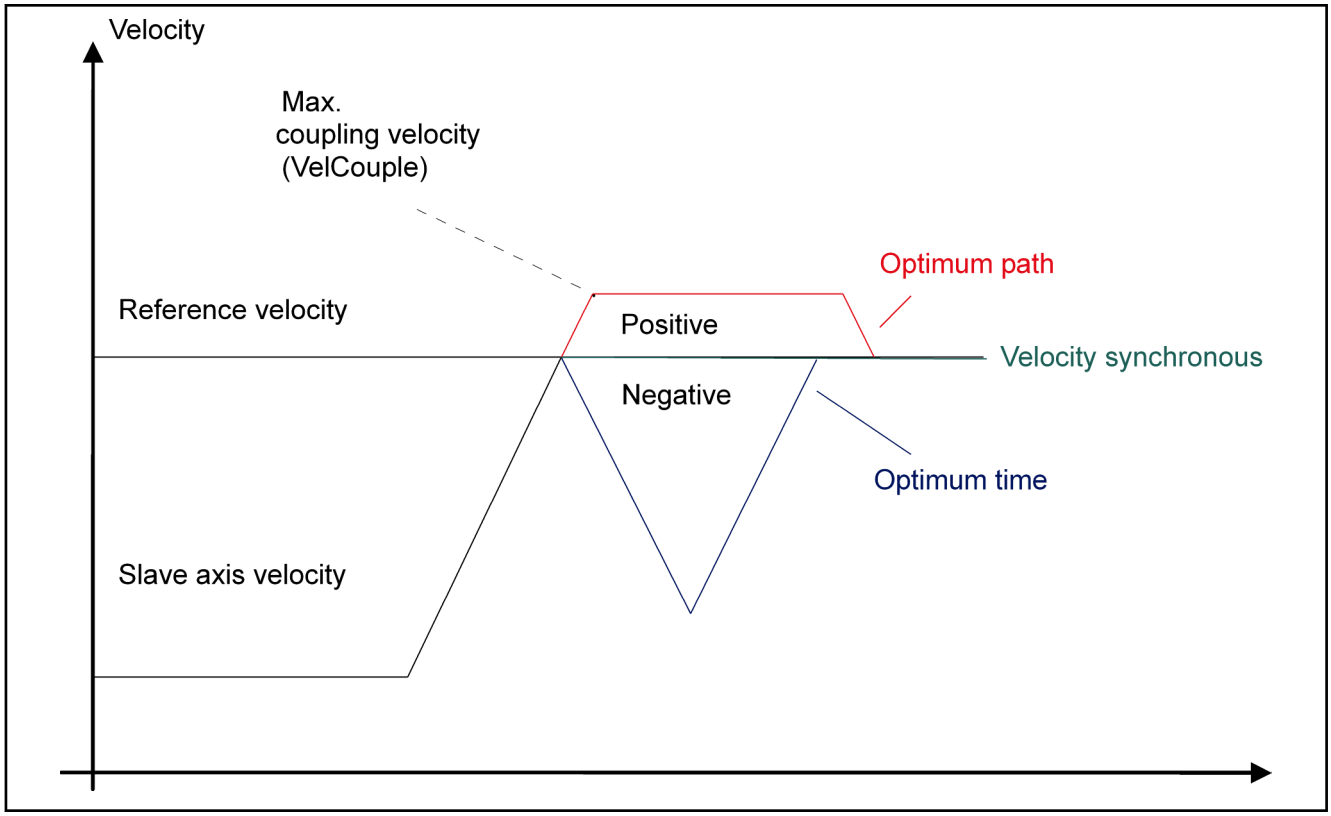


Fig. 14-32: Alternative coupling for modulo axes

The following coupling types are distinguished:

- Velocity compensation

  Instead of a position compensation, an offset of the master velocity is saved and carried along. This is the fastest coupling type that also couples and decouples the master axes and the slave axis without any motion during standstill.

  Upon deactivation of the coupling (SysAxCoupleCmd[i].OnOff=0), the axis maintains a relative coupling offset between the interpolation position (target position) and axis position ($target position). In the channel, this offset can be removed using the NC command "SACSYNC".

- Optimum path

  The shortest path is used for the position compensation.

  (Note: If the velocity is not limited, it is also fastest path)

and only for modulo axes:

- Optimum time

  The fastest path is selected considering the specified slave axis limitations.

- Positive

  The position compensation is performed out in positive velocity direction.

- Negative

  The position compensation is performed in negative velocity direction.



The synchronization mode influences the slave axis motion when coupling and decoupling as well as when activating and deactivating the coupling. With the path-optimized, time-optimized or direction-optimized synchronization, there is - in case of a position lag caused by a coupling - a compensation motion in case of an axis standstill.

Select the SyncMode "velocity-synchronous" to avoid the compensation motion in case of an axis standstill. After deactivation of the coupling, transfer the axis position into the channel of the axis using the SACSYNC NC command.

The synchronization mode is specified for the respective slave axis in the system date **SysAxCoupleCmd[i].SyncMode**.

Note:

0 - Shortest path

1 - Optimum path with regard to time

2 - Velocity-synchronous

3 - Positive path

4 - Negative path

For the velocity coupling (2 - velocity-synchronous), the slave axis has - in case of synchronism - an offset by one with regard to the absolute reference value. This offset is stored in **SysAxCoupleSta[i].ActOffset** and can for example be used to offset a repeated absolute coupling. The reference value offset is updated after each coupling change of the respective slave axis. Thus, it is to be stored in the synchronous state.



After a decoupling in the velocity mode, the synchronous position is lost. Even if offsetting the reference value shift, a position-specific coupling cannot be restored for the modulo axes. Using the reference value offset, a tooth-exact coupling can be performed for any tooth if there is a tooth pitch (GearTeeth > 1).

Program:

```
;-------------------------------------------------
; Couple slave axis CL velocity synchronous to
; B master axis
;-------------------------------------------------
1 SD.SysAxCoupleCmd[CL%].SyncMode          = 2
1 SD.SysAxCoupleCmd[CL%].Master[1].Enable   = TRUE
1 SD.SysAxCoupleCmd[CL%].Master[1].AxIndex  = B%
1 SD.SysAxCoupleCmd[CL%].Master[1].Offset   = 0
1 SD.SysAxCoupleCmd[CL%].Master[1].N_Input  = 21
1 SD.SysAxCoupleCmd[CL%].Master[1].N_Output = -1
;-------------------------------------------------
; Data transfer
;-------------------------------------------------
CALL SAC_VALID
;-------------------------------------------------
; Wait synchronism
;-------------------------------------------------
CALL SAC_LEADSYNC[CL%]
;-------------------------------------------------
; Store recoupling reference value offset
;-------------------------------------------------
OFFSET! = SD.SysAxCoupleSta[CL%].ActOffset
;-------------------------------------------------
; Command and decouple CL axis
;-------------------------------------------------
...
1 SD.SysAxCoupleCmd[CL%].Master[1].Enable   = FALSE
...
;-------------------------------------------------
; time-optimum and teeth-exact recoupling
```

```
;-------------------------------------------------
1 SD.SysAxCoupleCmd[CL%].SyncMode           = 1
1 SD.SysAxCoupleCmd[CL%].GearTeeth          = 21
1 SD.SysAxCoupleCmd[CL%].Master[1].Enable   = TRUE
1 SD.SysAxCoupleCmd[CL%].Master[1].Offset   = OFFSET!
...
```

Synchronous run monitoring of the slave axis:

Synchronous run behavior in the group is decisive for the processing quality since deviations from the coupling relation have a direct influence on the workpiece accuracy. For monitoring purposes, the ACTUAL master value is compared th the ACTUAL slave axis position (**SysAxCoupleCmd[i].PosWinSel**: 1,2,3). The ACTUAL master value is calculated from the ACTUAL position of the master axes, assessed by the coupling relation. The last command value of virtual master axes is always included in the comparison. The current difference of the ACTUAL master value to the ACTUAL slave axis position is saved in **SysAxCoupleSta[i].Lag**.

A command value monitoring can be configured as an alternative to the ACTUAL value monitoring (**SysAxCoupleCmd[i].PosWinSel**: 11,12,13). The control checks if the slave axis can follow the master value within the framework of the specified accuracy limit by complying with the maximum velocity and acceleration. For example, this monitoring allows to tolerate an interrupted coupling with re-coupling (RecoupleEnable) only within the accuracy limit.

Two accuracy limits can be saved in SysAxCoupleCmd[i].PosWin[1..2]. SysAxCoupleCmd[i].PosWinSel selects the active accuracy limit as follows:

SysAxCoupleCmd[i]. PosWinSel	Description
0	Synchronous run monitoring disabled.
1	Synchronous run monitoring with accuracy limit 1 selected (PosWin[1]).
2	Synchronous run monitoring with accuracy limit 2 selected (PosWin[2]).
3	Synchronous run monitoring with both accuracy limits selected (PosWin[1] and PosWin[2], with hysteresis behavior). This mode is for example useful if there is a strong noisy actual position value.
11	Command value monitoring with accuracy limit 1 selected (PosWin[1]).
12	Command value monitoring with accuracy limit 2 selected (PosWin[2]).
13	Command value monitoring with the accuracy limits 1 and 2 selected (PosWin[1] and PosWin[2], with hysteresis behavior

Tab. 14-20: Selecting the accuracy limit of the synchronous run monitoring

The selected synchronous run monitoring is only active if there is one of the following states in the status element SysAxCoupleSta[i].State:

SysAxCoupleSta[i].State	Description
1	Initial coupling (HUNTING)

2	Recoupling (RECOUPLING)
3	Command values coupled (COUPLED)

Tab. 14-21: Prerequisite for the synchronous run monitoring

If the synchronous run monitoring is active, the control displays the result of the synchronous run monitoring in the status element **SysAxCoupleSta[i].InPos** (1 = Synchronous run, 0 = no synchronous run).

The falling edge of the InPos Signals results in error INPOS_VIOLATION (8) and is displayed in **SysAxCoupleSta[i].Error**. For possible exception handling, refer to chapter 14.5.6 "Exception Handling" on page 567.

For more information on the result of the synchronous run monitoring, refer to the status element SysAxCoupleSta[i].StateAddOn:

Mode of the synchronous run monitoring	Result of the synchronous run monitoring	
	SysAxCoupleSta[i].InPos (1 = synchronous run, 0 = no synchronous run)	SysAxCoupleSta[i]. StateAddOn
Inactive (SysAxCoupleCmd[i]. PosWinSel = 0 or SysAxCoupleSta[i]. State ≠ 1, 2, 3)	0	0
one ACTUAL accuracy limit is active (SysAxCoupleCmd[i]. PosWinSel = 1 or 2)	1: Difference between the actual reference value and the actual slave axis position is in the selected accuracy limits 1 or 2 (<= PosWin[<PosWinSel>]) 0: otherwise	0: No additional information 1000: Accuracy limit 1 is complied with 2000: Accuracy limit 2 is complied with
both ACTUAL accuracy limits are active (SysAxCoupleCmd[i]. PosWinSel = 3)	Hysteresis response 1: Difference between the actual reference position and the actual slave axis position is in the two accuracy limits 1 and 2 (<= PosWin[1] and <= PosWin[2]) 0: Difference between the actual reference position and the actual slave axis position is not in the two accuracy limits 1 and 2 (> PosWin[1] and > PosWin[2]) No change: Difference between the actual reference value and the actual slave axis position is in the accuracy limits 1 and 2	0: No additional information 1000: Accuracy limit 1 is complied with 2000: Accuracy limit 2 is complied with 3000: Accuracy limits 1 and 2 are maintained

one command value accuracy limit is active (SysAxCoupleCmd[i]. PosWinSel = 11 or 12)	1: Difference between the slave axis master value and the slave axis position is within the selected accuracy limits 1 or 2 (<= PosWin[<PosWinSel>]) 0: otherwise	0: No additional information 1000: Accuracy limit 1 is complied with 2000: Accuracy limit 2 is complied with
both command value accuracy limits are active (SysAxCoupleCmd[i]. PosWinSel = 13)	Hysteresis response 1: Difference between the slave axis master value and the slave axis position is within the two accuracy limits 1 and 2 (<= PosWin[1] and <= PosWin[2]) 0: Difference between the slave axis master value and the slave axis position is not in the two accuracy limits 1 and 2 (> PosWin[1] and > PosWin[2]) No change: Difference between the slave axis master value and the slave axis position is within the accuracy limits 1 and 2	0: No additional information 1000: Accuracy limit 1 is complied with 2000: Accuracy limit 2 is complied with 3000: Accuracy limits 1 and 2 are maintained

Tab. 14-22: Result of the synchronous run monitoring

Examples of synchronous run monitoring

Program:

```
; example actual position monitoring of B1 axis with hysteresis:
1 SD.SysAxCoupleCmd[B1%].PosWinSel   = 3   : REM actual position monitoring
1 SD.SysAxCoupleCmd[B1%].PosWin[1]   = 0.6 : REM Inpos=0 (win1 will be left)
1 SD.SysAxCoupleCmd[B1%].PosWin[2]   = 0.3 : REM InPos=1 (win2 will be reached)

; example command position monitoring of ZA axis:
1 SD.SysAxCoupleCmd[ZA%].RecoupleEnable = true     : REM supress error UNCOUPLED(3)
1 SD.SysAxCoupleCmd[ZA%].PosWinSel      = 11       : REM command position monitoring
1 SD.SysAxCoupleCmd[ZA%].PosWin[1]      = 0.05     : REM ensure distance <= 0.05 mm
1 SD.SysAxCoupleCmd[ZA%].AbortEnable    = 16777216 : REM bit #24, INPOS_VIOLATION(8)
```

14.5.3 Activation/deactivation

The prepared data from **SysAxCoupleCmd[i]** is simultaneously transferred to the interpolator using the control structure **SysAxCoupleCtr**. The following CPL example program shows the data transfer with subsequent error analysis:

Program:

```
SACVALID

1 DIM ERRSTR$(100)
WAIT
1 SD.SysAxCoupleCtr.Validate=1
1 ERRNO%  = SD.SysAxCoupleCtr.CfgErrNum
1 AXIS%   = SD.SysAxCoupleCtr.CfgErrAx
1 MASTER% = SD.SysAxCoupleCtr.CfgErrMaster
1 CASE ERRNO% OF
1   LABEL 0:  ERRSTR$ = "NoError"
1   LABEL 1:  ERRSTR$ = "Circularity"
1   LABEL 2:  ERRSTR$ = "Max Depth"
1   LABEL 3:  ERRSTR$ = "Illegal Axis"
```

```
1   LABEL 4:  ERRSTR$ = "Master enable without AxIndex"
1   LABEL 5:  ERRSTR$ = "Master enable without N_INPUT"
1   LABEL 6:  ERRSTR$ = "Master enable without N_OUTPUT"
1   LABEL 7:  ERRSTR$ = "PosWinSel out of range [0..2]"
1   LABEL 8:  ERRSTR$ = "ASUP Number out of range [0,1..8]"
1   LABEL 9:  ERRSTR$ = "ASUP Channel invalid [0,1..MAX_CHAN]"
1   LABEL 10: ERRSTR$ = "Invalid AbortAction combination"
1   LABEL 11: ERRSTR$ = "Invalid SyncMode for axis type"
1   LABEL 12: ERRSTR$ = "Invalid VelCouple (NAN)"
1   LABEL 13: ERRSTR$ = "Invalid AccCouple (NAN)"
1   LABEL 14: ERRSTR$ = "Invalid N_Input (NAN)"
1   LABEL 15: ERRSTR$ = "Invalid N_Output (NAN)"
1   LABEL 16: ERRSTR$ = "Invalid Offset (NAN)"
1   LABEL 17: ERRSTR$ = "Invalid RetractPos (NAN)"
1   LABEL 18: ERRSTR$ = "Invalid PosWin[1] (NAN)"
1   LABEL 19: ERRSTR$ = "Invalid PosWin[2] (NAN)"
1   LABEL 20: ERRSTR$ = "Tab not found"
1   LABEL 21: ERRSTR$ = "Invalid SwitchCond"
1   LABEL 22: ERRSTR$ = "Invalid SwitchMin (NAN)"
1   LABEL 23: ERRSTR$ = "Invalid SwitchMax (NAN)"
1   LABEL 24: ERRSTR$ = "Invalid switch range"
1   LABEL 25: ERRSTR$ = "Invalid tab/master modulo range to slave axis"
1   LABEL 26: ERRSTR$ = "Invalid master modulo range to tab"
1   OTHERWISE ERRSTR$ = "Unkown system axis couple configuration error"
1 ENDCASE
1 IF MASTER% = 0 THEN
1   ERRSTR$="SAC("+STR$("##",AXIS%)+"):"+ERRSTR$
1 ELSE
1   ERRSTR$="SAC("+STR$("##",AXIS%)+")"+STR$(".Master(#):",MASTER%)+ERRSTR$
1 ENDIF
1 IF ERRNO% <> 0 THEN
1   SETERR(ERRSTR$)
1 ENDIF
```

The control data SysAxCoupleCmd of the system axis coupling can be changed without influencing the current coupling. The changes are checked and applied only after validation in the control structure by:

SysAxCoupleCtr.Validate = 1

Incorrect data is not applied and acknowledged by the control with the index of the SAC in **SysAxCoupleCtr.CfgErrAx**, the index of the master **SysAxCoupleCtr.CfgErrMaster** and a respective error code (>0) in **SysAxCoupleCtr.CfgErrNum**. At the end, the data validation resets the transfer job with **SysAxCoupleCtr.Validate = 0**.

The system data currently active in the interpolator is also visible in the status structure **SysAxCoupleSta[i].Cmd** after a successful transfer.

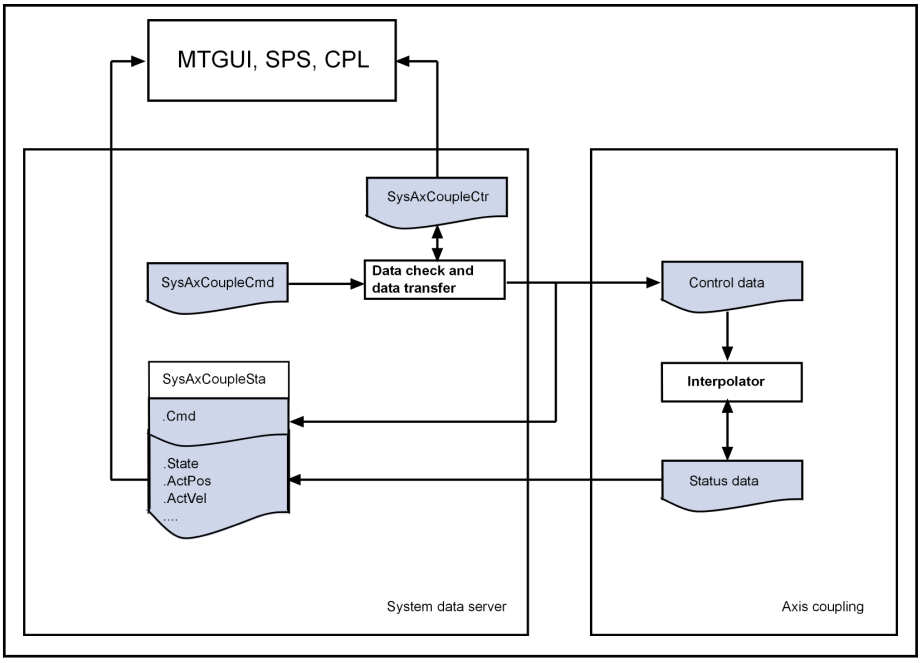


Fig. 14-33: System data transfer

Real-time switching of a master axis

For an interpolator-exact switching of the coupling rule in a specified position or velocity range, a transfer condition can be specified for each master in ***SysAxCoupleCmd[i].Master[m].SwitchCond****:*

- 0 - Unconditional switching
- 1 - Position-conditional switching
- 2 - Velocity-conditional switching
- 100..131 - Externally-triggered switching

The conditional switching comprises the coupling data:

- SysAxCoupleCmd[i].Master[m].AxIndex
- SysAxCoupleCmd[i].Master[m].N_Input
- SysAxCoupleCmd[i].Master[m].N_Output
- SysAxCoupleCmd[i].Master[m].Offset
- SysAxCoupleCmd[i].Master[m].Tab

Without switching condition (SwitchCond=0), coupling data with data transfer (Validate=1) becomes active immediately.

For a position- and velocity-conditional switching, the following trigger range is determined

SysAxCoupleCmd[i].Master[m].SwitchMin and

SysAxCoupleCmd[i].Master[m].SwitchMax.

The switching-relevant position or velocity derives from the specified system axis in SysAxCoupleCmd[i].Master[m].AxIndex. Trigger positions are also possible outside the modulo interval of the master axis so that it can only be switched after n revolutions of the axis.

An external trigger condition can also be used from the bit mask SysAxCoupleCtr.ExtCond. The switching condition is met if the bit "Switch-Cond-100" was replaced in the mask "ExtCond" by the PLC for example.

The switch SysAxCoupleCmd[i].Master[m].Enable is absolutely (uncondition-ally) effective. The active master axis specifications are deleted with

SysAxCoupleCmd[i].Master[m].Enable=0 and enabled with SysAxCoupleCmd[i].Master[m].Enable=1 with regard to the switching condition.

The active real-time coupling data is stored in the system date SysAxCoupleSta[i].ActMaster[m].

A pending transfer of the master axis switching is displayed in SysAxCoupleSta[i].State with PENDING (+20).

Example program on real-time switching:

```
; ---------------------------------------------------------------------
; function:     SAC_REALTIME
;
;    Description: Real-time table switching in the position
;                  range 180°-270° at active system axis coupling
; ---------------------------------------------------------------------
; --- system axis index of slave axis Z:
1 Z%=3
; --- preparing table switching in the range 180°-270°
1 SD.SysAxCoupleCmd[Z%].Master[1].SwitchCond = 1
1 SD.SysAxCoupleCmd[Z%].Master[1].SwitchMin  = 180.0
1 SD.SysAxCoupleCmd[Z%].Master[1].SwitchMax  = 270.0
1 SD.SysAxCoupleCmd[Z%].Master[1].Tab        = "stroke.fct"
; --- Data transfer
WAIT
1 SD.SysAxCoupleCtr.Validate = 1
; --- Ensure activation in the interpolator
WAIT
; --- synchronization to conditioned switching
SACSYNC(Z=3)
```

14.5.4 Synchronization

The coupling starts immediately after the control data transfer and is performed asynchronously to the sequence of the part programs in the channels. Positioning motions can be performed parallel to coupling/decoupling processes to reduce non-productive times.

Synchronism has to be ensured prior to the machining or the positioning of the workpiece. The status of the system axis coupling (SysAxCoupleSta[i].State) can be queried with the SACSYNC NC command as waiting condition:

Program:

```
SACSYNC_COUPLED

1 SLAVE% = P1%
;-----------------------------------------------
; Wait for synchronous run of the slave axis
;-----------------------------------------------
SACSYNC([AXP(AXIS%,3,0)])
```

In the main program, the cycle can be used for any slave axis:

```
;Set-up (spanning) axis CR synchronous to the counter spindle CL ?

1 CALL SACSYNC_COUPLED(CR%)

;Clamp workpiece in CL

...

;Set-up (spanning) axis CL synchronous to workpiece axis B?

1 CALL SACSYNC_COUPLED(CL%)

;Machining with hobbing cutter

...
```

Additionally, synchronous run monitoring can be included if required.

The "SACSYNC" NC command is able to monitor several different SAC states at the same time:

```
SACSYNC(TO=200,CL=3,CR=5,A=0)
```

It is expected in a time window of 200 ms that the cl-axis is coupled (3), the CR-axis stopped (5) and the A axis coupling switched-off (0). If all conditions are met within 200 ms, processing is continued. Otherwise, the control outputs a runtime error.

For deactivated axes, SACSYNC eliminates the decoupling offsets created by SyncMode=2 (velocity-synchronous). Thus, the channel axes can be positioned absolutely again.

It can be waited for the conditional activation of the master axis data with-SACSYNC(...<axis|spindel>=<state>...), since a pending conditional transfer (<state>+20) does not fulfill the continuation condition.

Example: Deactivate coupling offset

;Deactivate coupling without position compensation:	
1 SD.SysAxCoupleCmd[XB%].SyncMode = 2	Decouple velocity-synchronously
1 SD.SysAxCoupleCmd[XB%].OnOff = 0	Deactivate coupling
1 SD.SysAxCoupleCtr.Valid = 1	Activation
SACSYNC	;Wait for the deactivation of all channel axes and ; Applying coupling offset
G1 G90 XB=100 F1000	Absolute axis positioning

Synchronization with status "synchronous run monitoring"

The SACSYNC command can also wait that the accuracy limit of the synchronous run monitoring is reached when specifying the SAC state. This allows for example to continue the program processing earlier if the accuracy limit (INPOS1, INPOS2) was reached, but the coupling is no yet present.

The following examples show the different effects of the waiting conditions:

`SACSYNC(XB=0)`	;Wait until the coupling of the xb-axis is deactivated.
`SACSYNC(XB=3)`	;Wait until the xb-axis is command value-coupled.
`SACSYNC(XB=5)`	;Wait until the coupling of the xb-axis is stopped.
`SACSYNC(XB=1000)`	;Wait until the be-axis reached the INPOS1 window. Reaching of the INPOS2 window as well as the other coupling states are not analyzed.
`SACSYNC(XB=3000)`	;Wait until the xb-axis reached the INPOS1 window and the INPOS2 window. The remaining coupling states are not analyzed.
`SACSYNC(XB=2003)`	;Wait until the xb-axis is command value-coupled and reached the INPOS2 window. Reaching the INPOS1 window is not analyzed.
`SACSYNC(XB=3003)`	;Wait until the xb-axis is command value-coupled and reached the INPOS1 and the INPOS2 window.

14.5.5 System axis coupling with spindles

Spindle modes

Spindles are handled like endlessly rotating rotary axes in the system axis coupling. Slave spindles can only be coupled in the axis mode "position con-

trol" (refer to the manual "MTX Functional Description - Basics", chapter "Switching to position-controlled spindle").



For master spindles, changing the axis mode (speed control/position control) causes a transition from the actual position to the command position and vice versa. Depending on the velocity of the master spindle and the coupling rule of the slave spindle/slave axis, decoupling can be caused.

If possible, change the axis mode only at standstill. In speed control, neither switch master nor slave spindles of synchronous groups.

	Spindle in Speed control	Spindle in Position control	C-axis mode
Master spindle	Actual position	Command position	Command position
Slave spindle	Not permitted	Command position	Command position

Tab. 14-23: Axis mode-dependent master and slave spindle position

For position-controlled slave spindles, switching the axis mode and especially switching the drive parameter causes the system axis coupling to switch off. Only switching to the C-axis mode in standstill without position jump is allowed. Note that switching can cause a change in velocity and acceleration limit values.

Special features and restrictions

The following spindle functions are not supported

- Speed mode (M3, M4, M5)
- Positioning (M19)
- Thread cutting (G33)
- Tapping (G63)
- Constant cutting velocity (G96)
- Spindle coupling (SPCC)

exclusively create and use the interpolation position without considering the coupling parts of the system axis.

The spindle state signals of position-controlled slave spindles

- "Speed reached" (iSp_ProgSpReach)
- "Spindle positioning active" (iSp_OrientateAct)
- "Spindle positioning finished" (iSp_OrientateAct)
- "Spindle counterclockwise rotation active"(iSp_TurnM4)
- "Speed active" (iSp_TurnCmd)
- "Synchronous window 1/2" (iSp_Synchr1/2)

refer to the interpolation position without the coupling parts of the system axis if the system axis coupling is active.

The signal "Spindle Stop" (iSp_Stop) still shows the drive state (S-0-0331: n_actual=0).

Spindle reset (qSp_Reset) does not complete the system axis coupling. Due to the active reference specification, the spindle remains in motion.

As for the axes, a slave spindle with SyncMode=2 without compensation motion can also be decoupled.

The caused switch-off offset can be reduced with M19 (positioning spindle) if the system axis coupling is active. If the SPCC spindle coupling is active, the switch-off offset remains.

Master spindle interpolators are not limited by slave axes. When exceeding the maximum velocity or the maximum acceleration, decoupling results.

Spindle coupling examples

System axis-coupled slave axes can be programmed. This allows superimposed speed specifications. In the example, spindle S2 with a constant relative speed to the workpiece should drill a bore in spindle S1:

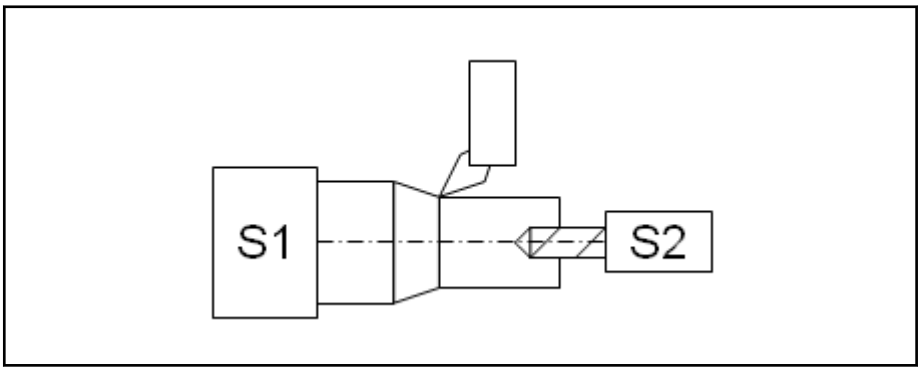


Fig. 14-34: Spindle coupling

S2 is continuously superimposed with the changing speed of the spindle S1.

Program:

```
...
0100 S1%=3 : S2%=4
N110 SPM(S2=1)          ;switch slave spindles to position control
...
                        ;coupling on
0140 SD.SysAxCoupleCmd[S2%].OnOff              =    1
                        ;S2 activate interpolation specification
0150 SD.SysAxCoupleCmd[S2%].IpoEnable          = TRUE
                        ;activate S1 override
0160 SD.SysAxCoupleCmd[S2%].Master[1].Enable   = TRUE
0170 SD.SysAxCoupleCmd[S2%].Master[1].AxIndex  =  S1%
                        ;negative feedback S2 = -S1
0180 SD.SysAxCoupleCmd[S2%].Master[1].N_Input  =    1
0190 SD.SysAxCoupleCmd[S2%].Master[1].N_Output =   -1
                        ;data application
0200 SD.SysAxCoupleCtr.Validate   = 1
N210 SACSYNC(S2=3)      ;wait till override is active
N230 G97 M203 S2=500    ;boring speed 500 rpm
N240 G95 G1 Z=-20 F2    ;feed at 100% S2 potentiometer 1000 mm/min
...
                        ;decoupling
0280 SD.SysAxCoupleCmd[S2%].OnOff = 0
0290 SD.SysAxCoupleCtr.Validate   = 1
N300 SACSYNC(S2=0)      ;waiting for decoupling
...
```

Another use case of the system axis spindle coupling is the polygonal turning. Tool spindle and workpiece spindle have to be coupled synchronously in an integer ratio depending on the geometry and the number of edges:

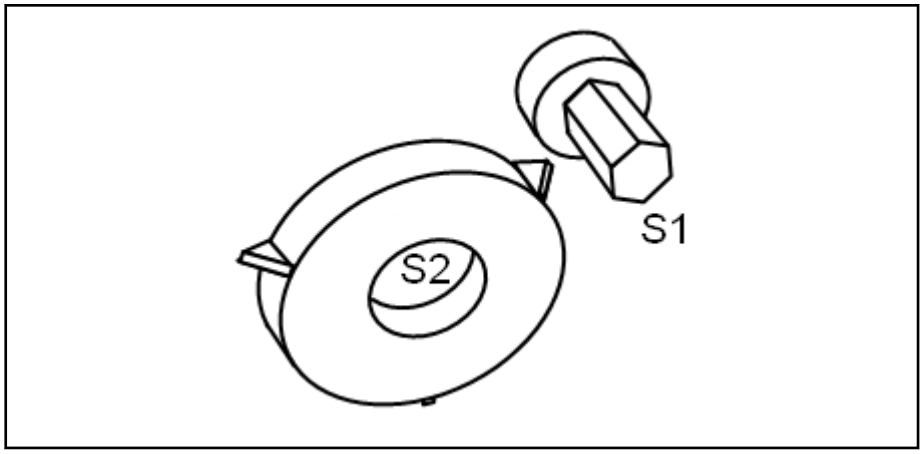


Fig. 14-35: Polygonal machining

In the example, the coupling ratio of tool and workpiece is 2:1 to machine a hexagon geometry.

Program:

```
0100 S1%=3 : S2%=4
N110 SPM(S1=1,S2=1) ;switch master and slave spindle to position control
...
                          ;enable coupling
0140 SD.SysAxCoupleCmd[S2%].OnOff              =     1
                          ;no S2 interpolation specification
0150 SD.SysAxCoupleCmd[S2%].IpoEnable          = FALSE
                          ;activate master spindle S1
0160 SD.SysAxCoupleCmd[S2%].Master[1].Enable   =  TRUE
0170 SD.SysAxCoupleCmd[S2%].Master[1].AxIndex  =   S1%
                          ;coupling ratio S1:S2
0180 SD.SysAxCoupleCmd[S2%].Master[1].N_Input  =     1
0190 SD.SysAxCoupleCmd[S2%].Master[1].N_Output =     2
0200 SD.SysAxCoupleCtr.Validate                =     1
N210 SACSYNC(S2=3)        ;waiting until coupling is active
...
```

14.5.6 Exception Handling

Exception handlings such as an emergency retract or deactivation of the drive can be performed by the drive and by the NC.

Drive-controlled standstill and drive-controlled retract motion:

Via the drive parameter P-0-0119 "optimum deactivation", a drive can be stopped or brought to a certain position.

During retraction, the drive generates a position command profile to travel the traversing distance in case of error. In case of error, a relative traversing block is activated which is defined by the parameters "P-0-0055, retract path", "P-0-0056, retract velocity", "P-0-0057, retract acceleration" and "P-0-0058, retract jerk".

For details, refer to the drive documentation.

Control-controlled reactions:

In a coupling cascade, an error can occur in each coupling element which is intended to cause an emergency reaction. Also external events, as from a tool rupture monitoring for example, can trigger an emergency reaction. A system axis coupling has to be defined for each axis which is to participate in an exception handling. To this end, the "Abort SAC" has to reference an SAC group device as master axis or an axis of the channel which is to be monitored (AxIndex=i). The reference value can be deactivated (Master[m].Enable=FALSE) so that no "real" coupling exists. The "Abort SAC" is in the group after activation (OnOff=1) and is automatically informed in case of malfunctions.

Malfunctions in a couple link can be treated explicitly in each SAC object; they can also be ignored.

Occurring malfunctions are grouped into the signal sources:

- Axis error
- Coupling error
- Channel error
- External error
- Slave axis or slave group error
- Master axis or master group error
- Formula error
- Exiting the selected synchronous run window

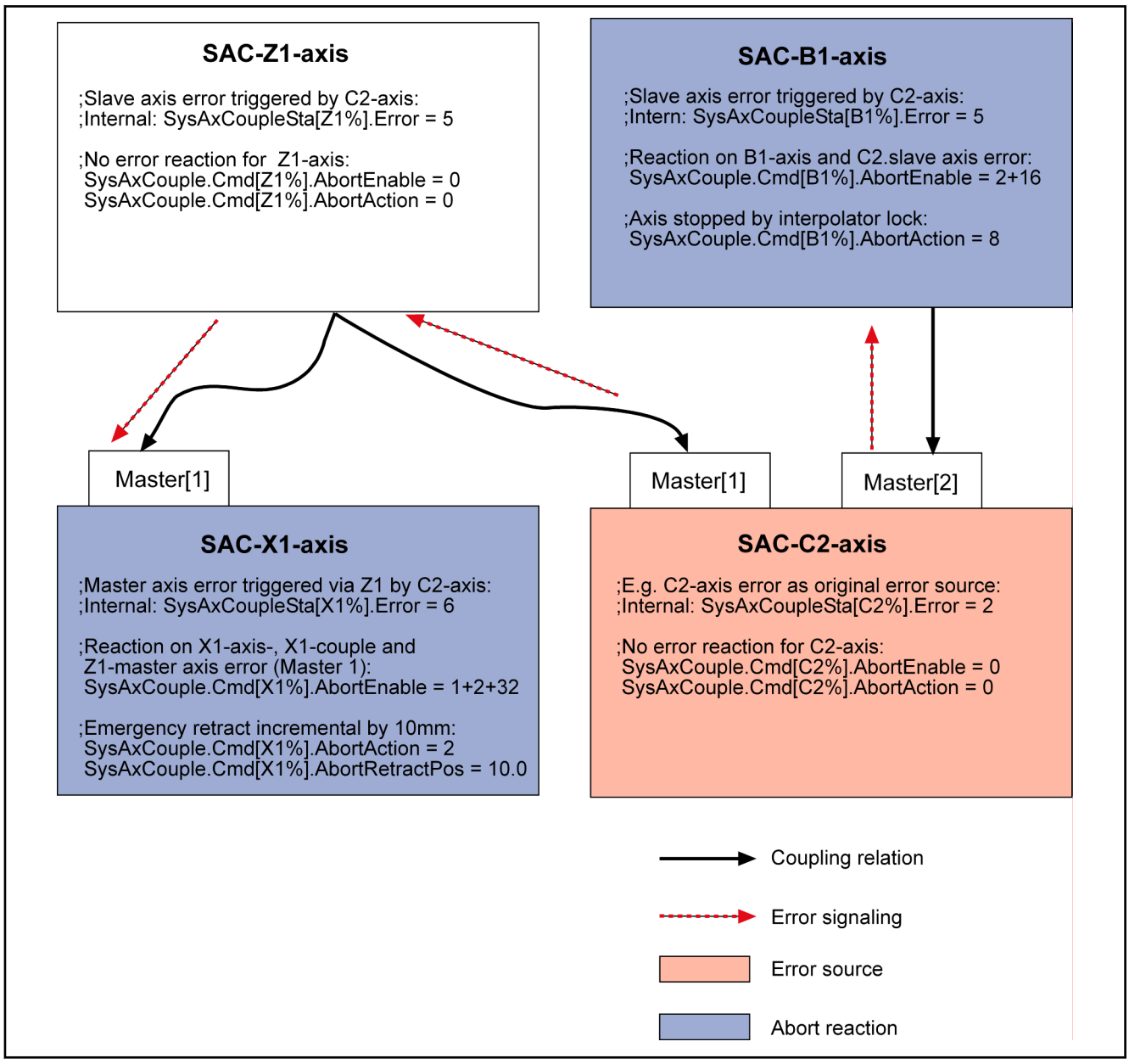


Fig. 14-36: Error processing in a coupling cascade

*Enable the individual signal sources via the system date **SysAxCoupleCmd[i].AbortEnable**:*

- 0
  Execute no error reaction

Direct error sources (Error=1..4)

- Bit 0 (# 1)
  Reaction in case of drive error
- Bit 1 (# 2)
  Reaction in case of coupling error
- Bit 2 (# 4)
  Reaction in case of external error (ExtError<>0)
- Bit 3 (# 8)
  Reaction in case of error in interpolation channel (channel error)

Indirect error sources (Error=5,6)

- Bit 4 (# 16)
  Reaction in case of error of an SAC slave axis
- Bit 5 (# 32)
  Reaction in case of error of the SAC master axis 1
- Bit 6 (# 64)
  Reaction in case of error of the SAC master axis 2
- Bit 7 (#128)
  Reaction in case of error of the SAC master axis 3
- Bit 8 (#256)
  Reaction in case of error of the SAC master axis 4
- Bit 9 (#512)
  Reaction in case of error of the SAC master axis 5

Formula error (Error=7):

- Bit 16 (#65536)
  Reaction in case of zero division (FormulaDiag=1)
- Bit 17 (#131072)
  Reaction in case of overflow (FormulaDiag=2)
- Bit 18 (#262144)
  Reaction in case of definition range violation (FormulaDiag=3)
- Bit 19 (#524288)
  Reaction in case of underflow (FormulaDiag=4)
- Bit 20 (#1048576)
  Reaction to processing errors (FormulaDiag=5..9)

InPos error (Error=8)

- Bit 24 (#16777216)
  Reaction to violation of InPos condition

All other leading and slave SAC objects are automatically informed upon occurrence of a malfunction at an SAC in the couple link. If the signal source is enabled, the exception handling starts in the same interpolation cycle.

If this is not desired, delay it by **SysAxCoupleCmd[i].AbortDelay** ms.

*After the delay elapsed, the (bit-coded) exception handling is initiated from **SysAxCoupleCmd[i].AbortAction**:*

- (BIT 0)

  Retraction absolute to the position **SysAxCoupleCmd[i].AbortRetractPos**

  - or -

- (BIT 1)

  Retraction incremental by **SysAxCoupleCmd[i].AbortRetractPos**

  - or -

- (BIT 2)

  Issue channel error with malfunction type

  - or -

- (BIT 3-6)

  Continue normal or modified coupling

In case of an axis error, neither the abort type "retract" nor "coupling" can be activated. Therefore, only a channel error with the type of the malfunction is output.

NOTICE **Abort reactions with incorrect parameterizations can cause tool and machine damage!**

Test the abort reaction without tool action outside the danger area.

Parallel to the above described reactions, an asynchronous program can be triggered from **SysAxCoupleCmd[i].AbortAsupNo = [1..8]** as follows:

- in the axis channel **SysAxCoupleCmd[i].AbortAsupChan = 0**

  - or -

- In channel **SysAxCoupleCmd[i].AbortAsupChan = [1..60]**

The general functional description applies for the installation and the execution of an ASUP.

During the abort handling, ABORT (+10) superimposes the SAC status in SysAxCoupleSta[i].State.

Sequence of the abort methods from SysAxCoupleCmd[i].AbortAction:

1. Retraction

   The current position at the time of interruption is saved in **SysAxCoupleSta[i].AbortPos**.

   It is retracted with maximum velocity and acceleration to the specified SysAxCoupleCmd[i].AbortRetractPos position. The retract position is specified as the new and only reference value. The SAC object is interpolated via ABORT-RECOUPLE to ABORT-COUPLED to the retract position.

   A channel error subsequently shows the end of the retraction. When abort ASUP is active, the channel error is generated only after ASUP end.

2. Channel error

   If the SAC axis is a channel axis, an error is reported to the channel, also specifying the malfunction source. The SAC object immediately goes into the ABORT-ERROR status without consideration of the master values where the axis is braked with maximum delay to a velocity of zero.

3. Coupling

   This abort method applies if no other abort method is specified. The axis coupling continues with the relations from the system data. Additionally, individual or all reference values can be locked via the bits #3-#6 in SysAxCoupleCmd[i].AbortAction. The exception coupling is aborted with SysAxCoupleCmd[i].Reset = 1. As during the retraction, a channel error message considering a started ASUP is issued.

After the error handling, the SAC object is always in the ABORT-ERROR state. From this state, it can only be set to inactive (SysAxCoupleSta[i].State=0) after error deletion and a subsequent reset with **SysAxCoupleCmd[i].Reset = 1**.

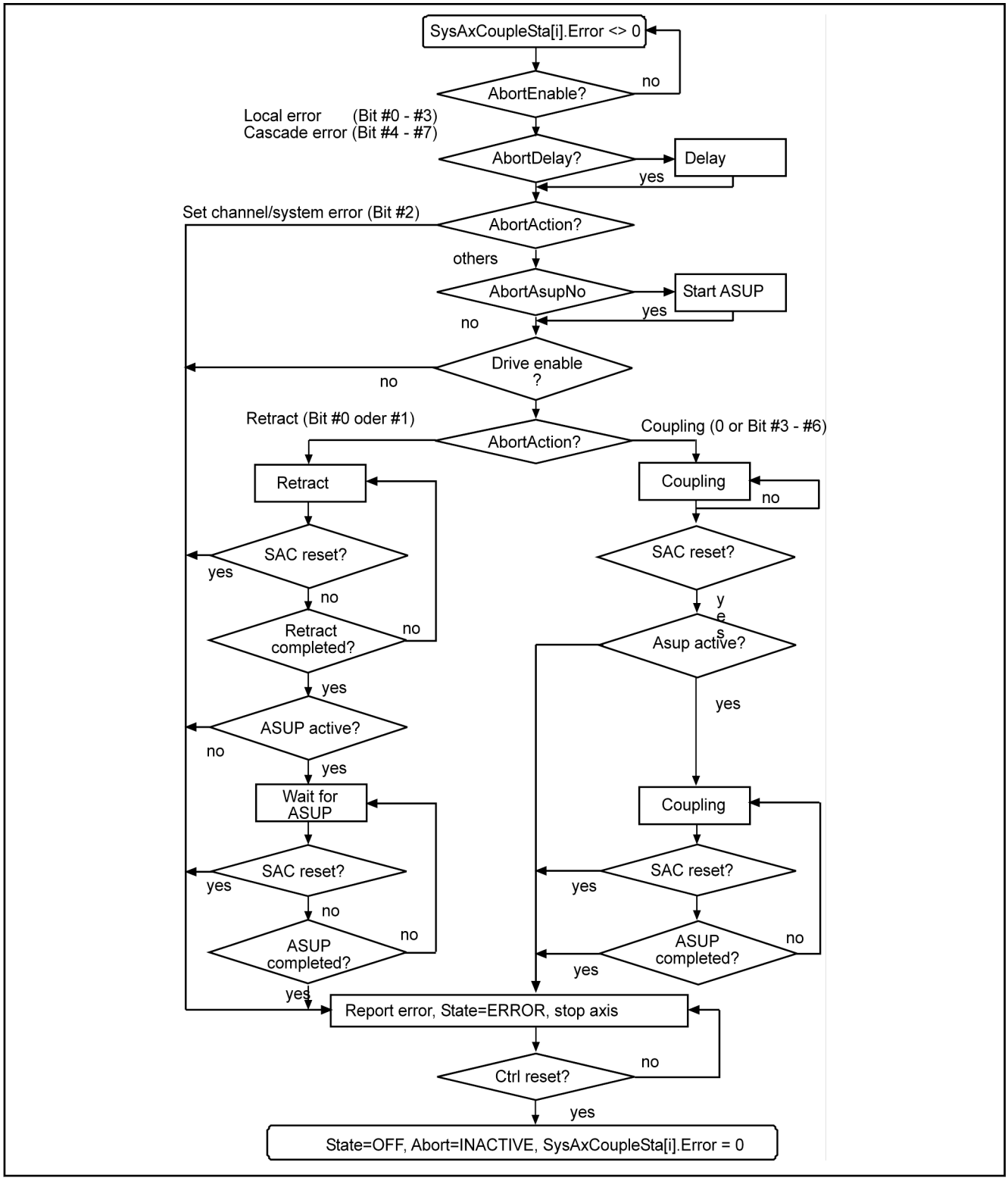


Fig. 14-37: Error processing sequence

An ASUP program may only be activated once in a channel. Therefore, parameterize only one SAC element in the group for ASUP activation in each respective channel. Errors of other SACs in the group are automatically exchanged among each other as slave or master axis errors.

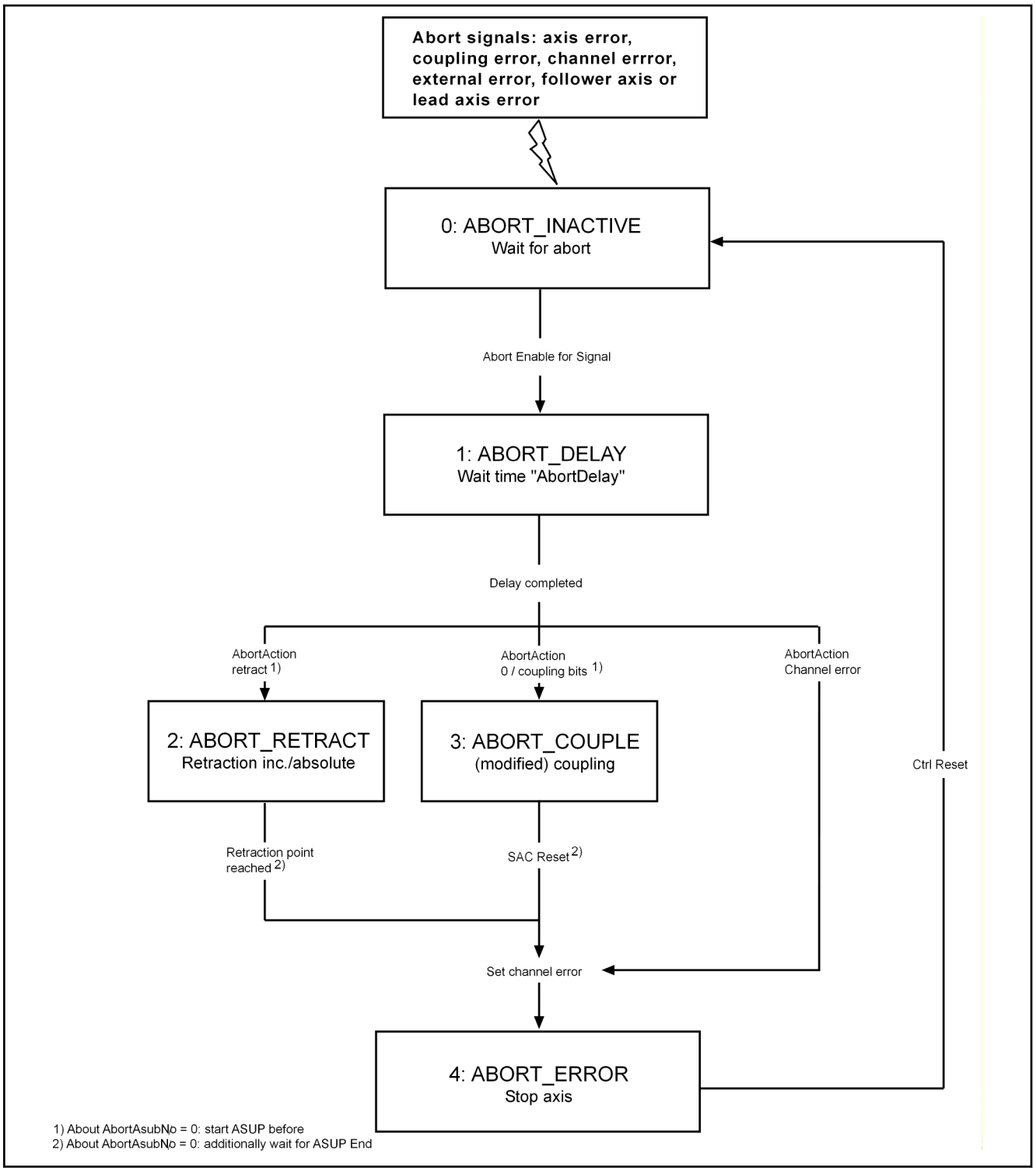


Fig. 14-38: State diagram of the exception handling

Abort states in SysAxCoupleSta[i].Abort:

0	ABORT_INACTIVE	-	The abort mechanism is inactive.
1	ABORT_DELAY	-	Abort initiated, waiting time active.
2	ABORT_RETRACT	-	Retraction of the system axis active.
3	ABORT_COUPLE	-	System axis remains in (modified) coupling.
4	ABORT_ERROR	-	The exception handling is completed and waits for reset.

Tab. 14-24: Abort states

Example of an error handling:

Program:

```
;--------------------------------------------------
;    Coupling relation:
;               +---------+
;               |Z axis   |
;               +---------+
;                    |
;                    +-------------+
;                    |             |
;                    V             V
;               +---------+   +---------+
;               |YB axis  |   |VM axis  |
;               +---------+   +---------+
;                    |             |
;                    +-------------+
;                    |
;                    V
;               +---------+
;               |XB axis  |
;               +---------+
;
;
; System data for SAC error treatment:
;
; 1. Enabling abort
; ------------------
;    AbortEnable: (parallel)
;           0 - not enabled
;    Bit #Val                    (variable)
;     0  #  1 - Axis error       (AXENA)
;     1  #  2 - Coupling error   (COENA)
;     2  #  4 - External error   (EXENA)
;     3  #  8 - Channel error    (CHENA)
;     4  # 16 - Follower group   (SLENA)
;     5  # 32 - M1 group         (M1ENA)
;     6  # 64 - M2 group         (M2ENA)
;     7  #128 - M3 group         (M3ENA)
;
;     Example: Enabling abort for
;              axis error, external error and channel error
;     1 SD.SysAxCoupleCmd[1].AbortEnable = 1+4+8
;
; 2. Abort action
; ----------------
;    AbortAction:
;       0 - Coupling continuous
;    BIT #Val (alternative ----)  (variable)
;    ------------------------------
;      0 # 1 - Retract absolute   (ABSRT)
;    ------------------------------
;      1 # 2 - Retract relativw   (INCRT)
;    ------------------------------
;      2 # 4 - Set channel error  (CHERR)
;    ------------------------------
;      3 # 8 - IPO Disable        (DISIP)
;      4 #16 - Master1 Disable    (DISM1)
;      5 #32 - Master2 Disable    (DISM2)
;      6 #64 - Master3 Disable    (DISM3)
;   ------------------------------
;
; Example:    absolute retract motion when abort:
;             1 SD.SysAxCoupleCmd[1].AbortAction = 1
;
;             Axis stop when abort:
;             1 SD.SysAxCoupleCmd[1].AbortAction = 8+16+32+64
```

```
;
;--------------------------------------------------
; System axis indices of axes Z, XB, YB and VM:
1 Z%=3 : XB%=4 : YB%=5 : VM%=64
;--------------------------------------------------
; AbortEnable Defines
1 AXENA%=1  : COENA%=2  : EXENA%=4
1 CHENA%=8  : SLENA%=16
1 M1ENA%=32 : M2ENA%=64 : M3ENA%=128
;--------------------------------------------------
; AbortAction Defines
1 ABSRT%=1 : INCRT%=2  : CHERR%=4
1 DISIP%=8 : DISM1%=16 : DISM2%=32 DISM3%=64
;--------------------------------------------------
; Slave axis XB: Master YB and VM
;--------------------------------------------------
;
; After 100ms report channel error in channel XB at YB / VM
; or Z error and start ASUP 1 in channel 3
1 SD.SysAxCoupleCmd[XB%].AbortEnable     = M1ENA%+M2ENA%
1 SD.SysAxCoupleCmd[XB%].AbortDelay      = 100
1 SD.SysAxCoupleCmd[XB%].AbortRetractPos = 0.0
1 SD.SysAxCoupleCmd[XB%].AbortAction     = CHERR%
1 SD.SysAxCoupleCmd[XB%].AbortAsupNo     = 1%
1 SD.SysAxCoupleCmd[XB%].AbortAsupChan   = 3
;
;--------------------------------------------------
; Slave axis YB: Master Z (CH1)
;--------------------------------------------------
;
; If external error (SysAxCoupleCmd[YB%].ExtError <> 0,
; YB axis is retracted by 10 mm
1 SD.SysAxCoupleCmd[YB%].AbortEnable     = EXENA%
1 SD.SysAxCoupleCmd[YB%].AbortDelay      = 0
1 SD.SysAxCoupleCmd[YB%].AbortRetractPos = -10
1 SD.SysAxCoupleCmd[YB%].AbortAction     = INCRT%
1 SD.SysAxCoupleCmd[YB%].AbortAsupNo     = 0
1 SD.SysAxCoupleCmd[YB%].AbortAsupChan   = 0
;
;--------------------------------------------------
; Slave axis: Master Z (CH1)
;--------------------------------------------------
;
; Immediate stop of axis VM
; if error of master and slave axes     (XB, YB, Z)
1 SD.SysAxCoupleCmd[VM%].AbortEnable     = SLENA%+M1ENA
1 SD.SysAxCoupleCmd[VM%].AbortDelay      = 0
1 SD.SysAxCoupleCmd[VM%].AbortRetractPos = 64.0
1 SD.SysAxCoupleCmd[VM%].AbortAction     = DISIP%+DISM1%
;
;--------------------------------------------------
; Slave axis Z: no master, IpoEnable=TRUE
;--------------------------------------------------
;
; Immediate absolute retract of Z axis if error
; of slave axes
1 SD.SysAxCoupleCmd[Z%].AbortEnable     = SLENA%
1 SD.SysAxCoupleCmd[Z%].AbortDelay      = 0
1 SD.SysAxCoupleCmd[Z%].AbortRetractPos = 100.0
1 SD.SysAxCoupleCmd[Z%].AbortAction     = ABSRT%
;
```

The error in the SAC is saved in SysAxCoupleSta[i].Error. This error is entered irrespective of the enabling of the exception handling (AbortEnable) and transferred to the dependent slave and master axis members.

Errors in SysAxCoupleSta[i].Error:

0	NO_ERROR	-	There is no error.
1	UNCOUPLED	-	The coupling cannot be maintained and no recoupling is enabled (RecoupleEnable=FALSE).
2	AXIS_ERROR	-	There is an diagnostics class 1 error for the slave axis.
3	CHAN_ERROR	-	There is an error in the channel for the slave axis.
4	EXT_ERROR	-	An error code <> 0 was entered in SysAxCoupleCmd[i].ExtError.

5	SLAVE_ERROR	-	An error occurred in a dependent slave axis or in the slave axis group.
6	MASTER_ERROR	-	An error occurred in a master axis or in the master axis group.
7	FORMU-LA_ERROR	-	Error when calculating the formula in: SysAxCoupleCmd[i].Formula The detailed error code is stored in SysAxCoupleSta[i].FormulaDiag.
8	INPOS_VIOLA-TION	-	The synchronous run monitoring determined that the actual position of the slave axis already exited the selected accuracy window and that no recoupling enabled (RecoupleEnable=FALSE) is present.

Tab. 14-25: Errors in SysAxCoupleSta[i].Error

The parameterized exception handling for the above named errors is performed only after activation (AbortEnable). As soon as the malfunction is eliminated, the error can be deleted with SysAxCoupleCmd[i].Reset.

14.5.7 Application cycles

Transfer of system data:

Program:

```
;-----------------------------------------------------
;Program name: SAC_VALID
;Function:     SAC - system data transfer
;Parameters:    None
;-----------------------------------------------------
;
1 DIM ERRSTR$(100)
1 DIM FORMEL$(100)
WAIT
1 SD.SysAxCoupleCtr.Validate=1
1 ERRNO%  = SD.SysAxCoupleCtr.CfgErrNum
1 AXIS%   = SD.SysAxCoupleCtr.CfgErrAx
1 MASTER% = SD.SysAxCoupleCtr.CfgErrMaster
1 IF ERRNO% <> 0 THEN
1   CASE ERRNO% OF
1     LABEL 1:  ERRSTR$ = "Circulation (auto-coupling)"
1     LABEL 2:  ERRSTR$ = "Nesting too deep"
1     LABEL 3:  ERRSTR$ = "Axis cannot become slave axis (e.g. spindle)"
1     LABEL 4:  ERRSTR$ = "Master enabled with invalid master index"
1     LABEL 5:  ERRSTR$ = "Master enabled without master input revolution"
1     LABEL 6:  ERRSTR$ = "Master enabled without master output revolution"
1     LABEL 7:  ERRSTR$ = "Monitoring window index invalid"
1     LABEL 8:  ERRSTR$ = "ASUP number is invalid"
1     LABEL 9:  ERRSTR$ = "ASUP channel does not exist"
1     LABEL 10: ERRSTS$ = "AbortAction contains an invalid combination"
1     LABEL 11: ERRSTR$ = "Syncmode (for axis type) not allowed"
1     LABEL 12: ERRSTR$ = "Invalid coupling velocity"
1     LABEL 13: ERRSTR$ = "Invalid coupling acceleration"
1     LABEL 14: ERRSTR$ = "Invalid input speed"
1     LABEL 15: ERRSTR$ = "Invalid output speed"
1     LABEL 16: ERRSTR$ = "Invalid offset"
1     LABEL 17: ERRSTR$ = "Invalid retract position"
1     LABEL 18: ERRSTR$ = "Invalid InPos window1"
1     LABEL 19: ERRSTR$ = "Invalid InPos window2"
1     LABEL 20: ERRSTR$ = "Table is not loaded"
1     LABEL 21: ERRSTR$ = "Invalid switching condition"
1     LABEL 22: ERRSTR$ = "Invalid lower switching limit"
1     LABEL 23: ERRSTR$ = "Invalid upper switching limit"
1     LABEL 24: ERRSTR$ = "Switching range too small"
1     LABEL 25: ERRSTR$ = "Invalid modulo range: Table to slave axis"
1     LABEL 26: ERRSTR$ = "Invalid modulo range: Master to table"
1     LABEL 27: ERRSTR$ = "Invalid master axis shift"
1     LABEL 28: ERRSTR$ = "Invalid master value mode"
1     LABEL 29: ERRSTR$ = "Formula syntax error"
1     LABEL 30: ERRSTR$ = "Formula timeout"
1     LABEL 31: ERRSTR$ = "Formula interpretation error"
1     OTHERWISE ERRSTR$ = "Error at system axis coupling data transfer"
1   ENDCASE
```

```
1   IF MASTER% = 0 THEN
1     ERRSTR$="SAC("+STR$("##",AXIS%)+"):"+ERRSTR$
1   ELSE
1     IF (ERRNO% = 29) OR (ERRNO% = 31) THEN
1       FORMEL$=SD.SysAxCoupleCmd[AXIS%].Formula
1       START%=1
1       IF (MASTER% > 40) THEN
1            START%=MASTER%-40
1       ENDIF
1       ERRSTR$="SAC("+STR$("##",AXIS%)+")
                "+MID$(FORMEL$,START%,MASTER%)+"^ "+ERRSTR$
1   ELSE
1     ERRSTR$="SAC("+STR$("##",AXIS%)+")
                "+STR$(".Master(#):",MASTER%)+ERRSTR$
1       ENDIF
1       ENDIF
1   SETERR(ERRSTR$)
1       ENDIF
```

Synchronizing the deactivation:

Program:

```
;-----------------------------------------------------
; program name: SAC_OFFSYNC
; function:     At SAC - State OFF and/or ERROR
;               synchronization
; parameters:    P1% - Number of system axes
;               P2% - 0: All SAC in OFF state
;                        or ERROR
;                     1: All SAC in OFF state
;-----------------------------------------------------
1 MAXAX% = P1%
1 NOERR% = P2%
1 REPEAT
1 WAIT(,20)
1   SYN% = 0
1 FOR I%=1 TO MAXAX%
1     CASE SD.SysAxCoupleSta[I%].State OF
1 LABEL 0: SYN%=SYN%
1 LABEL 7: SYN%=SYN%+NOERR%
1 LABEL 17: SYN%=SYN%+NOERR%
1       OTHERWISE SYN%=SYN%+1
1   ENDCASE
1 NEXT
1 UNTIL (SYN%=0)
```

Resetting the system axis coupling and deleting system data:

Program:

```
;-----------------------------------------------------
; program name: SAC_OFF
; function:     Switch off all SAC couplings and
;              completely delete the SAC system data
; parameters:    None
;-----------------------------------------------------
;
1 DIM AXINFO%(3)
1  VERSION=0
1 ERR_VAR%=MCODS(45,-1,VERSION,AXINFO%,3)
1 IF ERR_VAR% <> 0 THEN
1   SETERR("Unknown system axis count")
1 ENDIF
1 MAXAX%=AXINFO%(1)
;-----------------------------------------------------
; Initialize system data, enabling abort
; lock, activate SAC reset
;-----------------------------------------------------
1 FOR I%=1 TO MAXAX%
1   SD.SysAxCoupleCmd[I%].OnOff           = 0
1   SD.SysAxCoupleCmd[I%].IpoEnable       = FALSE
1   SD.SysAxCoupleCmd[I%].VelCouple       = 0.0
1   SD.SysAxCoupleCmd[I%].AccCouple       = 0.0
1   SD.SysAxCoupleCmd[I%].AbortEnable     = 0
1   SD.SysAxCoupleCmd[I%].AbortAsupNo     = 0
1   SD.SysAxCoupleCmd[I%].AbortAsupChan   = 0
1   SD.SysAxCoupleCmd[I%].AbortDelay      = 0
1   SD.SysAxCoupleCmd[I%].AbortAction     = 0
1   SD.SysAxCoupleCmd[I%].AbortAction     = 0.0
1   SD.SysAxCoupleCmd[I%].Reset       = 1
```

```
1   SD.SysAxCoupleCmd[I%].ExtError        = 0
1   SD.SysAxCoupleCmd[I%].RecoupleEnable  = TRUE
1   SD.SysAxCoupleCmd[I%].GearTeeth       = 0
1   SD.SysAxCoupleCmd[I%].SyncMode        = 2
1   SD.SysAxCoupleCmd[I%].Rigidity        = 0
1   SD.SysAxCoupleCmd[I%].PosWinSel       = 0
1   SD.SysAxCoupleCmd[I%].PosWin[1]       = 0.0
1   SD.SysAxCoupleCmd[I%].PosWin[2]       = 0.0
1   FOR J%=1 TO 3
1     SD.SysAxCoupleCmd[I%].Master[J%].Enable   = FALSE
1     SD.SysAxCoupleCmd[I%].Master[J%].AxIndex  = 0
1     SD.SysAxCoupleCmd[I%].Master[J%].N_Input  = 0
1     SD.SysAxCoupleCmd[I%].Master[J%].N_Output = 0
1     SD.SysAxCoupleCmd[I%].Master[J%].Tab = ""
1     SD.SysAxCoupleCmd[I%].Master[J%].SwitchCond = 0
1     SD.SysAxCoupleCmd[I%].Master[J%].SwitchMin  = 0.0
1     SD.SysAxCoupleCmd[I%].Master[J%].SwitchMax  = 0.0
1 NEXT
1 NEXT
1 CALL SAC_VALID
1 CALL SAC_OFFSYNC[MAXAX%,0]
;
;-------------------------------------------------
; Still active pending exception treatments
; abort
;-------------------------------------------------
1 FOR I%=1 TO MAXAX%
1   IF SD.SysAxCoupleSta[I%].Abort = 3 THEN
1   SD.SysAxCoupleCmd[I%].Reset       = 1
1 ENDIF
1 NEXT
1 CALL SAC_VALID
1 CALL SAC_OFFSYNC[MAXAX%,0]
;-------------------------------------------------
; Delete pending errors after reset
;-------------------------------------------------
1 FOR I%=1 TO MAXAX%
1   SD.SysAxCoupleCmd[I%].Reset       = 1
1 NEXT
1 CALL SAC_VALID
1 CALL SAC_OFFSYNC[MAXAX%,1]
;-------------------------------------------------
; Revoke reset
;-------------------------------------------------
1 FOR I%=1 TO MAXAX%
1   SD.SysAxCoupleCmd[I%].Reset       = 0
1 NEXT
1 CALL SAC_VALID
;-------------------------------------------------
; Apply coupling offset in preparation
;-------------------------------------------------
SACSYNC
WAIT
```

Parameterizing a slave axis coupling:

Program:

```
;-------------------------------------------------
; Program name: SAC_INIT
; Function:     Preparing and transferring a coupling cascade as example:
;
;                      Z axis +
;                             |
;          BV axis -> B axis +->CL axis ->CR axis
;
; Parameters:    none
;-------------------------------------------------
;
1 CALL SAC_OFF
;
;-------------------------------------------------
; System axis index of the Z-, B-, X- and CL axis:
1 Z%=3 : B%=8 : CL%=9 : CR%=10 : BV%=12
; Synchronization modes
1 SYNSHORT%=0 : SYNTIME%=1 : SYNVEL%=2
1 SYNPOS%=3   : SYNNEG%=4
;-------------------------------------------------
;
;-------------------------------------------------
; Slave axis B (Master BV)
;-------------------------------------------------
```

```
1 SD.SysAxCoupleCmd[B%].OnOff              = 1
1 SD.SysAxCoupleCmd[B%].IpoEnable          = TRUE
1 SD.SysAxCoupleCmd[B%].VelCouple          = 5600000.0
1 SD.SysAxCoupleCmd[B%].AccCouple          = 120.0
1 SD.SysAxCoupleCmd[B%].GearTeeth          = 1
1 SD.SysAxCoupleCmd[B%].SyncMode           = SYNVEL%
1 SD.SysAxCoupleCmd[B%].RecoupleEnable     = TRUE
; ---------- Master 1 (B) active --------------------
1 SD.SysAxCoupleCmd[B%].Master[1].Enable   = FALSE
1 SD.SysAxCoupleCmd[B%].Master[1].AxIndex  = BV%
1 SD.SysAxCoupleCmd[B%].Master[1].Offset   = 0
1 SD.SysAxCoupleCmd[B%].Master[1].N_Input  = 1
1 SD.SysAxCoupleCmd[B%].Master[1].N_Output = 1
; ---------- No error reaction ----------------
1 SD.SysAxCoupleCmd[B%].AbortEnable        = 0
1 SD.SysAxCoupleCmd[B%].AbortDelay         = 0
1 SD.SysAxCoupleCmd[B%].AbortRetractPos    = 0.0
1 SD.SysAxCoupleCmd[B%].AbortAction        = 0
1 SD.SysAxCoupleCmd[B%].AbortAsupNo        = 0
1 SD.SysAxCoupleCmd[B%].AbortAsupChan      = 0
; ---------- Synchronous running window CR ----------------
1 SD.SysAxCoupleCmd[B%].PosWin[1]          = 0.02
1 SD.SysAxCoupleCmd[B%].PosWin[2]          = 0.01
1 SD.SysAxCoupleCmd[B%].PosWinSel          = 1
;
;--------------------------------------------------
; Slave axis CL (master B)
;--------------------------------------------------
1 SD.SysAxCoupleCmd[CL%].OnOff              = 1
1 SD.SysAxCoupleCmd[CL%].IpoEnable          = TRUE
1 SD.SysAxCoupleCmd[CL%].VelCouple          = 5600000.0
1 SD.SysAxCoupleCmd[CL%].AccCouple          = 120.0
1 SD.SysAxCoupleCmd[CL%].GearTeeth          = 21
1 SD.SysAxCoupleCmd[CL%].SyncMode           = SYNTIME%
1 SD.SysAxCoupleCmd[CL%].RecoupleEnable     = TRUE
; ---------- Master 1 (B) active --------------------
1 SD.SysAxCoupleCmd[CL%].Master[1].Enable   = FALSE
1 SD.SysAxCoupleCmd[CL%].Master[1].AxIndex  = B%
1 SD.SysAxCoupleCmd[CL%].Master[1].Offset   = 0
1 SD.SysAxCoupleCmd[CL%].Master[1].N_Input  = 21
1 SD.SysAxCoupleCmd[CL%].Master[1].N_Output = -1
; ---------- Master 1 (Z) active --------------------
1 SD.SysAxCoupleCmd[CL%].Master[2].Enable   = FALSE
1 SD.SysAxCoupleCmd[CL%].Master[2].AxIndex  = Z%
1 SD.SysAxCoupleCmd[CL%].Master[2].Offset   = -0.42
1 SD.SysAxCoupleCmd[CL%].Master[2].N_Input  = 1
1 SD.SysAxCoupleCmd[CL%].Master[2].N_Output = 0.321
; ---------- No error reaction ----------------
1 SD.SysAxCoupleCmd[CL%].AbortEnable        = 0
1 SD.SysAxCoupleCmd[CL%].AbortDelay         = 0
1 SD.SysAxCoupleCmd[CL%].AbortRetractPos    = 0.0
1 SD.SysAxCoupleCmd[CL%].AbortAction        = 0
1 SD.SysAxCoupleCmd[CL%].AbortAsupNo        = 0
1 SD.SysAxCoupleCmd[CL%].AbortAsupChan      = 0
; ---------- Synchronous running window CR ----------------
1 SD.SysAxCoupleCmd[CL%].PosWin[1]        = 0.02
1 SD.SysAxCoupleCmd[CL%].PosWin[2]        = 0.01
1 SD.SysAxCoupleCmd[CL%].PosWinSel        = 1
;
;--------------------------------------------------
; Slave axis CR (master CL)
;--------------------------------------------------
1 SD.SysAxCoupleCmd[CR%].OnOff     = 1
1 SD.SysAxCoupleCmd[CR%].IpoEnable = TRUE
1 SD.SysAxCoupleCmd[CR%].VelCouple = 5600000.0
1 SD.SysAxCoupleCmd[CR%].AccCouple = 120.0
1 SD.SysAxCoupleCmd[CR%].GearTeeth = 0
1 SD.SysAxCoupleCmd[CR%].SyncMode  = SYNTIME%
1 SD.SysAxCoupleCmd[CR%].RecoupleEnable     = TRUE
; ---------- Master 1 (B) active --------------------
1 SD.SysAxCoupleCmd[CR%].Master[1].Enable   = FALSE
1 SD.SysAxCoupleCmd[CR%].Master[1].AxIndex  = CL%
1 SD.SysAxCoupleCmd[CR%].Master[1].Offset   = 0
1 SD.SysAxCoupleCmd[CR%].Master[1].N_Input  = 1
1 SD.SysAxCoupleCmd[CR%].Master[1].N_Output = 1
; ---------- Error reactions CL(none) ------------
1 SD.SysAxCoupleCmd[CR%].AbortEnable        = 0
1 SD.SysAxCoupleCmd[CR%].AbortDelay         = 0
1 SD.SysAxCoupleCmd[CR%].AbortRetractPos    = 0.0
1 SD.SysAxCoupleCmd[CR%].AbortAction        = 0
1 SD.SysAxCoupleCmd[CR%].AbortAsupNo        = 0
1 SD.SysAxCoupleCmd[CR%].AbortAsupChan      = 0
; ---------- Synchronous running window CL ----------------
```

```
1 SD.SysAxCoupleCmd[CR%].PosWin[1]        = 0.02
1 SD.SysAxCoupleCmd[CR%].PosWin[2]        = 0.01
1 SD.SysAxCoupleCmd[CR%].PosWinSel        = 1
;
;-------------------------------------------------
; Apply coupling data
;-------------------------------------------------
CALL SAC_VALID
```

Activating a master axis coupling

Program:

```
;-------------------------------------------------
; Program name: SAC_LEADON
; function:
;   couple leading axis P2% to follower axis P1%.
;   require already executed SAC
;   parameterization:
;
;   SD.SysAxCoupleCmd[P1%].OnOff    = 1
;   SD.SysAxCoupleCmd[P1%].N_INPUT  = input speed
;   SD.SysAxCoupleCmd[P1%].N_OUTPUT = output speed
;   SD.SysAxCoupleCmd[P1%].AxIndex  = system axis index
;
; parameter:    P1% - index of follower axis
;               P2% - index of leading axis
;-------------------------------------------------
1 AXIS%   = P1%
1 MASTER% = P2%
1 SD.SysAxCoupleCmd[AXIS%].Master[MASTER%].Enable = TRUE
1 CALL SAC_VALID
```

Synchronizing a master axis coupling

Program:

```
;-------------------------------------------------
; Program name: SAC_LEADSYNC
; function:     wait until follower axis P1%
;               is reference value synchron.
;
; parameter:    P1% - index of follower axis
;-------------------------------------------------
1 REPEAT
1  WAIT(,20)
1 UNTIL (SD.SysAxCoupleSta[SLAVE%].State=3)
WAIT
```

14.5.8 System data overview

SAC control structure: SysAxCoupleCtr

Element	Type	Unit	Value range	Description
.Validate	Byte	-	0, 1	0: Transfer completed 1: Activate transfer

.CfgErrNum	Byte	-	-	0: No error 1: Circulation (auto-coupling) 2: Nesting too deep 3: Axis cannot become slave axis (e.g. spindle) 4: Master enabled at invalid master index 5: Master enabled without input master revolution 6: Master enabled without master output revolution 7: Monitoring window index invalid 8: ASUP number invalid 9: ASUP channel does not exist 10: AbortAction contains an invalid combination 11: Sync mode (for axis type) not allowed 12: Invalid coupling velocity 13: Invalid coupling acceleration 14: Invalid input speed 15: Invalid output speed 16: Invalid offset 17: Invalid retract position 18: Invalid InPos window 1 19: Invalid InPos window 2 20: Table is not loaded 21: Invalid switching condition 22: Invalid lower switching limit 23: Invalid upper switching limit 24: Switching range too small 25: Modulo range of table slave axis invalid 26: Modulo range of master table invalid 27: Invalid master axis offset 28: Invalid reference value mode 29: Syntax error in the formula expression 30: Formula runtime timeout 31: Formula interpretation error
.CfgErrAx	unsigned byte	-	0..250	1-250: Index of the incorrect SAC

.CfgErrMaster	Byte	-	0..250	0 - Not master relevant 1..5 - Incorrect master Formula error (CfgErrNum 29,31): 1..250 - Syntax error position in formula
.ExtCond	Unsigned Int	-		Bit mask for external real-time switching: Bit 2^0- Enables SwitchCond 100 Bit 2^1- Enables SwitchCond 101 ... Bit 2^{31}- Enables SwitchCond 131

Tab. 14-26: Elements of the "SysAxCoupleCtr" structure

SAC control structure:

SysAxCoupleCmd[i]

Element	Type	Unit	Value range	Description
.OnOff	Byte	-	0.1	0: Deactivate coupling 1: Activate coupling
.Reset	Byte	-	0.1	0: No reset 1: SAC reset
.IpoEnable	Boolean	-	TRUE, FALSE	TRUE: The interpolated axis value is added to the reference value FALSE: The axis value is not taken into consideration
.RecoupleEnable	Boolean	-	TRUE, FALSE	TRUE: Recoupling permitted FALSE: Interruption of coupling triggers error
.AccCouple	Double	m /s^2 1000°/s^2	0: AxMaxAcc >0.. AxMaxAcc	Maximum acceleration during coupling and decoupling
.VelCouple	Double	mm/min °/min	0: AxMaxVel >0.. AxMaxVel	Maximum velocity during coupling and decoupling
.GearTeeth	Int	-	0.. MaxInt	0: Absolute coupling >1: Number of teeth of the slave axis gearwheel
.SyncMode	Short	-	0..4	Coupling mode: 0: Optimum distance 1: time-optimized 2: Velocity-synchronous 3: Positive velocity direction 4: Negative velocity direction

.Rigidity	Byte	-	0.1	0: Soft coupling with max. AccCouple and VelCouple 1: Hard coupling with parameterized max. axis acceleration and max axis velocity
.PosWinSel	Byte	-	0..3	0: no synchronous run monitoring 1: Lag monitoring with PosWin[1] 2: Lag monitoring with PosWin[2] 3: Lag monitoring with PosWin[1] and PosWin[2]; Hysteresis response
.PosWin[i], i = [1,2]	Double	mm,°	>0..MaxPos	Two accuracy windows for synchronous run determination
.ExtError	Unsigned short	-	0..65535	Report external error 0: No error >0: External error code
.AbortEnable	Int	BITMASK	0 or Sum Bit 0..24	Reaction enabled for error source: 0: no error handling # 1 Bit 0: Error in drive # 2 Bit 1: Error in axis coupling # 4 Bit 2: External error GUI/PLC/CPL # 8 Bit 3: Error in interpolation channel # 16 Bit 4: Error in slave axis (cascade) #32 Bit 5: Error in first master axis (cascade) #64 Bit 6: Error in second master axis (cascade) #128 Bit 7: Error in third master axis (cascade) #256 Bit 8: Error in fourth master axis (cascade) #512 Bit 9: Error in fifth master axis (cascade) # 65536 Bit 16: Formula error zero division # 131072 Bit 17: Formula error zero overflow # 262144 Bit 18: Formula error zero undefined # 524288 Bit 19: Formula error zero underflow # 1048576 Bit 20: Formula error "processing #16777216 Bit 24: Exiting the selected accuracy window of the synchronous run monitoring
.AbortAsupNo	Byte	-	0..8	0: No ASUP start 1..8: Start asynchronous subroutine in case of error
.AbortAsupChan	Byte	-	0..60	0: Channel of the axis (for synchr. axes) 1..60: Channel in which ASUP is to be started.
.AbortDelay	Int	ms	0..MaxInt	Abort waiting time in ms before an abort reaction (AbortAction) is triggered.

.AbortAction	Int	BITMASK	0 or Bit 0..6	Abort treatment after waiting time: 0: Coupling keeps on running - or - #1 Bit 0: Retraction absolute after AbortRetractPos - or - #2 Bit 1: Retraction relative by AbortRetractPos - or - # 4 Bit 2: Report channel error or a combination of the bits 3-8: # 8 Bit 3: Switch off interpolator reference value # 16 Bit 4: Switch off master axis reference value 1 # 32 Bit 5: Switch off master axis reference value 2 # 64 Bit 6: Switch off master axis reference value 3 # 128 Bit 7: Switch off master axis reference value 4 # 256 Bit 8: Switch off master axis reference value 5 Remark: Upon deactivation of all reference values, the slave axis is stopped.
.AbortRetractPos	Double	mm,°		Retract position
.Formula	String	-	Max. 250 characters	Reference value formula
.Master[m], m=[1..5]				Substructure for master axis definition

	.Enable	Boolean	-	TRUE, FALSE	TRUE: Consider reference value for slave axis FALSE: Reference value not considered
	.AxIndex	unsigned byte	-	0..250	0: Master axis not defined 1..250: System axis index of master axis
	.Switch-Cond	Short	-	0, 1, 2, 100, ..., 131	0 - Unconditional switching 1 - Position-conditional switching 2 - Velocity-conditional switching 100 - Switching trigger ExtCond bit 2^0 ... 131 - Switching trigger ExtCond bit 2^{31}
	SwitchMin	Double	mm,° mm/min, °/min		Lower switching position or switching speed
	SwitchMax	Double	mm,° mm/min, °/min		Upper switching position or switching speed
	.Offset	Double	mm, °		Offset of the slave axis position compared to the master axis.
	.N_Input	Double	-		Number of master axis revolutions
	.N_Output	Double	-		Number of slave axis revolutions
	.Shift	Double	mm,°		Master axis shift (phase shift for table)
	.Mode	Unsigne-dInt	-	0.1	0 - Modulo interval of the reference value depends on the slave axis 1 - Modulo interval of the reference value depends on the master axis
	.Tab	String	-	Max. 30 characters	Name of the coupling table without path "" - No table coupling "<Tab>" - Table coupling with <Tab> "<Place>" - Table coupling with SysAxCoupleTab[<Place>]

Tab. 14-27: Elements of the structure SysAxCoupleCmd[i]

SAC status structure:

SysAxCoupleSta[i]

Element	Type	Unit	Value range	Description
.Cmd	Struct SysAxCoupleCmd	-		Active control data in the interpolator.

.State	Byte	-	0..7 11..17 21..27	Status of the system axis coupling: 0: Off 1: Coupling (HUNTING) 2: Recoupling (RECOUPLING) 3: Synchronous (COUPLED) 4: Stopping 5: Stopped 6: Decoupling (DISCOUPLE) 7: Error 1x: Abort (ABORT-x) 2x: Conditional switching (PENDING-x)
.StateAddOn	Int	-	0, 1000, 2000, 3000	Additional information on the state of the system axis coupling: 0: No additional information 1000: Accuracy limit 1 of the synchronous run monitoring is not violated 2000: Accuracy limit 2 of the synchronous run monitoring is not violated 3000: Accuracy limits 1 and 2 of the synchronous run monitoring are not violated
.Abort	Byte	-	0..4	Status of abort treatment: 0:- Inactive 1: Abort delay 2: Retraction active 3: (Modified) coupling active 4: Abort terminated, error state
.FormulaDiag	Byte	-	0..9	Formula runtime error: 0: No error 1: Zero division 2: Overflow (result > 1.0E99) 3: Value range not defined 4: Underflow (result < 1.0E-99) 5-9: Formula canceled (internal error)
.FormulaDiagPos	Byte	-	0..250	Operator or operand position in SysAxCoupleSta[axis].Cmd.Formula resulting in an error.

.Error	Int	-	0..8	Abort error and/or error source: 0: No error 1: Coupling torn off 2: Axis error 3: Channel error 4: Error due to external event 5: Error in a slave axis 6: Error in a master axis 7: Error in the formula calculation 8: Error in the synchronous run monitoring. The actual position of the slave axis exited the selected accuracy window and no recoupling enable (RecoupleEnable=FALSE) is active.
.InPos	Int	-	0..1	State of synchronous run monitoring: a) PosWinSel = 1 or 2: InPos = 0, if |actual $_{\text{reference value}}$ - actual $_{\text{slave axis}}$|>PosWin[PosWinSel] InPos = 1 if |actual $_{\text{reference value}}$ - actual $_{\text{slave axis}}$|<=PosWin[PosWinSel] b) PosWinSel = 3: InPos = 0, if |actual$_{\text{reference value}}$ - actual $_{\text{slave axis}}$|>PosWin[1] and |actual $_{\text{reference value}}$ - actual $_{\text{slave axis}}$|>PosWin[2] InPos = 1 if |actual$_{\text{reference value}}$ - actual $_{\text{slave axis}}$| <= PosWin[1] and |actual $_{\text{reference value}}$ - actual $_{\text{slave axis}}$| <= PosWin[2] Otherwise, InPos keeps its current value Remark: For PosWinSel = 0 or SysAxCoupleCmd[i].State = 0 or > 3, InPos is always set to 0.
.Lag	Double	mm, °		Difference |actual $_{\text{reference value}}$ - actual $_{\text{slave axis}}$
.AbortPos	Double	mm, °	-	Position at time of last retract trigger.
.FormulaWatch	Double	-		Partial result of the formula (WATCH function)
.ActPos	Double	mm, °	-	Current command position of the slave axis
.ActVel	Double	mm/min, °/min	-	Current velocity of the slave axis
.ActOffset	Double	mm, °		Current position difference of the slave axis to the reference value.
.ActMaster[m], m=[1..5]				Substructure with active master

	.AxIndex	Short	-	0..250	Active master axis: 0: Master axis not defined 1..250: System axis index of master axis
	.Offset	Double	-		Active reference value offset
	.N_Input	Double	-		Active input speed
	.N_Output	Double	-		Active output speed
	.Shift	Double	mm,°		Active master axis shift
	.Tab	String	-	30 characters	Active coupling table

Tab. 14-28: Elements of the structure SysAxCoupleSta[i]

SAC table management

SysAxCoupleTab[i]

Element	Type	Unit	Value range	Description
.FileName	String	-	100 characters	Source file name of the coupling table with path specification.
.State	Short	-	0..4	Table state 0 - Not assigned 1 - Table is loaded 2 - Table is loaded 3 - Table is used 4 - Table is removed
.FileTime	String	dd:mm:yy hh:mm:ss	18 characters	Time stamp of the loaded coupling table file.
.CacheSize	Int	Bytes		Memory assigned by table.

Tab. 14-29: Elements of the structure SysAxCoupleTab[i]

14.6 Channel axis coupling

14.6.1 Description

Function

The channel axis coupling establishes a fix relation between the axis coordinates (ACS) of a master and a slave axis.

Master and slave axes are coupled to an axis group. Each axis group has exactly one master axis and a maximum of seven slave axes. All axes of one axis group have to be located in one channel. Multiple axis groups can also belong to one channel.

If the slave axes are not in the coupling position when the coupling is activated, they are moved to this position by an internally generated linear traversing motion. The active path feed and the override potentiometer are applied.

Generally, the channel axis coupling can also be combined with other coupling variants. The order of the couplings is:

- Channel axis coupling
- System axis coupling (see chapter 14.5 "System axis coupling" on page 528)
- Gantry coupling, (see chapter 14.1 "Gantry axes" on page 485)

In an axis group, there can be parallel axes, electronic gears and other arbitrary coupling relations at the same time.

It is differentiated between linear and arbitrary coupling of axis positions with each other.

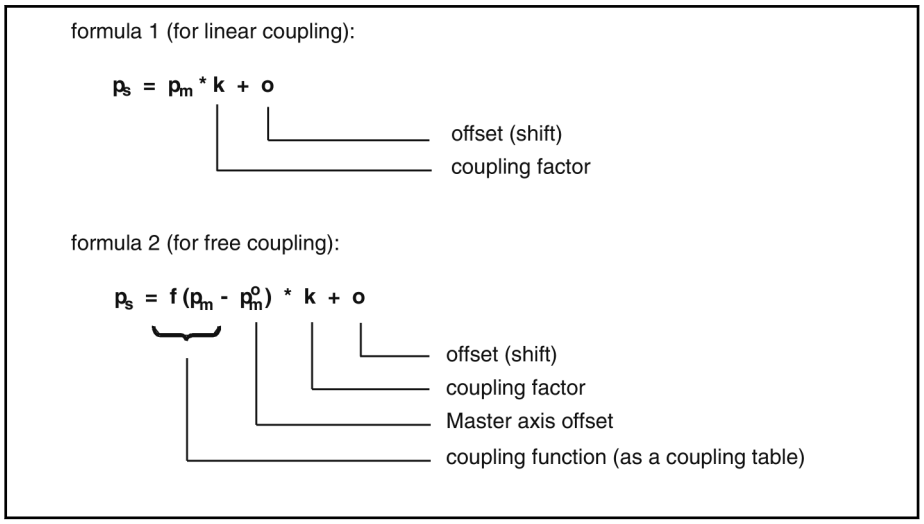


Fig. 14-39: Formulas for axis coupling

The function f(p) is stored as function table (coupling table) in the NC memory.

Structure and mode of action of coupling tables are described in detail in chapter 14.4 "Coupling tables" on page 524.

An axis group consists of one master axis and one or multiple slave axes. Each slave axis has its individual coupling relation to the master axis according to (formula 1) or (formula 2).

The right sides of the formulas 1 and 2 represent the reference values parameterized with regard to the master position. These values are used as position specification for the slave axes in each interpolation cycle.

Example of coupling relations:

- Parallel axes (parallely arranged machining tables)
- Electronic gears (drives moving towards each other in a specified ratio)

Special features and restrictions

- A member of an axis transformation cannot be a slave axis of an axis coupling.
- A Hirth axis cannot be a slave axis of an axis coupling.
- A gantry axis cannot be a slave axis of an axis coupling.
- A slave axis cannot be the master axis of another axis coupling at the same time.
- Linear coupling:

  If the master axis is a modulo axis, the slave also has to be a modulo axis. The following restriction applies to the modulo value:

  (Modulo value $_{Master}$ * Coupling factor) modulo modulo value $_{Slave}$ = 0
- Any coupling:

  If the master axis is a modulo axis, the coupling table has to be periodically (table entry #20 = 1). The modulo value of the master axis has to be a multiple of the period of the table.

Modulo value $_{Master}$ modulo modulo value $_{Table}$ = 0

- There is no lag monitoring.

Relevant NC functions

AxCouple(..)	Establish/cancel channel axis coupling
CoupleSplineTab(..)	Compile coupling table

Tab. 14-30: Relevant NC functions, channel axis coupling

Relevant CPL functions

SD(582,0,m)	Channel axis number m if the axis m is a master axis of an axis coupling. 0: If m is not a master axis.
SD(582,s,0)	Number of the master axis for which the axis s is a slave axis. 0: If s is not a slave axis.
SDR(582,s,1)	Programmed slave axis offset in the current programming unit (mm, inch or degree). 0: If s is not a slave axis.
SDR(582,s,2)	Programmed coupling factor in the current programming unit. 0: If s is not a slave axis.
SDR(582,s,3)	Programmed master axis offset in the current programming unit. 0: If s is not a slave axis.
with: m: Number of the master axis 1..8 s: Number of the slave axis 1..8	

Tab. 14-31: Relevant CPL functions, channel axis coupling

Relevant IF signals

iAx_MasterAxIndex_00 .. iAx_MasterAxIndex_06

The values of these seven signals show the binary-coded system axis number of the master axis of a channel axis coupling or a gantry coupling.

- If the axis is not coupled, the value of the master axis number is 0.
- If the axis is the master, the master axis number is identical to the axis number.

Examples

```
N100 AXC(C0,A(),B())
```

Coupling with the distance 0.

C: Master axis

A/B: Slave axes

Both slave axes run with "linear" coupling.

The default values "coupling distance 0.0" and "coupling factor 1.0" are applied.

	`N100 AXC(C0,A(ACT),B(ACT,2))`	Coupling at current axis positions. C: Master axis A/B: Slave axes Both slave axes run with "linear" coupling. A and B are coupled to C without approaching motion. B is operated with coupling factor 2.
	`N100 AXC(C0,B(,0.5,,T_B))`	Create axis group C: Master axis B: Slave axis Slave B runs with "any" coupling.
Creating axis coupling	`N100 AXC(C0,B())`	B is coupled to C as slave axis. Slave axis B runs with "linear" coupling.
Extending axis coupling	`N150 AXC(C1,A(4.5,2))`	A is added as slave axis. Slave axis A runs with "linear" coupling.
Reducing axis coupling	`N200 AXC(C-1,A())`	Slave axis A is removed from axis group c.
Canceling axis coupling	`N250 AXC(C-1)`	The entire axis group c is deleted.

14.6.2 Applying

If coupling tables are required for a coupling group, they have to be created accordingly.

For more information on the coupling tables, refer to chapter 14.4 "Coupling tables" on page 524.

14.6.3 Activating

By programming the NC function "AxCouple (...)", the function is activated (see examples above).

When working with coupling tables, they can be checked before and compiled with the NC function "CoupleSplineTab(...)".

14.6.4 Deactivating

The "AxCouple" function can switch off a specified coupling or all couplings:

- AxCouple(<Masteraxisname>, -1) completes the coupling with the programmed master axis.
- AxCouple() completes all couplings.

15 Measuring functions

15.1 Traveling against touch probe (G75)

15.1.1 Description

Summary

G75 is used to automatically measure workpiece positions. Instead of a workpiece, a touch probe is chucked. For a G75 block, the NC travels to the programmed direction until the touch probe hits the workpiece. The position of the touch probe is immediately stored and the motion is stopped.

The NC is provided with the following two G75 functions:

- Travel against the first touch probe (G75.1 or G75)
- Travel against the second touch probe (G75.2)

Terms for G75

Touch probe:

- All sensors checking the workpiece position, e.g. mechanical touch probes, echo sensors, etc.
- Only knows two states: "workpiece" or "no workpiece"
- Converts these states into an electrical output signal

Travel axes:

- They are programmed as axis coordinates in the G75 block
  (e.g. for G75.1 X100 Y200 F1000, X and Y are travel axes)
- Can simultaneously be a measuring axis (optional)

Measuring axes:

- Only Sercos drives supporting the touch probe function
- Are accordingly defined by a machine parameter (MP **EnablProbeFunc** "touch probe function available")
- Store the measured value when receiving the edge of the touch probe
- Can simultaneously be a travel axis (optional)

The measured value:

- is the value stored by the measuring axis when receiving the edge of a touch probe
- always derives from the source determined before via by S-0-0426 "Signal selection, touch probe 1" or S-0-0427 "Signal selection, touch probe 2"
- knows only the value of the active encoder system as source

What happens in the G75 block?

G75 travels and measures.

G75 starts the touch probe function in all measuring axes of the channel.

The travel axes move linearly with the programmed feed towards the point resulting from the programmed coordinates.

All measuring axes receiving a touch probe edge in the meantime, store the detected measured value immediately.

At the same time, they send the signal "touch probe edge detected" to the NC.

The NC permanently checks whether it receives the signal "Touch probe edge detected" from the measuring axis with the measuring axis.

If yes: The NC immediately stops all travel axes, stores the measured values of all measuring axes and deletes the distance to go.

If no: The travel axes move up to the programmed coordinates and stop.

Finally, the NC continues with the next program line.

Why did the touch probe not trigger?

Possible reasons:

- The search depth of the workpiece indicated by the axis coordinates is too small.
- The touch probe triggered but the measuring axes do not receive a touch probe edge; there is possibly a wiring error. - In this case, the touch probe can be damaged.

What happens after the G75 block?

Following the G75 block, the user has to:

- check whether the touch probe triggered
- ensure that the measured values are applied to the part program
- ensure that the required warnings and error messages are generated

For this purpose, appropriate CPL functions are available.

PLC Interface

The PLC can determine whether a G75 block is active. For this purpose, specify a corresponding bit at the "NC function-specific bit interface". This bit becomes active for the period of the G75.1 (G75) or G75.2 block.

Relevant NC functions

G75.1 (or G75)	Travel against first touch probe
G75.2	Travel against second touch probe

Tab. 15-1: Relevant NC functions

Relevant CPL functions

SD(9)	Provides information on whether the touch probe triggered.
PPOS(), PROBE(), PCSPROBE()	Provides the measured value of the (synchronous) measuring axis (but differently).

Tab. 15-2: Relevant CPL functions

Relevant machine parameters (MP)

Probe1Edge	Path: MEAS / Dr[i] / Probe1Edge Name: "Edge of touch probe 1" (corresponds to Macoda-ID 1003 00011)
EnablProbeFunc	Path: MEAS / Dr[i] / EnablProbeFunc Name: "Touch probe function available" (corresponds to Macoda-ID 1003 00012)

Probe2Edge	Path: MEAS / Dr[i] / Probe2Edge Name: "Edge of touch probe 2" (corresponds to Macoda-ID 1003 00013)
NcFunc[1..24]	Path: PLC / NcFuncBitIf / NcFunc[1..24] Name: "NC function-specific bit interface" (corresponds to Macoda ID 3020 00001/ 3020 00002)

Tab. 15-3: Relevant machine parameters (MP)

15.1.2 Commissioning

Machine parameters (MP)

Determining measuring axes:

All system axes the user likes to use as measuring axes have to be determined as such via the machine parameter **EnablProbeFunc** "touch probe function available", i.e.:

- if the system axis is only needed for **G75.1** (G75), set

  MEAS / Dr[i] / EnablProbeFunc = "only touch probe 1" in the IndraWorks Engineering
- if the system axis is needed for **G75.1** (G75) and for **G75.2**, set

  MEAS / Dr[i] / EnablProbeFunc = "both touch probes" in the IndraWorks Engineering

Determining touch probe edges:

As soon as one system axis is defined as measuring axis, specify on which touch probe it is supposed to react. This is accomplished via IndraWorks Engineering.

- **G75.1** (G75) always belongs to touch probe 1 whose edge is selected via the machine parameter **Probe1Edge** "edge, touch probe 1".

  There are only two options:

  MEAS / Dr[i] / Probe1Edge = "positive edge"

  - or -

  MEAS / Dr[i] / Probe1Edge = "negative edge"
- **G75.2** always belongs to touch probe 2 whose edge is selected via the machine parameter **Probe2Edge** "edge, touch probe 2". - There are only two options:

  either

  MEAS / Dr[i] / Probe2Edge = "positive edge"

  or

  MEAS / Dr[i] / Probe2Edge = "negative edge"

At least one measuring axis:

- For **G75.1-** (G75) block, the following applies:

  The channel has to have at least one measuring axis.
- For the **G75.2**block, the following applies:

  The channel has to have at least one measuring axis which supports touch probe 2.
- Otherwise, the result is:

  Error 2105 "No channel axis is enabled for touch probe enabled."

Unless the channel is in test mode.

Same touch probe edge:

- All measuring axes whose measured values are required including the measuring axis with the smallest system axis index have to trigger at the same touch probe edge.

  The settings of the user are **not** checked by the NC.

- Background information:

  For a G75 block, all corresponding measuring axes are enabled for the next measurement:

  – Irrespective of the touch probe edge set
  – Irrespective of whether all measuring axes actually receive the signal from one and the same touch probe
  – Irrespective of whether the measuring axis is also the travel axis
  – Irrespective of whether the measured value of each measuring axis is actually used afterwards

Determining the signal "G75 active":

- The user can specify a maximum of one "active" bit at the channel-specific PLC interface for the same G75 function.

  Thus, there are four alternatives for G75:

  – No bit; neither for G75.1 nor for G75.2
  – Only one bit for G75.1
  – Only one bit for G75.2
  – Two bits; one for G75.1 and one for G75.2

- It is specified in IndraWorks Engineering via:

  PLC / NcFuncBitIf / **NcFunc[1..24]**

  The user writes the name of the G75 function into the respective path. The index number corresponds to the bit number, i.e. by entering e.g. The ”G75.2" under NcFunc[21], the signal "G75.2 active" is assigned to Bit21 of the NC function-specific interface.

- The NC functions G75.1 and G75 are identical and therefore use the same bit - irrespective of whether the bit was specified as "G75" or "G75.1".

- If two or more "active" bits were specified for the same G75 function, the NC only uses the bit with the smallest index. The others are not applied.

Sercos parameters

The touch probe function of the drive is specified by Sercos international. The NC uses the following Sercos parameters for G75 blocks:

Sercos parameters:		Assigned:		Relevant for
IDN	Name	G75.1	G75.2	SCS file
S-0-0130	Measured value 1, positive edge	X	-	Optional
S-0-0131	Measured value 1, negative edge	X	-	Optional
S-0-0132	Measured value 2, positive edge	-	X	Optional
S-0-0133	Measured value 2,negative edge	-	X	Optional

Sercos parameters:		Assigned:		Relevant for
S-0-0169	Control parameter for touch probe	X	X	-
S-0-0170	Command for touch probe cycle	X	X	-
S-0-0405	Probe 1 enable	X	-	-
S-0-0406	Touch probe 2 enabled	-	X	-
S-0-0409	Touch probe 1 positively recorded	X	-	Optional
S-0-0410	Touch probe 1 negatively recorded	X	-	Optional
S-0-0411	Touch probe 2 positively recorded	-	X	Optional
S-0-0412	Touch probe 2 negatively recorded	-	X	Optional
S-0-0426	Signal selection probe 1	X	-	Mandatory
S-0-0427	Signal selection of touch probe 2	-	X	Mandatory

X Yes
\- No

Tab. 15-4: Sercos parameters for G75 blocks

Remark:

- If the drive is provided with the Sercos parameter required for the G75 block, it also supports the respective touch probe function.
- If not, the respective drive cancels the Sercos startup and refers to the missing Sercos IDN with a corresponding error message and thus to the missing touch probe function.
- Nevertheless, additional settings and checks at the drive can be necessary. The user is responsible for those settings and checks before operating the touch probe function of the NC. For detailed information, refer to the documentation of the drive manufacturer.

SCS file or Sercos file

The SCS file is used to set the NC for each measuring axis:

- the mandatory Sercos parameters and
- the optional Sercos parameters (required by the user).

SCS files always have to be adjusted (either by the machine tool manufacturer or by the user). Thus, each editor is responsible.

Mandatory parameters in SCS file

- For part programs with G75.1 (G75) blocks

  S-0-0426 "Signal selection, touch probe 1" has to be parameterized in the SCS file.
- For part programs with G75.2 blocks

  S-0-0427 "Signal selection touch probe 2" must be parameterized in SCS file.
- Both parameters S-0-0426 and S-0-0427 can be set independently from each other -

either on

S-0-0051 "Actual position value encoder 1"

or

S-0-0053 "Actual position value encoder 2".

Optional parameters in SCS file

G75.1:

If there are no optional parameters in the SCS file which are assigned to the G75.1 (G75) block, the following is performed automatically for each G75.1 block:

- the real-time status bit 1 is used for

  "touch probe 1 recorded positively" (S-0-0409) or

  "recorded negatively for touch probe 1" (S-0-0410) -

  depending on whether the machine parameter **Probe1Edge** specifies a positive or negative touch probe edge.
- the measured value is read via the (Sercos) service channel, i.e. either

  "measured value 1 positive edge" (S-0-0130) or

  "measured value 1 negative edge" (S-0-0131) -

  depending on whether the machine parameter **Probe1Edge** specifies a positive or negative touch probe edge.

Thus, in this case, there is neither S-0-0409, S-0-0410, S-0-0130 nor S-0-0131 in the SCS file.

G75.1:

To read the measured value cyclically from the G75.1 (G75) block, since it is faster than via the service channel, set

S-0-0015 "Telegram type" and

S-0-1050.1.6 "Configuration list AT" as follows:

- **S-0-0015** to "Freely configurable telegram"
- **S-0-1050.1.6** adding of

  "measured value 1 positive edge" (**S-0-0130**) or

  "measured value 1 negative edge" (**S-0-0131**) -

  depending on whether the machine parameter **Probe1Edge** specifies a positive or negative touch probe edge.

In this case, the measured value is not read via the service channel.

G75.1:

In order not to use the real-time status bit 1 for the G75.1 (G75) block but for other purposes, set

S-0-0015 "Telegram type"

S-0-1050.1.6 "Configuration list AT"

S-0-0026 "Configuration list, signal status word" and

S-0-0328 "Bit assignment list, signal status word" in the SCS file as follows:

- **S-0-0015** to "Freely configurable telegram"
- **S-0-1050.1.6** adding signal status word (**S-0-0144**)
- **S-0-0026** adding of

"touch probe 1 recorded positively" (**S-0-0409**) or

"touch probe 1 recorded negatively" (**S-0-0410**) -

depending on whether the machine parameter **Probe1Edge** specifies a positive or negative touch probe edge.

- **S-0-0328** adding "0" at the same list position which incorporates S-0-0409 or S-0-0410 in the S-0-0026 list.

G75.2:

If there are no optional parameters in the SCS file which are assigned to the G75.2 block, the following is performed automatically for each G75.2 block:

- the real-time status bit 1 is used for

  "recorded positively for touch probe 2" (S-0-0411) or

  "recorded negatively for touch probe 2" (S-0-0412) -

  depending on whether the machine parameter **Probe2Edge** specifies a positive or negative touch probe edge.
- the measured value is read via the (Sercos) service channel, i.e. either

  "measured value 2 positive edge" (S-0-0132) or

  "measured value 2 negative edge" (S-0-0133) -

  depending on whether the machine parameter **Probe2Edge** specifies a positive or negative touch probe edge.

Thus, in this case, there is neither S-0-0411, S-0-0412, S-0-0132 nor S-0-0133 in the SCS file.

G75.2:

To read the measured value cyclically from the G75.2 block, since it is faster than via the service channel, set

S-0-0015 "Telegram type" and

S-0-1050.1.6 "Configuration list AT" as follows:

- **S-0-0015** to "Freely configurable telegram"
- **S-0-1050.1.6** adding of

  "measured value 2 positive edge" (**S-0-0132**) or

  "measured value 2 negative edge" (**S-0-0133**) -

  depending on whether the machine parameter **Probe2Edge** specifies a positive or negative touch probe edge.

In this case, the measured value is not read via the service channel.

G75.2:

In order not to use the real-time status bit 1 for the G75.2 block but for other purposes, set

S-0-0015 "Telegram type"

S-0-1050.1.6 "Configuration list AT"

S-0-0026 "Configuration list, signal status word" and

S-0-0328 "Bit assignment list, signal status word" in the SCS file as follows:

- **S-0-0015** to "Freely configurable telegram"
- **S-0-1050.1.6** adding signal status word (**S-0-0144**)

- S-0-0026 adding of

  "recorded positively for touch probe 2" (S-0-0411) or

  "Touch probe 2, negatively recorded" (S-0-0412) -

  depending on whether the machine parameter Probe2Edge specifies a positive or negative touch probe edge.
- S-0-0328 adding "0" at the same list position which incorporates S-0-0411 or S-0-0412 in the S-0-0026 list.

Irrelevant Parameters for SCS File

The following is performed automatically at the Sercos phase startup:

- generated from the machine parameters EnablProbeFunc, MP Probe1Edge and MP Probe2Edge of the "Touch probe control parameters" (S-0-0169) and written to the drive
- using the machine parameter EnablProbeFunc in the drive

  the real-time control bit 1 with "touch probe 1 enabled" (S-0-0405) and

  the real time control bit 2 with "touch probe 2 enabled" (S-0-0406) is assigned or
- the "Command touch probe cycle" (S-0-0170) is not created from the machine parameter EnablProbeFunc and written for drive.

Thus, S-0-0169, S-0-0170, S-0-0405 and S-0-0406 may never be in the SCS file.

15.1.3 Handling instruction: Measuring with the touch probe functions G75.1 and G75.2

Note

The following example shows how to start the touch probe functions G75.1 and G75.2.

The example is only a template. You have to adjust it to your requirements.

The commissioning of the touch probe functions G75.1 and G75.2 is only an addition to the regular commissioning.

Task

The following system axes are given: 1, 2, 3, 4, 5 and 6.

- The system axes 2, 3, 5 and 6 has to be defined as measuring axes for G75.1 and react to the positive touch probe edge.
  - The internal encoder is used as source for the measured value.
  - The measured values are cyclically transferred.
  - No separate real-time status bit.
  - "G75.1 active" is output for channel 1 at bit 23 of the NC function-specific PLC interface.
- The system axes 5 and 6 have to be additionally defined as measuring axes for G75.2 and react to the negative touch probe edge.
  - The external encoder is used as source for the measured value.
  - The measured values are not cyclically transferred.
  - No separate real-time status bit.
  - "G75.2 active" is output for channel 1 at bit 24 of the NC function-specific PLC interface.

Drives

- Ensure that the drives assigned to the system axes 2, 3, 5 and 6 support the touch probe functionality. Use the drive documentation provided by the manufacturer.
- Ensure that the touch probe functionality is provided in the drives as requested by the drive manufacturer. For this purpose, additional activities can be necessary.
- Ensure that the signal for touch probe 1 is generated correctly and connected to the correct hardware input of drives 2, 3, 5 and 6.
- Ensure that the signal for touch probe 2 is generated correctly and connected to the correct hardware input of drives 5 and 6.

Machine parameters (MP)

Set the machine parameters in the listed IndraWorks Engineering paths to the defined values.

Changes have to be saved.

MEAS / Dr[1] /

- EnablProbeFunc = "No"
- Probe1Edge = irrelevant
- Probe2Edge = irrelevant

MEAS / Dr[2] /

- EnablProbeFunc = "Only touch probe 1"
- Probe1Edge = "positive edge"
- Probe2Edge = irrelevant

MEAS / Dr[3] /

- EnablProbeFunc = "Only touch probe 1"
- Probe1Edge = "positive edge"
- Probe2Edge = irrelevant

MEAS / Dr[4] /

- EnablProbeFunc = "No"
- Probe1Edge = irrelevant
- Probe2Edge = irrelevant

MEAS / Dr[5] /

- EnablProbeFunc = "Both touch probes"
- Probe1Edge = "positive edge"
- Probe2Edge = "negative edge"

MEAS / Dr[6] /

- EnablProbeFunc = "Both touch probes"
- Probe1Edge = "positive edge"
- Probe2Edge = "negative edge"

For channel 1:

PLC / NcFuncBitIf /

- NcFunc[23] = "G75.1"
- NcFunc[24] = "G75.2"

SCS files or Sercos files

Add the following to the SCS files of the drives 1 to 6 and save the modified SCS files.

SCS file drive 1:

- Ensure that there are no Sercos parameters in the SCS file relevant for G75 (this means none parameters listed in tab. 15-4 "Sercos parameters for G75 blocks" on page 596).

SCS file drive 2:

- S-0-0426 = (51) ; S-0-0051 "Actual position value encoder 1"
- S-0-0015 = (0b111) ; binary 7: "Freely configurable telegram"
- S-0-1050.1.6 = (..., 130, ...) ; add S-0-0130 "measured value 1 positive edge"
- Ensure that the SCS file does not contain any of the parameters listed in tab. 15-4 "Sercos parameters for G75 blocks" on page 596 except for S-0-0426 and S-0-0130.

SCS file drive 3:

- As SCS file drive 2.

SCS file drive 4:

- As SCS file drive 1.

SCS file drive 5:

- S-0-0426 = (51) ; S-0-0051 "Actual position value encoder 1"
- S-0-0427 = (53) ; S-0-0053 "Actual position value encoder 2"
- S-0-0015 = (0b111) ; binary 7: "Freely configurable telegram"
- S-0-1050.1.6 = (..., 130, ...) ; add S-0-0130 "measured value 1 positive edge"
- Ensure that the SCS file does not contain any of the parameters listed in tab. 15-4 "Sercos parameters for G75 blocks" on page 596 except for S-0-0426, S-0-0427 and S-0-0130.

SCS file drive 6:

- As SCS file drive 5.

Final Activity

Restart control and drives.

During the Sercos startup, the control transfers the SCS files to the corresponding drives.

Activating

A measurement can only be started in the part program. It is required to program the NC functions G75.1 or G75.2.

The following example shows the determination of the workpiece position of the system axes 2 and 3 (in the part program named as Y and Z) using touch probe 1. The touch probe is configured and thus creates a positive edge when contacting the workpiece.

Program:

```
N10 G0 Y100 Z0                ; Approaching starting position for measurement
N20 F500                      ; Set feed for measurement
N30 G75.1 Y200 Z100           ; Approach position in workpiece
N40 WAIT                      ; Synchronization with NC preparation
110 IF SD(9)=0 THEN           ; if touch probe triggered
120   YPOS = PPOS("Y") ; Storing measured value of y-axis in YPOS
```

```
130   ZPOS = PPOS("Z") ; Storing measured value of z-axis in ZPOS
N140  MSG(CONTACT)           ; Output message
150 ELSE                     ; Touch probe not triggered
160 SETWARN("NO CONTACT!") ; Outputting warning
N170  M0                     ; Waiting on NC start
180   WAIT                   ; Synchronization with NC preparation
190   CLRWARN()              ; Delete warning
200 ENDIF
```



When the touch probe triggers (also contact with the workpiece), the NC stops the motion to impede damages at the touch probe. The programmed target position is thus not reached.

15.2 Flying measurement (FME)

15.2.1 Description

Summary

FME supplements a travel command (G0, G1, etc.). During the travel motion, the FME measures the workpiece position and is thus called "flying measurement". Instead of a workpiece, a touch probe is chucked. If the touch probe hits the workpiece while traveling, the position of the touch probe is immediately saved. Measuring is only possible for one single axis. The travel command is always completely accomplished.

The NC has two FME functions:

- Initializing flying measurement (InitMeas or IME) and
- flying measurement (FlyMeas or FME).

Both exclusively use the first touch probe.

Terms Used with FME

Touch probe:

- All sensors checking the workpiece position, e.g. mechanical touch probes, echo sensors, etc.
- Only knows two states: "workpiece" or "no workpiece"
- Converts these states into an electrical output signal

The measuring axis:

- can only be such a Sercos drive supporting the touch probe function
- is accordingly defined via a machine parameter
  (MP **EnablProbeFunc** "Touch probe function available") -
  and initialized via the IME block
- stores the measured value when receiving the touch probe edge
- is also supposed to be travel axis at the same (optional)

The measured value:

- is the value stored by the measuring axis when receiving the edge of a touch probe
- derives from the source defined before -
  via S-0-0426 "Signal selection, touch probe 1"
- knows only the value of the active encoder system as source

What happens in the IME block?

IME (InitMeas) **initializes FME** (FlyMeas)**:**

The measuring axis programmed in FME has to be initialized first using the "InitMeas" (IME) function.

What happens in the FME block?

FME (FlyMeas) **supplements a travel command:**

The travel command is accomplished.

Parallel to this command, FME starts the touch probe function of the programmed measuring axis.

If the measuring axis receives a touch probe edge, it immediately saves the detected measured value.

The travel command is completed when the programmed position is.

FME checks whether the touch probe triggered.

If yes:

The NC reads the measured value of the measuring axis and continues with the next program line.

If not:

The NC outputs a warning and waits until the measurement event occurs.

Why did the touch probe not trigger?

Possible reasons:

- The coordinates in the travel command are not correctly selected and the workpiece was not reached by the touch probe.
- The touch probe triggered but the measuring axis does not receive a touch probe edge; there is possibly a wiring error.

What happens after the FME block?

After the FME block, the user has to:

- Ensure that the measured value is applied to the part program.

For this purpose, appropriate CPL functions are available.

PLC Interface

The PLC is in the position to determine whether an FME block is active or not. For this purpose, specify a corresponding bit at the "NC function-specific bit interface". This bit is then active during the FME block.

Restrictions

- Flying measurement cannot be used together with the function "Quick stop via probe input".

  In case of an enabled drive function "Quick stop via probe input" the drive independently decelerates the measuring axis when the measurement event occurs. This behavior contradicts the function "Flying measurement".

Relevant NC functions

InitMeas (IME)	Initializing flying measurement
FlyMeas (FME)	Flying Measurement

Tab. 15-5: Relevant NC functions

Relevant CPL functions

PPOS(), PROBE()	Provides the measured value of the (synchronous) measuring axis (but in different forms).

Tab. 15-6: Relevant CPL functions

Relevant machine parameters (MP)

Probe1Edge	Path: MEAS / Dr[i] / Probe1Edge Name: "Edge of touch probe 1" (corresponds to Macoda-ID 1003 00011)
EnablProbeFunc	Path: MEAS / Dr[i] / EnablProbeFunc Name: "Touch probe function available" (corresponds to Macoda-ID 1003 00012)
NcFunc[1..24]	Path: PLC / NcFuncBitIf / NcFunc[1..24] Name: "NC function-specific bit interface" (corresponds to Macoda ID 3020 00001/ 3020 00002)

Tab. 15-7: Relevant machine parameters (MP)

15.2.2 Commissioning

Machine parameters (MP)

Determining the measuring axes:

- All system axes the user intends to use as measuring axes have to be specified as such via the machine parameter **EnablProbeFunc** "Touch probe function available", i.e.: the following has to be set in IndraWorks Engineering:

  MEAS / Dr[i] / EnablProbeFunc = "Only touch probe 1".
- If, however, in IndraWorks Engineering,

  MEAS / Dr[i] / EnablProbeFunc = "Both touch probes" is set,

  no error results, although the second touch probe cannot be used for the flying measurement.

Determining the touch probe edge:

- As soon as one system axis is defined as measuring axis, specify on which touch probe edge it reacts. This is accomplished via IndraWorks Engineering.
- Flying measurement always belongs to touch probe 1

  whose edge is selected via the machine parameter **Probe1Edge** "Edge of touch probe 1". There are only two options:

  either

  MEAS / Dr[i] / Probe1Edge = "positive edge"

  or

  MEAS / Dr[i] / Probe1Edge = "negative edge"

Determining the signal "FME active":

- For the FME function, the user can specify up to one "active" bit at the channel-specific PLC interface.
- It is specified in IndraWorks Engineering via:

  PLC / NcFuncBitIf / NcFunc[1..24]

The user writes "FME" into the respective path. The index number corresponds to the bit number, i.e. by entering e.g. "FME" under NcFunc[21], the signal "FME active" is assigned to the Bit21 of the NC function-specific interface.

- If two or more "active" bits are declared for the FME function, the NC only uses the bit with the smallest index. The others are not applied.

Sercos parameters

The touch probe function of the drive is specified by Sercos international. This means that the NC controls the touch probe function of the drive via Sercos parameters. For the flying measurement, the NC uses the following:

Sercos parameters:		Relevant for
IDN	Name	SCS file:
S-0-0130	Measured value 1, positive edge	Optional
S-0-0131	Measured value 1, negative edge	Optional
S-0-0169	Control parameter for touch probe	-
S-0-0170	Command for touch probe cycle	-
S-0-0405	Probe 1 enable	-
S-0-0409	Touch probe 1 positively recorded	Optional
S-0-0410	Touch probe 1 negatively recorded	Optional
S-0-0426	Signal selection probe 1	Mandatory

- No

Tab. 15-8: Sercos parameters for the flying measurement

Remark:

- If the drive has the Sercos parameters required for the flying measurement, it also supports the touch probe function.
- If not, the respective drive cancels the Sercos startup and refers to the missing Sercos IDN with a corresponding error message and thus to the missing touch probe function.
- Nevertheless, additional settings and checks at the drive can be necessary. The user is responsible for those settings and checks before operating the touch probe function of the NC. For detailed information, refer to the documentation of the drive manufacturer.

SCS file or Sercos file

The NC sets the following for the measuring axis using the SCS file:

- the mandatory Sercos parameter and
- the optional Sercos parameters (required by the user).

SCS files always have to be adjusted (either by the machine tool manufacturer or by the user). Thus, each editor is responsible.

Mandatory parameters in SCS file

- For part programs with flying measurement (FME),

  S-0-0426 "Signal selection, touch probe 1" has to be parameterized in the SCS file.
- Parameter S-0-0426 is set -

  either on:

S-0-0051 "Actual position value encoder 1"

or

S-0-0053 "Actual position value encoder 2".

Optional parameters in SCS file

If there are no optional parameters for flying measurement:

- for each IME block automatically:

  the real-time status bit 1 is automatically specified with

  "touch probe 1 recorded positively" (S-0-0409) or

  "Touch probe 1, negatively recorded" (S-0-0410) -

  depending on whether the machine parameter **Probe1Edge** specifies a positive or negative touch probe edge.
- for each FME block automatically:

  the measured value is read via the (Sercos) service channel, i.e. either

  "measured value 1 positive edge" (S-0-0130) or

  "measured value 1 negative edge" (S-0-0131) -

  depending on whether the machine parameter **Probe1Edge** specifies a positive or negative touch probe edge.

Thus, in this case, there is neither S-0-0409, S-0-0410, S-0-0130 nor S-0-0131 in the SCS file.

FME:

To read the measured value cyclically, since it is faster than via the service channel, set

S-0-0015 "Telegram type" and

S-0-1050.1.6 "Configuration list AT" as follows:

- **S-0-0015** to "Freely configurable telegram"
- **S-0-1050.1.6** adding of

  "measured value 1 positive edge" (**S-0-0130**) or

  "measured value 1 negative edge" (**S-0-0131**) -

  depending on whether the machine parameter **Probe1Edge** specifies a positive or negative touch probe edge.

In this case, the measured value is not read via the service channel.

FME:

If the real-time status bit 1 for flying measurement is to be used for other purposes, set

S-0-0015 "Telegram type"

S-0-1050.1.6 "Configuration list AT"

S-0-0026 "Configuration list, signal status word" and

S-0-0328 "Bit assignment list, signal status word" in the SCS file as follows:

- **S-0-0015** to "Freely configurable telegram"
- **S-0-1050.1.6** adding signal status word (**S-0-0144**)
- **S-0-0026** adding of

  "touch probe 1 recorded positively" (**S-0-0409**) or

"touch probe 1 recorded negatively" (**S-0-0410**) -

depending on whether the machine parameter **Probe1Edge** specifies a positive or negative touch probe edge.

- **S-0-0328** adding "0" at the same list position which incorporates S-0-0409 or S-0-0410 in the S-0-0026 list.

Irrelevant Parameters for SCS File

During the Sercos phase startup,

- the "Touch probe control parameter" (S-0-0169) is automatically generated and written to the drive from the machine parameters **EnablProbeFunc** and MP **Probe1Edge**
- the real-time control bit 1 is assigned or not assigned with "Touch probe1 enabled" (S-0-0405) in the drive using the machine parameter **EnablProbeFunc**
- the "Command touch probe cycle" (S-0-0170) is not created from the machine parameter **EnablProbeFunc** and written for drive.

Therefore, S-0-0169, S-0-0170 and S-0-0405 may never be in the SCS file.

15.2.3 Handling instruction: Flying Measurement

Note

The following example shows how to start the touch probe function "flying measurement".

The example is only a template. You have to adjust it to your requirements.

The commissioning of the touch probe function "flying measurement" is only an addition to the regular commissioning.

Task

The following system axes are given: 1, 2, 3, 4, 5 and 6.

- System axis 2 has to be defined as measuring axis for FME and react to the positive touch probe edge.
  - The internal encoder is used as source for the measured value.
  - The measured values have to be transferred cyclically.
  - No separate real-time status bit.
  - "FME active" is output for channel 1 at bit 22 of the NC function-specific PLC interface.

Drives

- Ensure that the drive, assigned to the system axis 2, supports the touch probe functionality. Use the drive documentation provided by the manufacturer.
- Ensure that the touch probe functionality is provided in the drive as requested by the drive manufacturer. For this purpose, additional activities can be necessary.
- Ensure that the signal for touch probe 1 is generated correctly and connected to the correct hardware input of drive 2.

Machine parameters (MP)

Set the machine parameters in the listed IndraWorks Engineering paths to the defined values.

Changes have to be saved.

MEAS / Dr[1] /

- EnablProbeFunc = "No"
- Probe1Edge = irrelevant

MEAS / Dr[2] /

- EnablProbeFunc = "Only touch probe 1"
- Probe1Edge = "positive edge"

MEAS / Dr[3] /

- EnablProbeFunc = "No"
- Probe1Edge = irrelevant

MEAS / Dr[4] /

- EnablProbeFunc = "No"
- Probe1Edge = irrelevant

MEAS / Dr[5] /

- EnablProbeFunc = "No"
- Probe1Edge = irrelevant

MEAS / Dr[6] /

- EnablProbeFunc = "No"
- Probe1Edge = irrelevant

For channel 1:

PLC / NcFuncBitIf /

- NcFunc[22] = "FME"

SCS files or Sercos files

Add the following to the SCS files of the drives 1 to 6 and save the modified SCS files.

SCS file drive 1:

- Ensure that there are no Sercos parameters in the SCS file which are relevant for the flying measurement. This means no parameters listed in tab. 15-8 "Sercos parameters for the flying measurement" on page 606.

SCS file drive 2:

- S-0-0426 = (51) ; S-0-0051 "Actual position value encoder 1"
- S-0-0015 = (0b111) ; binary 7: "Freely configurable telegram"
- S-0-1050.1.6 = (..., 130, ...) ; add S-0-0130 "measured value 1 positive edge"
- Ensure that the SCS file does not contain any of the parameters listed in tab. 15-8 "Sercos parameters for the flying measurement" on page 606 except for S-0-0426 and S-0-0130.

SCS file drive 3:

- As SCS file drive 1.

SCS file drive 4:

- As SCS file drive 1.

SCS file drive 5:

- As SCS file drive 1.

SCS file drive 6:

- As SCS file drive 1.

Final Activity

Restart control and drives.

During the Sercos startup, the control transfers the SCS files to the corresponding drives.

Activating

A measurement can only be started in the part program. For that purpose, first the NC function InitMeas (IME) and then FlyMeas (FME) has to be programmed.

The following example shows the determination of the workpiece position of the system axes 2 (in the part program named as Y) using the touch probe. The touch probe is configured and thus creates a positive edge when contacting the workpiece.

Program:

```
N10 G0 Y100 Z0               ; Approaching starting position for measurement
N20 IME(MpiAxis 2)           ; Initialize flying measurement for system axis 2
N20 F500                     ; Set feed for measurement
N30 G1 FME(MpiAxis 2) Y200 Z100 ; Approach position behind the workpiece
N40 WAIT                     ; Synchronization with NC preparation
110 IF SD(9)=0 THEN          ; if touch probe triggered
120   YPOS = PPOS("Y") ; Storing measured value of y-axis in YPOS
N130  MSG(CONTACT)           ; Output message
140 ELSE                     ; Touch probe not triggered
150 SETWARN("NO CONTACT!") ; Outputting warning
N160  M0                     ; Waiting on NC start
170   WAIT                   ; Synchronization with NC preparation
180   CLRWARN()              ; Delete warning
190 ENDIF
```

☞ In contrast to G75, the measuring motion is completely processed during flying measurement. Complete processing irrespective of whether the touch probe triggers (being in contact with the workpiece).

15.3 Measuring on a contour (MOC)

15.3.1 Description

Summary

Measuring on a contour is similar to the functions "Travel against touch probe (G75)" (see page 593) and "Flying Measurement (FME)" (see page 603). As the described function, the `MOC` also measures the workpiece position during the travel motion. In contrast to `G75` and `FME`, any path can be used as measuring distance while measuring on a contour.

Measuring on a contour is a modal NC function activated at the beginning of the measuring distance (an `MOC` with parameters) and deactivated at the end of the measuring distance (a `MOC` with empty parentheses or the next `MOC`). Any travel commands can be programmed while this function is active. However, other measuring functions are not allowed.

The trigger used for the measurement can be a touch probe, a customer input in the channel interface or a high-speed input. If a touch probe is used, it is engaged instead of the tool. If the touch probe hits the workpiece while traveling, the position of the touch probe is immediately stored.

Measuring is only possible with synchronous axes of one channel.

MOC-Specific Terms

Touch probe:

- All sensors checking the workpiece position, e.g. mechanical touch probes, echo sensors, etc.
- Only knows two states: "workpiece" or "no workpiece"
- Converts these states into an electrical output signal

Measuring signal source:

- Can be a touch probe

  - or -
- a customer input in the channel interface (`qCh_Custom1..8`)

  - or -
- a high-speed input

Measurement event:

- Is a point in time at which the measuring signal source experiences a change in level (edge) and this edge was programmed as trigger in `MOC`.
- Starts determining the measured values.
- Triggers different additional reactions of `MOC` (see chapter "Potential reactions to the measurement event" on page 611)
- Is registered only once along the measuring distance. `MOC` no longer monitors the signal source after the first measurement event.

Measuring axis:

- Measuring axes can only be synchronous axes of the channel.
- Measuring axes are specified in the `MOC` block (unless programmed, all configured channel axes are measured).
- Measuring axes may not be removed along the measuring distance from the channel.
- Measuring axes have to meet the following additional requirements if a touch probe is used as signal source:
  - They can only be such Sercos drives that support the touch probe function.
  - They have to be enabled via the machine parameter "Touch probe function available" (`MEAS/Dr[i]/EnablProbeFunc`).

The measured values:

- are always available for all channel axes if the customer interface or the high-speed input is used.
- are only available if the touch probe is used for the measuring axes.
- are stored in a system date.
- are converted into program coordinates (PCS) when all channel axes were measured.
- Can also be queried in the part program via CPL functions (`SD(9)`, `PROBE()`, ...).

Potential reactions to the measurement event

When a measurement event occurs, NC function `MOC` can execute an additional reaction (apart from storing the measured values). This reaction is programmed as a numerical code for the `MOC` command at the start of the measuring distance.

1. **Velocity change**

   In general, the NC travels with the programmed feed (F-value). In G94 mode, a second feed can be programmed (F2-value) which is used instead if a PLC bit (qCh_Feedrate2) is set.

   If reaction 1 (velocity change) is programmed in MOC, the NC uses the second feed to travel at the start of the measuring distance. The NC does this irrespective of whether the PLC bit is active. The normal logic is again active after the measurement event. Thus, the PLC bit decides again which feed is used.

2. **Execution of an asynchronous subroutine**

   As reaction 2, MOC can execute an asynchronous subroutine (refer to the manual "MTX 14VRS Functional Description Basics", chapter "Asynchronous Subroutines") in case of an measuring event. The subroutine has to be logged into the system (e.g., via ASPSET) before MOC is called.

   If an asynchronous subroutine is started in "buffered NC block specification" mode, all blocks are lost after completion of the subroutine. If the complete measuring distance was transferred to the NC in a message, the measuring distance is therefore only be traveled to the point where the measurement event occurred.

3. **Reset**

   The third reaction by MOC triggers a channel reset when the measurement event occurs.



Since the part program is aborted at a channel reset, the measurement event is no longer followed by end of the measuring distance (MOC()) and measuring on a contour remains active. That is why we recommend to write MOC() to the "state after channel reset" (machine parameter CHAN/Ch[k]/Ini/ChResetState).

4. **Flying measurement**

   If reaction 4 is programmed, MOC does not carry out any other actions apart from measuring. In other words, the measuring distance is traveled completely (without any intermediate stop).

5. **Skipping the remaining measuring path**

   If reaction 5 is programmed, MOC can skip the measuring distance after the measurement event has occurred. In this case, the NC stops on the path when the measurement event occurs. Thereafter, the remaining blocks are automatically skipped without any motion. However, all CPL and auxiliary functions of these blocks are processed.

   If the end block of the measuring distance (MOC()) contains a motion, this motion is carried out subsequently. If this block does not contain any programmed motion, measuring on a contour causes a traveling to the end point of the measuring distance using the last active motion function.

☞ If reaction 5 is programmed, the NC function in the end block of the measuring distance causes the machine to travel to the last position of the measuring distance even if a motion was not programmed.

We recommend to program a safe absolute position in the end block via a straight line (`G90 G0`). It has to be ensured that this position can be reached from any point of the contour without collision.

☞ We recommend to incorporate the end of the measuring path (`MOC()`) into the "state after channel reset" (machine parameter `CHAN/Ch[k]/Ini/ChResetState`) because, otherwise, runtime errors may occur during the reset if an error occurred while blocks were being skipped.

If reaction 5 is selected, it is not allowed to switch transformation and/or placements along the measuring distance or to change the active zero point offsets.

Measurement event not occurred

If the measurement event did not occur along the complete measuring distance, the reaction of the measuring on a contour function may vary depending on what is programmed.

0 There is no reaction. (specification)
1 A warning is output informing the user that the measurement event did not occur.
2 A runtime error is initiated.

Determining the measured values via system date

The measured values are written to system date "`/SysMeasPos`" (unit: millimeters) whenever measurement takes place. Users can only use `MOC` if they created the system date before (also refer to the MTX Functional Description - Special Functions, chapter System data). For a template of the SD structure, please refer to "/feprom/schemas/sdmoc.xsd_". The system date is to be defined as field with the size of the channel (see "/feprom/SDDefMoc.xml_").

/SysMeasPos[Channel]/ ACSPos[1..8]	double	Measured positions of the channel axes in mm
/SysMeasPos[Channel]/ PCSPos[1..8]	double	Measured positions of the channel axes in mm, converted into program coordinates
/SysMeasPos[Channel]/Version	uint	Consecutive version number of the measurement
/SysMeasPos[Channel]/Valid	uint	Is 1 if the measured values are valid

Tab. 15-9: System date structure for measuring on a contour

The measured values are entered in fields ACSPos and PCSPos under the coordinate number. This means that the measured position of the first coordinate can be found in the channel under ACSPos[1].

Axis positions that were not measured are set to 1000000 km. The PCS position is only calculated if all axes of the channel were measured. Corrections are based on the current actual positions.

The version number can be used to verify whether a measurement has been triggered. In this case, the version number is incremented by 1.

An external application can subscribe to the modification event of this system date (OPC item `NC.SystemDataEvent`, (also refer to the "MTX OPC Communication") and can therefore be called once the measurement is complete.

Determining the measured values via CPL functions

CPL function `SD(9)` can be used to verify in the part program whether or not the measurement event occurred. Functions `PPOS()`, `PROBE()` and `PCSPROBE()` provide the measured value of an axis.



In general, a `WAIT` command has to be programmed for the synchronization between the end of the measuring distance and the first CPL function which refers to the measurement.

Special features and restrictions

When selecting the measurement signal source "Probe (MT)" and enabled drive function "Quick stop via probe input" automatically decelerates the measuring axes when the measurement event occurs (chapter 15.3.6 "MOC with Quick Stop via Probe Input" on page 618).

All MOC reactions that do not require an axis standstill can be used together with "Quick stop via probe input". The MTX reports a runtime error once the measurement event occurs.

Reactions:

- 1 Change velocity
- 4 Flying measurement

Relevant NC functions

MPEDGE	Reconfigures the measuring edge when the touch probe is used
MOC	Measuring on a contour

Tab. 15-10: Relevant NC functions

Relevant CPL functions

SD(9)	Determines whether the measuring event occurred.
PPOS(), PROBE(), PCSPROBE()	Provides the measured value of the (synchronous) measuring axis (but in different forms).

Tab. 15-11: Relevant CPL functions

Relevant machine parameters (MP)

EnablProbeFunc	Path: MEAS / Dr[i] / EnablProbeFunc Name: "Touch probe function available" (corresponds to Macoda-ID 1003 00012)
Probe1Edge	Path: MEAS / Dr[i] / Probe1Edge Name: "Edge of touch probe 1" (corresponds to Macoda-ID 1003 00011)
Probe2Edge	Path: MEAS / Dr[i] / Probe2Edge Name: "Edge of touch probe 2" (corresponds to Macoda-ID 1003 00013)

Tab. 15-12: Relevant machine parameters (MP)

15.3.2 Applying

The "`/SysMeasPos`" system date has to be created with the corresponding structure (refer to the MTX Function Description - Special Functions, chapter “System data”). For structure and definition templates, please refer to "/feprom/schemas/sdmoc.xsd_" and "/feprom/SDDefMoc.xml_".

If a touch probe is to be used as signal source, all measuring axes have to be configured accordingly. This is performed as in the "Travel against touch probe (G75) ", see page 593 and "Flying Measurement (FME)", see page 603. The touch probe should be connected to the touch probe input of all measuring axes.

If a customer input in the channel interface is to be the signal source, this input has to be provided with the appropriate measuring edges by the PLC program.

If a high-speed input is to be the signal source, this input has to be wired appropriately.



The end of the measuring distance (`MOC()`) should be written to the "state after channel reset" (machine parameter `CHAN/Ch[k]/Ini/ChResetState`) in order to ensure that a reset stops the measurement if any is in progress.

15.3.3 Activation

At the start of the measuring distance, an `MOC` with parameters is programmed in the part program. The measurement starts at this block. If the block itself contains a motion, the start is "flying" (i.e., without stop).

Example

```
N100 MOC(MT1, ES1, RE2, AS4, ER1, X, Z)
                ; The positions of axes X and Z are to be measured if touch probe 1 registers a falling edge. In this case, the asynchronous subroutine 4 is to be additionally executed. If the touch probe does not register any falling edge along the measuring distance, a warning is to be output.
```

If the `MOC` block contains a motion, the start is "flying" (without stop). Once the motion is started, the measurement is active.

15.3.4 Deactivation

A block containing an MOC without parameters (with empty parentheses) represents the end of the measuring distance. This is the latest point where the measuring logic is deactivated. Depending on the programmed error reaction, a warning or an error message is generated in this block if the measurement event did not occur.

Example

```
N500 MOC()                    ; End of the measuring distance
```

If the MOC block contains a motion, the end is "flying" (without stop). Once the motion is started, the measurement is stopped.

A measuring distance can also be completed by a subsequent measuring distance. In this case, an MOC with parameters represents both the end of the old measuring distance and the start of the new one.

Example

```
N100 MOC(CI6, ES3, RE1)       ; Measurement 1 is started
...                           ; Measurement 1 is running
N200 MOC(HS2, ES2, RE4,       ; Measurement 1 is stopped, measurement 2 is
ER2)                          started
...                           ; Measurement 2 is running
N300 MOC()                    ; Measurement 2 is stopped
```

15.3.5 Handling instruction: Measuring on a contour

The following examples show how programming should be done in the part program. The examples are based on the assumption that all measuring axes are correctly configured and the system date is created.

The first example shows the velocity change. The PLC bit for the second feed is to be continuously set to zero.

Example velocity change:

```
N10 G01 F100 G94              ; Sets the programmed velocity F to 100 mm/s
N20 F2=500                    ; Sets the second velocity F2 to 500 mm/s
N20 MOC(MP1, ES1, RE1)        ; Starting at this block, F2 = 500 is used for travel-
                              ing; as soon as touch probe 1 registers a falling
                              edge, F = 100 is used
N30 X… Y… Z…                  ; F2 = 500 is used for traveling if an edge was not
                              received from MP1 yet, or F = 100 if an edge was
                              registered
N40 X… Y… Z… F=70             ; F2 = 500 is used for traveling if an edge was not
                              received from MP1 yet, or F = 70 if an edge was
                              registered
N50 X… Y… Z… F2=400           ; F2 = 400 is used for traveling if an edge was not
                              received from MP1 yet, or F = 70 if an edge was
                              registered
N60 MOC()                     ; Starting at this point, F = 70 is used for traveling
```

The second example shows the "Execute asynchronous subroutine" reaction. This variant can, e.g., be used for crash monitoring. Based on the measurement position, a departure strategy can be determined in the subroutine.

Example of an asynchronous subroutine

N10 ASPSET(3, SAFEPOS01, 11)	; Log in asynchronous subroutine "SAFEPOS01" to the system as number 3.
N20 MOC(MP2, ES2, RE2, AS3)	; If touch probe 2 registers a rising edge, asynchronous subroutine 3 (SAFEPOS01) is executed
N30 X… Y… Z…	; Measurement is in progress
N40 X… Y… Z…	; Measurement is in progress
N50 X… Y… Z…	; Measurement is in progress
N60 X… Y… Z…	; Measurement is in progress
N70 MOC()	; Measurement is completed (the touch probe is deactivated)

The following example illustrates "flying" measurement and shows how the measured values are read out with CPL.

Example of flying measurement

N10 G1 F1000 X... Y... Z...	; Move to start position
N20 MOC(MP2, RE4, X, Z) X... Z...	; If touch probe 2 provides the preconfigured edge, measurement takes place; the start is "flying"
N30 X… Y… Z…	; Measurement is in progress
N40 X… Y… Z…	; Measurement is in progress
N50 X… Y… Z…	; Measurement is in progress → the touch probe might trigger at this point • The measured values for X and Z are latched • The latched values are written to the system date • The motion is not interrupted
N60 X… Y… Z…	; Measurement is no longer in progress
N70 MOC() X… Y… Z…	; Measurement is stopped "on-the-fly"
N80 WAIT	; Synchronization
90 XPOS = PPOS(1)	; Stores the measured value of the X-axis
100 ZPOS = PPOS(3)	; Stores the measured value of the Z-axis

The following example illustrates reaction 5 (skip remaining measuring distance).

Example of skipping the remaining measuring distance

N10 G1 F1000	; Initialization
N20 MOC(HS5, ES3, RE5)	; If high-speed input 5 provides any edge, the motion is stopped; this is followed by block N70
N30 X… Y… Z…	; Measurement is in progress
N40 X… Y… Z…	; Measurement is in progress

N50 X… Y… Z…	; Measurement is in progress → the touch probe might trigger at this point • There is a stop on the path (Feed hold) and the measured values are latched • The latched values are written to the system date • The axis travels along a straight line to point X10 Y20 Z30 in rapid traverse mode according to the specification in the end block of the measuring distance
N60 X… Y… Z…	; This motion is not carried out and the block is skipped
N70 MOC() G90 G0 X10 Y20 Z30	; Measurement is stopped, and a safe point is approached
N80 X… Y… Z…	

15.3.6 MOC with "Quick Stop via Probe Input"

Function

In some cases a particularly quick axis stop after probing a workpiece is required by a probe. The function "Quick stop via probe input" can be enabled in IndraDrive for touch probe 1.

Restrictions

All MOC reactions that do not require an axis standstill can be used together with "Quick stop via probe input"(chapter "Special features and restrictions" on page 614).

Reaction to measuring result

The MTX checks if a quick stop can be executed for some or all axes of the channel once measurement event occurs.

If one or several axes can be stopped with a quick stop, the MTX reacts as follows:

1. Deceleration of path motion. The remaining axes not stopped by a quick stop are decelerated
2. Waiting for standstill of all axes and applying the actual position of the "quick stop" axes
3. Continuing machining

If all axes are stopped with a quick stop, the MTX reacts as follows:

1. Canceling the path motion
2. Waiting for standstill of all axes and applying the actual position of the "quick stop" axes
3. Continuing machining

Applying

Prerequisite: Probe 1 has been commissioned.

Via P-0-0226 "Probe, extended control word", the function is enabled and configured.

The functional scope depends on the used IndraDrive firmware.

The description of the used firmware always applies!

S-0-0124 "Standstill window" has to be set to a valid value.

15.4 Digitizing (recording measured values)

15.4.1 Description

Function

The "Digitizing" function provides for the technical prerequisites for taking up a surface profile and for storing logged axis positions:

- Logging of the workpiece in x-y-plane and z-direction with **automatic level adjustment** by simultaneous **online correction** of the z-axis:

  To log a specified surface (e.g. freeform surfaces), a point grid with the maximum possible density must be taken up from the surface. A suitable sensor (e.g. laser) travels along the workpiece in a meandering fashion. Meanwhile, the sensor feed axis Z is constantly adjusted "online" to the distance line of the surface profile ("axis distance control"; see also under "digitizing procedure").

- **Logging axis positions in real-time** via Sercos:

  By recording the axis positions X, Y and the axis position Z corrected in-real-time, the distance lines are digitized by lining points.

  When the 5-axis transformation is active, recording of the surface profile can also be performed together with logging of the axis positions of five axes.

- **Storing real-time data** (axis data and Z correction values) in a recording file:

  Storing on the hard disk of the PC panel or of an external PC via Ethernet network.

- The digitizing **process can be controlled** via the NC program:

  Via NC command, logging of axis positions can be started, interrupted, continued and terminated in the program.

- Influencing of digitizing **via programmable default settings**.

- **Synchronization of an external device** via recording of the measured values within the NC. To this end, one of the high-speed digital outputs of the control can be used.

- Additional information (**scaling functions**) for subsequent processing of the real-time data is provided in the log file.

- Programming of the laser path on which the workpiece is "keyed" is **not** part of the "digitizing" function. From a programming point of view, this means that the recorded data has to be suitable for subsequent processing with a CAD system.

Application options:

- **Takeover of the data** into a suitable CAD/CAM system (visualize and modify the workpiece, realize design ideas).
- **Generation of an NC program** or "direct processing" **from the log data** (copying of the part on an identical or different machine).

Hardware components required for digitizing:

- Sercos drive with additional input for an external measuring system.
- Internal measuring systems (drives) for taking up the real-time positions
- external measuring system (e.g. laser) for online correction
- Ethernet network (TCP/IP) for data transfer to the PC
- NC with a free RAM memory of maximum possible size

- PC panel for operation
- Free hard disk memory on PC panel to save real-time data if there is no external PC (via NFS)
- ext. PC with NC network to store real-time data on the hard disk.

Example:

Structure and Connection of the NC and Drive Component for Digitizing on a 3-Axis Machine.

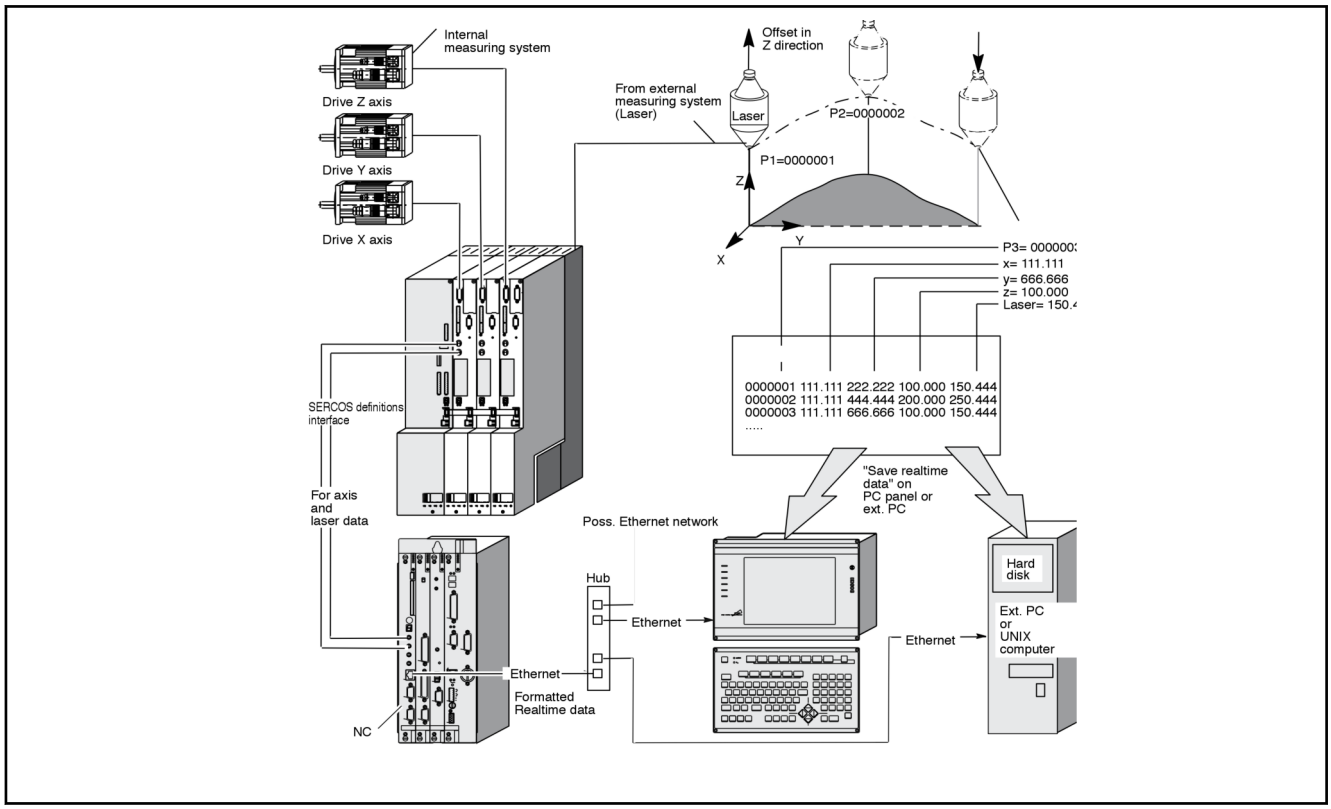


Fig. 15-1: Structure and connection of the NC and drive component for digitizing

Digitizing process:

- The digitizing process has to be initiated in the NC program. The digitizing process itself is program-controlled, i.e. the process can be started, interrupted, continued and stopped in the program.
- For digitizing, you need a sensor (e.g. laser) which keys the workpiece contact free on a meandering path (see the fig. below). The laser has a measuring system of its own which is used to keep a preset distance (offset) to the surface **constant** during keying, i.e. the signals of the laser measuring system are offset in real-time with the internal measuring system of the laser infeed axis (e.g. z-axis). In this way, the z-axis of the distance line of the surface can be tracked in IPO cycle (**axis distance control**).



Extended options are available in connection with the (laser) infeed axis (e.g. collision detection, hole detection) by using the "Axis distance control for digitizing" function (see chapter 15.6 "Axis distance control for digitizing " on page 635).

The "External measuring system" (laser) is applied at the drive module (see figure above).

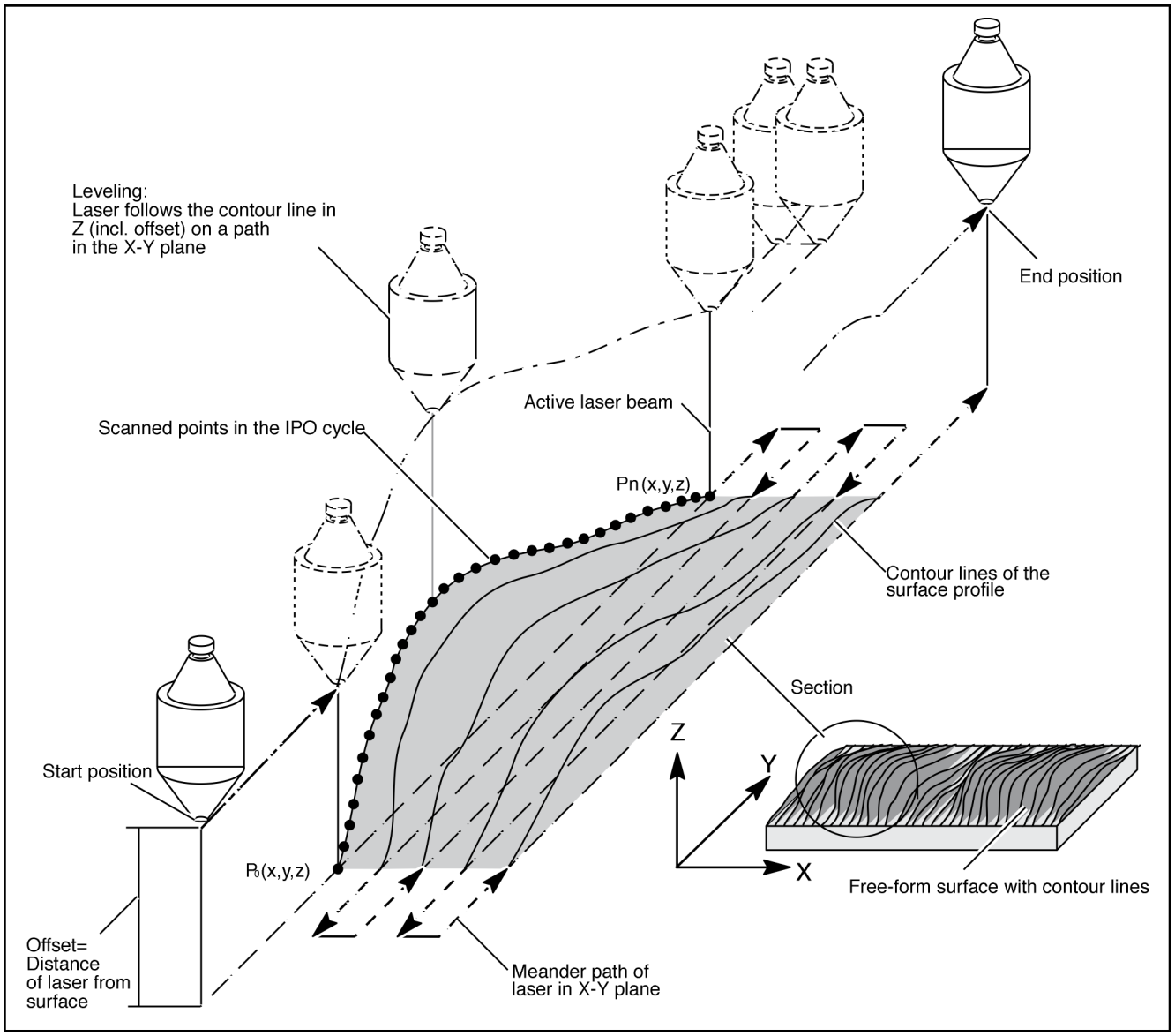


Fig. 15-2: External measuring system

- During keying, all required axis positions (e.g. X, Y, and z-axis) are logged in the interpolator cycle (or its multiple) and stored to the hard disk of a PC panel or an external PC.
- A so-called digitalization buffer executes the saving action. The real-time position data coming from the drive are written in portions into the free RAM memory sector of the NC. Already "filled" buffers are stored in the output file (see RECFILE) to the hard disk of a connected PC panel or an external PC (via network). The duration of the storing process depends on the network load or system load of the control at the given moment.

  During this time, several buffers can be filled by the "real-time part" (data received via Sercos) (see the following figure "already filled buffers").

Example:

Buffering of incoming and outgoing real-time data

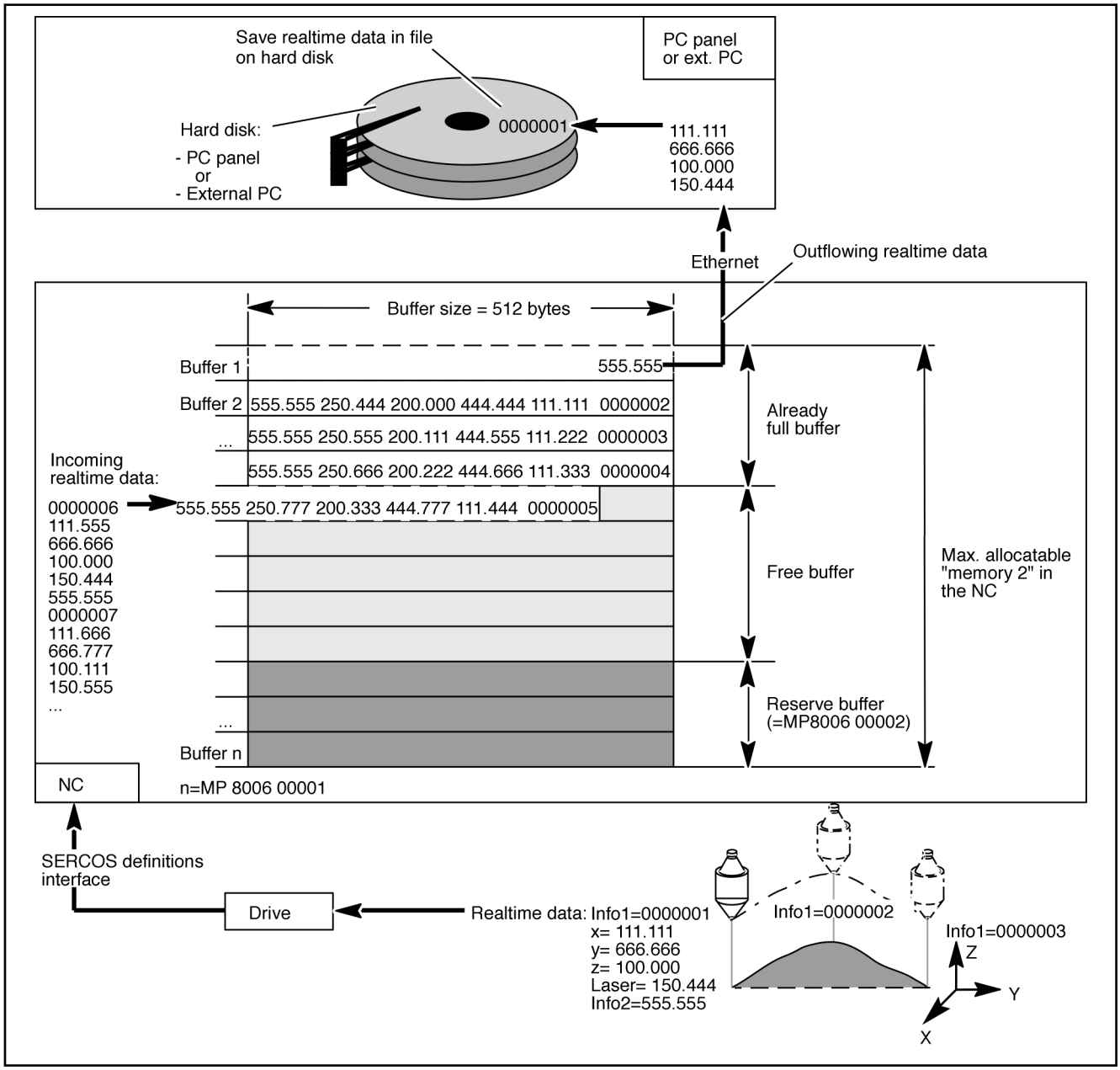


Fig. 15-3: Example - Buffering of incoming and running off real-time data

With high network load, the data is only later at the hard disk. Then, a data jam occurs in the digitizing buffer, i.e. more data is coming in than can be stored.

If so many buffers are needed that the number of free buffers falls below the "number of reserve buffers", the control reacts according to the parameterization in the command

RECFILE(, ,[<File-I/O>]) and RECTIME(, [<AutoStop>],)] of the RecordSet:

- – The control sets the channel to zero to interrupt reading in of the real-time data. Filled buffers can be saved first.

  (see RECFILE(, ,<File-I/O=1 or 2>)

– With each standstill of the channel axes, reading in of real-time data is interrupted automatically. Thus, already filled buffers can be stored (see RECTIME(,<AutoStop=1>).
– If the "number of reserve buffers" is not observed and the data cannot be stored in time, the digitizing buffer "overflows".

  Remedy:

  Increase the number of digitizing buffers in MP 8006 00001 (see RECFILE(, ,<File-I/O=0>) or change the RECTIME parameter in the NC function "RECORDSet".

Restrictions

- In the NC, the "digitizing" function may only be active in the system once at one given time. Multiple starting (e.g. in different channels) is not permitted.
- Incorrect statements (e.g. the specified ident number is not configured in the cyclic axis telegram) or problems with file processing (e.g. insufficient memory space on the hard disk) result in a runtime error.
- If the desired log file already exists, it is overwritten by a new digitizing (depending on parameterization).
- Axis numbers can be selected within the entire system. There are no restrictions regarding the axes of the channel in which the digitizing program has been started.
- Active digitizing is terminated at the latest at the end of program. The log file is closed.
- With the RecordBreak instruction, digitizing is stopped, i.e. the control interrupts read-in of real-time data. The opened log file remains open.
- When target feed of the channel and the actual feed of all axes of this channel equals 0, the generation of redundant data is avoided (see RECTIME(..,[AutoStop] parameter).
- If storing is not possible due to excessive network load, the channel feed is set to zero (see above) and the log data is interrupted at standstill. Then, buffers already filled with real-time data can be stored despite any potential network loads (see RECFILE(, ,[File-I/O]).
- Up to 8 "measuring points" (axis numbers, axis positions,..) can be defined (see RECPROBE1... RECPROBE8).
- The recorded values can be weighed with a factor for further data processing (e.g. for a CAD system).
- Formatted output of the recorded values is possible.
- Up to 8 additional "measuring points" (e.g. offsets,..) can be defined (RECPROBE101 ...RECPROBE108) which can be arithmetically (+/-) connected to RECPROBE1 ... RECPROBE8.

  These measuring points are not stored in the file.
- In case of missing inversion range compensation in the drive (e.g. by third party manufacturers) incorrect actual position values are output.

Initialize "Digitizing": The "Digitizing" function is initialized via the NC functions "RecordSetProbe" and "RecordSet" and stands at the start of the NC program which contains the traversing motions for recording of the tool surface.

One or several NC blocks are programmed with RecordSetProbe to set the measuring points, followed by an NC block programmed with RecordSet to set the saving operation.

```
N10 RecordSetProbe RECPROBE1(..)
N20 RecordSetProbe RECPROBE2(..)
N30 RecodSet RECFILE(..) RECTIME(..)
```

*NC function **"RecordSetProbe"***

- By means of RecordSetProbe, measuring points (position data) are parameterized **before** the RecordSet command. RecordSetProbe can be programmed repeatedly. Already parameterized RECPROBEx cannot be overwritten.
- All parameters are applied using the RecordSet command. Thus, digitizing is initialized and can be started.

RecordSetProbe `RECPROBEx(..)`

Tab. 15-13: NC function RecordSetProbe

Parameters of RecordSetProbe

`RECPROBEx(`<Number>, <Name>, {<Factor>}, {<Format>}, {<±RECPROBEy>}`)`		
with		
x: 1 .. 8 and 101 .. 108		
<Number>	0:	For measuring points not derived from axes
	1..64	Axis number or indication of the physical axis name
<Name>	With the <Name> parameter, the position data of the axis denominated under <Number> is transferred There are the following options:	
	S-0-xxxx	Standard Sercos parameter
	P-0-xxxx	Product-specific Sercos parameter
	I-ACTPOS	Actual drive position - LSEC - CCOMP - temp. comp. in the unit 0.0001 mm
	I-HSINPUTx	With x: 1..8 High-speed input of the NC
	I-CMDPOS	Axis command position in the unit 0.0001 mm
	@aaaaa	A permanent CPL variable
<Factor>	**Optional** The unit of the incoming position data is converted with a conversion factor into another unit (e.g. 1 mm = 0.001 mm). In the new unit, the position data is stored to the log file. The result is rounded.	

<Format>	**Optional, default: "%d"**, format of the incoming position data in one of the following options:	
	%d	4 byte integer also %<Length>d. Example: %6d
	%f	4 byte float also %<Length>.<Decimal positions>f. Example: %10.4f
	%e	4-byte float (exponential representation)
<±RECPROBEy>	**Optional** RECPROBEy (y=101, 102, ... ,108). The value of the position is compensated by "+" or "-" RECPROBEy, and the result is stored in the logging file (e.g. offset of the laser to the surface). Correction is indicated via the RECPROBE number "y" (e.g. RECPROBE1(1, I-ACTPOS, , , +101) adds RECPROBE101 to Probe1). The corrected value is stored in the log file.	

Tab. 15-14: Parameter of the NC function RecordSetProbe

- A Sercos parameter specified as measuring point has to be specified in the cyclic drive telegram.
- RECPROBE101 ... RECPROBE108 are not stored in the log file. The measuring points (e.g. offset) can only be used for the offset with RECPROBE1 to RECPROBE8.
- In case of simultaneous recording of I-ACTPOS and I-CMDPOS it has to be taken into consideration that the measured values refer to different points in time.

NC function **"RecordSet"**

Using RecordSet, the saving operation and the NC behavior during digitalizing is specified.

- All RecordSet parameters have to be written in one line.
- After RecordSet, digitizing is initialized and can be started.

RecordSet `RECFILE(..) RECTIME(..) RECSEPARATOR(..)`

Tab. 15-15: NC function RecordSet

RecordSet parameters

`RECFILE(`<Name>, {<Rewrite>}, {<File-I/O>}`)`	
<Name>	**Default: /mnt/<FileName>** Name of the log file incl. path (max. 100 characters)

{<Rewrite>}	**Optional, Default: 0**, Options: "Save file" under the given <Name> overwrites (=1) the existing file of the same name, or is refused (=0)	
{<File-I/O>}	**Optional, Default: 1** If storing is impossible because of network load, the channel feed is set to zero. When the channel axes are at standstill, read-in of the real-time data is interrupted. Thus, already filled buffers can be stored. The "number of reserve buffers" has to be sufficiently large to buffer all data obtained during the braking process. Options:	
	0:	The feed is not set to zero
	1	After storing of already filled buffers, the process is automatically continued
	2	After storing of the filled buffers, the process is only continued after NC start
{RECTIME(<Time>, {<AutoStop>}, {<Poti>})}		
	RECTIME is **optional**. If no RECTIME is programmed, <Time> is set to the current NC cycle time, [<AutoStop>] and [<Poti>] are set to zero	
<Time>	0.5 to 10000.0 ms. Indication of the time cycle in which the position data are to be read (e.g. in the interpolator cycle or its integer multiple. Value is automatically rounded to an integer multiple of the interpolator cycle	
{<AutoStop>}	**Optional, Default: 0** Options:	
	0	The saving process is not interrupted
	1	Digitizing interrupts storing of the data as long as the axes of the channel which has started digitizing are at standstill
{<Poti>}	**Optional, Default: 0** Options:	
	0	The value of the channel potentiometer has no effect
	1	Depending on the value of the channel potentiometer, the scanning frequency of digitizing is modified (e.g. potentiometer 50%: doubling of scanning time)
{RECSEPARATOR(<Char>)}		
	RECSEPARATOR is **optional**. If RECSEPARATOR is not programmed, a tab stop is used as separator	

<Char>	Delimiter between the logged position data, in case the data is to be **exported** later (e.g. into a Microsoft Excel spreadsheet). Attention: Always indicate the delimiter as a CPL string in the form of `RECSEPARATOR(["," ])`! Recommended characters: Space: "" Tab: "\t" Comma: "," Semicolon: ";" Colon: ":"
	If "Point" is used as a delimiter, incorrect interpretation of data may occur as the point is likewise used in the floating decimal point format (e.g. 10.102)

Tab. 15-16: Parameter definition of the NC function RecordSet

Control "Digitizing": The "RecordOn", "RecordOff", "RecordBreak", "RecordContinue" NC functions affect storing of real-time data during automatic processing of the NC program.

NC function "RecordOn"

RecordOn	Start "digitizing". Log file is opened. RecordOn starts recording and storing of the measured values (position data). The required parameters have to have been set previously with RecordSet. If the "digitizing" function is already active, further RecordSet instructions cause a runtime error.

Tab. 15-17: NC function RecordOn

NC function "RecordOff"

RecordOff	Finish "digitizing". RecordOff terminates recording of the measured values and closes the log file. After the RecordOff command, digitizing can be restarted with RecordOn, or reinitialized with RecordSet(...). Other functions for terminating digitizing: • Program end • Channel or system reset.

Tab. 15-18: NC function RecordOff

NC function "RecordBreak"

RecordBreak	Interrupt "digitizing". RecordBreak interrupts digitizing. Recording and saving of the measured values is stopped, but the log file remains open. This does not affect the feed so that processing of the NC programs continues. If "End of program" (e.g. M30), reset, or RecordOff is detected while recording is interrupted, the log file is closed.

Tab. 15-19: NC function RecordBreak

NC function "RecordContinue"

RecordContinue {RECTIME(<Time>, {<AutoStop>}, {<Potentiometer>})}	Continue "digitizing". RecordContinue continues digitizing, i.e. recording and storing of the measured values. When RecordContinue is programmed, a scanning time can be entered. This allows for a modification of the time specified in RECTIME. RECTIME parameter, see NC function "RecordSet".

Tab. 15-20: NC function RecordContinue

Additionally, digitizing can be influenced by the following settings:

- Automatic stop of "storing of real-time data" if no traversing motion exists:
  → This is set in RecordSet(..., RECTIME(<Autostop=**1**>),...)
- Close log file automatically at end of program:
  → Programming M30 in the NC program.

Relevant machine parameters (MP)

8006 00001	Number of digitizing buffers (buffer size 512 bytes). The maximum number of digitizing buffers follows from the formula: Max. number of buffers = "Size of free memory space 2" / 512.
8006 00002	Number of reserve buffers. Recommendation: Approx. 30% of MP 8006 00001. In case of data jam, the number of reserve buffers can be increased, and/or a smaller feed may be used if applicable.
4075 00102	Assignment of the digital outputs.

Tab. 15-21: Relevant machine parameters

15.4.2 Handling instruction: Digitizing (recording measured values)

Applying: IW Engineering/configuration: Editing parameters

- The "digitizing" function has to be activated via the parameter **dig** "digitizing" in the setup (SUP).
- In the parameter **NofDigBuff** "number of digitizing buffers" (8006 00001), the number of digitizing buffers (buffer size 512 byte) has to be entered.
  If the difference between LimitUsedBuffer and MaxUsedBuffer is very small in the recording file in the "[Statistics]" section, the value of **NofDigBuff** should be increased.

- In the parameter **NofReservBuff** "number of reserve buffers" (8006 00002), determine the number of reserve buffers.

  Recommendation: Approx. 30% of the parameter **NofDigBuff** "number of digitizing buffers" (8006 00001). In case of data jam, the number of reserve buffers can be increased, and/or a smaller feed may be used if applicable.
- In the parameter **DigIOAllocOut** “Allocation of the digital outputs” (4075 00102) set the value "digitizing"; in this way, the digital output can be used for the synchronization of the recording with an external device.

	Documentation	Instruction
Instruction:		Editing machine parameters
Documentation:	MTX Machine Parameters	Digitizing

IW Engineering/Sercos: Adapting SCS file to phase 2

With an accordingly configured axis telegram (Sercos), the actual axis value as well as the actual sensor value are available in the NC. To this end, modify the Sercos file by taking up the Sercos parameters for the actual sensor value of the connected external measuring system into the sector for the cyclic drive telegram.

	Documentation	Instruction
Instruction:		Editing SCS parameters
Documentation:	Rexroth IndraDrive Parameter Description	Digitizing

Activating:

Prerequisite:

Regarding machine configuration, RecordSet is set.

Creating the NC program:

1. Choosing a name for the NC program.
2. Insert the "parameterized RecordSet" at the start of the NC program.
3. Program the path.
4. By programming of the "RecordOn" command, specify the time from which position data is to be recorded.
5. Program the "RecordBreak" command to temporarily interrupt digitizing (e.g. for traveling outside of the workpiece surface).

   Program "RecordContinue" to continue digitization.
6. Program the end of digitizing with "RecordOff" or with "End of program M30".
7. **Processing:**

   Starting of the digitizing process with recording and storing of the position data.

☞ If the motion of the laser stalls during automatic processing, this may be due to a bottleneck in the transmission of the real-time data via the network.

For details on the cause of the malfunction, see the data from the "[Statistics]" area in the log file.

Frequently, problems like this can be solved by modification of the settings in "RecordSet".

☞ If the difference between LimitUsedBuffer and MaxUsedBuffer is very small in the recording file in the "[Statistics]" section, the value of the machine parameter **NofDigBuff** "Number of digitizing buffers"should be increased.

Example:

1 - Programming actual position with internal measuring point "I-ACTPOS" of the NC and (S-0-0053)

```
RecordSetProbe RECPROBE1(1,I-ACTPOS,0.0001,%9.3f)
RecordSetProbe RECPROBE2(2,I-ACTPOS,0.0001,%9.3f)
RecordSetProbe RECPROBE3(3,I-ACTPOS,0.0001,%9.3f)
RecordSetProbe RECPROBE4(3,S-0-53,0.0001,%9.3f)
RecordSet    RECFILE(/mnt/digit/beispiel.dig,1,0)    REC-
TIME(8,1,0)
```

The parameters have the following meaning:

- The real-time data is stored in the log file "example.dig". It is stored in the path "/mnt/digit".

  Any already existing file of this name is deleted before recording starts.
- The data is recorded every 8 ms.
- As <File-I/O> equals 0, no measures are taken in case of network problems to prevent overflow of memory.
- The axes with the physical axis numbers 1, 2 and 3 record the corrected actual positions.
- The axis 3 additionally records the "second" actual value.
- As <AutoStop> equals 1, no data is recorded when the axes are at standstill.
- As <Poti> equals 0, the potentiometer does not change the scanning time.

Example:

2 - Programming actual position with actual position of motor encoder (S-0-0051)

```
RecordSetProbe RECPROBE1(1,S-0-51)
RecordSetProbe RECPROBE2(2,S-0-51)
RecordSetProbe RECPROBE3(3,S-0-51)
RecordSet RECFILE(/mnt/digit/beispiel.dig,1) RECTIME(8)
```

The parameters have the following meaning:

- The real-time data is stored in the log file "example.dig". It is stored in the path "/mnt/digit". Any already existing file of this name is deleted before recording starts.
- The data is recorded every 8 ms.
- The drives with the physical axis numbers 1, 2 and 3 record the actual position of the motor encoder (S-0-51).
- As <File-I/O> is not specified, feed is reduced to zero in case of network problems. After storing of already filled buffers, the part program is continued automatically.
- As <AutoStop> equals 0, data is also recorded when the axes are at standstill.
- As <Poti> equals 0, the potentiometer does not change the scanning time.

Example:

---

3 - Programming actual position with internal measuring point "I-ACTPOS" of the NC and an Offset (S-0-0053) (actual position value 2)

```
RecordSetProbe RECPROBE1(X,I-ACTPOS,0.0001,%9.3f)
RecordSetProbe RECPROBE2(Y,I-ACTPOS,0.0001,%9.3f)
RecordSetProbe RECPROBE3(Z,I-ACTPOS,0.0001,%9.3f,-101)
RecordSetProbe RECPROBE101(Z,S-0-53,[FAKTOR])
RecordSet    RECFILE(/mnt/digit/beispiel.dig,1,2)    RECTIME(8,1,0) RECREPARATOR([","])
```

The parameters have the following meaning:

- The axes with the physical axis names X, Y and Z record the corrected actual positions.
- Ident number S-0-53 (actual position value 2) of the axis Z is assigned to the RECPROBE101. In the example, the value of the ident number S-0-053 is scaled in relation to RECPROBE3 with the help of the CPL variable FACTOR.
- For the z-axis, the value of RECPROBE101 is subtracted.
- The measured values are stored in the "/mnt/digit/example.dig" file. Any already existing file of this name is deleted before recording starts.
- As <File-I/O> equals 2, feed is reduced to zero in case of network problems. After storing of already filled buffers, the part program is continued only after "NC start".
- As <AutoStop> equals 1, no data is recorded when the axes are at standstill.
- As <Poti> equals 0, the potentiometer does not change the scanning time.
- A "," is used as separator.

Disabling: Digitizing can be terminated via

- RecordOff
- Program end
- Channel reset
- System reset

The "Digitizing" function provides for the technical prerequisites for taking up a surface profile and for storing logged axis positions.

15.5 The Log File

15.5.1 Description

Functions

File type: The digitizing result is stored in ASCII format as "log file".

File name and path: In "<Name>" of the **RECFILE** parameter (see initializing function "Record-Set"), specify the file name and the path under which the log file is to be stored.

Example: <Name> corresponds to **/mnt/digit/example.dig**

Open file (Start of recording): Open the log file: **RecordOn** command.

Incoming position data are stored in the structure detailed below.

Close file(End of recording): Close the log file: **RecordOff** command.

Data remaining in the buffer is stored.

File structure: The structure of the log file is broken down into three sections. The start of a section is identified with the following **keywords**:

1. **[Settings]**:

   Contains information on the configuration, log file and part program used to digitize the workpiece.

2. **[Data]**:

   Contains real-time data, organized in lines and columns.

   The columns contain the recorded values (e.g. the axis position of the x-axis) which have been defined with RECPROBE1..8.

   Each line contains a complete block of RECPROBE1..8 of a single scanning time. Within the data, programmed interruptions are marked with RecordBreak and a commentary. If no comment is specified with the RecordBreak command, this entry is not required.

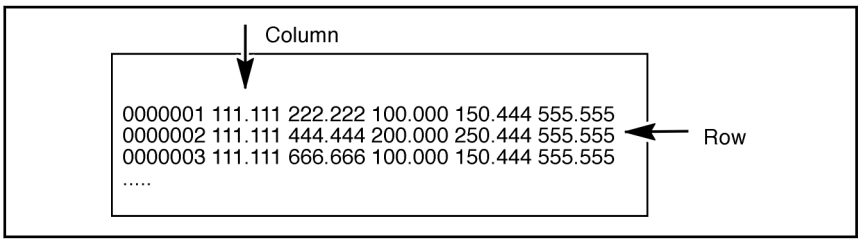


Fig. 15-4: File structure

3. **[Statistics]**:

   Contains statistic information which have occurred during digitizing.

Detailed description of the file structure:

[Settings]

[DATE]

<Creation data of output file>

[PROGRAM]

<File name of the part program>

[RECFILE]

(also see RecordSet RECFILE)

Name: <File name>, Rewrite: <Rewrite>, File-I/O: <File-I/O>

[RECTIME]

see RecordSet RECTIME

RECTIME: <Time>, AutoStop: <AutoStop>, Poti: <Poti>

[RECPROBE1]

see RecordSet RECPROBE1

Axis: <Number>, Name: <Name>, Factor: <Factor>, Format: <Format>

....

[RECPROBE8]

[RECSEPARATOR]

[Data]

<RECPROBE1(<Name>)> {<RECPROBE2(<Name>)} ... {<RECPROBE8(<Name>)}

[Statistics]

NmbOfSamples	Number of recorded lines
LimitUsedBuffers	Maximum number of available buffers The real-time data of digitizing are stored in buffers before they are written into the file.
MaxUsedBuffers	Max. buffer distance reached Depending on the system or network load, the storing of the data in the file takes place much slower than the recording of the real-time data. This results in a buffer distance between the buffer currently being written in the NC and the buffer currently being stored on the hard disk.
DownSlopesDetected	Number "Exceeding the critical buffer distance" If the above-named buffer distance exceeds a maximum value (=MP 8006 00002), the control - depending on the "File-I/O" parameter - attempts to stop the axis motions and then to interrupt digitizing.

DownSlopesExecuted	Number of executed decelerations till standstill (see AutoStop). After the critical buffer distance is exceeded, decelerating the axes is triggered. Once standstill is reached, digitizing is interrupted so that the real-time data can be written into the log file. The NC program can be continued when free buffers are available once more. The output number is an indication of the system or network load.
TimeAtEnd	Time "End of digitizing".
Additional information	Only for internal purposes

Tab. 15-22: Detailed description of the file structure

Example:

Content of the log file "example.dig"

```
[Settings]
[DATE]
6.11.2000 10:35
[PROGRAM]
Example.cnc
[RECFILE]
Name: /mnt/digit/example.dig, Rewrite: 1, File-I/O: 0
[RECTIME]
RECTIME: 8, AutoStop: 0, Potentiometer: 0
[RECPROBE1]
Axis: X, Name: I-ACTPOS, Factor: 1, Format: %d ,
[RECPROBE2]
Axis: Y, Name: I-ACTPOS, Factor: 1, Format: %d ,
[RECPROBE3]
Axis: Z, Name: I-ACTPOS, Factor: 1, Format: %d ,
[RECPROBE4]
Axis: Z, Name: S-0-53, factor: 1, Format: %d ,
[RECSEPARATOR]
<Tab>
[DATA]
111.111 222.222 100.000 150.444
111.111 444.444 200.000 250.444
111.111 546.345 122.000 350.444
...
...
[Statistics]
NumbOfSamples: 1000
LimitUsedBuffers: 50
MaxUsedBuffers: 30
DownslopesDetected: 0
DownslopesExecuted: 0
TimeAtEnd: 7.11.2000 11:50
Info
```

15.6 Axis distance control for digitizing

15.6.1 Description

Function

When a 3D contour is digitized, the coordinates of the surfaces concerned have to be determined and stored.

To this end, a part program is used which

1. initially brings a measuring device (e.g. laser) via the infeed axis (e.g. Z) to a predetermined distance to contour. In the course of the process, this reference is a reference value.
2. Switches on the "axis distance control for digitizing" function
3. Starts the digitizing process (see chapter 15.4 "Digitizing (recording measured values)" on page 619), and
4. afterwards, travels along the object to be digitized for example in a meandering fashion in that level which is defined by means of the other two axes (e.g. x-y-plane).

By means of the measuring device, changes of distance to the reference value are recorded permanently, fed to the NC, and converted to mm if required (standardization see MP 7050 00720). To this end, the measuring data has to be fed into the external encoder connection of a drive (e.g., as an incremental encoder signal) and transmitted cyclically via Sercos interface to the NC.

After activation of the axis distance control, the NC is able to control the position of the infeed axis in such a way that the distance between measuring device and object surface valid at the time of activation will remain constant (deviation to the reference value = 0). This is to ensure that the available working range of the measuring device is not exceeded.

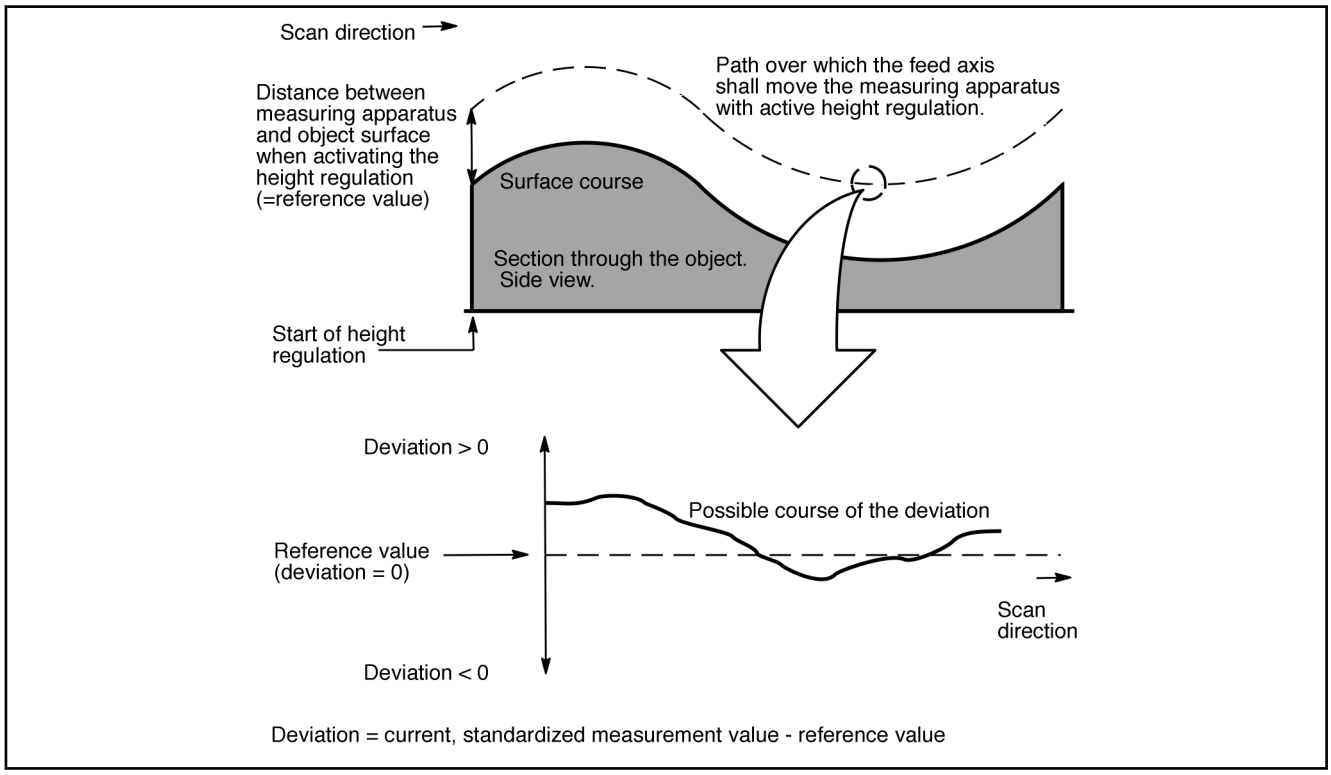


Fig. 15-5: Measuring device

In the framework of the function, you have the following options:

- Program-controlled activation, interruption, continuation and deactivation of axis distance control. See "relevant NC functions".



After control startup, axis distance control is always deactivated.

- Program-controlled fading-over of the related machine parameters. See "relevant NC functions".

Faded machine parameter configuration data becomes only effective after program deselection, channel or system reset!

- Monitoring of deviation via programmable tolerance band.

  In this case, collision and hole detection is possible (see chapter 15.6.2 "Monitoring deviations" on page 641).
- Limiting of the current standardized measured value in respect of velocity and acceleration.

  See MP 7050 00740 and MP 7050 00741.
- Smoothing the incoming measured values.

  Suppresses "outliers", see MP 7050 00730.

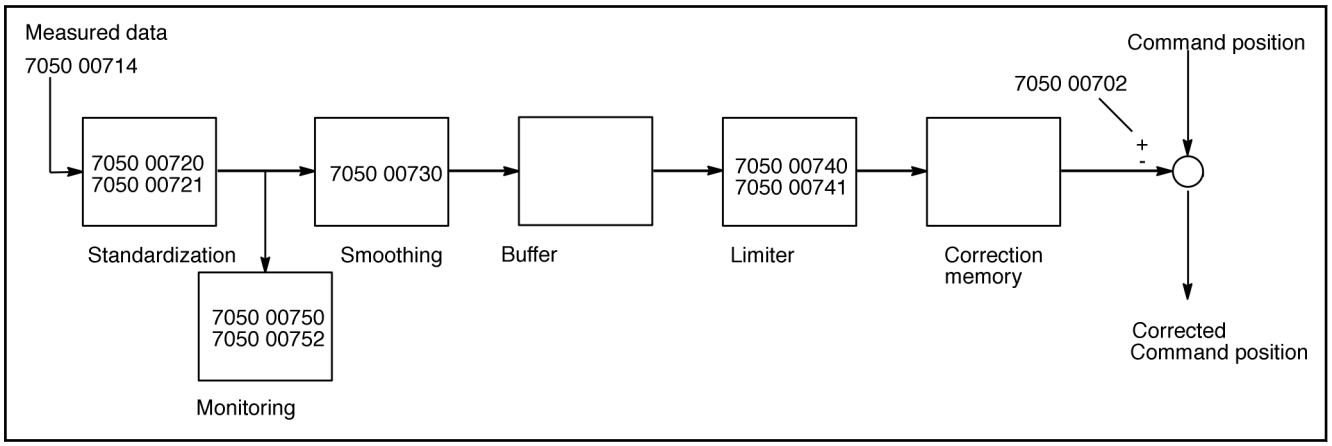


Fig. 15-6: Effect points of machine parameters in the measuring data processing path

Restrictions

- When axis distance control is activated, the feed axis must not be removed from the channel (e.g. via axis transfer).
- Axis distance control is available only for axes which are not involved in an axis transformation when coordinates programming is switched on.
- With axis distance control activated, the software limit switches of the infeed axis are out of function.
- When axis distance control is switched on, the infeed axis cannot be moved to a fixed machine position.

Relevant NC functions

<table>
<tr><td rowspan="20">DistCtrl/DCR</td><td colspan="3">Starts the axis distance control.
Additionally, the current distance between measuring device and surface is applied as reference value. Individually programmed, the configuration data defined via machine parameters are applied.
Optionally, some configuration data can be faded over by means of programming various additional commands. They are programmed within a parameter list of the "DistCtrl/DCR" function which is enclosed in round brackets:</td></tr>
<tr><td colspan="3">DCAXIS(<Axis>,<Corr>)</td></tr>
<tr><td></td><td colspan="2">Overrides MP 7050 00702.</td></tr>
<tr><td><Axis></td><td colspan="2">Name or number of the channel axis for which axis distance control is to be activated.</td></tr>
<tr><td rowspan="2"><Corr></td><td>+1 or 1:</td><td>Take into account the correction values in positive direction of motion</td></tr>
<tr><td>-1 :</td><td>Include correction values in negative direction of motion in the calculation</td></tr>
<tr><td colspan="3"></td></tr>
<tr><td colspan="3">DCFILTER(<Time>)</td></tr>
<tr><td></td><td colspan="2">Overrides MP 7050 00730.</td></tr>
<tr><td><Time></td><td>0:</td><td>Filter off</td></tr>
<tr><td></td><td>> 0:</td><td>Filter on, values in ms</td></tr>
<tr><td colspan="3"></td></tr>
<tr><td colspan="3">DCLIMIT([<Vel>] [<Acc>])</td></tr>
<tr><td></td><td colspan="2">Overrides MP 7050 00740 and MP 7050 00741.</td></tr>
<tr><td><Vel></td><td colspan="2">Overrides MP 7050 00740.
Value input using the unit mm/min or inch/min, as required by the active unit of measurement (G71, G70)</td></tr>
<tr><td><Acc></td><td colspan="2">Overrides MP 7050 00741.
Value input using the unit m/s^2 or 1000 inch/s^2, as required by the active unit of measurement (G71, G70).</td></tr>
<tr><td colspan="3"></td></tr>
<tr><td colspan="3">DCMON([<Collision>][<Hole>])</td></tr>
<tr><td></td><td colspan="2">Overrides MP 7050 00750 and MP 7050 00752</td></tr>
<tr><td><Collision></td><td colspan="2">Overrides MP 7050 00740.
Value input using the unit mm/min or inch/min, as required by the active unit of measurement (G71, G70)</td></tr>
<tr><td><hole></td><td colspan="2">Overrides MP 7050 00741.
Value input using the unit m/s"2" or 1000 inch/s"2", as required by the active unit of measurement (G71, G70).</td></tr>
</table>

DcBreak/DCB	Interrupts axis distance control. The current correction value remains active.
DcCont/DCC	Resumes axis distance control after an interruption. The NC controls the deviation from the reference value as fast as possible.
DistCtrl(0) or DCR(0)	Deactivates axis distance control, stores the current correction value and stops axis motion. If the command is programmed together with a traversing motion in one block, the NC deactivates axis distance control only after the motion.

Tab. 15-23: Relevant NC functions

Relevant CPL functions

PCS, WCS	Neither of these functions takes the correction value of axis distance control into account!
MCS	It provides the position of the coordinate including the correction value.
SPOS	It provides the command position of the axis including the correction value.
APOS, PPOS, PROBE	All CPL functions supplying a value based on the actual position or the measured value of a drive take the correction value of axis distance control into account.

Tab. 15-24: Relevant CPL functions

Relevant Sercos parameters

S-0-1050.1.6	Configuration list AT Determines which Sercos parameters are to be cyclically transmitted to the NC. The Sercos parameter specified in MP 7050 00714 has to be included (e.g. S-0-0053 actual position value 2 incremental). See the respective drive documentation for the realization of the additional evaluation of the external encoder.

Tab. 15-25: Relevant Sercos parameters

Relevant channel interface signals

iAx_Custom1 to iAx_Custom8 see "MP 7050 00708".

Relevant machine parameters (MP)

7050 00700	Axis distance control exists 0: No; 1: Yes
7050 00702	Axis distance control: Number of the channel axis and correction direction. Specifies the axis of the channel for which axis distance control is to be activated. An axis number • Without or with positive sign causes the correction values to be taken into account in the positive direction of motion of the axis for which axis distance control is activated. • With negative sign causes the correction values to be taken into account in the negative direction of motion of the axis for which axis distance control is activated.
7050 00708	Axis distance control: Interface signal, number of the customer output of the channel Specifies the output of the respective channel interface at which an activated axis distance control is to be recorded (output set: Axis distance control is active).
	-1 : No signal output
	0: i signal Ax_Custom1
	:
	7: i signal Ax_Custom8
7050 00710	Axis distance control: Sensor selection, device group
7050 00712	Axis distance control: Sensor selection, device number
7050 00714	Axis distance control: Sensor selection, special identifier (S-0-xxxx, P-0-xxxx, @xxxx) It indicates the drive parameter in which the encoder impulses coming from the measuring device are counted. The Sercos parameter entered here likewise has to be included in the drive telegram S-0-1050.1.6.
7050 00720	Axis distance control: Sensor increments → mm factor Together with MP 7050 00721, is used to convert the data transmitted to the NC (e.g. encoder impulses) into the unit millimeter.
7050 00721	Axis distance control: Sensor increments → mm divisor See "MP 7050 00720".
7050 00730	Axis distance control: Filter smoothing time in ms Parameterizes a rectangular filter Entry of 0 deactivates the filter.

7050 00740	Axis distance control: Limiter - velocity in mm/min Maximum velocity with which the current standardized measured value is permitted to change. In this way, the infeed axis, for example on traveling a hole, can be prevented from plunging into the whole at its maximum permitted velocity.
7050 00741	Axis distance control: Limiter - acceleration in m/s^2 Maximum acceleration with which the current standardized measured value is permitted to change.
7050 00750	Axis distance control: Monitoring - collision detection in mm (0.000=off) Specifies the tolerance limit for collision detection (in mm). If the deviation from the reference value exceeds the value specified, the NC signals a collision. The entry of 0.000 deactivates collision detection.
7050 00752	Axis distance control: Monitoring - hole detection in mm (0.000=off) Specifies the tolerance limit for hole detection (in mm). If the deviation from the reference value exceeds the value specified, the NC "freezes" the axis distance control. In this state, the infeed axis remains at its current position. Enter 0.000 to deactivate hole detection.

Tab. 15-26: Relevant machine parameters

15.6.2 Monitoring deviations

At first, the NC converts the data of the measuring device coming in via the Sercos interface to mm via MP 7050 00720 and MP 7050 00721.

Note that for correct interpretation of the measuring data

- A correct configuration of the external encoder interface in the drive is required (S-0-0115, bit 3 or bit 5),
- the correction value have to be taken into account

  (using MP 7050 00702: sign, axis number).

In the result, the reported position data have to:

- Get smaller in case of an increase of the distance between measuring device and surface, as the surface position (in relation to the measuring device) descends.
- Get larger in case of a decrease of the distance between measuring device and surface, as the surface position (in relation to the measuring device) rises.

Now, the standardized measured value is checked for meeting the maximum permissible tolerances.

This check is based on the distance between measuring device and object surface which is active when axis distance control is switched on. In the following, this reference value is used as zero line of a tolerance range.

If the currently standardized measured value is not identical with the reference value, there is a positive or negative deviation from the zero line.

In this case, traveling across a hole results in a jump to a higher deviation in negative direction, as the encoder suddenly reports a lower position of the scanned surface.

A suddenly rising contour, on the other hand, causes an abrupt increase in the reported surface position, resulting in a jump to a higher deviation in positive direction.

Example:

Monitoring deviations

- When axis distance control is switched on, the distance to the surface is 100 mm. Accordingly, the encoder reports e.g. the standardized value of 100 mm. From now on, this distance is used as a reference value:

  Deviation = current standardized measured value - master value = 100 mm - 100 mm = 0 mm

- Now, the surface contour falls by 60 mm. The encoder reports the position 40 mm:

  Deviation = current standardized measured value - reference value = 40 mm - 100 mm = -60 mm

  The NC immediately compensates fort he deviation by readjusting the infeed axis. This reduces the deviation to 0. At the same time, the standardized position reported by the sensor increases to 100 mm once more.

- Now, the surface contour abruptly rises from 40 mm to 100 mm. The encoder reports the position 160 mm:

  Deviation = current standardized measured value - master value = 160 mm - 100 mm = +60 mm

  The NC immediately compensates fort he deviation by readjusting the infeed axis. This reduces the deviation to 0. At the same time, the standardized position reported by the sensor falls to 100 mm once more.

The jump to a higher deviation in positive or negative direction can by used for **collision and hole detection**. To this end, the current deviation is permanently compared to MP 7050 00750 (collision detection) and MP 7050 00752 (hole detection).

When the current deviation exceeds the respective parameterized limit value, the NC reacts as follows:

- After collision detection:

  The NC generates a warning and stops the motion in the channel via feed hold.

  If no collision has occurred, the operator can continue the motion with NC start (possibly with lower position of the feed potentiometer).

- After hole detection:

  The NC interrupts the axis distance control until crossing of the hole is completed (effect comparable to DcBreak/DCB). Scanning of the object continues. During the process, the current correction value remains active, i.e. the infeed axis does not change its coordinate.

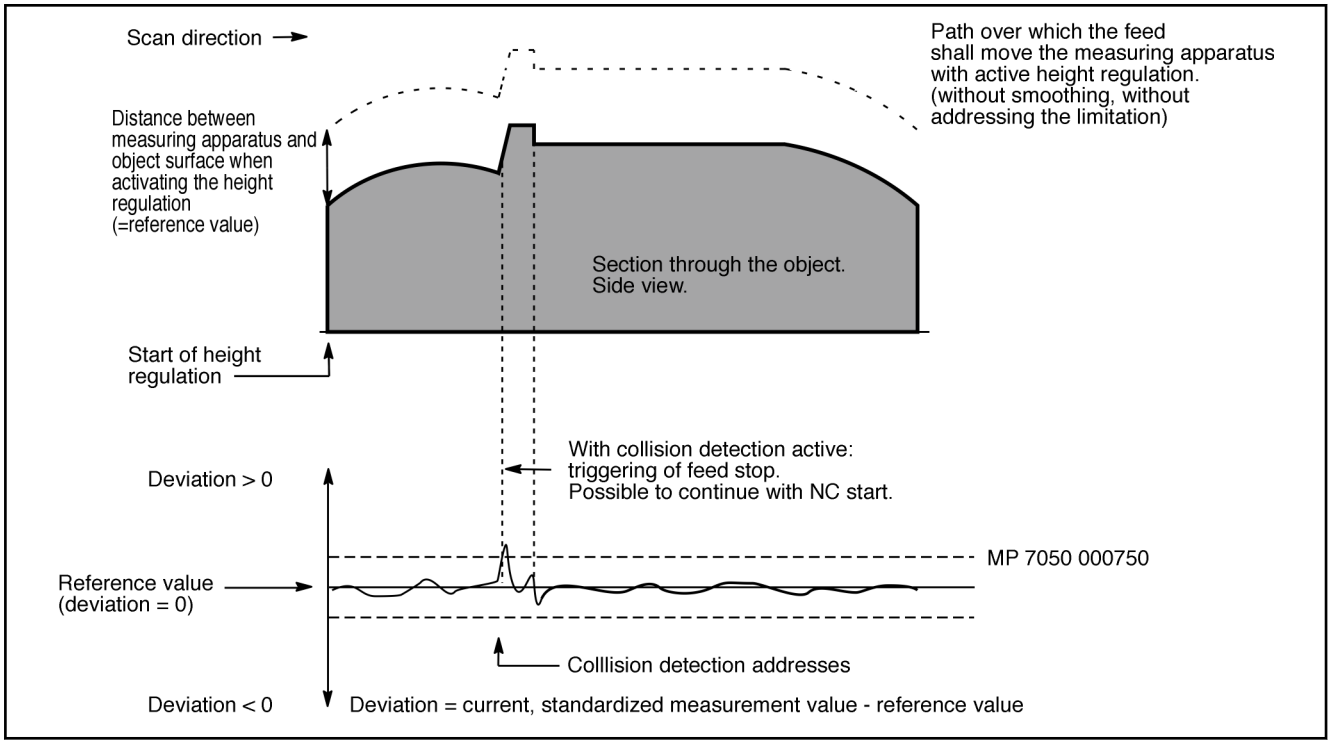


Fig. 15-7: Principle of collision detection

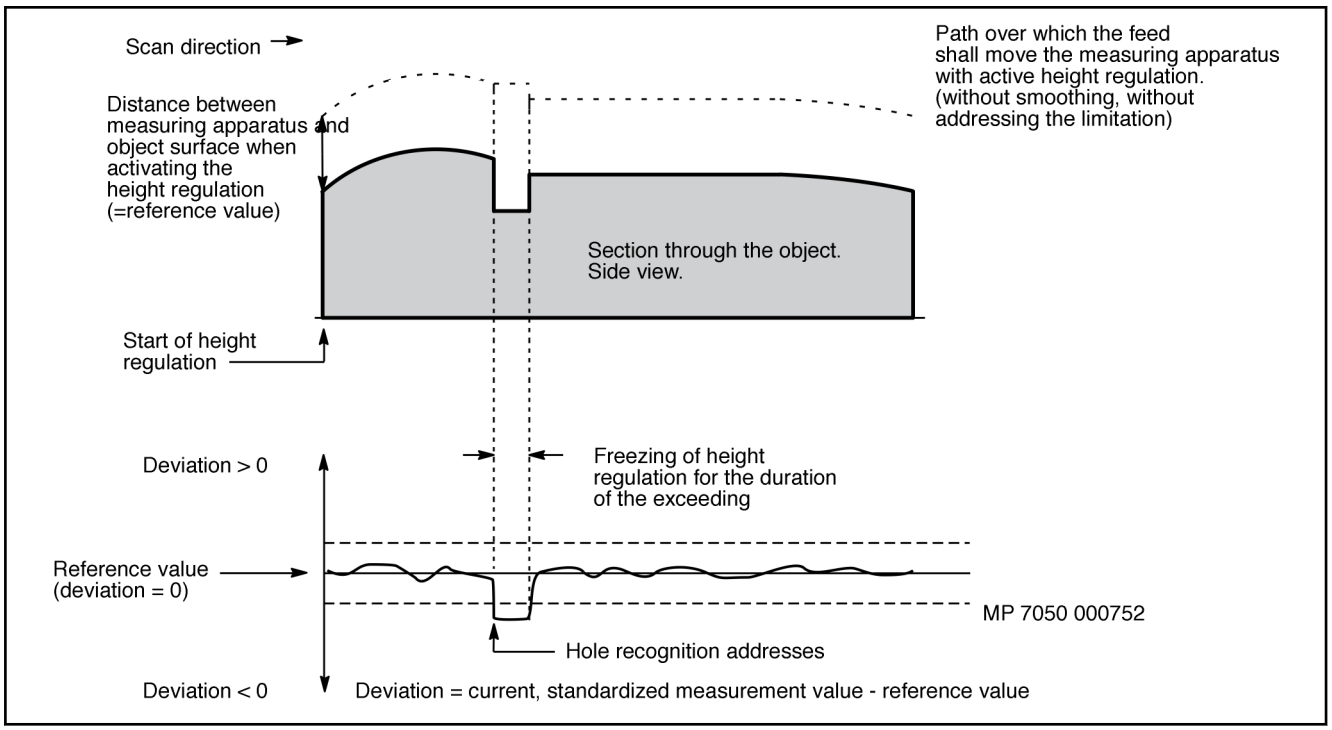


Fig. 15-8: Principle of hole detection

15.6.3 Handling instruction: Axis Distance Control for Digitizing

Applying:

1. Ensure that the measuring data signals of the measuring device are suitable for the external encoder connection of the drive.

2. The relevant Sercos parameters (see above) are correctly parameterized in the drive and saved, or - at Sercos initialization - transmitted by the control to the drive.

   For information on Sercos, refer to the manual "MTX Functional Description Special Functions", chapter "Functions for the Drive Parameterization".

3. Check whether the external measured value changes in the correct manner when the distance between the measuring device and the scanned surface changes.
4. In the drive telegram (S-0-1050.1.6), include the Sercos parameter which comprises the actual position value of the measuring device and is to be cyclically transferred to the NC.

   Check whether the correct telegram is configured in S-0-0015. Sercos parameters comprised in S-0-1050.1.6 are only transferred to the NC if the "freely configurable telegram" is selected in S-0-0015.
5. Configure the parameters listed under "relevant machine parameters" according to your application.
6. Startup control and drives again.

NOTICE **Workpiece or machine can be damaged!**

Ensure that the correction direction (MP 7050 00702) has been correctly parameterized.

☞ MP 7050 00702 and MP 7050 00730 to 7050 752 can also be influenced via the NC function "DistCtrl".

NOTICE **Workpiece or machine can be damaged!**

Do not reset the control during machining.

Switching on and off

Activation:

Program DistCtrl(...) or DCR(...) (with additional parameters if applicable).

Deactivation:

Program DistCtrl(0) or DCR(0).

After activation of the axis distance control, the NC is able to control the position of the infeed axis in such a way that the distance between measuring device and object surface valid at the time of activation will remain constant (deviation to the reference value = 0). This is to ensure that the available working range of the measuring device is not exceeded.

IW Operation: Preparing measuring data processing

The measuring data has to be fed into the external encoder connection of a drive (e.g. as an incremental encoder signal) and transmitted cyclically via the Sercos interface to the NC.

IW Operation/program: Adapt Sercos file to phase 2

- Enter the "Freely configurable telegram" into the Sercos parameter **S-0-0015** "Telegram parameter".
- Enter the Sercos parameter contained in parameter **ScsParMcsCorr** "Sercos parameter" (7050 00714) in the Sercos parameter **S-0-1050.1.6**

"Configuration list drive telegram" (e.g. S-0-0053 "actual position value 2 incremental").



See the respective drive documentation for the realization of the additional evaluation of the external encoder.

	Documentation	Instruction
Instruction:		Editing SCS parameters
Documentation:	Rexroth IndraDrive Sercos Parameter Description	Height control

IW Engineering/configuration: Editing parameters

- The distance control function has to be activated via the parameter **onl** "Distance control" in the setup (SUP).
- **EnaMcsCorr** "Activate distance control" (7050 00700)
- **ChAxMcsCorr** "Channel axis" (7050 00702)
- **DirMcsCorr** "Correction direction" (7050 00702)
- **ChIfSigMcsCorr** "Interface signal" (7050 00708) determines, via which customer channel interface signal an activated axis distance control is signaled.
- Right mouse button on the node **Device** "Sensor selection"
- "Change to"
- Select the desired function:
  - **Dr** "Drive" (7050 00710)
  - **CplVar** "Permanent CPL variable" (7050 00710)

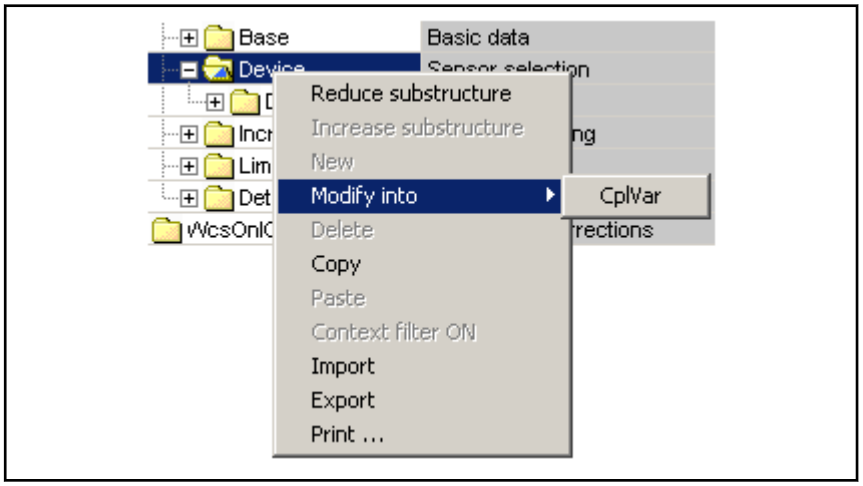


Fig. 15-9: Modifying the sensor selection

- Depending on the selected function, only some of the following parameters are active:
  - **DrIndMcsCorr**
    "Drive number" (7050 00712)
  - **ScsParMcsCorr**
    "Sercos parameter" (7050 00714)
  - **CplVarMcsCorr**
    "Variable name" (7050 00714)

- **FactMcsCorr**
  "Multiplier" (7050 00720)
- **DivMcsCorr**
  "Divisor" (7050 007210)
- **FilterMcsCorr**
  "Smoothing/filter" (7050 00730)
- **MaxVelMcsCorr**
  "Maximum velocity" (7050 00740)
- **MaxAccMcsCorr**
  "Maximum acceleration" (7050 00741)
- **CollDet**
  "Collision detection" (7050 00750)
- **HoleDet**
  "Hole detection" (7050 00752)



The parameter **DirMcsCorr** "Correction direction" (7050 00702) and the parameters **FilterMcsCorr** "Smoothening/filter" (7050 00730) to **HoleDet** "Hole detection" (7050 00752) can also be influenced via the NC function **"DistCtrl"**.

	Documentation	Instruction
Instruction:		Editing machine parameters
Documentation:	MTX Machine Parameters	Axis Distance Control for Digitizing

IW Operation/IndraLogic: Linking interface signals

If the signal output is active, the channel interface signal **((iAx_Custom1...iAx_Custom8)** that is declared in the parameter **ChIfSigMcsCorr** "Interface signal" (7050 00708) is linked.

	Documentation	Instruction
Instruction:		Editing PLC Signals
Documentation:	MTX PLC Interface	Axis Distance Control for Digitizing

15.6.4 Handling instruction: Adjust the Axis Distance Control for Digitizing Parameter via CPL

Then, after activation of the axis distance control, the NC is able to control the position of the infeed axis in such a way that the distance between measuring device and object surface valid at the time of activation will remain constant (deviation to the reference value = 0). This is to ensure that the available working range of the measuring device is not exceeded.

IW Engineering/configuration: Applying axis distance control

		Instruction chapter 15.6.3 "Handling instruction: Axis Distance Control for Digitizing" on page 643
Instruction:		Applying the axis distance control

IW Operation/NC programming: Modify the tool length correction

- **DistCtrl(<Fct>)** or **DCR(<Fct>)** overrides some function-specific configuration data in the machine parameters.

  - with -

  <Fct>

  – **DcAxis(<Axis>,<Corr>)** overrides the parameter **ChAxMcsCorr** "channel axis" (7050 00702) and the parameter **DirMcsCorr** "correction direction" (7050-00702):

    Short form: DCA(...)

    – **<Axis>**

      Name or number of the channel axis for which axis distance control is to be activated.

    – **<Corr>**

      Direction of motion in which the correction values are to be included in the calculation:

      +1 or 1:In positive direction of motion

      -1: In negative direction of motion

  – **DcFilter(<Time>)** overrides the parameter **FilterMcsCorr** "smoothening/filter" (7050 00730).

    Short form: **DCF(...)**

    – **<Time>** Filter parameterization for smoothening the sensor values.

      0: Filter off

      >0: Filter on, smoothing time in ms

  – **DcLimit({<Velocity>},{<Acceleration>})** overrides the parameter **MaxVelMcsCorr** "max. velocity" (7050 00740) and the parameter **MaxAccMcsCorr** "maximum acceleration" (7050 00741).

    Short form: **DCL(...)**

    – **<Vel>**

      Maximum change velocity of the correction value. Value input using the unit mm/min or inches/min, as required by the active unit of measurement (G71, G70).

    – **<Acc>**

      Maximum steepness (acceleration) of the correction value. Value input using the unit m/s2 or 1000 inch/s2, as required by the active unit of measurement (G71, G70).

  – **DcMon({<Collision>},{<Hole>})** overrides the parameter **CollDet** "collision detection" (7050 00750) and the parameter **HoleDet** "hole detection" (7050 00752). Short form: DCM(...)

    – **<Collision>**

Tolerance band for collision detection. Value input using the unit mm or inch, as required by the active unit of measurement (G71, G70).

0: Collision detection off.

– **<hole>**

Tolerance band for hole detection. Value input using the unit mm or inch, as required by the active unit of measurement (G71, G70).

0: Hole detection off.

– **DcBreak**

Interrupts axis distance control. The current correction value remains active. Short form: DCB

– **DcCont**

Resumes axis distance control after an interruption using DCB.

The NC controls the deviation from the reference value as fast as possible.

Short form: **DCC**

	Documentation	
Documentation:	MTX Programming Manual	Tool length correction

15.7 Measurement at fixed stop

15.7.1 Description

While the control travels all the programmed synchronous axes using linear interpolation and the given feed to the programmed end point, the current torque is monitored for a selected axis. If a torque limit is exceeded for this axis, then the path motions are decelerated to v=0.

The following handling instructions describe measuring at fixed stop.

15.7.2 Handling instruction: Measurement at fixed stop

IW Engineering/Configuration:

- The "Move to fixed stop" function has to be activated via the parameter **MPS** "Move to fixed stop" in the setup (SUP)
- **EnablMovePoStop** "Move to fixed stop can be activated (1003 00030)"
- **MaxVelPoStop**

  "Maximum axis velocity (1005 00030)"
- **MaxTorqPoStop**

  "Maximum torque limit (1003 00031)"

Instruction:		Editing machine parameters
Documentation:	MTX Machine Parameters	Measurement at fixed stop

IW Operation/NC programming:

- **FsProbe(MfsAxis<i>)<Coordinates><Feed>**

  "Measurement at fixed stop"
- **FsProbe(MfsAxis(<k>, {<Threshold>}))<Coordinates><Feed>**

  "Measurement at fixed stop" with torque threshold

– **<i>**

  System axis number of the measurement axis

– **<k>**

  System axis name or number of the measurement axis

– **<Threshold>**

  Torque threshold input value in % of maximum torque. If no <Threshold> is programmed, the parameter **MaxTorq** "Maximum torque limit (1003 00031)" is active

– **<Coordinate>**

  Position to be approached while measuring.

– **<Feed>**

  Desired path feed. Limited by the parameter **MaxVelPoStop** "Maximum axis velocity (1005 00030)" and the parameter **MaxVel** "Maximum axis velocity (1005 00002)".



- The interface signal **iAx_FxStopAct** "Fixed stop active" is output as long as the traversing motion to the fixed stop is active.
- The following functions are not permitted in the **FsProbe block**:
  – **G75** (Probe)
  – **MOB** (measuring on a block)
  – **InitMeas/FlyMeas** (flying measurement)
  – **RedTorque** (Torque reduction)
  – **FsMove / FsReset / FsTorque** (Move to fixed stop)

Documentation:	MTX Programming Manual	Measurement at fixed stop

15.8 Measuring in space

15.8.1 Description

The function "Measuring in space" contains two cycle commands:

- G629
- G628

The cycle **G629** determines the center point of a measuring ball. The touch probe is positioned and oriented in front of the ball using "Jogging in space". It points approximately into the direction of the ball center. The measuring cycle is started and measures the ball automatically by sampling five points. The position of the ball center is measured in the BCS and in the WCS. The tool vector does not have to be parallely to an axis.

The cycle **G628** computes the orientation of a plane using three ball center points determined using **G629**.

System Design

The ball center point can be measured in different transformation types. The transformations are shown in chapter 11 "Axis transformation" on page 183.

- **RRLLL**: The axis of the touch probe moves parallely to the x-/y-/z-coordinates in the BCS.

- **RLLLR**: The transformation angle Theta Θ determines the direction of motion of the touch probe axis.
- **LLLRR**: The transformation angles Theta Θ and Phi Φ determine the direction of motion of the touch probe axis.

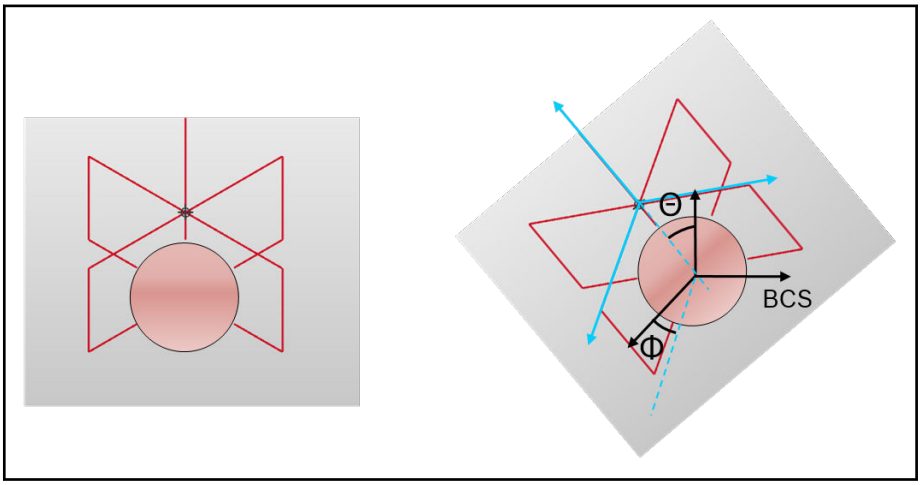


Fig. 15-10: Schematic representation of the direction of motion of the touch probe. a): RRLLL, b): RLLLR and RLLLR.

System data

Input parameters and intermediate data are saved in SD variables. The structure `SD.SysPrBall.Ball` includes the parameters of the balls to be measured. The parameters of up to nine balls can be logged during a measuring run. To compute the plane orientation in BCS `SD.SysPrBall.PLANE` and in WCS `SD.SysPrBall.PLANE_WCS`, three ball positions are required. The ball center points are determined using five measuring points per ball. The points are saved in the substructure `SD.SysPrBall.Ball.PNT`.

SD variables	Description
`SD.SysPrBall.PLANE`	Plane orientation in BCS
`SD.SysPrBall.PLANE_WCS`	Plane orientation in WCS
`SD.SysPrBall.Ball[*]`	Ball parameter for the measurement
`SD.SysPrBall.Ball[*].TRN`	Macoda parameters of the axis transformation
`SD.SysPrBall.Ball[*].TRS`	Transformation data (system date)
`SD.SysPrBall.Ball[*].PNO`	Index of the touch probe. 1: Probe 1; 2: Probe 2
`SD.SysPrBall.Ball[*].PDI`	Distance between the starting position and the first point of contact, positive values
`SD.SysPrBall.Ball[*].RAD`	Ball radius
`SD.SysPrBall.Ball[*].IFE`	Feeding velocity
`SD.SysPrBall.Ball[*].PFE`	Feed velocity of the measurement
`SD.SysPrBall.Ball[*].NPF`	Path of the measuring result
`SD.SysPrBall.Ball[*].ERT`	Maximum permitted deviation of the measuring result (deviation from the measured ball radius)

SD.SysPrBall.Ball[*].RES	Measuring result
SD.SysPrBall.Ball[*].RES.BCP	Measured position of the ball center point in the BCS
SD.SysPr-Ball.Ball[*].RES.BCP_WCS	Measured position of the ball center point in the WCS
SD.SysPrBall.Ball[*].RES.RAD	Measured radius Sum of the ball and touch probe radius
SD.SysPrBall.Ball[*].RES.ERT	Deviation from the measured ball radius (ABS(Measured calibrated ball radius – RES.RAD)/ RES.RAD)
SD.SysPrBall.Ball[*].PNT[**]	Intermediate data when measuring the surface points.
SD.SysPr-Ball.Ball[*].PNT[**].TouP	Position of contact on the ball surface under WCS
SD.SysPr-Ball.Ball[*].PNT[**].BCSP	Position of contact on the ball surface under BCS
SD.SysPr-Ball.Ball[*].PNT[**].Ori	Orientation when measuring the surface points.
SD.SysPr-Ball.Ball[*].PNT[**].StrP	Starting point of measuring motion
SD.SysPr-Ball.Ball[*].PNT[**].TarP	Target point of the measuring motion
SD.SysPr-Ball.Ball[*].PNT[**].IP1	Intermediate point 1 when feeding
SD.SysPr-Ball.Ball[*].PNT[**].IP2	Intermediate point 2 when feeding
SD.SysPr-Ball.Ball[*].PNT[**].MRD	Measured radius of the current point
SD.SysPr-Ball.Ball[*].PNT[**].ERT	Measuring radius deviation of the current point

* Index of the ball. Up to 9 balls can be measured during a run.

** Index of the points of contact of the ball surface. 5 points per measurement.

Tab. 15-27: System data for measuring cycles

Parameter

The cycle commands **G629** and **G628** require the following input parameters:

Input parameter	Description	Property	Data type	Unit/setting range
TRN	Macoda parameters of the axis transformation	Mandatory	INT	1 - 20
TRS	Transformation data (system date)	Optional	STRING	200
BNO	Ball index	Mandatory	INT	1 - 9

Input parameter	Description	Property	Data type	Unit/setting range
PNO	Index of the touch probe. 1: Probe 1; 2: Probe 2	Mandatory	INT	1 - 2
PDI	Measuring distance	Mandatory	DOUBLE	Positive
RAD	Ball radius	Mandatory	DOUBLE	mm
NPF	Path for the measuring result	Optional	STRING	Default "/usr/mtb"
IFE	Feeding velocity	Mandatory	DOUBLE	mm/min
PEF	Feed velocity during measurement	Mandatory	DOUBLE	mm/min
ERT	Maximum permitted deviation of the measuring result	Optional	DOUBLE	

Tab. 15-28: Input parameter for G629

Input parameter	Description	Property	Data type	Unit/setting range
BIDX	Index of the balls used to determine the plane orientation	Mandatory	INT	BIDX: 123-987 Three different balls have to be selected. The index has to be-tween 1 and 9. The index order influences the direction vector of the new x-axis related to the start coordinate system

Tab. 15-29: Input parameter for G628

15.8.2 Commissioning

Preparation

The following steps are required for the measurement:

1. Importing the file "SDDefProbeBall.xml" from /feprom to /usrfep
2. Activating the 5-axis transformation and specifying the parameters TRN with the transformation type used (see also “System Design "System Design" on page 649)
3. Setting the touch probe on the tool number N by PNO=N and activating the tool N. Setting the tool parameters L3 with a probe length (up to the center point of the touch probe).
4. Ensure that the touch probe is calibrated. The resulting measured calibrated ball radius SD.SysPrBase.PROBE[].GEO.R is used in the **G629** measuring cycle.

5. Activating the tool length correction G47. When activating the tool length correction, G41/G42 may **NOT** be used. As the position of the touch center point is required.
6. The radius RAD of the ball to be measured is required as input value.
7. Configuration and feed velocity during the measurement `ERT` and the feeding velocity **PFE**.

Commissioning

1. Index specification of the ball to be measured by the input parameter **BNO**
2. Manual moving of the touch probe close to the ball interface. This position is used as starting position.

☞
- 2.1 The longitudinal axis of the touch probe has to point towards the ball center point.
- 2.2 The distance between the probe and the ball surface should be smaller than the input parameter “Measuring distance PDI”.

3. Measurement start
4. Measurement repeated for the next ball.

☞ **Three** balls are required to determine the orientation of a surface

Measuring

1. The vector v_1 describes the distance between the starting position and the measured ball center point.
2. Measuring the first point of contact
   - 2.1 The probe moves along the tool orientation v_1 using the measuring cycle G75
   - 2.2 When the probe touches the ball surface, the motion is stopped.
   - 2.3 The point of contact is logged with the P_1
   - 2.4 The touch probe returns to the initial position P_0.
3. Measuring the second point of contact.
   - 3.1 The touch probe moves parallely to the xy-plane along the vector $-v_1$. The distance of the motion is *d = 2 * ® + r)*.
   - 3.2 The touch probe moves parallely to v_1 by v_1+*R*.
   - 3.3 Touch probe moves towards the ball center point v_2
   - 3.4 Measurement of the point of contact with the measuring cycle G75.
   - 3.5 The point of contact is logged with the P_2
   - 3.6 The touch probe returns to the initial position P_0.
4. Measuring the third point of contact as in step 3.
5. Measuring the fourth point of contact along v_4. The vector v_4 is vertical to v_1 and v_2.
6. Measuring the fifth point of contact.
7. Computing the ball center point using five ball surface points.
8. Checking the deviation of the ball radius. The measurement is aborted if the deviation is too high.

9. Computing the plane orientation using three previously determined ball center points.

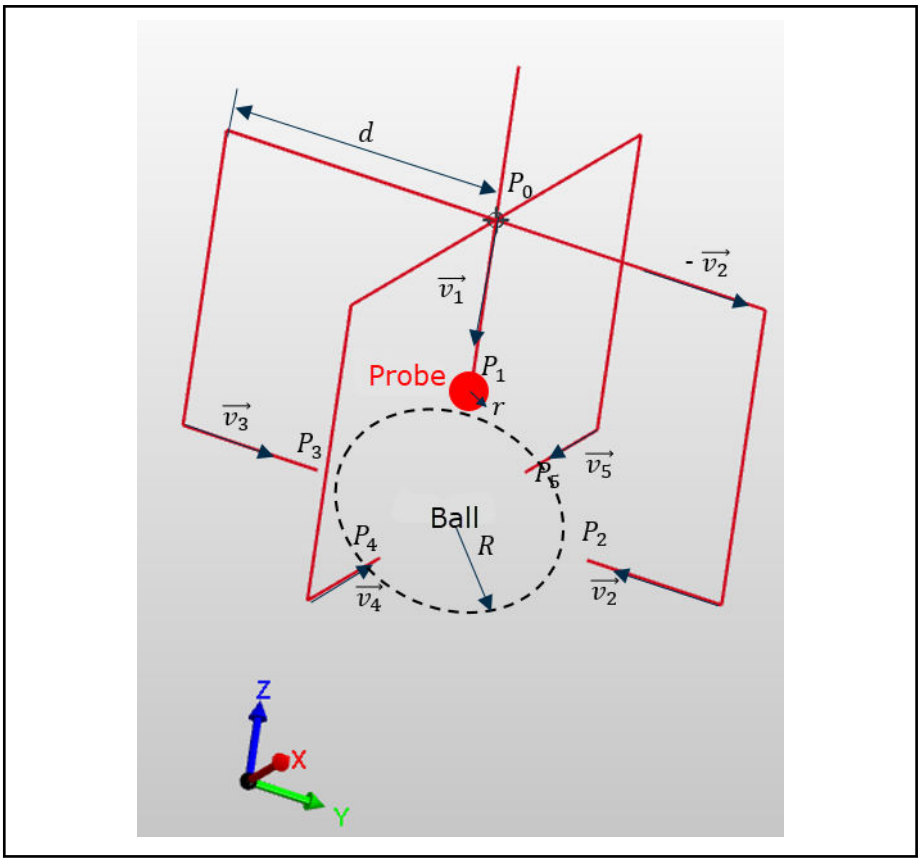


*Fig. 15-11: Path motion of measuring cycle **G629***

Use case

This section shows an NC program example to determine a plane orientation by measuring three ball center points. The measuring log is stored in the file "SysPrRes_ddmm_hhmmss.txt " in *PATH$* The plane orientation is saved in the structure "SD.SysPrBall.PLANE" and can be used as parameter for the “Inclined plane” function.

Example: NC program to determine a plane orientation:

Program:

```
;Step 1: Set probe parameters and active probe as a tool
;Prepare for the G629 cycle
;Probe parameters setting in simulation mode
;On real machine, please make sure you set the right probe parameters
10 SD.SysPrResult.CAL[1].AXIS[1].DIR[1].CALI = TRUE
20 SD.SysPrBase.PROBE[1].GEO.R = 3
30 SD.SysPrBase.PROBE[1].DRVINP =1
40 SD.SysPrBase.PROBE[1].TOOL=TRUE
;Start
N0 G48 G153 COORD()
N1 STOCK
N10 F10000 STOCKADD(1)
N50 G0 X0 Y0 Z0 A0 C0
1 TRN%=3 : PNO%=2 : PDI=16 : IFE=10000 : PFE=5000
;Active probe tool and only active Z tool length correction only
N60 Coord([TRN%]) // 3232203 RRLLL
N70 T[23]
N80 G47(,,ZTR)
50 DIM PATH$(100)
60 PATH$ = "/usr/user/result"
70 R1=40 : R2=35 : R3=30

;Step 2: Go to ball number 1 initial measurement point.
;Make probe orientation go through ball center.
```

```
; Program position should be probe center position
N90 X-200 Y-200 Z0 phi90 theta-45
;Measure first ball center position
N100 G629(TRN[TRN%],BNO1,PNO[PNO%],PDI[PDI],RAD[R1],IFE[IFE],PFE[PFE],
NPF[PATH$],ERT0.00001)

;Step 3: Go to ball number 2 initial measurement point.
;Make probe orientation go through ball center.
N110 X0 Y0 Z0 phi90 theta45
N100 G629(TRN[TRN%],BNO2,PNO[PNO%],PDI[PDI],RAD[R2],IFE[IFE],PFE[PFE],
NPF[PATH$],ERT0.00001)

;Step 4: Go to ball number 3 initial measurement point.
;Make probe orientation go through ball center.
N130 X0 Y200 Z10 phi30 theta-60
N100 G629(TRN[TRN%],BNO3,PNO[PNO%],PDI[PDI],RAD[R3],IFE[IFE],PFE[PFE],
NPF[PATH$],ERT0.00001)

;Step 5: Calculate orientation of target plane
;Calculate plane orientation by ball index(BINX) 1 2 3
N150 G628(BIDX 123)
N160 WAIT
80 PRN#(0, "(BCS) PHI: ####.####",SD.SysPrBall.PLANE.PHI,"
   THETA: ####.####",SD.SysPrBall.PLANE.THETA," PSI: ####.####",
            SD.SysPrBall.PLANE.PSI)
80 PRN#(0, "(WCS) PHI: ####.####",SD.SysPrBall.PLANE_WCS.PHI,"
   THETA: ####.####",SD.SysPrBall.PLANE_WCS.THETA," PSI: ####.####",
            SD.SysPrBall.PLANE_WCS.PSI)
N170 WAIT


;-----------------
;Set inclined plane paras and activate inclined plane
;G153 ; Activate inclined plane.
;-----------------
N180 Coord(0)
N190 G0 X0 Y0 Z0 A0 C0
N200 M30
```

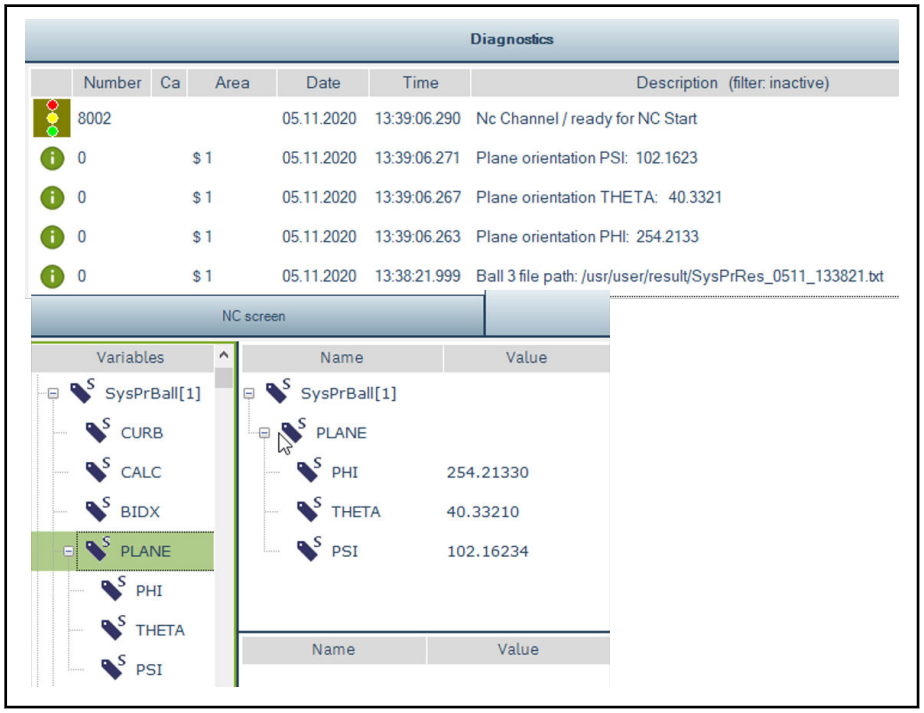


Fig. 15-12: An exemplary measuring result of the plane orientation

15.8.3 Troubleshooting

The measuring cycles **G629** and **G628** can generate the following error messages in the table:

ErrorID	Description
4304	Only use parameters <Parameter names> optionally
4311	Value range exceeded
4355	No edge active
4402	Tool length correction is not active
4407	
4410	Transformation not defined
4413	Transformation incorrect
4426	Tool radius correction active
4431	Axis transformation type is not supported by the cycle
4433	Mandatory parameter is not programmed
4461	SD.SysPrBase.PROBE[<Probe no.>].DRVINP is not 1 or 2
4553	No free logic file number available
4554	Invalid path name
4555	The radius of the ball to be measured is too small
4556	Error when writing the log file to the <directory name> directory
4557	Radius of ball or of probe is too small
4558	The probe <probe no.> is not a workpiece probe
4560	(Warning, no error) The measuring error at the ball <Index> exceeds the adjustment value.
4561	Error when calculating the displacement vector
4562	Error when calculating the center of the calibrating ball
4563	Path name <Parameter name> too long (maximum 70 characters)

*Tab. 15-30: Error codes of the measuring cycle **G629***

ErrorID	Description
4433	Mandatory parameter is not programmed
4311	Value range exceeded
4559	Error when calculating the normal vector of the plane

*Tab. 15-31: Error codes of the measuring cycle **G628***

16 Synchronizing functions

16.1 Introduction

There are different instances in the MTX. These instances are to be synchronized among each other. The most important instances are shown in the following figure.

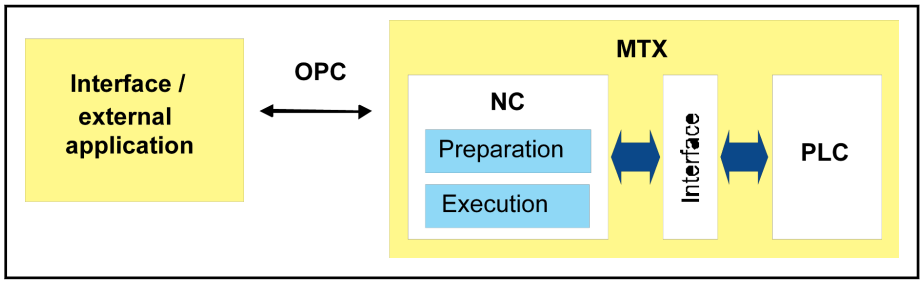


Fig. 16-1: MTX instances

A synchronization takes always place between two devices. One device waits on a specified event triggered by another device depending on its state.

This chapter describes functions to synchronize the NC preparation and the NC execution using other instances. NC or CPL functions are programmed in the part program triggering events while processing or stopping the process and waiting for events.

The table tab. 16-1 "NC functions to stop the NC preparation and NC execution" on page 657 lists the functions to synchronize towards the NC preparation or NC execution. That means that there are functions in a column which are used to wait for the preparation or the execution of an event.

		NC preparation	NC execution
NC preparation		-	PREP
NC execution	In the channel	WAIT	-
	Between channels	-	WEV, WPV
Interface		WAIT(BITIF(...))	WAITA/WAITO, feed hold
PLC		WAIT(BITIF(...))	WAITA/WAITO, feed hold
Positions		-	ASTOPA/ASTOPO, BSTOPA/BSTOPO, WSTOPA/WSTOPO
Interface (OPC)		-	-

Tab. 16-1: NC functions to stop the NC preparation and NC execution

Example:

Finding waiting function in table

The execution of blocks is to be interrupted in the part program in channel 2 at the position "N1010" until the x-axis (programmed in channel 1) exceeds the axis position 500.

Since the NC execution is supposed to wait, it has to be searched in the column "NC execution". For the NC function `ASTOPA`, go to the "Positions" line.

Enter the line `N1010 ASTOPA(1,"X">500)` into the part program of channel 2 to resolve the task. This stops the MTX the execution of channel 2 in this block until the axis position of the x-axis is greater than 500 mm.

☞ Most of the functions waiting during an NC execution (e.g. WEV, WAITA, WAITO, WPV, WPVE ...) implicitly cause a falling edge at the end of the block. Incorrectly set synchronization points can cause damages at the machine.

Check the program sequence for possible synchronization problems prior to the actual processing.

The functions used by the NC preparation and NC execution to send a synchronization event to other instances are listed in the table tab. 16-2 "NC functions to send signals from the NC preparation and NC execution" on page 658.

Example:

Finding sending function in the table

As soon as the NC preparation reaches the block "2020", the interface is to change its display. Since the NC preparation is supposed to trigger the event, it has to be searched in the column "NC preparation". For the "SD (CPL)" entry, refer to the "Interface (OPC)" line.

The task is resolved by

- creating a system date "SignalVar"
- subscribing the interface of the OPC item "NC.SystemDataEvent" and evaluating "NC.SystemData_U" accordingly (see documentation "MTX OPC Communication", chapter "Items") and
- entering the line `2020 SD.SignalVar=10` into the part program.

Thus, the system date is set to a value of 10 by the NC preparation if the block `2020` is reached. The interface is now enabled via the system data server and can query the value or react accordingly.

		NC preparation	NC execution
NC preparation		-	-
NC execution	In the channel	-	-
	Between channels	Perm. CPL variables	SEV, SPV
Interface		BITIF	Different NC functions
PLC		BITIF, PLC, SD (CPL)	M-functions, SSD
Interface (OPC)		SD (CPL)	SSD, SSDQ

Tab. 16-2: NC functions to send signals from the NC preparation and NC execution

☞ There are individual NC functions (e.g. `G61(IPS2)`) setting bits in the interface due to their states. Different instances can react on these bits.

16.2 Synchronizing with NC preparation

16.2.1 NC Preparation and NC Execution (WAIT)

Description

Function

The main task of the NC control is to process part programs and to convert the contained commands (NC blocks) into motions of the drives. For this purpose, the individual blocks have to be interpreted and prepared in the first step. This step is called **NC preparation**.

Currently prepared blocks are cached in a buffer (block preparation buffer). The buffer size is specified by the machine parameter `NCO/LookAh/Ch[k]/NofBlkPrep` (MP 7060 00110). Different MTX functionalities require a certain buffer size. An example is the block look-ahead.

Once a block has been processed completely (or traveled), the **NC execution** receives the next block from the block preparation buffer. The NC execution creates command values for drives based on the preparation data. The command values are finally sent to the drive controller with the drive bus. That means that the programmed motion takes only place at the execution point in time.

☞ All CPL blocks (blocks beginning with a number) are blocks of the NC processing. All commands and calculations of these blocks are executed during preparation time. Also refer to the documentation "MTX Programming Manual", chapter "Basic information on standard and CPL programming".

The NC preparation of a block can be a lot earlier (up to several minutes) than its execution. Since the temporal distance depends on many factors, it can be not predefined for a specific NC block. It is in general not even constant for the individual blocks at repetitive execution of a part program.

Thus, there has to be a synchronization between preparation and execution if the current machine- or process-related system states should be accessed. The **WAIT** command (without parameter) is used for this synchronization.

The WAIT command blocks the block preparation until the NC block programmed immediately before has been completely processed. The block preparation is thus synchronized with the active status (execution point in time) of the control. Then, processing continues with the block preparation of the subsequently programmed block. At this time, no other prepared blocks exist, since these have all already been processed.

☞ The `WAIT` function (without parameter) can be programmed in standard NC blocks (DIN) and in CPL blocks. In a CPL block that includes a WAIT instruction, a ":" must not be programmed. Subsequent CPL commands have to be written to a new CPL block.

Relevant NC functions

N10 WAIT ; WAIT as NC function

Relevant CPL functions

10 WAIT : REM WAIT as CPL function

Relevant machine parameters (MP)

The size of the block preparation buffer is specified by the machine parameter `NCO/LookAh/Ch[k]/NofBlkPrep` (MP 7060 00110).

Handling instruction

In the following example, the current workpiece definition of the axis x is output as message.

Since the CPL function WCS is processed at the preparation point in time, but the axis is moved at the execution point in time, a synchronization between both instances has to take place.

```
N10 G1 F2500 X234          ; Traversing Motion for X
20 WAIT                    ; Synchronization (preparation = execution)
30 XPOS = WCS("X")         ; Determining position at the preparation point in time
40 PRN#(0, "X = ", XPOS)   ; Outputting position
```

Without block 20, the position in block 30 is determined at the preparation point in time while the execution of N10 is still active. That means that a value other than "X=234" is output in this case.

16.2.2 NC Preparation and Interface/PLC (WAIT(BITIF(...)))

Description

Function

Many processes and sequences are controlled in the tool machine by the PLC. It is thus required to synchronize NC and PLC. The interface between the two instances is the PLC interface. This interface is described in detail in the documentation "MTX PLC Interface".

As already described in chapter 16.2.1 "NC Preparation and NC Execution (WAIT)" on page 659, there are two different points in time in the MTX to prepare and execute a block. Both points in time can be clearly separated (several minutes) from each other. Since the temporal distance depends on multiple factors, it cannot be predicted in the part program.

If the current machine- or process-related system states in the NC preparation (e.g. with CPL functions) should be accessed and these depend on the PLC, a synchronization between the NC preparation and the PLC has to take place.

For this purpose, the PLC sets a bit in the interface to signal that the respective system state has been reached. The CPL function `WAIT(BITIF(...))` has to be programmed in the part program so that this block waits on the OK of the PLC.

Relevant CPL functions

Using the CPL function WAIT(BITIF(...)), the NC block preparation waits on a PLC signal.

Relevant IF signals

It can be waited on any bit in the interface.

Handling instruction

In the following example, the axis position of the asynchronous axis XA is determined when a laser light barrier (managed by the PLC) is triggered. The PLC writes the state of the laser light barrier on the fourth customer output of the respective channel (`qCh_Custom4`).

PLC part It is written on the customer output of the PLC in the following program excerpt. The example is written in "Structured text (ST)". The function block should be executed cyclically.

```
VAR_INPUT                          // Declaration part
LaserInput : BOOL;
END_VAR

qCh_Custom4 := LaserInput          // Writing the output
```

Tab. 16-3: Program example: PLC program; Copying the laser light barrier to customer output 4

NC part The preparation of the NC has to be stopped now until the customer output 4 is set in the interface by the PLC. For this purpose, the CPL command `WAIT(BITIF(...))` is used.

If an error occurs in the laser light barrier, a timeout of two seconds is defined. If the PLC signal does not reach the NC within this period, an error message is output. The system does not remain in an undefined state but forwards an error message to the user.

```
10 WAIT(BITIF("qCh_Custom4"), 2000, ERG%)   ; Waiting for customer input 4
20 IF RES%=0 THEN                           ; Did the signal arrive?
30 XAPOS = ACS("XA")                        ; Determining position
40 ENDIF
50 IF ERG%=2 THEN                           ; Did the timeout arrive?
60 SETERR("Laser signal missing!")          ; Outputting error
70 ENDIF
```

Tab. 16-4: Program example CPL program, Waiting for customer input 4

16.2.3 Interface/PLC and NC Preparation (BITIF, PLC, SD)

Description

Function

Many processes and sequences are controlled in the tool machine by the PLC. It is thus required to synchronize NC and PLC. The interface between the two instances is the PLC interface. This interface is described in detail in the documentation "MTX PLC Interface".

In contrast to chapter 16.2.2 "NC Preparation and Interface/PLC (WAIT(BITIF(...)))" on page 660, the procedure from the NC preparation to the PLC is described. That means that the NC preparation sends a signal (via interface) that is received by the PLC. It can then adjust its sequence accordingly.

If the signal is to be sent from the NC preparation, three possible functions can be used:

- BITIF for signals to the interface
- PLC for signals to the flag
- System data for more complex information

The bits of the interface are known as global variables in the PLC and do not have to be declared additionally. The PLC cyclically queries the state of the interface bit or the flag and does thus detect the signal of the NC preparation.

The behavior is different when using system data:

- No effect on the interface
- The PLC can directly query system data (via the function block "MT_SD_RD" described in the documentation "MTX PLC Interface", chapter "System Data")

- System data has to be created additionally in the NC
- System data can contain individual values (e.g. INTEGER, REAL) as well as complex structures

Restrictions

The PLC can only be used to write flags if the machine parameter PLC/Plc is set to IL (= IndraLogic) (MP 2060 00200 to a value of 5).

Relevant CPL functions

- BITIF for signals to the interface
- PLC for signals to the flag
- CPL assignments for values in system data

Relevant machine parameters (MP)

The machine parameter PLC/Plc has to be set to IL (= IndraLogic) (MP 2060 00200 to a value of 5) to allow the PLC to write on flags.

Relevant IF signals

It can be written on all interface inputs with BITIF.

Handling Instruction: Interface

In this example, the NC is to report to the PLC when the program is started. For this purpose, the NC preparation is to set the customer input 2 at the beginning of the program. The PLC then switches on an LED.

☞ Instead of using BITIF or PLC, the input "iCh_ProgRun" can also be queried by the PLC in this example to determine whether a part program is processed.

NC part The preparation starts as soon as the program is started. If the `BITIF` command is reached, the NC preparation sets the respective bit.

```
10 BITIF("iCh_Custom2") = TRUE          ; Setting input
```

Tab. 16-5: Program example: NC program; Setting customer output 2 via BITIF

PLC part The PLC cyclically queries the state of the interface in a function block. As soon as the customer input 2 is enabled, an internal state is switched and the output is enabled for the LED.

```
VAR_OUTPUT                              // Declaration part
LEDOutput : BOOL;
END_VAR

IF iCh_Custom2 THEN                     // Checking input
LEDOutput := TRUE                       // Switching on LED
InterState := 4                         // Changing internal status
END_IF;
```

Tab. 16-6: Program example: PLC program; Copying the laser light barrier to customer output 4

Handling Instruction: Flag

This example has the same requirement than the previous one, but has to use a flag instead of an interface. The flag (RunningState) is created as global variable of type "Byte". The flag address is to be byte 23. Since this flag is also to be used for other signals, it is defined that the value 7 means that the program has started.

```
VAR_GLOBAL                          // Declaration part
RunningState AT %MB23 : BYTE;
END_VAR
```

Tab. 16-7: Program example: PLC program; Copying the laser light barrier to customer output 4

NC part The preparation starts as soon as the program is started. If the `PLC` command is reached, the NC preparation writes the value to the flag. It is important that the same address as in the declaration is used.

Write access to a flag is allowed if the machine parameter PLC/Plc is set to IL (= IndraLogic) (MP 2060 00200 to a value of 5).

```
10 PLC(3, , 23, 1) = 7              ; Setting flag to 7 at byte 32
```

Tab. 16-8: Program example: NC program; Setting the flag via PLC

PLC part The PLC cyclically queries the state of the flag in a function block. As soon as this value is 7, an internal state is switched and the output is enabled for the LED.

```
VAR_OUTPUT                          // Declaration part
LEDOutput : BOOL;
END_VAR

IF RunningState = 7 THEN            // Checking input
LEDOutput := TRUE                   // Switching on LED
InternState := 4                    // Changing internal status
END_IF;
```

Tab. 16-9: Program example :PLC program; Querying the flag

Handling Instructions: System Data

This example is also based on the same requirement as the previous one. Additionally to the processing status, the currently active feed as well as the active direction of rotation of the spindle to the PLC is to be reported. Thus, use system data.

At first, a user-specific system data structure called "StartState_t" is defined with the following elements:

- State : Byte_t
- CurFeed : Float_t
- SpindleDir : Int_t

Due to the hardware structure of modern computers, it is reasonable to define elements with a size of four bytes ("CurFeed" and "SpindleDir") at the beginning of the structure. Thus, access times decrease. The ready structure definition looks as follows in the xsd syntax:

Program:

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

    <xs:element name="CurFeed" type="Float_t">
      <xs:annotation>
        <xs:documentation>current feed</xs:documentation>
      </xs:annotation>
    </xs:element>

    <xs:element name="SpindleDir" type="Int_t">
      <xs:annotation>
        <xs:documentation>current spindle orientation</xs:documentation>
      </xs:annotation>
    </xs:element>

    <xs:element name="State" type="Byte_t">
      <xs:annotation>
        <xs:documentation>NC execution state</xs:documentation>
      </xs:annotation>
    </xs:element>

  </xs:sequence>
</xs:complexType>
```

After the new structure has been created, an SD variable "StartState" of type "StartState_t" can be declared.

Program:

```
<Variable Storage="volatile">
  <Name>StartState</Name>
  <Type>StartState_t</Type>
  <Comment>Control variable for NC after PLC</Comment>
</Variable>
```

NC part The preparation starts as soon as the program is started. CPL commands are used to determine the required data and to write them into the system data structure. To access the system data variable, the CPL keyword `SD.` is used.

```
10 SD.StartState.CurFeed = SDR(5,1,2)          ; Programmed Feed
20 SD.StartState.SpindleDir = SD(12,1)         ; Spindle direction
30 SD.StartState.State = 7                     ; "Running" state
...
```

Tab. 16-10: Program example: NC program; Assigning to system date

PLC part The PLC cyclically queries the state of the system date in a function block using the function block "MT_SD_RD". As soon as the element "State" is 7, an internal state is switched and the output is enabled for the LED. At this moment, the other two elements are also filled with valid values.

An instance of the function block "MT_SD_RD" with the name "SD_READ" as well as a flag called "ReadState" (type byte, initialization 1) had to be created before.

Program:

```
VAR_OUTPUT
  LEDOutput : BOOL;                                        // Declaration
END_VAR

IF ReadState = 1 THEN
  SD_READ(Start:=TRUE, XPath="/StartState/State",
          DestAdr:=ADR(InternState),
          Size:=SIZEOF(InternState));                      // Read SD
  ReadState := 2;                                          // Change status
END_IF;

IF (ReadState = 2) AND (NOT SD_READ.Active) THEN
  IF SD_READ.Error THEN
    ...                                                    // Error handling
  ELSEIF InternState = 7 THEN                              // in State stand 7
    LEDOutput := TRUE;                                     // switch on  LED
    ReadState := 3;                                        // Query next value
  ELSE
    ReadState := 1;                                        // Read SD again
  END_IF;
  SD_READ(Start:=FALSE);                                   // Reset FB
END_IF;

IF ReadState = 3 THEN
  SD_READ(Start:=TRUE, XPath="/StartState/CurFeed",
          DestAdr:=ADR(CurrentFeed),
          Size:=SIZEOF(CurrentFeed));                      // Read SD
  ReadState := 4;                                          // Change status
END_IF;

IF (ReadState = 4) AND (NOT SD_READ.Active) THEN
...
```


The syntax in the programming example corresponds to IndraLogic 1G. At IndraLogic 2G, inputs and outputs of the function block are named differently.

16.2.4 Interface (OPC) and NC Preparation (SD)

Description

Function

The interface (IndraWorks Operation and IndraWorks Engineering) communicates with the NC via OPC. Furthermore, external applications using the OPC interface can be used to query the NC state and to send commands to the NC.

To display the current NC state in the interface, it has to be transferred from the NC. Apart from several status information available in the OPC interface by default (see documentation "MTX OPC Communication"), specific information can be send from the part program.

The system data is used to synchronize between interface and NC preparation (refer to the manual "MTX Machine Parameters", chapter "System Data"). System data is available in the NC as well as in the interface (via OPC).

The NC preparation can easily write on system data by a CPL assignment. The keyword "SD." has to be placed in front of the variable.

An interface of the OPC item "NC.SystemDataEvent" can be subscribed and "NC.SystemData_U" can be evaluated accordingly (see documentation "MTX OPC Communication", chapter "Items"). This calls the application (e.g. a delegate is called). It can determine the value of the system date and change its status.

Relevant CPL functions

System data via CPL is changed as value assignment. The keyword "SD." has to be programmed in front of the system date.

Relevant OPC Items

NC.SystemData_U: The OPC item `NC.SystemData_U,<XPath>` (see "MTX OPC Communication", chapter "System Data, Item NC.SystemData_U") provides the content of the specified system data structure as XML stream.

NC.SystemDataEvent: Via the OPC item `NC.SystemDataEvent,<XPath>` (see "MTX OPC Communication", chapter "System Data, Item NC.SystemDataEvent"), it is possible to subscribe to a change event of a system data structure. A (CallBack) function is always called by the OPC layer if the specified system data changes.

Handling instruction

The NC processes a loop in the part program in the following example. A subroutine containing the actual machining is repetitively called.

Now, the NC preparation has to report the current processing iteration to the interface. Thus, the system date "/PPState" (type Byte_t) is to be used.

NC part At first, the system date has to be created (see "MTX Machine Parameters", chapter "System data"). Since a default type is used, no schema file (*.xsd) has to be created. An "SDDefMTB.XML" file is created in the usrfep in which the system date is declared.

```
...
<Variable Storage="volatile">
<Name>PPState</Name>
<Type>Byte_t</Type>
<Comment>Synchronisation</Comment>
</Variable>
...
```

Tab. 16-11: Declaring the system date "PPState"

Then, restart the MTX to create the system date.

The second step modifies the part program to ensure that the number of the iteration is written to the system date.

```
...
100 FOR NR%=1 TO COUNT%          ; Loop
110 WAIT                         ; Synchronizing with execution
120 SD.PPState = NR%             ; Iteration → System date
N130 P SubProgram(NR%)           ; Subroutine call
140 NEXT I%
...
```

Tab. 16-12: Part program with loop iteration after system date

As soon as the NC preparation block 120 is reached, the system date is changed.



`WAIT` is required in block 110, since multiple loop iterations might have already been processed by the NC preparation before the NC execution reaches the first block in the subroutine. Also refer to chapter 16.2.1 "NC Preparation and NC Execution (WAIT)" on page 659. Alternatively, the NC function `SSD` can also be used for the assignment in block 120. In this case, `WAIT` can be omitted.

Interface part

The steps of the interface are only described here (not listed as program code), since the specific syntax depends on the program code and the environment.

First, the interface has to subscribe the OPC item "NC.SystemDataEvent". Therefore, a group containing a change event (here "NCStateChange") is created. The item "NC.SystemDataEvent" is included in this group.

```
...
// Initialization
Create "NCState" group
Change event "NCStateChange" for "NCState" group
Create "NC.SystemDataEvent" item in "NCState" group
...
```

Tab. 16-13: Preparing SD query via OPC

The OPC item "NC.SystemData_U" is then queried in the event function. The value written by the NC preparation in block 120 is read there.

```
...
// Change event function "NCStateChange"
Query item "NC.SystemData_U,/PPState"
...
```

Tab. 16-14: SD query via OPC

☞ It is strongly recommended NOT to query the OPC item "NC.SystemData_U,/PPState" cyclically. The item is not optimized for that purpose and thus creates a high system load.

16.3 Synchronizing with NC execution

16.3.1 NC Execution and NC preparation

Description

Function

If the part program contains a critical passage with very short blocks (in terms of time), it should be ensured that the prepared block buffer is filled as much as possible before this passage is processed. As a result, there is no break (processing cannot get a new block, since none of the blocks has been completely prepared yet) if block preparation takes more time than block processing.

It may be likewise unfavorable if the MTX starts the execution of a program as soon as the first block is completely prepared. In this case, the preparation buffer is empty.

The MTX allows to transfer a program to the control via buffered NC block specification and to have it executed there. The communication, however, it treated with a lower priority and can be prone to errors. For this reason, the transfer of individual blocks might take considerably longer.

In any of the three application scenarios, execution can be purposefully stopped prior to the critical section, until a sufficient number of blocks has been completely prepared by block preparation. Therefore, the NC function PREP is used.

Restrictions

- The block preparation buffer must at least be as big as the programmed number of blocks to be prepared. It can be configured channel by channel in the machine parameters.

- The function only waits in manual data input mode, in buffered NC block specification mode, and in automatic mode. It does not wait in any of the other operating modes (e.g., single block).
- If WAIT or the program end is found while waiting, the blocks are completely prepared till there. Then, the execution continues.
- It is never waited at reset.
- The PREP command may not be used in an asynchronous subroutine. If it was programmed in an asynchronous subroutine, a runtime error is generated.
- If the buffered NC block specification is used, either an auto start flag or a PREP may not be contained in the first block of a buffered NC block message.
- The velocity imperatively decreased to zero at the end of a block in which PREP was programmed.

Relevant NC functions

The NC function PREP can stop the execution of a block until the programmed number of blocks is prepared.

Relevant machine parameters (MP)

The machine parameter NCO/LookAh/Ch[k]/NofBlkPrep "Total number of blocks for block preparation and interpolation" (MP 7060 00110) configures the memory size of the block preparation buffer. The completely prepared blocks are stored in this memory. Thus, the programmed value of the PREP function may never exceed the value of the machine parameter.

Handling instruction

This example shows a critical passage with 200 complex spline blocks in a part program. It is required for processing that this passage is processed with constant (programmed) path velocity. However, WAIT has to be programmed right before the passage, since a machine state is queried there. It is thus highly probable that the spline blocks have not yet been fully prepared when their execution starts.

To ensure that all 200 blocks are prepared before the passage is executed, `PREP` is programmed.

```
100 WAIT                          ; Synchronizing for machine status
... Querying machine status ...
N150 PREP(200)                    ; Wait till all spline blocks are prepared
N160 G6 F1234
... 200 spline blocks ...         ; Critical passage
```

16.3.2 NC execution between channels (WEV/WREV/SEV, WPV/SPV)

Description

Function

One of the most occurring synchronizations is the synchronization between channels. In general, each channel is a closed work unit. If two working areas of these units superimpose each other or if the working sequences have to be adjusted (e.g. axes have to change channels), synchronization is required.

The MTX provides two different tools to synchronize between channels:

- Bit events (WEV/WREV/SEV/REV) and
- Permanent variables (WPV(E)/SPV(E) or CPL)

Bit events

96 global and system-wide bit events are created in the MTX. These events are the levels ("0" or "1") which can be modified or used with the NC functions `REV` (ResetEvent), `SEV` (SetEvent), `WEV` (WaitForEvent) and `WREV`.

With the `WEV` function, it is waited at runtime (or the continuous block switching is impeded) until a certain bit event occurs. For this purpose, it is checked whether the respective level is "1". If the level is "1" when WEV has been reached, it is not waited. In this case, it is decelerated to a velocity of 0 at block end.

The `WREV` function runs alike. All bit events programmed in the waiting condition were deleted after waiting.

A bit event is triggered (set) at runtime using the `SEV` function. Thus, the level of the programmed event is set to "1" (irrespective whether the value was "0" or "1").

A bit event is deleted (set to the level "0") at runtime using the `REV` function.

If several functions are programmed in one block, they are applied in the following sequence:

1. REV
2. SEV
3. WEV/WREV

With the block `N10 REV(10) WEV(10)` can thus be waited for the edge of event 10.

Permanent variables

Permanent variables of type INTEGER can be accessed under the names `@1` to `@100`. Additionally, further permanent variables can be defined in the MTX. Definable permanent variables are not automatically declared as a component of the system software but have to be declared manually via user entry in the files named "wmhperm.dat" (for MTB-specific data) and "anwperm.dat" (for end user-specific data) (see "MTX Programming Manual", chapter "Variable programming").

Permanent variable are system-wide valid variables with simple types (e.g. INTEGER, REAL, DOUBLE, STRING). Data can be directly written to them at the preparation point in time via CPL assignment or at the execution point in time using `SPV` and `SPVE`.

```
...
100 @MYVAR=10                    ; Preparation point in time
N110 SPV[@MYVAR=20]              ; Execution point in time
...
```

Tab. 16-15: Assigning permanent variables

The NC block execution can be stopped by programming the NC function `WPV` or `WPVE`. A condition for a permanent variable can be specified in this function. The NC execution is stopped until the condition is fulfilled.

Restrictions

The MTX does not monitor whether invalid states occur. That means if two channels mutually wait for a synchronization signal, the NC stands still (deadlock).

Relevant NC functions

WEV/WREV/SEV/REV for bit events

WPV(E)/SPV(E) for permanent variables

Relevant CPL functions

CPL assignments for permanent variables

Handling instruction: bit events

In the following example, the channels 1 and 2 should be synchronized. The x-axis of channel 1 passes the working area of channel 2 when it moves towards the tool change. Thus, channel 1 should wait until channel 2 signals that the path is free. Bit event 42 should stand for "Path free".

Then, channel 1 moves towards the tool change. In this state, channel 2 may not return since a collision might occur. Thus, a second bit event 44 is used for the state "K2 return".

```
...
N100 WEV(42)                    ; Wait for "Path free"
N110 REV(44)                    ; Delete "K2 return"
N120 M6 T4                      ; Tool change
                                ; Critical area was exited
N130 SEV(44)                    ; Signal "K2 return"
...
```

Tab. 16-16: Program for channel 1

```
...
N100 G0 X -50 Y 200             ; Leave collision zone
N110 SEV(42)                    ; Signal "Path free"
...                             ; Machining outside collision zone
N180 WEV(44)                    ; May K2 return?
...                             ; Machining inside collision zone
```

Tab. 16-17: Program for channel 2



A loop can not simply be used in the given example. At an inconvenient sequence, channel 1 could wait for 42 in N100 while channel 2 waits for 44 in N180.

Handling instruction: permanent variables

In this case, there should be the same situation as in chapter "Handling instruction: bit events" on page 670. Now, a permanent variable @CH_SYNC should be used for synchronization. 6 stands for "Path free" and 7 for "K2 return".

The variable has to be created in the first step. For this purpose, the following line has to be added to the file "wmhperm.dat" (in USRFEP or root):

DEF INT @CH_SYNC;

The programs for both channels look as follows:

```
...
N100 WPV[@CH_SYNC=6]          ; Wait for "Path free"
N120 M6 T4                    ; Tool change
                              ; Critical area was exited
N130 SPV[@CH_SYNC=7]          ; Signal "K2 return"
...
```

Tab. 16-18: Program for channel 1

```
...
N100 G0 X -50 Y 200           ; Leave collision zone
N110 SPV[@CH_SYNC=6]          ; Signal "Path free"
...                           ; Machining outside collision zone
N180 WPV[@CH_SYNC=7]          ; May K2 return?
...                           ; Machining inside collision zone
```

Tab. 16-19: Program for channel 2

16.3.3 NC Execution and Interface/PLC (WAITA/WAITO)

Description

Function

Many processes and sequences are controlled in the tool machine by the PLC. It is thus required to synchronize NC and PLC. The interface between the two instances is the PLC interface. This interface is described in detail in the documentation "MTX PLC Interface".

If the execution of the part program (and thus the movement of the mouse) depends on the PLC, NC execution and PLC have to be synchronized.

It is waited with the NC functions `WAITA` and `WAITO` at runtime (active point in time, also refer to chapter "Function" on page 660) until one or several of a maximum of 16 interface signals assumed a specified value.

Depending on the operation, the following can be programmed for several interface signals:

- `WAITA`: "AND operation" of the individual signals

  Wait until **all** interface signals assumed the specified value.
- `WAITO`: "OR operation" of the individual signals

  Wait until **at least one** interface signal assumed the specified value.

☞ If `WAITA` and `WAITO` are programmed in one NC block, block execution is stopped until both conditions have been fulfilled. The WAITO condition is evaluated first.

☞ The PLC can also stop the execution of the part program with the channel interface bit "qCh_FeedHold". However, it is not known in this case where that happens in the program.

Relevant NC functions

WAITA: "AND operation" of the individual signals

WAITO: "OR operation" of the individual signals

Relevant IF signals

Each bit in the interface can be queried using the function BITIF. Thus, WAITA and WAITO can wait for any bits in the interface.

Handling instruction

Two PLC-controlled spindles should exist in an example machine. To process the part program, both spindles have to run with their command speeds.

PLC part

If the spindles are controlled by the PLC, the PLC has to detect when the command speed is reached. It signals it (separate for each spindle) in the channel interface. The command speeds should be stored in the global variables "SP1_SET" and "SP2_SET".

At an NC start, the NC program is automatically started and the spindles start.

```
VAR_OUTPUT                          // Declaration part
SP1_ON: BOOL;                       // Starting bit of spindle 1
SP2_ON: BOOL;                       // Starting bit of spindle 2
END_VAR

IF qCh_NCStart THEN                 // Is NC start pressed?
SP1_ON := TRUE                      // Starting spindle 1
SP2_ON := TRUE                      // Starting spindle 2
qCh_Custom4 := 0                    // Speed not reached
qCh_Custom5 := 0                    // Speed not reached
END_IF;
```

Tab. 16-20: Program example PLC program, Starting spindle at NC start

It is then cyclically checked in a function block whether the command value has been reached and the bits are set in the interface if necessary.

```
VAR_INPUT                                   // Declaration part
SP1_FEED: REAL                              // Current speed Sp 1
SP2_FEED: REAL                              // Current speed Sp 2
END_VAR

IF ABS(SP1_FEED-SP1_SET) < EPSILON THEN     // Checking speed
qCh_Custom4 := 1                            // Spindle 1 running
END_IF;

IF ABS(SP2_FEED-SP2_SET) < EPSILON THEN     // Checking speed
```

```
qCh_Custom5 := 1                                    // Spindle 2 running
END_IF;
```

Tab. 16-21: Program example PLC program, Querying the spindle speed

NC part

As soon as the two bits that display the spindle status are active, the part program should start. For this purpose, the NC function "WAITA" is programmed at the beginning of the program. It checks both bits. Only if both are active, the execution of the part program is continued.

```
... CPL calculations if necessary ...

N10 WAITA[BITIF("qCh_Custom4"), BITIF("qCh_Custom5")]      // Waiting for bits

... Part program:
```

Tab. 16-22: Program example NC program, Waiting for interface bits

16.3.4 NC execution and positions (*STOPA/*STOPO)

Description

Function

If two working areas of axes superimpose each other or if the working sequences have to be adjusted, a synchronization is required. In this chapter, the synchronizations of the NC execution are described based on the axis positions. Thus, the respective functions can stop the execution until the programmed position is reached or exited.



In the broadest sense, it is a channel synchronization as described in chapter 16.3.2 "NC execution between channels (WEV/WREV/SEV, WPV/SPV)" on page 668. In contrast to the use cases listed there, the location in the part program does not have to be known when synchronizing with positions.

The basic mode of operation of these synchronization commands is as follows:

1. *STOPA/*STOPO is programmed in channel A.
2. The conditions specified refer to the (synchronous) channel axes of channel A.
3. In *STOPA/*STOPO, a channel B is specified that is stopped.
4. Channel B may continue running if the conditions of *STOPA/*STOPO are met.

In the following, channel A (to which the axes belong to) is called "control channel" and channel B (that is stopped) "channel to be controlled".

The functions described differ in the type of the monitored coordinate system. For a detailed explanation of the coordinate systems of the MTX, refer to the manual "MTX Functional Description", chapter "Coordinate systems in the channel". The following functions are provided:

- `ASTOPA`/`ASTOPO`: Axis coordinate system (ACS)
- `BSTOPA`/`BSTOPO`: Basic workpiece coordinate system (BCS)
- `WSTOPA`/`WSTOPO`: Workpiece coordinate system (WCS)

One or several conditions to stop the channel can be specified for each channel to be controlled:

- `*STOPA`: As long as **all** conditions are fulfilled, the synchronous motion of the channel to be controlled is stopped (logical AND operation).
- `*STOPO`: As long as **at least one** condition is fulfilled, the synchronous motion of the channel to be controlled is stopped (logical OR operation).

Relevant NC functions

	AND operation	OR operation
Axis coordinate system (ACS)	ASTOPA	ASTOPO
Basic workpiece coordinate system (BCS)	BSTOPA	BSTOPO
Workpiece coordinate system (WCS)	WSTOPA	WSTOPO

Tab. 16-23: Relevant NC functions

Handling instruction

In the following example, a grinding machine machines workpieces on a conveyor belt. The belt continuously moves the parts through the grinding machine in y-direction. The rotating grinding disk is reciprocated by the machine in x-direction.

Where the tool is engaged, travel with constant path velocity. The x-axis was programmed to move further than the workpiece in each direction. During this period (tool not engaged), the belt is to continue moving the workpiece to machine the next grinding path.

The program for the conveyor belt motion runs in channel 1. The length of the workpiece is transferred to a system date as parameter.

```
N10 G0 G90 Y0                       ; On starting position
20 WAIT
30 LENGTH=SD.PartLength             ; Part length from system date
40 IF LENGTH = 0 THEN               ; No new part?
50 GOTO .EXIT                       ; Then complete
60 ENDIF
N70 P GetWorkpiece                  ; Workpiece on belt
N80 Y100                            ; moving into the machine
N90 G91 Y[LENGTH]                   ; Grinding process
N100 Y100                           ; Moving out of the machine
N110 P RemoveWorkpiece              ; Removing workpiece from belt
120 GOTO N10                        ; Loop return
```

Tab. 16-24: Program example: NC program in channel 1; Moving the conveyor belt

The tool is moved in channel 2. It moves reciprocates along the x-axis. To synchronize with the motion in channel 1, the NC function `WSTOPA` is programmed. Whenever the WCS position of X is between 200 and 800 mm, the tool is engaged. The belt is to be at standstill during this period.

```
N10 G0 X0                           ; On starting position
N20 WSTOPA(1,"X">200,"X"<800)       ; Stop condition for channel 1
N30 G1 F800 X1000                   ; move towards (grinding)
N40 X0                              ; return (grinding)
N50 GOTO N30                        ; start again from beginning
```

Tab. 16-25: Program example: NC program in channel 2; Grinding motion and control in channel 1

16.3.5 PLC and NC Execution (Auxiliary Function SSD)

Description

Function

Many processes and sequences are controlled in the tool machine by the PLC. It is thus required to synchronize NC and PLC.

This chapter describes the case opposite to chapter 16.3.3 "NC Execution and Interface/PLC (WAITA/WAITO)" on page 671 The NC execution is supposed to send a message to the PLC to report its position in the part program. The PLC can then react and adapt its sequence accordingly.

The MTX provides two communication options:

- Auxiliary functions for single bit events
- System data for more complex status information

For detailed information on how to apply and use auxiliary functions, refer to the manual "MTX Functional Description Basics", chapter "Auxiliary functions". Thus, it is not explained here in detail. This chapter explains the use of system data as communication medium between PLC and NC execution.

System data is used in the following steps:

1. System data (including type if necessary) have to be created in the NC
2. The NC functions SSD/SSDE assign values to system data at runtime (active point in time)
3. The PLC can directly query system data (via the function block "MT_SD_RD" described in the documentation "MTX PLC Interface", chapter "System Data")

System data can contain individual values (e.g. INTEGER, REAL) and complex structures.

Relevant NC functions

The NC functions SSDE and SSD assign values to system data at runtime (active point in time). They can thus be used to synchronize with the NC execution.

Handling Instructions: System Data

The part program processes a loop in this example. The NC execution has to inform the PLC about its current iteration via the system date "CycleNo" of type "Int_t".

The SD variable has to be declared before. Since it is a standard type, no user-specific type has to be created.

Program:

```
<Variable Storage="volatile">
  <Name>CycleNr</Name>
  <Type>Int_t</Type>
  <Comment>Durchlaufnummer</Comment>
</Variable>
```

NC part

The value of the current iteration is written on the system date using the NC function SSDE within the loop. This function has to be used in order not to assign before runtime (active point in time). Thus, this case differs from the example in chapter "Handling Instructions: System Data" on page 664.

```
...
100 FOR NR%=1 TO COUNT%          ; Loop
N110 SSDE[SD.CycleNr = NR%]      ; Iteration → System date
```

```
N120 P SubProgram(NR%)                ; Subroutine call
130 NEXT I%
...
```

Tab. 16-26: Part program with loop iteration after system date

PLC part

The PLC cyclically queries the state of the system date in a function block using the function block "MT_SD_RD". It can then execute different reactions based on "CycleNo".

An instance of the function block "MT_SD_RD" with the name "SD_READ" as well as a flag called "ReadState" (type byte, initialization 1) had to be created before.

Program:

```
IF ReadState = 1 THEN
  SD_READ(Start:=TRUE, XPath="/CycleNr",
          DestAdr:=ADR(InternCycle),
          Size:=SIZEOF(InternCycle));          // Read SD
  ReadState := 2;                              // Change status
END_IF;

IF (ReadState = 2) AND (NOT SD_READ.Active) THEN
  IF SD_READ.ERROR THEN
    ...                                        // Error handling
  ELSE
    ...                                        // Work with InternCycle
  END_IF;

  SD_READ(Start:=FALSE);                       // Reset function block
  ReadState := 1;                              // Read SD again
```

The syntax in the programming example corresponds to IndraLogic 1G. At IndraLogic 2G, inputs and outputs of the function block are named differently.

16.3.6 Interface (OPC) and NC Execution (SSD/SSDE/SSDQ)

Description

Function

The interface (IndraWorks Operation and IndraWorks Engineering) communicates with the NC via OPC. Furthermore, external applications using the OPC interface can be used to query the NC state and to send commands to the NC.

To display the current NC state in the interface, it has to be transferred from the NC. Apart from several status information available in the OPC interface by default (see documentation "MTX OPC Communication"), specific information can be send from the part program.

System data synchronizes interface and NC execution (cf. "MTX Machine Parameter", chapter "System data"). System data is available in the NC as well as in the interface (via OPC).

SSD and SSDE

In contrast to chapter 16.2.4 "Interface (OPC) and NC Preparation (SD)" on page 665, the NC execution has to change the system date at runtime (active point in time). Therefore, the NC functions `SSD` and `SSDE` are used.

Both functions write a CPL expression into a system date at runtime. `SSDE` accepts any expression calculated during preparation time and assigned at the active point in time. The `SSD` function can only assign simple expressions. However, these are then calculated at the active point in time (also refer to the documentation "MTX Programming Manual", chapter "Writing a System Date at Execution Time").

☞ It is recommended to use SSDE. Since it is already calculated during the preparation, the runtime of the command is shorter.

SSDQ

The more complex NC function `SSDQ` ("writing into a system data queue") provides more comfort for the message transmission. If SSDQ is programmed, it writes the message specified by the user (an integer) at the **execution point in time** into a queue in the system data (an "Int_t" system data array). The queue is used as ring buffer. If writing is done to the last element of the system data array, the function enters the programmed message in the first element of the queue when called the next time. In all other cases, it is written to the subsequent element.

At the same time, the `SSDQ` function modifies a control structure (of type "SSDQCtrl_t") that is also stored in the system data. Thus, the user can determine on which queue element has been written or which messages have been newly received by accessing system data (see "MTX OPC Communication", chapter "System Data"). The control structure consists of the two "Int_t" elements **LastFilled** and **LastGet**. If the function wrote a programmed message into the queue, the **LastFilled** element is set to the index of the modified entry in the queue. This index is one-based.

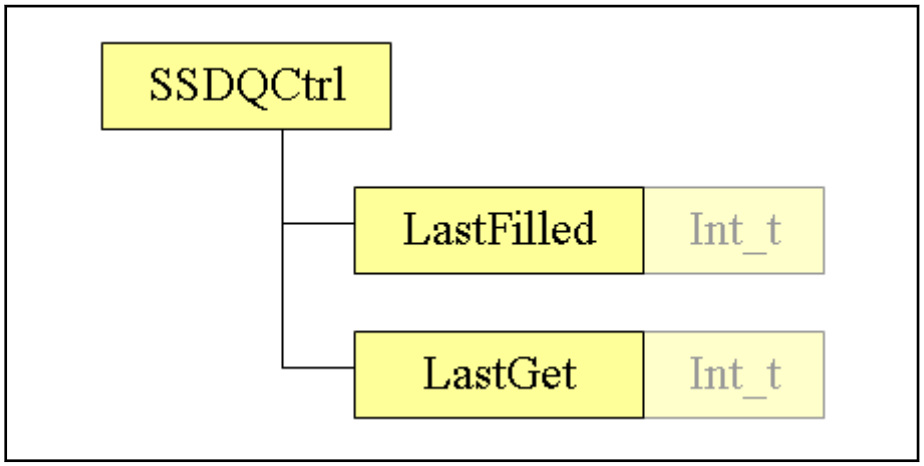


Fig. 16-2: Structure of the SSDQ control structure

While writing, a change event is triggered for the control structure. The queue is modified, but a change event is, however, not triggered. Thus, the user can activate the external application by subscribing to the change event of the control structure. This can, for example, be performed with the OPC item "NC.SystemDataEvent" (see "MTX OPC Communication", chapter "System Data, ").

After the SDDQ function has been executed, the user is responsible for reading the message from the queue. The **LastGet** element in the control structure has to be set to the index of the latest queue element read.

If the user reads the control structure and the values of **LastFilled** and **LastGet** differ, new messages arrived in the queue.

If **LastFilled** is larger than **LastGet**, the new messages are positioned in the entries of the queue from **LastGet**+1 to **LastFilled** (in the order programmed).

If **LastFilled** is smaller than **LastGet**, the ring buffer behavior of the queue is applied. In this case, the new messages are located in **LastGet**+1 up to the end of the queue as well as from the beginning of the queue to **LastFilled** (in the order programmed).

If the user does not succeed in setting **LastGet** in due time, the queue could be full and messages could be lost. For this reason, the SSDQ function

checks whether there are still free elements in the queue (next writing position unequal **LastGet**). If this is not the case, a runtime error is created.

With the **SSDQInit** function, it is possible to initialize a control structure. When programming SSDQInit, **LastGet** and **LastFilled** are set to 1 at the execution time.

Restrictions

SSD

- Only simple CPL expressions can be assigned.

SSDQ

- The queue has to comprise at least three elements.
- The size of the message queue depends on the free memory space of the target.
- If the execution of the programmed motion of a block with SSDQ requires less than one IPO cycle, there is a "jerk".
- As the communication via NCS to Windows is only low priority, there may be delays when reading messages.



As one single query of a system date (e.g. via OPC) requires relatively extensive management efforts, it is recommended to load whole structures or whole fields to the PC in one query.

Relevant NC functions

The MTX provides the following three NC functions to communicate from the NC execution to an interface:

- `SSD`: Assigning simple CPL expressions in system data
- `SSDE`: Assigning any CPL expressions in system data
- `SSDQ`: Write into a system data queue

SSDQ Via SSDQ (see "MTX Programming Manual", chapter "NC Functions with General Language Syntax", subchapter "Writing into a System Data Queue SSDQ), a programmed message is written into a queue in the system data. At the same time, a control structure is modified so that the user can determine the messages newly arrived in the queue. Both, the queue and the control structure can be specified via an XPath. If no XPaths are programmed, the SSDQ tries to use the system date

"/SSDQueue" as queue and

"/SSDQCtrl" as control structure.

SSDQInit: Initialize a control structure via SSDQInit. Both elements of the structure (LastFilled and LastGet) are set to 1. This is to be done before using SSDQ for the first time.

Relevant OPC Items

NC.SystemData_U: The OPC item `NC.SystemData_U,<XPath>` (see "MTX OPC Communication", chapter "System Data, Item NC.SystemData_U") provides the content of the specified system data structure as XML stream.

NC.SystemDataEvent: Via the OPC item `NC.SystemDataEvent,<XPath>` (see "MTX OPC Communication", chapter "System Data, Item NC.SystemDataEvent"), it is possible to subscribe to a change event of a system data structure. A (CallBack) function is always called by the OPC layer if the specified system data changes.

Handling Instruction: SSDE

The NC processes a loop in the part program in the following example. A subroutine containing the actual machining is repetitively called.

Now, the NC preparation has to report the current processing iteration to the interface. Thus, the system date "/PPState" (type Byte_t) is to be used.

NC part

At first, the system date has to be created (see "MTX Machine Parameters", chapter "System data"). Since a default type is used, no schema file (*.xsd) has to be created. An "SDDefMTB.XML" file is created in the usrfep in which the system date is declared.

```
...
<Variable Storage="volatile">
<Name>PPState</Name>
<Type>Byte_t</Type>
<Comment>Synchronisation</Comment>
</Variable>
...
```

Tab. 16-27: Declaring the system date "PPState"

Then, restart the MTX to create the system date.

The second step modifies the part program to ensure that the number of the iteration is written to the system date.

```
...
100 FOR NR%=1 TO COUNT%          ; Loop
N110 SSDE[SD.PPState = NR%]      ; Iteration → System date
N120 P SubProgram(NR%)           ; Subroutine call
130 NEXT I%
...
```

Tab. 16-28: Part program with loop iteration after system date

As soon as the NC execution block 110 is reached, the system date is changed.

Interface part

The steps of the interface are only described here (not listed as program code), since the specific syntax depends on the program code and the environment.

First, the interface has to subscribe the OPC item "NC.SystemDataEvent". Therefore, a group containing a change event (here "NCStateChange") is created. The item "NC.SystemDataEvent" is included in this group.

```
...
// Initialization
Create "NCState" group
Change event "NCStateChange" for "NCState" group
Create "NC.SystemDataEvent" item in "NCState" group
...
```

Tab. 16-29: Preparing SD query via OPC

The OPC item "NC.SystemData_U" is then queried in the event function. The value written by the NC preparation in block 120 is read there.

```
...
// Change event function "NCStateChange"
Query item "NC.SystemData_U,/PPState"
...
```

Tab. 16-30: SD query via OPC

☞ It is strongly recommended NOT to query the OPC item "NC.SystemData_U,/PPState" cyclically. The item is not optimized for that purpose and thus creates a high system load.

Handling Instruction: SSDQ

Applying

MTX

In the MTX, the system data for the queue and the control structure has to be created. The queue is a one-dimensional array of Int_t elements, the control structure looks as shown in fig. 16-2 "Structure of the SSDQ control structure" on page 677.

The user can choose where to create the structures (as root elements or substructures). For detailed information on how to define system data, refer to the manual "MTX Machine Parameters", chapter "System data". In the following example, the structure "/SSDQueue" is created as queue and the structure "/SSDQCtrl" as control structure with number of channel dimensions. These are also the values specified for SSDQ.

Example:

Creating system data structures "/SSDQueue" and "/SSDQCtrl"

In the following excerpt from an XML definition file, the system data for a queue and a control structure is created.

Program:

```
<SDDef>
  <Variable Storage="volatile" Dimension="Channel,30">
    <Name>SSDQueue</Name>
    <Type>Int_t</Type>
    <Comment>Single queue on default name</Comment>
  </Variable>

  <Variable Storage="volatile" Dimension="Channel">
    <Name>SSDQCtrl</Name>
    <Type>SSDQCtrl_t</Type>
    <Comment>Single management structure on default name</Comment>
  </Variable>
</SDDef>
```

If the definition files were supplemented, restart the system to create the system data.

Windows application

It is assumed in this description that the application to be synchronized with the execution of the part program runs under Windows and uses OPC to communicate with the MTX.

After the Windows application has established the OPC communication to the MTX, the change event has to be subscribed to the control structure used.

For this purpose, the OPC item `NC.SystemDataEvent,<XPath>` is to be used.

As soon as the event occurs, the application has to read the control structure and the queue. If the queue is not too large (below 4kB), this is to be performed one access each.



Communication becomes also faster if the items for reading system data have already been created in advance.

Afterwards, the data is analyzed. This is shown in the following pseudo code.

Program:

```
<SDDef>
  <Variable Storage="volatile" Dimension="Channel,30">
    <Name>SSDQueue</Name>
    <Type>Int_t</Type>
```

```
    <Comment>Single queue on default name</Comment>
  </Variable>

  <Variable Storage="volatile" Dimension="Channel">
    <Name>SSDQCtrl</Name>
    <Type>SSDQCtrl_t</Type>
    <Comment>Single management structure on default name</Comment>
  </Variable>
</SDDef>
```

Finally, the application has to write back the modified value of LastGet to the MTX. Thus, the value specifies the index of the last message read.

The following sequence has to be executed by the Windows application:

1. Log in to OPC; establish connection with MTX
2. Subscription to the change event by the control structure used
3. As soon as the event occurs, read control structure and queue
4. Analyze data, read new messages
5. Write back the modified value of "LastGet" to the MTX

Activating

By programming SSDQ, a message is written into the queue and the Windows application can receive and analyze it.

Example:

Excerpt of a part program

N10 SSDQInit

...

N310 X123.32 Y43.20 Z54.11 SSDQ(M27)

...

N410 X128.55 Y40.23 Z23.26 SSDQ(M28)

...

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

0 … 9

2-dimensional delta kinematics with additional rotary axis in the point (RRR)
Activate ... 241
Deactivate ... 241
2/3-axis transformation: Face
Activate ... 392
Apply ... 390
Deactivate ... 392
3-axis transformation LLR ... 337
Activation ... 343
Apply ... 340
3-axis transformation: Cylinder jacket
Activate ... 366
Apply ... 364
Deactivate ... 367
3-axis transformation: LLR
Activate ... 400
Apply ... 239, 397
Deactivate ... 401
3-axis transformation: Scara
Deactivate ... 210
3-axis/4-axis transformation: Scara
Activate ... 210
3/4-axis transformation: Bipod
Activate ... 202
Apply ... 198
Deactivate ... 205
3/4-axis transformation: SCARA
Apply ... 208
4-axis cylinder jacket transformation
Activate ... 374
Apply ... 372
Deactivate ... 375
4-axis transformation - Type 1: SCARA on linear axis ... 219
5-axis cylinder jacket transformation
Apply ... 380
Deactivate ... 383
5-axis transformation - type 1: LLLCB
with linear motion of the rotary axes ... 258
with linear orientation ... 258
with vector orientation ... 257
5-axis transformation - Type 1: LLLCB
Feed ... 262
Tool correction ... 262
5-axis transformation - Type 1: LLLRR ... 256
North pole singularity ... 262
5-axis transformation - Type 1:LLLCB
Axis connection vectors ... 259
5-axis transformation - Type 2: RLLLR ... 273
Ambiguous axis positions ... 279
Linear orientation ... 276
North pole singularity ... 279
Tool correction ... 278
Vector orientation ... 276
5-axis transformation - Type 2: RLLLR (with rotary table) ... 273
5-axis transformation - type 3: RRLLL
North pole singularity ... 288
5-axis transformation - Type 3: RRLLL ... 283
Ambiguous axis positions ... 288
Linear orientation ... 286
Reference position ... 286
Tool correction ... 287
Vector orientation ... 286
with swivel/rotary table ... 283
5-axis transformation - type 4: Cardanic head
Properties ... 295
Reference position ... 296
5-axis transformation - Type 4: Cardanic head
Ambiguous axis positions ... 297
Equator singularity ... 298
Feed ... 299
North pole singularity ... 298
Tool correction ... 297
5-axis transformation - Type 4.1: Cardanic head with linear orientation motion ... 303
Inclined plane ... 304
5-axis transformation - Type 4.1:Cardanic head with linear orientation motion
North pole ... 304
5-axis transformation - Type 5: Less-equipped RRLLL ... 308
Activating ... 313
Ambiguities ... 310
Apply ... 311
Define data of reference position ... 311
Optional transformation parameters ... 309
Reference position ... 311
Tool correction ... 311
Transformation branch ... 310
Value ranges ... 310
5-axis transformation - Type 5: Less-equipped RRLLL with swivel/rotary table ... 308
5-axis transformation - Type 6: Cardanic RLLLR ... 317
Ambiguous axis positions ... 322
Equator singularity ... 323
Linear orientation ... 321
North pole singularity ... 322
Reference position ... 322
Tool correction ... 322
Vector orientation ... 321
5-axis transformation - Type 6:Cardanic RLLLR
Properties ... 320
5-axis transformation - Type 7: Tripod transformation with wobble plate ... 326
Geometric limits ... 328
Reference position ... 327

Tool correction..... 329
5-axis transformation - Type 8: Tripod transformation with wobble plate..... 332
5-axis transformation LLLCA - Type 4/4.1: Cardanic head with vector or linear orientation motion..... 305
5-axis transformation LLLCB - Type 4: Cardanic head with vector orientation motion..... 293
5-axis transformation RRLLL with swivel/rotary table..... 283
5-axis transformation: Cardanic LLLRR
Activate..... 302
Apply..... 299
Deactivate..... 303
5-axis transformation: Cardanic RLLLR
Activate..... 325
Apply..... 323
Deactivate..... 326
5-axis transformation: LLLAB
Activating..... 272
Applying..... 270
Deactivating..... 273
5-axis transformation: LLLRR
Activate..... 266
Apply..... 263
Deactivate..... 267
5-axis transformation: RLLLR
Activate..... 282
Application..... 280
Deactivate..... 283
5-axis transformation: RRLLL
Activating..... 292
Applying..... 289
Deactivating..... 293
5-axis transformation: Tripod with wobble plate LLLLL
Activate..... 331, 335
Apply..... 329
Deactivate..... 332, 337
5-axis transformation: Tripod with wobble plate LLLLL CAx
Apply..... 333
6-axis transformation..... 343
Feed..... 352
Orientation of the TCS - x-coordinate..... 351
Orientation of the TCS - y-coordinate..... 352
Orientation of the TCS - z-coordinate..... 352
Reference position..... 349
Tool correction..... 353
With linear motion of rotary axes..... 345
6-Axis transformation
Type 3033001..... 345
Type 3333301..... 344
Type and position of the rotary axes..... 346
With tensor orientation..... 344
6-axis transformation with linear motion of the rotary axes..... 345
6-axis transformation with tensor orientation.... 344
6-axis transformation: LLLRRR
Activate..... 358
Apply..... 354
Deactivate..... 359

A

Abbreviations..... 9
Abort block..... 42
About this documentation..... 1
Information representation..... 7
Validity of the documentation..... 1
Accuracy..... 111
Accuracy programming..... 111
Accuracy programming..... 111
Action block..... 53
Activate system data queue..... 681
Activate the axis transformation at AT2..... 188
Activate the axis transformation via system date..... 188
Activating axis collision monitoring
Activate..... 102
Activating/deactivating axis collision monitoring
Deactivate..... 103
Activation of the axis transformation at AT1..... 188
Active..... 51
ActTool.K1..... 54
ActTool.K2..... 54
ADJ..... 54
Akima method..... 124
ANSI Z535.6-2006..... 7
Apply system data queue..... 680
MTX..... 680
Windows applications..... 680
Applying axis collision monitoring
Apply..... 101
Approximate B-splines with tolerance specification (compressor)..... 141
Automatic detection of corners and lines..... 149
Axes
Gantry..... 485
Linear modulo axis..... 13
Axis collision monitoring..... 98
Axis configuration in the channel..... 465
Axis transfer..... 465
Axis distance control for digitizing..... 635
Apply..... 643
Monitoring deviations..... 641
Switch off..... 644
Switch on..... 644
Axis kinematic calibration..... 407
Calibration cycle..... 420
Calibration file..... 415
Direct measuring..... 408
Indirect measuring..... 409
Measuring methods..... 408
NC commands..... 411
Selecting the axis transformation..... 407

Axis kinematic calibration - NC commands..... 411
Calculate optimum parameter block..... 412
Forward transformation with specified parameters..... 414
Axis kinematics calibration
Optimizable configuration parameters..... 407
Axis position monitoring..... 103
Activate..... 104
Apply..... 104
Axis transfer..... 465
Activate..... 473
Apply..... 471
Assignment of a operational importance..... 468
Axis assignment..... 465
Axis displays..... 469
Axis group..... 465
Change axes between axis groups..... 465
Deactivate..... 482
Integrate axis..... 467
Remove axis..... 466
Rename axis names..... 468
Axis transformation..... 183
2(3)-axis transformation..... 401
3-axis cylinder jacket transformation..... 359
3-axis transformation LLR..... 337
4-axis cylinder jacket transformation..... 367
4-axis transformation - Type 1: SCARA on linear axis..... 219
4-axis transformation - Type 2: LLLR..... 242
4-axis transformation - Type 3: LLLR..... 248
5-Achs-Transformation - Type 4.1: Cardanic head with linear orientation motion..... 303
5-Achs-Transformation - Type 6: Cardanic RLLLR..... 317
5-Achs-Transformation LLLCA - Type 4/4.1: Cardanic head with vector or linear orientation motion..... 305
5-axis cylinder jacket transformation..... 375
5-axis transformation - Type 1: LLLR..... 256
5-axis transformation - Type 2: RLLLR (with rotary table)..... 273
5-axis transformation - Type 3: RRLLL (with swivel/rotary table) 283
5-axis transformation - Type 5: Less-equipped RRLLL (with swivel/rotary table)..... 308
5-axis transformation - Type 7: Tripod transformation with wobble plate..... 326
5-axis transformation - Type 8: Tripod transformation with wobble plate..... 332
5-axis transformation LLLCB - Type 4: Cardanic head with vector orientation motion..... 293
6-axis transformation..... 343
Axis kinematic calibration..... 407
Bipod axis transformation..... 196
Display active transformation data..... 190
Face transformation..... 383
ID for axis arrangements..... 185
Input variables..... 186
Non-orthogonal 2- and 3-axis transformation..... 210
Output variables..... 186
Points of action..... 186
Principle..... 183
SCARA-axis transformation..... 205
Tabular overview..... 192
Terms..... 184
Type ID..... 184
Axis transformations
Activate and deactivate..... 188
Configure..... 187

B

B-splines
NURBS 118
Backward transformation..... 183
BAP..... 142
Bessel method..... 123
Bipod axis transformation
Positioning the spindle (M19)..... 203
Programming the spindle c-axis tracking..... 203
Bipod axis transformations..... 196
3-axis bipod..... 196
4-axis bipod..... 196
Define data of reference position..... 198
Block pre-run..... 40
Block pre-run applications..... 59
Block pre-run approach..... 43
Configuring the block pre-run..... 47
Examples..... 68
Functional restrictions..... 66
General information about the block pre-run..... 40
Interface signals..... 65
Look ahead..... 65
Spindles..... 62
Switch modal states in the action block..... 58
Switch modal states in the adjustment program..... 58
System data..... 63
Block pre-run applications
Init adjustment program..... 61
Manual input..... 59
Reentry marks RCB marks..... 61
Block pre-run approach
Enabling the block pre-run..... 43
Operations during the computation..... 45
Block switching via high-speed
Block switching on-the-fly..... 76
Block switching via high-speed signal..... 74
Activate..... 77
Apply..... 77
Cancel distance to go..... 76
Parameter HS..... 75

Parameter HSSTOP..... 75
Boundary conditions and dependencies
Entering subroutines..... 66
Feed programming (per rev.)..... 66

C
CalibDirect.cnc..... 421
CalibIndirect.cnc..... 424
Calibrate axis kinematics - NC commands
Backward transformation with current machine parameters..... 415
Calibrating axis kinematics - NC commands
Read current parameters from NC..... 413
Write optimized parameters into NC..... 414
Calibration cycle..... 420
CalibDirect.cnc..... 421
CalibIndirect.cnc..... 424
Direct measuring method..... 420
Indirect measuring method..... 422
Calibration file..... 415
Cardanic 5-axis transformation..... 293
With linear orientation motion..... 303
Cardanic 5-axis transformation LLLCA 305
Cardanic 5-axis transformation RLLLR..... 317
Centripetal parameterization..... 125
Change axes between axis groups..... 465
Channel control..... 31
Chord parameterization..... 125
Circular rounding of corners..... 441
Activate..... 443, 445
Apply..... 443, 444
Deactivate..... 444, 445
RoundEps (RND)..... 442
RoundEps (RNE)..... 442
CLD..... 151
Collision
Axis collision monitoring..... 98
Complaints..... 9
Configure scaling factors for rotary axis velocities..... 83, 86
Configuring the block pre-run
Adjusting modal NC functions..... 54
System data..... 47
Coordinate coupling
Selective additive..... 516
Coupling tables..... 524
Approximate of f(pm) between the data points..... 527
Create spline tables..... 527
Syntax..... 525
Couplings..... 485
Coupling tables..... 524
Gantry axes..... 485
Selective additive coordinate coupling..... 516
Selective additive coordinate coupling with tables..... 520
System axis couplings..... 528
Criticism..... 9
Cubic Hermite splines..... 128
Curve parameter interpolation..... 88
Customer Feedback..... 9
Cylinder jacket transformation..... 359
Plane projection..... 361
Reference position..... 363
Tool correction..... 363
Unwinding..... 359

D
De-Boor boundary condition..... 131
Deactivating..... 92
Definition interval..... 120
Denominator polynomial..... 121
Deselecting an axis transformation..... 189
Digitizing..... 619
Activate..... 629
Apply..... 628
Axis distance control..... 635
Control digitizing..... 627
Deactivate..... 631
Initialize digitizing..... 623
Display active transformation data..... 190
SysAxTrafo_t..... 190
SysAxTrafo1..... 190
SysAxTrafo2..... 190
Distance control for digitizing
Collision detection..... 642
Hole detection..... 642
Distance to end point..... 105
Distance to endpoint
Apply..... 105
Deactivate..... 106
Documentation
Change record..... 1
Drives (axes, spindles)..... 13
Hirth axis..... 15
Linear modulo axis..... 13
Dynamics and velocity control..... 83

E
EBC..... 124
End point, distance to..... 105
Equidistant parameterization..... 125
Euler angle..... 162
Mixed rotation around the respective new coordinate which is fixed in space..... 160
Rotation around coordinates fixed in space..... 159
Rotation around the new coordinate..... 158
Euler angles
Euler angle orientation - variants..... 158
Example
Tools..... 72
Examples
Adjusting the spindle..... 68

Auxiliary functions... 73
Axis configuration and coordinate configuration... 68
Individual axis adjustment with G77... 73
NC functions... 73
Spindle coupling... 70

F
Face transformation... 383
2-axis transformation... 384
3-axis transformation... 384
Automatic branch change... 388
Reference position... 386
Tool length correction... 386
Tool radius correction... 387
Transformation branches... 387
False position method... 124
Feedback... 9
FeedMode... 52
FeedVal... 52
feprom/schemas/sdaxtrf.xsd... 188
Flying measurement (FME)... 603
Commission... 605

G
G33... 435
Gantry axes... 485
Activate... 505
Actual value offset... 487
Apply... 498
Command value coupling... 486
Commissioning monitoring... 496
Compensation motion, signal sequence ... 490
Current monitoring... 497
Deactivate... 506
Determine the correct reference dimensions... 503
Inclined position compensation... 489
Inclined positions... 487
Measuring system... 486
Monitor the reference switch... 496
Monitoring functions ... 496
Reference... 486
Sercos setting... 497
Torque monitoring... 497
General information about the block pre-run
Term definition... 40
General information about the block search
Defining the scope of the block search... 41

H
Handwheel
In workpiece coordinates... 23
Handwheel in workpiece coordinates... 23
Activate... 29
Apply... 27
Coordinate selection in the WCS... 24
Deactivate... 30
Handwheel selection... 24
Hazard warnings... 7
Helpdesk... 683
High-speed signal... 74
Hirth axis... 15
Place programming... 15
Programmed in degrees... 16
Hotline... 683

I
Inclined plane... 162
Information representation
Names and abbreviations... 9
Integrate axis... 467
Intended use
Areas of application... 11
Introduction... 11
Use cases... 11
Interface signals, block pre-run... 65

J
Jog
Setup mode in workpiece coordinates... 19
Jog axes
In workpiece coordinates... 19
Jog in workpiece coordinates
Activate... 22
Apply... 20
Deactivate... 23
Jogging in workpiece coordinates... 19
Continuous jogging... 19
Incremental jogging... 19

L
Laser machining... 457
Activate... 460
Apply... 460
Deactivate... 463
Path velocity-dependent laser power control... 457
Laser power control, path velocity-dependent... 457
Leave and return to contour... 31
Return to contour options... 34
Select approach mode... 35
Select approach point... 34
Leaving and return to contour
Return to contour via PLC... 38
Leaving and returning to contour
Apply... 37
Leave contour via PLC... 37
Less-equipped 5-axis transformation with swiveling/rotary table... 308
Linear modulo axis... 13
Linear orientation... 174
LockTargBlock... 51

LockWarning..... 52
Log file
Close file..... 632
Digitizing..... 632
File name and path..... 632
File structure..... 633
Open file..... 632
Look ahead..... 65

M
Manual mode..... 19
Measure on a contour (MOC)..... 610
Measuring functions..... 593
Axis distance control for digitizing..... 635
Digitizing 619
Flying measurement (FME)..... 603
Log file..... 632
Measure on a contour (MOC)..... 610
Travel against touch probe (G75)..... 593
Measuring on a contour (MOC)
Activation..... 615
Applying..... 615
Deactivation..... 616
Determining measured values via CPL functions..... 614
Determining measured values via system date..... 613
Reactions to the measurement result..... 611
MOC..... 610
Modulo axis
Linear..... 13
Monitoring functions
Gantry..... 496
Monitoring position..... 103
Movement of orientation..... 157

N
NameOfAsup..... 52
NameOfInitAdjPrg..... 53
NC functions with high level language syntax
Adjusting modal NC functions..... 54
NC.SystemData_U..... 666, 678
NC.SystemDataEvent..... 666, 678
Nibble..... 451
Non-orthogonal 2-axis and 3-axis transformation..... 210
Application - non-orthogonal 2-axis transformation in xy-plane..... 216
Application - non-orthogonal 2-axis transformation in xz-plane..... 217
Non-orthogonal 2/3-axis transformation
Activate..... 218
Apply..... 214
Deactivate..... 219

O
OPC items, relevant..... 666, 678
NC.SystemData_U..... 666, 678
NC.SystemDataEvent..... 666, 678
Operating functions..... 19
Handwheel in workpiece coordinates..... 23
Jog in workpiece coordinates..... 19
OptFeedVal..... 52
Orientation..... 166
Orientation motion, linear 174
Orientation tensor..... 164
Motion..... 178
Orientation with the 3x3 orientation tensor..... 164
Rotation speed..... 179
Singularities..... 180
Orientation types..... 166
Linear orientation..... 174
Tensor orientation..... 176
Vector orientation..... 166
Orientation vector..... 162
Orientation motion..... 168
Orientation motion by specifying the end orientation vector..... 168
Orientation motion with programmed rotary axis..... 168
Programming the orientation motion..... 168
Programming with spline..... 167
Rotating speed..... 170
Singularity..... 172

P
Path length parameter interpolation..... 88
Path motion..... 115
Approximate B-splines with tolerance specification..... 141
Automatic detection of corners and lines..... 149
Cubic Hermite splines..... 128
Spline..... 115
Spline programming..... 143
Spline with checkpoint programming (B-spline)..... 135
Spline with data point programming and curvature-continuous transitions..... 130
Path movement
Automatic linear circular spline conversion..... 152
Periodic boundary condition..... 131
Points of action..... 186
AT1..... 186
AT2..... 187
Polar kinematics
Activate..... 406
Apply..... 404
Deactivate..... 406
Polynomial curve..... 88
Position monitoring..... 103
PrecProg..... 112

Program continuation with block pre-run after program abort... 80
Punch... 451
Punch and nibble... 450
Punching and nibbling
Activate... 454
Apply... 453
Deactivate... 454

R

Rapid traverse, reduced... 92
Apply... 93
Deactivate... 93, 94
RCB... 80
RECFILE... 625
RecordBreak... 628
RecordContinue... 628
Recording file
Structure... 632
RecordOff... 627
RecordOn... 627
RecordSet... 625
RecordSetProbe... 624
RECPROBE... 624
RECSEPARATOR... 626
RECTIME... 626
Removing the axis... 466
Rename axis names... 468
Retract vector as system date... 106
Activate... 109
Deactivate... 109
Rotary axis arrangement... 347
Rotation matrix ... 164
Round corners
With splines... 445
Round corners with splines
Interaction between modal and local NC function... 448
Local (non-modal) NC function... 448
Modal NC function... 447
Rounding of corners
Circular... 441
Rounding of corners with splines... 445

S

Safety instructions... 7
Safety/monitoring... 91
Axis position monitoring... 103
Distance to end point... 105
One-dimensional axis collision monitoring... 98
Rapid traverse, reduced... 92
Retract vector as system date... 106
Test feed... 94
Test rapid traverse... 94
Torque reduction... 91
SBC... 124
Scale rotary axis velocities... 83
American setting... 83
Scale rotary axis velocities, configuration... 83
European setting... 83
General setting... 84
Scale rotary axis velocities, programming... 86
Handling instruction... 87
Scaling of rotary axis velocities, configuration
Apply... 85
SCARA on linear axis... 219
SCARA-axis transformation... 205
Selective additive coordinate coupling... 516
Disabling... 519
Enabling... 519
Selective Additive Coordinate Coupling
Apply... 518
Selective additive coordinate coupling with tables... 520
Activation... 524
Create a spline table file... 522
Deactivation... 524
Establish a coupling... 522
Service hotline... 683
Setup mode
Jog in workpiece coordinates... 19
Signal alert symbol... 7
Signal words... 7
Spline
Akima method... 124
Angle condition, absolute... 150
Angle condition, relative... 150
Approximate B-splines with tolerance specification (compressor)... 141
Approximating B-splines with tolerance specification (compressor)... 118
Automatic detection of corners and lines... 149
B-spline... 135
B-spline approximation... 141
B-spline, APT format... 140
B-spline, change node vector... 138
B-spline, change point weights... 139
B-spline, checkpoints ... 135
B-spline, converting a CAD/CAM nurbs sequence into the NC program format of the MTX... 140
B-spline, definition and properties ... 137
B-spline, description for developers... 137
B-spline, MTX format... 140
B-spline, point weights... 136
B-splines with checkpoint programming... 118
Bessel method... 123
C1-continuous cubic splines... 122
C1-continuous cubic splines with data point programming... 117
C2-continuous cubic... 130
C2-continuous cubic splines with data point programming... 117
Coefficient programming... 119
Cubic Hermite splines... 128

Distance condition, absolute..... 150
Distance condition, relative..... 150
False position method..... 124
Integral monomial splines with coefficient programming..... 119
Linear orientation motion as C1-continuous cubic spline..... 126
Motion of the tool orientation..... 126
Motion of the tool orientation as C2-continuous cubic spline 131
Point sequences..... 115
Programming..... 143
Rational monom splines..... 121
Rational monomial splines..... 122
Spline parameter length..... 120
Spline with data point programming and tangential transitions..... 122
Tangent calculation..... 123
Types..... 116
Vector orientation..... 126
With checkpoint programming (B-spline) 135
With coefficient programming..... 116
Spline with data point programming and curvature-continuous transitions..... 130
SSDQ..... 678
SSDQInit..... 678
Suggestions..... 9
Support..... 683
Symbols..... 9
SysAxTrafo_t..... 188
System axis coupling..... 528
Abort methods from SysAxCoupleCmd[i].AbortAction..... 570
Abort statuses in SysAxCoupleSta[i].Abort..... 574
Activation..... 560
Apply..... 532
Control-controlled reaction..... 567
Coupling ratios..... 537
Create, load and remove a spline table file..... 540
Deactivation..... 560
Drive-controlled deactivation..... 567
Drive-controlled retract motion..... 567
Formula coupling..... 541
Master coupling with tables..... 538
Real-time switching..... 562
Slave axis, synchronization mode..... 555
Slave axis, synchronous run monitoring..... 558
Spindle coupling..... 564
States..... 536
Synchronization..... 563
Synchronization, synchronous run monitoring..... 564

T

Taking up a surface 619
Tangent calculation..... 123
Tangential tool guidance..... 180
Tangential tool orientation..... 455
Activation..... 456
Deactivate..... 457
TangTool..... 180
TangToolOri..... 180
Tapping..... 427
Activate..... 434
Apply..... 432
Deactivate..... 434
Retract from tapping (without compensation chuck)..... 428
TargDNr..... 54
TargEd..... 54
Target groups..... 1
Technology..... 427
Circular rounding of corners..... 441
Laser machining..... 457
Punch and nibble..... 450
Round corners with splines..... 445
Tangential tool orientation..... 455
Tapping..... 427
Thread cutting..... 435
Tensor orientation..... 176
Test feed..... 94
Activate..... 96, 97
Deactivate..... 97, 98
Test rapid traverse..... 94
Activate..... 97
Deactivate..... 97, 98
Thread cutting..... 435
Fast retract..... 437
Timeout..... 53
Tool guidance, tangential..... 180
Tool orientation..... 157
Torque reduction..... 91, 92
Activate..... 92
Apply..... 92
Travel against touch probe (G75)..... 593
Commission..... 595
Tripod transformation with wobble plate..... 326
Tripod transformation with wobble plate and additional c-axis for a tangential tool guidance..... 332
Type and position of the rotation axes..... 346
Type ID
Axis transformation..... 184

U

Unintended use..... 12
Consequences, disclaimer..... 11
usrfep/SDDefMTB.xml..... 189

V

Vector orientation..... 166
Velocity control..... 83

Velocity specification for asynchronous linear endless axes..... 13

W

Warnings..... 7
Write into a system data queue
Apply..... 680
Relevant OPC items..... 666, 678
SSDQ..... 678
SSDQInit..... 678

Notes

Notes

rexroth
A Bosch Company

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



R911393316

DOK-MTX***-NC*F*EX*V15-RE05-EN-P