

MTX 14VRS

Functional Description - Basics

Reference Book
R911342626

Edition 06



Title	MTX 14VRS Functional Description - Basics
Type of Documentation	Reference Book
Document Typecode	DOK-MTX***-NC*F*BA*V14-RE06-EN-P
Internal File Reference	RS-8ee2b2420cd132430a6846a501dc0e89-7-en-US-3
Change record	Edition 06, 2020-10 Refer to tab. 1-1 "Change record" on page 1
Copyright	© Bosch Rexroth AG 2020 All rights reserved, also regarding any disposal, exploitation, reproduction, editing, distribution, as well as in the event of applications for industrial property rights.
Liability	The specified data is intended for product description purposes only and shall not be deemed to be a guaranteed characteristic unless expressly stipulated in the contract. All rights are reserved with respect to the content of this documentation and the availability of the product.
Editorial Department	Engineering Automation Systems Control Functions Motion and CNC, HaBu (MiSc/MePe)

Table of Contents

	Page
1 About this documentation.....	1
1.1 Validity of the documentation.....	1
1.2 General information.....	2
1.3 Required and supplementing documentation MTX.....	3
1.3.1 Selection/compilation.....	3
1.3.2 Configuration.....	4
1.3.3 Commissioning.....	4
1.3.4 Operation.....	5
1.3.5 OEM Engineering.....	6
1.3.6 AddOns.....	6
1.4 Information representation.....	7
1.4.1 Safety instructions.....	7
1.4.2 Symbols used.....	7
1.4.3 Names and abbreviations.....	7
1.5 Customer feedback.....	8
2 Important instructions on use.....	9
2.1 Intended use.....	9
2.1.1 Introduction.....	9
2.1.2 Areas of use and application.....	9
2.2 Unintended use.....	10
3 System basics.....	11
3.1 System levels: Drives - NC	11
3.2 Terms.....	12
4 Drives (axes, spindles).....	15
4.1 Overview.....	15
4.1.1 General information.....	15
4.1.2 Sercos.....	15
4.1.3 Drive-integrated safety technology.....	17
4.2 Linear axis	18
4.2.1 Description.....	18
4.2.2 Handling instruction: Applying a Sercos drive as linear axis.....	19
4.3 Rotary axis	21
4.3.1 Description.....	21
4.3.2 Handling instruction: Applies a Sercos drive as rotary axis.....	22
4.4 Endless axis (rotary axis with modulo calculation)	23
4.4.1 Description.....	23
4.4.2 Handling instruction: Applies a Sercos drive as rotary axis with modulo calculation.....	25
4.5 Spindle without Sercos interface.....	27
4.5.1 Description.....	27
4.5.2 Applying.....	30

Table of Contents

	Page
4.6 Spindle with Sercos interface.....	36
4.6.1 Description.....	36
4.6.2 Handling instruction: Applying a Sercos Drive as spindle.....	37
4.6.3 Handling instruction: Spindle reset.....	41
4.7 Spindle/C-axis	41
4.7.1 Description.....	41
4.7.2 Switching from spindle operation to C-Axis operation.....	44
4.7.3 Switching from c-axis operation to spindle operation.....	46
4.7.4 Handling instruction: Applies a Sercos drive as c-axis (spindle and rotary axis functionality).....	47
4.8 Spindle/turret axis with motor encoder	49
4.8.1 Description.....	49
4.8.2 Application.....	53
4.8.3 Synchronization after control startup.....	54
4.8.4 Determining reference point of turret axis.....	56
4.8.5 Examples: Switching between spindle and turret mode.....	56
4.9 Virtual drive.....	58
4.9.1 Description.....	58
4.9.2 Handling instruction: Applying a virtual drive.....	59
5 Operating functions (manual mode)	61
5.1 Override	61
5.1.1 Description.....	61
5.1.2 Handling instruction: Override for auxiliary axes (asynchronous axes).....	63
5.1.3 Handling instruction: Channel feed override.....	64
5.1.4 Handling instruction: Override for spindles.....	65
5.2 Approaching reference point.....	65
5.2.1 Description.....	65
5.2.2 Handling instruction: Reference point coordinates via NC syntax using the function G74.....	69
5.2.3 Handling instruction: Traveling to reference point via NC syntax using the function G74 (Home)....	70
5.2.4 Handling instruction: Reference point traveling with PLC via digital interface.....	71
5.3 Jogging axes.....	73
5.3.1 Description.....	73
5.3.2 Handling instruction: Jogging in axis coordinates.....	75
5.4 Handwheel for Axes.....	78
5.4.1 Description.....	78
5.4.2 Handling instruction: Activating handwheel in axis coordinates.....	80
6 Channel control.....	85
6.1 Reset.....	85
6.1.1 Description.....	85
6.1.2 Handling Instructions: System reset.....	86
6.1.3 Handling instruction: Channel reset.....	88
6.2 Channel initialization program (Init program)	89
6.2.1 Description.....	89
6.2.2 Handling instruction: Init programs.....	94

	Page
6.3 Program selection	95
6.3.1 Description.....	95
6.3.2 Handling Instruction: NC part program selection via PLC.....	98
6.4 Automatic program reselection	99
6.4.1 Description.....	99
6.4.2 Handling instruction: Automatic program reselection.....	99
6.4.3 Automatic program selection after system startup.....	100
6.5 Canceling distance to go.....	101
6.5.1 Description.....	101
6.5.2 Handling instruction: Cancel distance to go.....	102
6.6 Asynchronous subroutines	103
6.6.1 Description.....	103
6.6.2 Handling instruction: Asynchronous subroutines.....	107
6.7 Creating channels.....	113
6.7.1 Description.....	113
6.7.2 Handling instruction: Creating a channel.....	113
6.7.3 Handling instruction: Creating a channel interface.....	113
7 Dynamics and velocity control	115
7.1 Axis jump capability.....	115
7.1.1 Description.....	115
7.1.2 Applying	116
7.1.3 Activating	116
7.1.4 Deactivating	116
7.2 Limiting the path acceleration	116
7.2.1 Description.....	116
7.2.2 Applying.....	117
7.3 Look-ahead	117
7.3.1 Description.....	117
7.3.2 Handling instruction: Look-ahead - Configuring look-ahead.....	119
7.4 Path slope.....	120
7.4.1 Description.....	120
7.4.2 Handling instruction: Path slope.....	122
7.5 Basics of the axis-specific jerk limitation.....	124
7.5.1 Description.....	124
7.5.2 Applying.....	126
7.5.3 Activating.....	126
7.5.4 Deactivating.....	127
7.6 Parameters of the axis-specific jerk limitation "JerkControl, JKC".....	127
7.6.1 Description.....	127
7.6.2 Applying.....	131
7.6.3 Activating.....	131
7.6.4 Deactivating.....	131
7.7 Dynamic limit values	131
7.7.1 Description.....	131
7.7.2 Handling instruction: Changing the axis acceleration.....	132

Table of Contents

	Page
7.8 Feed Group.....	134
7.8.1 Description.....	134
7.8.2 Activating.....	135
7.8.3 Disabling.....	135
8 Safety/monitoring.....	137
8.1 Electronic limit switches (software limit switches).....	137
8.1.1 Description.....	137
8.1.2 Handling instruction: Electronic limit switches - Software limit switches.....	138
9 Accuracy.....	141
9.1 Exact stop.....	141
9.1.1 Description.....	141
9.1.2 Handling instruction: Exact stop in feed mode.....	147
9.2 Radius adjustment for circles.....	149
9.2.1 Description.....	149
9.2.2 Handling instruction: Radius adjustment of circles for center point programming.....	151
9.2.3 Handling instruction: Radius adjustment of circles for radius programming.....	151
10 Axes, coordinates, coordinate systems.....	153
10.1 Overview.....	153
10.2 System axes and system coordinates.....	153
10.3 Coordinate systems in the channel	155
10.3.1 General information.....	155
10.3.2 Overview on coordinate systems	156
10.3.3 Axis coordinate system ACS	156
10.3.4 Machine coordinate system MCS.....	157
10.3.5 Workpiece coordinate system WCS.....	157
10.3.6 Basic workpiece coordinate system BCS.....	157
10.3.7 Program coordinate system (PCS).....	158
10.4 Transformations between coordinate systems	158
10.5 Axes and coordinates in the channel	160
10.6 NC functions and their relation to coordinate systems	166
11 Zero offsets (ZO).....	169
11.1 Basics.....	169
11.1.1 Description.....	169
11.2 Structure of zero offset tables.....	172
11.2.1 Description.....	172
11.2.2 Handling Instruction: Zero Offsets.....	173
12 Coordinate transformations.....	175
12.1 Overview.....	175
12.2 Placement: Workpiece position correction BcsCorr (BCR).....	175

	Page
12.2.1 Description.....	175
12.2.2 Handling Instruction: Workpiece Position Correction BcsCorr (BCR).....	176
12.3 Placement: Inclined Plane G152.1...G159.5.....	177
12.3.1 Description.....	177
12.3.2 Handling Instruction: Placement (coordinate transformation).....	181
12.4 Program Coordinate Offset Trans, (TRS); ATrans, (ATR).....	185
12.4.1 Description.....	185
12.4.2 Handling Instruction: Program Coordinate Offset "Trans""ATrans"	185
13 Input helps.....	187
13.1 Mirroring, scaling, rotating, shifting	187
14 Tool corrections	189
14.1 Overview.....	189
14.2 Basics.....	189
14.2.1 Tool reference points	189
14.2.2 Tool coordinate system	190
14.2.3 Tool coordinate system with active axis transformation.....	190
14.2.4 Applying.....	196
14.2.5 Activating.....	196
14.2.6 Deactivating.....	196
14.3 Specifying tool correction values	197
14.3.1 Description.....	197
14.3.2 D-correction.....	198
14.3.3 Handling instruction: Activating the D-correction.....	199
14.3.4 Handling instruction: Deactivating the D-correction.....	200
14.3.5 Handling instruction: Creating a D-correction table.....	200
14.3.6 External tool correction (ED)	202
14.3.7 Handling instruction: Activating the ED-correction.....	204
14.3.8 Handling instruction: Deactivating the ED-correction.....	205
14.3.9 Handling instruction: Writing the ED-tool correction.....	205
14.3.10 Tool length correction.....	205
14.3.11 Handling instruction: Activating the tool length correction (G47/G48).....	208
14.3.12 Handling instruction: Deactivating the tool length correction (G47/G48).....	211
14.3.13 Handling instruction: Applying the tool length correction (G47/G48).....	212
14.3.14 Tool radius correction	213
14.3.15 Handling instruction: Activating the 3D tool radius correction (G141/G142).....	242
14.3.16 Handling instruction: Deactivating the 3D tool radius correction (G141/G142).....	245
15 Inch/metric switching	247
15.1 Basics.....	247
15.1.1 Description.....	247
15.1.2 Handling instruction: Inch/metric switching in the manual data input or automatic mode.....	249
15.1.3 Handling instruction: Inch/metric switching to jog mode.....	249
15.1.4 Handling instruction: Inch/metric switching of the user interface.....	249

Table of Contents

	Page
16 Spindle functions	251
16.1 Overview.....	251
16.2 Channel spindles and channel spindle groups.....	252
16.2.1 Description.....	252
16.3 Assigning system spindles to channel spindles.....	253
16.3.1 Description.....	253
16.3.2 Applying.....	255
16.4 Assigning channel spindles to spindle groups.....	255
16.4.1 Description.....	255
16.5 Motion functions.....	257
16.5.1 Description.....	257
16.6 Gear function.....	258
16.6.1 Description.....	258
16.7 Path motions and spindle functions.....	260
16.7.1 Description.....	260
16.7.2 Applying.....	264
16.7.3 Activating.....	264
16.7.4 Disabling.....	264
16.8 Spindle speed specification.....	265
16.8.1 Description.....	265
16.9 Programming system spindles.....	266
16.9.1 Description.....	266
16.10 Actuating the system spindle via interface.....	268
16.10.1 Description.....	268
16.11 Motion management of system spindles.....	269
16.11.1 Description.....	269
16.11.2 Activating reservation in the channel.....	269
16.11.3 Enabling reservation of a "spindle in the channel".....	270
16.11.4 Transferring a reserved spindle to another channel.....	270
16.11.5 Conditional spindle enabling.....	271
16.11.6 Applying a conditionally enabled spindle using a different channel.....	271
16.11.7 Switching off spindle motion management for specified channels.....	271
16.12 Special spindle functions.....	271
16.12.1 Main spindle switching	271
16.12.2 Spindle functions with position interface	272
16.12.3 Position-synchronous spindle operation	273
16.13 Gear shifting with "spindle spinning"	280
16.13.1 Description.....	280
16.13.2 Example.....	282
16.13.3 Applying.....	283
16.13.4 Enabling/disabling.....	284
16.14 Gear shifting with "winding switching".....	284
16.14.1 Description.....	284
16.14.2 Applying.....	286
16.14.3 Compatibility	286
16.15 Tool-specific limitations.....	286

	Page
16.15.1 Description.....	286
16.15.2 Applying.....	290
16.15.3 Activating.....	290
16.15.4 Deactivating.....	290
17 Auxiliary functions.....	293
17.1 Basics.....	293
17.2 Channel-dependent and channel-independent auxiliary functions.....	293
17.2.1 Channel-independent auxiliary function.....	293
17.2.2 Channel-dependent auxiliary function.....	293
17.3 Bit-coded auxiliary functions.....	294
17.4 BCD, integer, float and double-coded Auxiliary Functions.....	294
17.5 Output response of auxiliary functions.....	295
17.6 Auxiliary functions (AUXF) – Basic parameters -.....	295
17.6.1 Auxiliary function basics.....	295
17.6.2 Overview.....	297
17.6.3 Auxiliary class definition (ClassDef).....	298
17.6.4 Acknowledgement behavior for bit-coded auxiliary functions (BitAckn)	309
17.6.5 Auxiliary function group definition (GrDef).....	312
17.6.6 Global output behavior (OutBehav)	313
17.6.7 Number of usable auxiliary functions (NofAuxFunc).....	314
17.6.8 Addresses of auxiliary functions in the PLC interface (Plclf).....	319
17.6.9 Defining a new auxiliary function.....	324
17.6.10 Parameterizing example: Auxiliary functions.....	327
18 Service and support.....	335
Glossary.....	337
Index.....	339

1 About this documentation

Editions of this documentation

Edition	Release date	Note
01	2014-03	First edition
02	2014-10	Customer-specific definition of the spindle names Extending the system limits of MTX
03	2015-10	Drive-integrated safety technology
04	2017-01	Chapters "5.3 Jogging axes", "5.4 Handwheel for axes", "6 Channel control", "14 Tool corrections" and "16.12.3 Position-synchronous spindle operation"
05	2020-02	New: chapter "Relevant system data" on page 37 , chapter "Parameter <AdditionalNCBlock-Mem> in the startup.xml file" on page 118 Revised: chapter 6.6 "Asynchronous subroutines " on page 103 Product backlog item 151170: Additional function SPHOME supplemented (chapter 4.8 " Spindle/ turret axis with motor encoder " on page 49) Revised: chapter 9.2 "Radius adjustment for circles" on page 149
06	2020-10	Revised: chapter 6 "Channel control" on page 85 , Defdb00206453 resolved Revised: chapter 9 "Accuracy" on page 141 , Defdb00206453 resolved

Tab. 1-1: Change record

1.1 Validity of the documentation

Purpose	This documentation describes the functions of the 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.

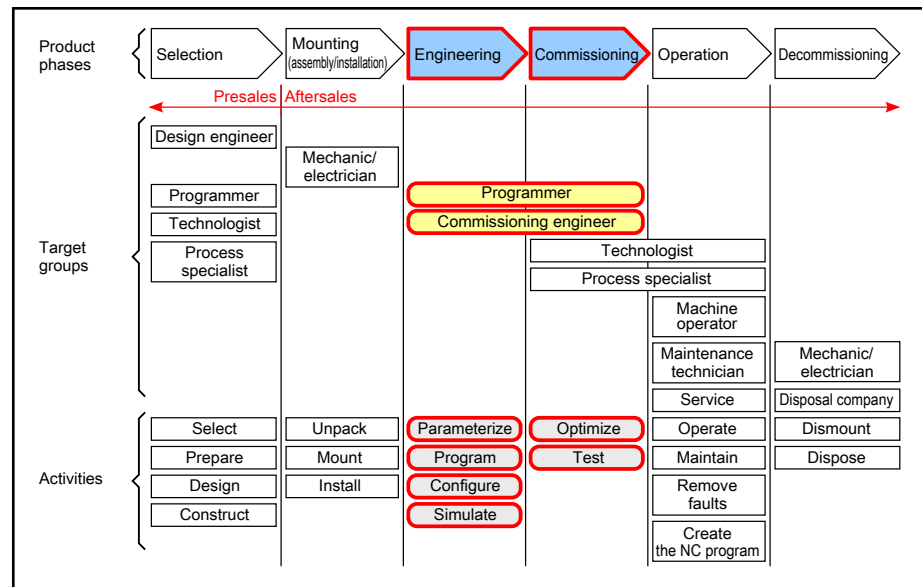


Fig. 1-1: Assigning this documentation to the target groups, product phases and target group activities

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:

- Functional description: "IndraMotion MTX Functional Description, Extension" and "IndraMotion MTX Functional Description, Special Functions"
- Machine parameters: "MTX Machine Parameters"
- NC and CPL functions: "MTX Programming Manual"
- Interface signals: "MTX PLC Interface"

Furthermore, have the drive documentation at hand.

You require good knowledge of the

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

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

CAUTION

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

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

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

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

1.3 Required and supplementing documentation MTX

1.3.1 Selection/compilation

Documentation titles with type codes and part numbers

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

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

1.3.2 Configuration

Documentation titles with type codes and part numbers

MTX 14VRS Machine Parameters DOK-MTX***-MA*PAR**V14-RERS-EN-P, R911342630 This documentation describes structure and the adjustment of available parameters of the MTX. It also includes the functions of the NC configurator and its operation.
MTX 14VRS PLC interface DOK-MTX***-PLC*INT*V14-PRRS-EN-P, R911342622 This documentation describes the interface signals and the program function blocks for the integrated PLC.
MTX 14VRS Functional Description - Basics DOK-MTX***-NC*F*BA*V14-RERS-EN-P, R911342626 This documentation describes the basic functions of the MTX. The basic commissioning steps and the control functions are provided as description and handling instruction.
MTX 14VRS Functional Description - Extension DOK-MTX***-NC*F*EX*V14-RERS-EN-P, R911342628 This documentation describes the extended functions of the MTX. The basic commissioning steps and the control functions are provided as description and handling instruction.
MTX 14VRS Functional Description - Special Functions DOK-MTX***-NC*F*SP*V14-RERS-EN-P, R911342624 This documentation describes the special functions of the MTX. The basic commissioning steps and the control functions are provided as description and handling instruction.
MTX Free Form Surface Milling DOK-MTX***-FREEFORM***-APRS-EN-P, R911341435 This documentation describes the free form surface milling process with the MTX control. CNC programs generated by a CAD/CAM system are used as basis for the entire process. Overview over the following topics: Description of the MTX or CNC function for freeform surface milling, strategy during the NC parameterization process, boundary conditions when generating CNC programs with CAM software.
MTX Conversion of MTX Projects DOK-MTX***-PROCONV****-PRRS-EN-P, R911342484 This documentation provides support during the conversion of MTX 1.x projects to MTX 2G. This project planning manual provides information on how to convert a project and identifies potential difficulties during the conversion.

Tab. 1-3: MTX documentation overview - Configuring

1.3.3 Commissioning

Documentation titles with type codes and part numbers

IndraWorks 13VRS Simulation DOK-IWORKS-SIMU****V13-APRS-EN-P, R911339256 This documentation describes the functions of the simulation components View3D, virtual control panel, virtual control and its operation in IndraWorks.
MTX 14VRS Commissioning DOK-MTX***-STARTUP*V14-CORS-EN-P, R911342620 This documentation describes the commissioning of the MTX control. Apart from a complete overview, commissioning and configuration of the axes and the user interface as well as the PLC data are described.

IndraWorks 14VRS Basis Libraries IndraLogic 2G DOK-IL*2G*-BASLIB**V14-LIRS-EN-P, R911343920 This documentation describes the system-comprehensive PLC libraries.
IndraWorks 14VRS Field Buses DOK-IWORKS-FB*****V14-APRS-EN-P, R911341485 This documentation describes the field bus and local periphery connections supported by the XLC, MLC and MTX systems. The focus of this documentation lies in the configuration, parameterization, commissioning and diagnostics of different periphery connections. It is the basis for the online help.
IndraWorks 14VRS WinStudio 7.4 DOK-IWORKS-WINSTUD*V14-APRS-EN-P, R911341585 This "User Manual and Technical Reference Book" supports the user in achieving the best results with the "Rexroth WinStudio"™ software. The document provides technical information and step-by-step instructions to create web-enabled HMI/SCADA programs.
IndraWorks 14VRS Software Installation DOK-IWORKS-SOFTINS*V14-CORS-EN-P, R911344286 This documentation describes the IndraWorks installation.
IndraWorks 14VRS Engineering DOK-IWORKS-ENGINEE*V14-APRS-EN-P, R911343566 This documentation describes the use of IndraWorks in which the engineering tools are integrated. It includes instructions on how to work with IndraWorks and how to operate the oscilloscope function.
IndraWorks 12VRS FDT Container DOK-IWORKS-FDT*CON*V12-APRS-EN-P, R911334398 This documentation describes the IndraWorks FDT Container functionality. It includes the activation of the functionality in the project and working with DTMs.
IndraWorks 14VRS PLC Programming System IndraLogic 2G DOK-IWORKS-IL2GPRO*V14-APRS-EN-P, R911343571 This documentation describes the PLC programming tool IndraLogic 2G and its use. The documentation includes the basic use, first steps, visualization, menu items and editors.
IndraWorks 14VRS HMI DOK-IWORKS-HMI*****V14-APRS-EN-P, R911343569 This documentation describes the functions, configuration and operation of the user interfaces IndraWorks HMI Engineering and IndraWorks HMI Operation.

Tab. 1-4: MTX documentation overview - Commissioning

1.3.4 Operation

Documentation titles with type codes and part numbers

MTX 09VRS Shop Floor Programming Turning and Milling DOK-MTX***-SF*PROG*V09-AWxx-EN-P, R911324377 This documentation describes the operation and programming of the graphic NC programming of turning and milling as well as the workpiece simulation.
MTX 12VRS Block Pre-Run DOK-MTX***-BLK*RUN*V12-APRS-EN-P, R911334379 This documentation explains to the machine manufacturer how to setup the "Block pre-run" function at the machine for the end user.

MTX 14VRS Programming Manual

DOK-MTX***-NC**PRO*V14-RERS-EN-P, R911342634

This documentation describes the standard programming of the MTX control. Apart from the basics of NC programming, the use of NC functions according to DIN 66025 as well as the NC functions with high-level language syntax and CPL functions are described.

MTX 14VRS Standard NC Operation

DOK-MTX***-NC*OP***V14-APRS-EN-P, R911342632

This documentation describes the operation of the standard user interface of the NC control of the MTX. It includes the operation of the interface, the NC program development as well as the tool management.

MTX 14VRS Multitouch

DOK-MTX***-MULTI***V14-APRS-EN-P, R911384735

The Multitouch user interface of the NC control MTX is described in this documentation.

MTX 14VRS Standard NC Cycles

DOK-MTX***-NC*CYC**V14-PRRS-EN-P, R911342638

This documentation describes the application of the standard cycles of the different technologies for the MTX control.

MTX 14VRS NC Simulation

DOK-MTX***-NC*SIM**V14-APRS-EN-P, R911379629

This documentation describes the NC simulation for the MTX control.

xx

Corresponding edition

*Tab. 1-5:**MTX documentation overview - Operating*

1.3.5 OEM Engineering

Documentation titles with type codes and part numbers

MTX 13VRS Automation Interface

DOK-MTX***-AUT*INT*V13-APRS-EN-P, R911337274

This documentation describes the script-based access to IndraWorks project data via the interface of the Automation Interface. Different objects including code examples are described. The Automation Builder is also described in this manual.

MTX 14VRS OPC Communication

DOK-MTX***-OPC*COM*V14-PRRS-EN-P, R911342636

This documentation describes the syntax and the structure of the items for the communication with Bosch Rexroth devices.

IndraWorks OPC UA Communication

DOK-IWORKS-OPC*UA*****-APRS-EN-P, R911379309

This documentation describes the OPC UA communication of the MLC and MTX control systems.

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

1.3.6 AddOns

Documentation titles with type codes and part numbers

MTX 11VRS Action Recorder

DOK-MTX***-ACR*****V11-APxx-EN-P, R911329943

This documentation describes the MTX action recorder. It includes the installation and commissioning as well as interface signals, application and operation.

MTX 14VRS Efficiency Workbench MTX cta, MTX ega

DOK-MTX***-EWB*****V14-APRS-EN-P, R911343916

This documentation describes the mode of operation and the area of application of the analysis tool MTX cta and MTX ega.

MTX Remote Condition Monitoring

DOK-MTX***-RCM*****V01-APRS-EN-P, R911334383

This documentation describes the operation of the Remote Condition Monitoring System.

MTX visIREC User Documentation

DOK-MTX***-VISIREC*V01-APRS-EN-P, R911344242

This documentation describes the analysis tool visIREC used to optimize the free form surface milling process. 2D or 4D display of path-related data. 2D or 4D display of coordinate-related data. Analysis of critical areas (path and orientation deviation). Comparing the programmed NC blocks to the interpolated NC blocks.

xx

Corresponding edition

Tab. 1-7:

MTX documentation overview - AddOns

1.4 Information representation

1.4.1 Safety instructions

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

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

The signal graphics (warning triangle with exclamation mark), added in front of the signal words Danger, Warning and Caution refers to hazards to persons.

CAUTION

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

NOTICE

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

1.4.2 Symbols used

Note Notes are represented as follows:



This is a note for the user.

Tip

Tips are represented as follows:



This is a tip for the user.

1.4.3 Names and abbreviations

Term	Explanation
IWE	IndraWorks Engineering
IWO	IndraWorks Operation
OWG	Optical waveguide

NC	Numerical Control
OEM	Original Equipment Manufacturer
Profibus	Communication connection
Sercos	Communication connection
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 use shall solely bear all risks from unintended use of the products.

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

- Anybody handling Bosch Rexroth products in any way is obliged to read and consent to the relevant safety instructions and the intended use.
- Hardware products may not be altered and have to remain in their original state; i.e. no structural changes are permitted. It is not permitted to decompile software products or alter source codes.
- Do not install damaged or defective products or use them in operation.
- It has to be ensured that the products have been installed as described in the relevant documentation.



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

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

2.1.2 Areas of use and application

The MTX control is used to

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

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



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

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

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

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

2.2 Unintended use

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

The CNC system MTX must not be used if ...

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

3 System basics

3.1 System levels: Drives - NC

Part of the task of the "Drives - NC" system (see the following figure) is that one or more drives move to a position defined by a point in the Cartesian space (usually in a right-handed, right-angled coordinate system). The Cartesian point crosses different levels in which it is calculated back to the drive positions. The levels are located on NC and drive level. The **NC level** differentiates whether the point can be based on the machine axes (axis level) or if it represents a point in the three-dimensional space (coordinate level) irrespective of the machine axis alignment. On the **drive level**, the point "transformed" by the NC is transmitted as drive command positions to the relevant drives via a Sercos telegram. The drives put the axis body into the accurate position to ensure that the point is reached in the Cartesian space.

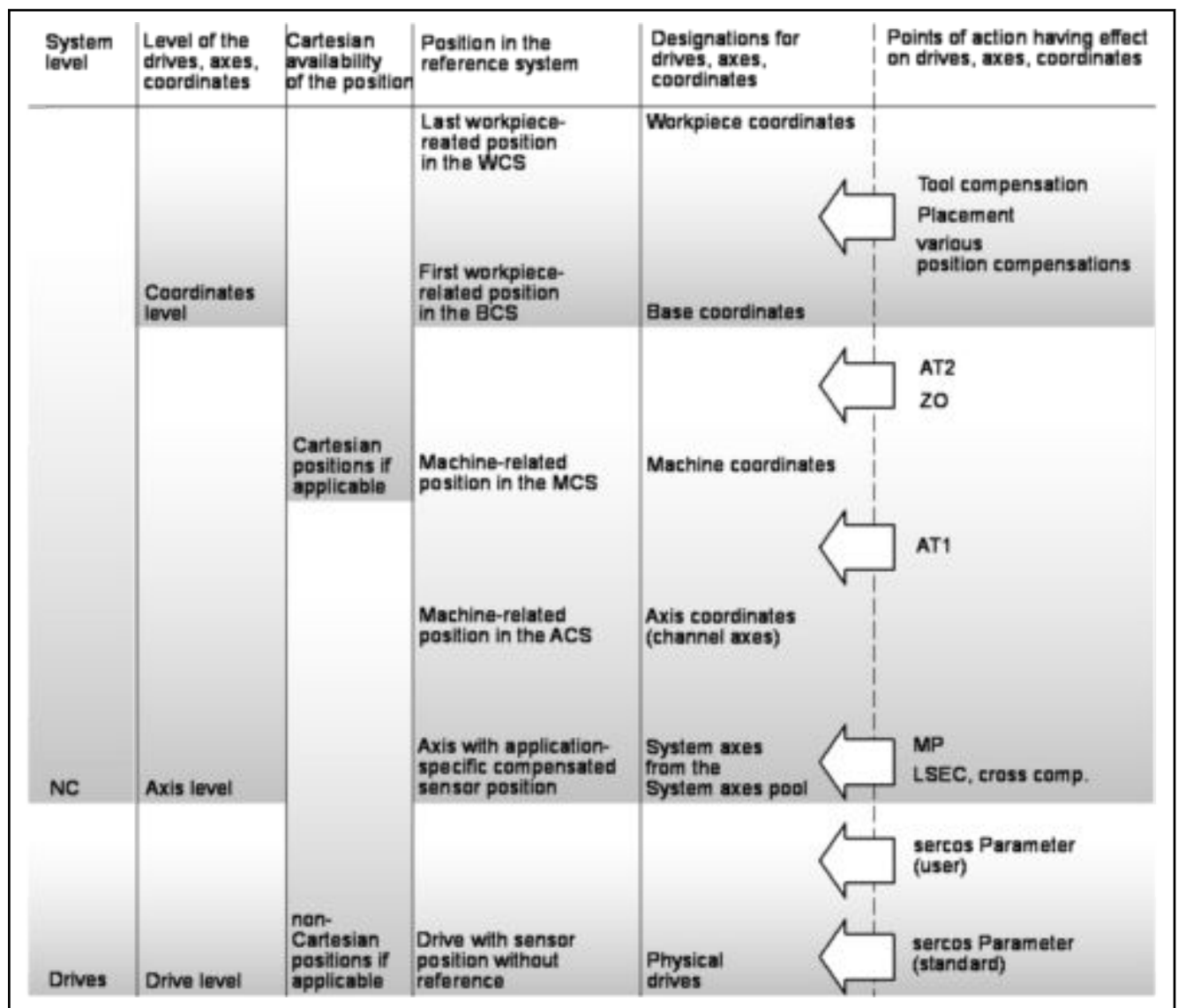


Fig. 3-1: System: Drives-NC

Upon transition and within the **system levels**, various offsets, transformations (at different points of action) and corrections are active which settle the Cartesian point on the drive position.

3.2 Terms



Detailed explanations on the following terms are provided in the next chapters.

Drives: In a machine, the drives are assigned to the physical machine axes. It is not significant on the drive level whether several drives with their physical machine axes are later combined in a "logic axis" (e.g. parallel axes), since this is required by the kinematics of the machine.

By default, the drives are only preconfigured at their minimum and thus do not have a reference to the machine.

On the drive level, user-specific Sercos parameters now define the special characteristics required to specify the drive for the physical machine axis. This includes information on the measuring system responsible for positioning the axis body that moves the drive.

A reference to the machine is only established after the drives have been defined as system axes from the **system axes pool** (pool of all possible drives for this machine). The machine parameters (MP) apply this definition. In this phase, the drives become axes and can be used by the NC.

Axes: A drive has to be defined as axis to "exist" for the NC. Due to the definition of the system axes in the machine parameters, the axis is provided with a name or a number used to control it. The geometric position in the machine and the various kinematic and dynamic parameters are specified in the machine parameters. For the NC, the machine exists first with its physical axes. Thus, an axis position can be approached at the lowest level in the axis coordinate system (ACS). This can be a Cartesian point. It does not have to be one, since the axes in this phase do not have to be rectangular to each other.

After an axis transformation (AT) has been implemented at this position (point of action 1), non-Cartesian axes become Cartesian machine coordinates in the machine coordinate system (MCS). Even though the axes move at an inclined angle towards each other, the NC can approach a defined point with transformed machine axes in the Cartesian machine coordinate system (MCS). This point is defined by the three axis coordinates X, Y and Z.

In order not to approach this point any longer via the axis coordinates x, y and z, but in the Cartesian space via an orientation with angle specification and linear motion in x, y and z instead (tool in space oriented at an inclined angle), another axis transformation is required (point of action 2). On this level, no axes, but "coordinates" are moved.

With a simple machine, at which all axes are rectangular to each other and no tool orientation is at an inclined angle in space (that means no AT1 and/or no AT2), the points at the workpiece are approached in x-, y-, z-axis coordinates of the ACS.

Coordinates: A coordinate mirrors the transformed (with AT2) direction of one or several machine axes. With an active axis transformation, the motion in a coordinate is no longer identical to the direction of a physical axis at the machine. The NC converts the Cartesian position of the coordinate to the physical axis position of one or several axes required to reach this position. The drive converts the physical axis position to the required positions of all involved drives. There are coordinates with active axis transformation AT1 in the MCS and coordinates with active axis transformation AT2 in the basic workpiece coordinate system (BCS).

Channels:	Specified axes and/or coordinates from the system axis pool are combined in one channel if there is a specified type of machining purpose (such as machining with synchronous axes). Axes and/or coordinates can be lent from this channel to other channels. By changing the channel of an axis/coordinate, a point in the Cartesian space can get a different relative position if there is another reference system.
Coordinate systems:	Coordinate systems determine reference points and reference directions. The machine zero point is the reference point at a machine for example. The position of the coordinate system determines the positive/negative travel direction of the axes/coordinates. Each offset, transformation and correction refers to specified coordinate systems. The reference position of all axes in the transformation is for example related to the BCS (first workpiece coordinate system). Thus, the position in space of a point in the Cartesian space is fixed, but its relative position changes according to the reference coordinate system when a function (e.g. offset, transformation) changes the coordinate system. The main coordinate systems are: • ACS (Axis Coordinate System) • MCS (Machine Coordinate System) • BCS (Basic Coordinate System) • WCS (Workpiece Coordinate System)
Points of action AT1 and AT2:	Each axis transformation acts at a defined position to convert position values of a reference system to position values of another reference system. Therefore, the NC provides two points of action (1 and 2). Point of action 1 "active axis transformation (AT1)" converts non-Cartesian axis positions into the ACS to Cartesian axis positions in the MCS. Point of action "active axis transformation (AT2)" converts Cartesian axis positions into the MCS to Cartesian axis positions in the BCS. The NC automatically selects the point of action when an axis transformation is selected.
Axis transformations:	An axis transformation is responsible for the kinematics of a machine. This is always required if the position of the physical axes does not directly indicate a Cartesian point in space (e.g. axes are not rectangular to each other or the longitudinal tool axis is not rectangular to the workpiece coordinate system). Use the axis transformation to program a point in the Cartesian space without computing the actual position of one or more physical axes that would be required to approach the Cartesian point. Each machine requiring a transformation of position data, is provided with a customized axis transformation. The axis transformation is explicitly switched on and off by programming.
Coordinate transformations on axis and coordinate level:	Apart from the axis transformations, there are various reference point offsets (coordinate transformations) for a point in the Cartesian space on axis and coordinate level. On the NC level, the zero point offsets (ZO) act on axis positions in the MCS. As the ZO acts on axis positions in the MCS, this offset is also transformed at the point of action 2 at a subsequent axis transformation. Thus, an active ZO might not be traceable at an active axis transformation.

Functions, such as placement (inclined plane), correction offsets and work-piece corrections are Cartesian offsets on coordinate positions in the BCS and WCS.

Compensations on NC level: Lead screw error compensation (LSEC),

Cross compensation (CCOMP):

These compensations are effective on NC level. The drives or the system axes are subject to a certain purpose-oriented correction to compensate inaccuracies in the mechanical system (gear, spindle pitch) for example. Compensate by correcting at a specified position using the correction size previously determined for this position. Thus, the drive position - without errors due to corrections in the NC - exactly represents the point in the Cartesian space.

Asynchronous axes: An axis that is not assigned to any channel is called "asynchronous axis". An asynchronous axis can be controlled from each channel if its axis mode is set to program mode in the axis interface (interface signal "qAx_OpModeSel"). Asynchronous means that the motion of an asynchronous axis is started block-synchronously, but the specified distance is traveled independently without synchronizing with other motions. Asynchronous axes are very limited in their functionality. Only the target position, the axis feed and - if applicable - the SHAPE order of the axis can be parameterized in the part program. Besides, only the NC functions "Approach to reference point" G74 (Home) and "Positive Stop" FsMove, FsTorque and FsReset have an effect on the asynchronous axes.

Set the programming unit (mm/inch) for asynchronous linear axes globally via a configuration date (NCO/CorrUnit/AsynchrAxUnit). The unit can be switched block-locally via the programming attributes INCH(...) and MM(...).

Asynchronous rotary axes are programmed in degrees.

Due to the axis transfer commands GAX(...) or WAX(...), an asynchronous axis can be applied to a channel. Thus, this axis becomes a regular channel axis.

Vice versa, remove a channel axis from a channel with the RAX(...) command. This axis is now an asynchronous axis.

4 Drives (axes, spindles)

4.1 Overview

4.1.1 General information

The MTX distinguishes axes according to their movement type:

- Linear axis
- Rotary axis
- Endless axis (rotary axis with modulo calculation)
- Linear modulo axis
- Hirth axis
- Spindles
- Spindle/C-axis
- Spindle/turret axis
- Virtual drive

and their movement programming:

- Synchronous axes (channel axes)
- Asynchronous axes (auxiliary axes)

4.1.2 Sercos

Data exchange

The cyclic data exchange between the NC and the drives is executed via the master data telegram and the drive telegram.

- The drive telegram is configured via the Sercos parameter S-0-1050.1.6.

The ident number S-0-0135 has to be entered first in S-0-1050.1.6. Additionally, a maximum of 10 Sercos ident numbers can be entered. These identification numbers may require a maximum of 32 bytes of memory.

- The master data telegram is configured via S-0-1050.0.6.

The ident number S-0-0134 has to be entered first in S-0-1050.0.6. Additionally, a maximum of 8 sercos ident numbers can be entered. These identification numbers may require a maximum of 32 bytes of memory.

Example: S-0-1050.1.6 = (S-0-135, S-0-51, S-0-189, S-0-84, S-0-144)

S-0-051 "Actual position value encoder 1" and S-0-189 "Lag distance " are assigned to 4 bytes each.

S-0-084 "Torque/actual force value " and S-0-144 "Signal status word " are assigned to 2 bytes each.

The drive telegram contains 4 ident numbers. They are assigned to 12 bytes in total.

S-0-1050.0.6 = (S-0-134, S-0-47, P-0-100, S-0-108)

S-0-047 "Position command value " is assigned to 4 bytes.

P-0-100 "Position command value extension " and S-0-108 "Feedrate override" are assigned to 2 bytes each.

The master data telegram contains 3 ident numbers. They are assigned to 8 bytes in total.

Scaling physical data

The NC expects that the physical quantities exchanged with the drive are provided with a specified unit and a specified number of decimal places.

The Sercos parameters of the drives that allow a change in unit and number of decimal places may thus not be set arbitrarily.

A subset of preference scalings is supported. For position data and velocity data, the parameter scaling can be used restrictively.

Preference scaling

Position data specified via S-0076

Scaling type	Measuring unit	Time unit	S-0-0076
Translatory	Meter	-	..xx00.0001
Rotary	Degree	-	..xx00.0010

Tab. 4-1: Position data, preference scaling, S-0-76

Velocity data, specified via S-0044

Scaling type	Measuring unit	Time unit	S-0-0044
Translatory	Meter	Minute	..0x00.0001
Rotary	Revolution	Minute	..0x00.0010
Rotary	Revolution	Second	..0x10.0010

Tab. 4-2: Velocity data, preference scaling, S-0-44

Acceleration data, specified via S-0160

Scaling type	Measuring unit	Time unit	S-0-0160
Translatory	Meter	Second	..0x00.0001
Rotary	Radiant	Second	..0x00.0010

Tab. 4-3: Acceleration data, preference scaling, S-0-160

Torque force data, specified via S-0086

Scaling type	Measuring unit	Time unit	S-0-0086
Percentage	Percent	-	..0x00.0000

Tab. 4-4: Torque force data, preference scaling, S-0086

Parameter scaling for position data

The parameter scaling may be restrictively used for position data.

When using the preference scaling, a position resolution of 10^{-7} m (= 0.0001 mm) or 10^{-4} degrees result. This resolution can be too low for some applications.

By setting bit 3 in S-0-0076, the position resolution is determined by the content of the Sercos parameters S-0-0077 (scaling factor translatory Position data) and S-0-0078 (Scaling exponent translatory Position data) or S-0-0079 (Rotary position resolution).

- Translatory scaling:
 - S-0-0076 (scaling type for position data) has to be set to ..xx00.1001.
 - S-0-0077 (scaling factor translatory position data) has to be set to 1.
 - S-0-0078 (scaling exponent translatory position data) may be set to the values -6, -7, -8 or -9.

The resolution of the position data results from $S-0-77 * 10^{S-0-78}$ m.

S-0-0077 = 1 and S-0-0078 = -7 corresponds to the preference scaling.

- Rotary scaling:
 - S-0-0076 (scaling type for position data) has to be set to ..xx00.1010.
 - S-0-0079 (rotary position data resolution) may be set to the values 360.000, 3.600.000, 36.000.000 or 360.000.000.

The resolution of the position data results in 360 degrees/S-0-0079.

S-0-0079 = 3.600.000 corresponds to the preference scaling.



The maximum possible traversing range of an axis can be restricted when using the parameter scaling.

At a position resolution of 10^{-9} m (= 0.000001 mm = 1 nm), the maximum possible traversing range is at $\pm 2147,483647$ mm.

Parameter scaling for velocity data

The parameter scaling can be restrictively used for velocity data.

Using a reference scaling results in a velocity resolution of 10^{-6} m/min (= 0.001 mm/min), 10^{-4} rpm or 10^{-6} rps. For a translatory velocity scaling in m/s, use the parameter scaling.

By setting bit 3 in S-0-0044, the velocity resolution is determined by the content of the Sercos parameter S-0-0045 (scaling factor for velocity data), S-0-0046 (scaling exponent for velocity data) and by bit 5 of S-0-0044 (scaling type for velocity data).

- S-0-0044 (scaling type for velocity data) has to be set to "..0x00.1001" for the time unit "minute" and "..0x10.1001" to the time unit "second"
- S-0-0045 (scaling factor for velocity data) has to be set to 1.
- S-0-0046 (scaling exponent for velocity data) is user-defined

The resolution of velocity data results in $S-0-45 \cdot 10^{S-0-46}$ m/min, m/s, rpm or rps.



Only use the allowed measuring and time units.

- Translatory data is always specified in the measuring unit "meter"
- Torque/force values always in the measuring unit "percent"

4.1.3 Drive-integrated safety technology

The MTX supports the drive-integrated safety engineering of IndraDrive.

The MTX supports the safety functions in special mode "Safe standstill" and in special mode "Safe Motion SMM". The supported number of safe motions depends on the MTX firmware:

- Up to MTX V14V06:
In special operation "Safe Motion", the MTX supports up to 2 spindles (SMM1 and SMM2).
- From MTX 14V08:
In special operation "Safe Motion", the MTX supports up to 16 spindles (SMM1 .. SMM16).

Constrictively, the following applies:

For the up to 16 safe motions, 2 different safely limited velocity levels or the maximum velocity can be used as safely limited velocity (SLS).

Data exchange To support the safe standstill and the safely limited velocity, MTX requires information about the active and the selected operating state of the safety engineering. To transmit this information, the drive telegram is parameterized.

The parameters for drive systems with safety engineering option module "S2" are:

- P-0-3213, Selected operating state
- P-0-3215, Selected operating state

Templates for scs-files, can be found under /feprom/scsindra/english/safety/.

The parameters for drive systems with safety engineering option module "S3" and "S4" are:

- P-0-3231, SMO: Operating state
- P-0-3262, SMO: Selected operating state

Templates for scs-files, can be found under /feprom/scsindra/english/safetySMO/.



More information about the drive-integrated safety engineering can be found in the following manuals:

- MTX Commissioning
- IndraDrive Integrated Safety Engineering according to IEC 61508
- IndraDrive Integrated Safety Engineering "Safe Motion"

4.2 Linear axis

4.2.1 Description

Function

Applies a connected Sercos drive as linear axis.

A linear axis is interpreted as a machine axis which

- only moves along a straight path.
- usually operates in the position control mode.
- only travels within a definite range.
- is programmed in the unit "mm" or "inch".

Relevant Sercos parameters

S-0-1050.1.6	Configuration list AT Depending on the encoder, the Sercos parameter S-0-0051 (Actual position value 1 motor encoder) or S-0-0053 (Actual position value 2 external encoder) has to be entered.
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.

Tab. 4-5: Relevant Sercos parameters

Relevant machine parameters (MP)

1001 00001	Drive function type
1003 00001	Name of the system axes
1003 00002	Channel assignment
1003 00004	Axis movement type

Tab. 4-6: Relevant machine parameters

4.2.2 Handling instruction: Applying a Sercos drive as linear axis

Integrating Sercos drives as linear axis

Drive: Drive commissioning

- Sercos parameters have been correctly parameterized in the drive or are transmitted by the NC to the drive in the course of Sercos initializing (addressing, timing, operation mode, weightings). Thus, for example, the scaling for a linear axis has to be set to "translatory" in the Sercos parameter S-0-0076 (scaling type for position data).

- and -

- "Commissioning" the drive. When doing so, parameterize the drive according to its requirements in respects of velocity, dynamics, etc.

- and -

- Ensure that the axis is correctly incorporated in the PLC program in respect of its interface signals.

	Documentation	
Documentation:	MTX Functional Description	Drive commissioning

IW Engineering/configuration: Editing parameters

- Creating an axis in the NC basic configuration:
 - <Right mouse key> on the node **MAIN** "NC Basic Configuration"
 - "New"
 - AxFunc[<i>i</i>]** (for axis)

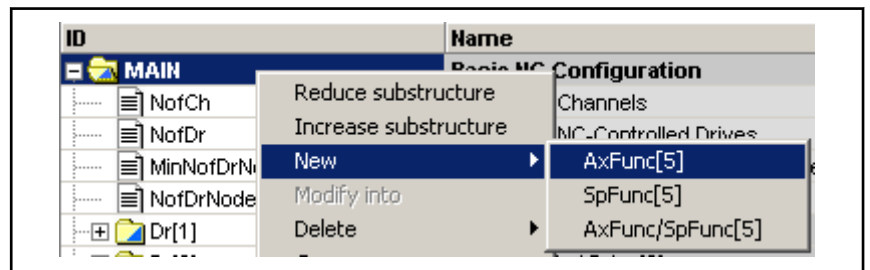


Fig. 4-1: Create axis

- SysDrName**
"Physical axis name" (1003 00001): Enter the system axis name.

Drives (axes, spindles)

- **EnablDr**
"Enable drive" (1001 00001): Enable relevant drive with the value **"Yes"**.
- **SysAxType**
"Axis type" (1003 00004): Select the value **linear axis**.
- **DefaultCh** "Channel Assignment" (1003 00002):
Assign the axis to the desired channel
- **DrType**
"Drive type" (1050 00001): Specify the type for the drive used (e.g. IndraDrive).
- **EnablScsCom**
"Activate drive (in the ring)" (1050 00002)
- **ScsIf**
"Number of the Sercos ring" (1050 00003)
- **ScsAddr**
"Sercos address" (1050 00004)



Adapt unit if necessary.

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

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

- **qAx_OpModeSel_00** "Axis Mode Bit 0"
- **qAx_OpModeSel_01** "Axis Mode Bit 1"
- **qAx_Reset** "Basic axis setting"
- **qAx_DrvOn** "Drive Enable"
- **qAx_DrvLock** "Feed lock"

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

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

- **iAx_RefKnow** "Reference point known"
- **iAx_RefReached** "Reference point reached"
- **iAx_TrvCmd** "Travel command"
- **iAx_Run** "Axis running"
- **iAx_TrqExceed** "Standstill error"
- **iAx_DryRun** "Test mode"
- **iAx_DrvPower** "Enabled for power activation"
- **iAx_DrvReady** "Drive operative"
- **iAx_DrvAct** "Drive active"

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

IW Operation/program: SCS file Adapt phase 2

- **S-0-1050.1.6** "Config. list master data telegram":
Depending on the encoder used in each case, the Sercos parameter **S-0-0051** "actual position value1 motor encoder" or **S-0-0053** "actual position value2 external encoder" has to be entered.
- **S-0-1050.0.6** "Configuration list MDT":
Enter the Sercos parameter **S-0-0047** "Position command value".
- **S-0-0032** "Main operation 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-0-0044** "Scaling type for velocity data" = **translatory**
- **S-0-0076** "Scaling type for position data" = **translatory**
- **S-0-0160** "Scaling type for acceleration data" = **translatory**

	Documentation	Instruction
Instruction:		Editing Sercos parameters
Documentation:	IndraDrive Sercos interface	Linear axis

	Documentation	Instruction
Instruction:		Editing machine parameters
Documentation:	MTX Machine Parameters	Modulo calculation

4.3 Rotary axis

4.3.1 Description

Function

Applies a connected Sercos drive as rotary axis.

A rotary axis is interpreted as a machine axis which

- Performs only rotary movements;
- usually operates in the position control mode.
- only travels within a definite range.
- is programmed in the unit "degree".

Relevant Sercos parameters

S-0-1050.1.6	Configuration list AT Depending on the encoder, the Sercos parameter S-0-0051 (Actual position value 1 motor encoder) or S-0-0053 (Actual position value 2 external encoder) has to be entered.
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.

Tab. 4-7: Relevant Sercos parameters

Relevant machine parameters (MP)

1001 00001	Drive function type
1003 00001	Name of the system axes
1003 00002	Channel assignment
1003 00004	Axis movement type

Tab. 4-8: Relevant machine parameters

4.3.2 Handling instruction: Applies a Sercos drive as rotary axis

Applies a Sercos drive as rotary axis

NC: Creating axis

The same steps as in the creation of a linear axis have to be completed. Only axis type and axis designation have to be differently selected.

		Instruction
Instruction:		Create axis

IW Engineering/configuration: Editing parameters

- **SysDrName** "Physical drive name" (1003 00001):
Entering system axis name
- **SysAxType** "Axis Type" (1003 00004):
select the value **Rotary axis**.



- In the Sercos parameter **S-0-0076** "scaling type for position data", the scaling type has to be set to "rotary" for a rotary axis.
- Adapt unit if necessary.

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

	Documentation	Instruction
Instruction:		Editing machine parameters
Documentation:	MTX Machine Parameters	Modulo calculation

4.4 Endless axis (rotary axis with modulo calculation)

4.4.1 Description

Function

Applies a connected Sercos drive as an endless axis (rotary axis with modulo calculation).

An endless axis is interpreted as a machine axis which

- performs only rotary movements
- usually operates in the position control mode
- can endlessly be traveled in both rotary directions
- is programmed in the unit "degree".

Using the modulo value (also referred to as the "modulo factor"), the current position of the axis is retransformed after reaching its command position into the position range between 0 and the modulo value.

The modulo value is stored in the drive (S-0-0103) and is automatically read by the NC upon startup.

For modulo axes, a parameterizable positioning logic is available. Using the setting, predefine the strategy which the modulo axis has to use to approach the target position.

Speed specification for asynchronous endless axes

In case of asynchronous endless axes, programming the speed in position control mode is possible. This can especially be used for master axes in electronic gears (system axis coupling).

The speed specification is conducted with the S attribute (<axis>=S<speed>) in rpm. When the speed specification is reached, the axis interface signal "iAx_ProgSpReach" is set. The speed specification can be changed as desired and can be transferred on-the-fly into a positioning movement.

The maximum possible speed limit value results from the modulo range set in the drive (S-0-0103) and the parameterized Sercos cycle time SCS/ScsIf[1..2]/ScsCycTime (S-0-0001) with:

$$S_{\max} = \frac{S-0-0103 \cdot 60}{2 \cdot S-0-0001 \cdot 10^{-6} \cdot 360} \left[\frac{U}{MIN} \right]$$

Fig. 4-2: Speed limit value

Example:

Speed limit value for endless axes

S-0-0103 = 360°, S-0-0001 = 4000µs → S max = 7500 rpm

Due to a larger modulo range or a shorter cycle time, the speed limit can be increased for rotary axes.

S-0-0103 = 3600°, S-0-0001 = 4000 µs → S max = 75000 rpm

S-0-0103 = 360°, S-0-0001 = 1000 µs → S max = 30000 rpm

Restrictions

- As long as G91 is active (relative dimension programming), modulo calculation and positioning logic are suppressed internally in the control for synchronous modulo axes under the This ensures that the axis moves exactly along the preset path.
- Unless the system is in positioning type "limited traversing range", the software limit switches is always to be hidden for endless axes. This can be performed statically via machine parameters or dynamically via the bit interface of the axis.

Relevant NC functions

PosMode(...)	Switching the positioning logic
PosMode()	Reset positioning logic to the setting of MP 1003 00005.
DC(..)	For the current block, change the positioning logic to "shortest path".
ACP/ACN(..)	For the current block, change the positioning logic to "sign logic".
G90/G91	Move axis in an absolute (with positioning logic) or incremental (without positioning logic) manner (only synchronous endless axes).
AC(..)/IC(..)	Move axis locally in an absolutely (with positioning logic) or incrementally (without positioning logic).
Modulo(...)	Modulo calculation for NC-internal axis positions for incrementally traveling endless axes.
S(...)	Speed specification in rpm (only asynchronous endless axes)

Tab. 4-9: Relevant NC functions

Relevant Sercos parameters

S-0-1050.1.6	Configuration list AT Depending on the encoder, the Sercos parameter S-0-0051 (Actual position value 1 motor encoder) or S-0-0053 (Actual position value 2 external encoder) has to be entered.
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 If "modulo format" is set as machining format in S-0-0076 (bit 7=1), parameterize 360 degrees as modulo value.

Tab. 4-10: Relevant Sercos parameters

Relevant IF signals

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

Tab. 4-11: Relevant IF signals

Relevant machine parameters (MP)

1001 00001	Drive function type
1003 00001	Name of the system axes
1003 00002	Channel assignment
1003 00004	Axis movement type
1003 00005	Positioning logic
1003 00050	Permit switching the positioning logic of endless axes using PosMode (PMD).
1020 00001	Software limit switch, positive: By means of this parameter, the software limit switches can be statically hidden.

Tab. 4-12: Relevant machine parameters

4.4.2 Handling instruction: Applies a Sercos drive as rotary axis with modulo calculation

Applying a Sercos drive as rotary axis with modulo calculation (endless axis).

Prerequisite:

Sercos parameters were correctly parameterized in the drive or are transmitted by the NC to the drive during Sercos initialization (addressing, timing, operation mode, scalings, etc.). Information on Sercos, refer to the manual "MTX Functional Description 14VRS Special Functions", chapter "Functions for Drive Parameterization".

Thus, for example, the scaling type has to be set to "rotary" and the processing format to "rotary" in the Sercos parameter S--0-0076 (scaling type for position data). Furthermore, the modulo value has to be parameterized in S-0-0103 with the value 360 degrees.

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"**.
- **SysAxType**
"Axis type" (1003 00004): Select the value **rotary axis**.

- **EnablModCalc**
"Enable modulo calculation" (1003 00004): Set to **"Yes"**.
- **ModePosLog**
"Positioning logic" (1003 00005): Select desired positioning logic.
- **EnablChgPosLog**
"Positioning logic can be switched per NC prog." (1003 00050): Determines, whether the positioning logic in the NC program should be switchable via the NC command **PosMode(..)**.



- Using the modulo value (also referred to as the "modulo factor"), the current position of the axis is retransformed after reaching its command position into the position range between 0 and the modulo value.
 - Assign the Sercos parameter **S-0-0103** "modulo value" to the modulo value; this value is read out by the NC automatically at startup.
 - Set the Sercos parameter **S-0-0076** "scaling type for position data" for a modulo axis to the scaling type "rotary" and the machining format to "modulo format".
- As long as **G91** is active (incremental data input), modulo calculation is suppressed internally in the NC for synchronous modulo axes under the 'Processing' and 'Manual data input' operation modes, with the exception of the output command value and its indication.
- Adapt unit if necessary

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

Relevant NC functions

- **PosMode(...):**
Switching the positioning logic
- **PosMode():**
Reset positioning logic to the set machine parameter "Positioning logic" (1003 00005)
- **DC(..):**
For the current block, change the positioning logic to "shortest path".
- **ACP/ACN(..):**
For the current block, change the positioning logic to "sign logic".

	Documentation	
Documentation:	MTX Programming Manual	Positioning logic

4.5 Spindle without Sercos interface

4.5.1 Description

General information

There are five different spindle variants (SP/SpFunc/Sp[]/Base/SpType):

1. Sercos spindle (1)
The MTX directly controls a connected Sercos drive. The commissioning is described in [chapter 4.6 "Spindle with Sercos interface" on page 36](#).
2. External spindle (2)
3. Analog spindle (3)
4. External spindle with encoder (4)
5. Analog spindle with encoder (5)

Function

External spindle The MTX sends the programmed auxiliary functions to the PLC and the PLC controls the spindle.

Spindle interface qSp

The signal qSp_Reset sets the NC-internal motion state to stop (M5). The remaining signals are not processed.

Spindle interface iSp

The spindle interface is not operated.

Analog spindle The MTX controls this spindle variant directly via a DA converter controlled via Profibus.

All spindle functions that can operate without position control, speed and position feedback, can be used.

Spindle interface qSp

All signals without reference to Sercos specific functions are processed.

Spindle interface iSp

All signals without reference to Sercos specific functions and without requiring speed or position feedback, are set.

- External spindle with encoder**
- The programmed auxiliary function have to effects on this spindle type:
 1. The MTX sends the auxiliary functions to the PLC and the PLC controls the spindle.
 2. The MTX internally starts a model spindle.
 - The actual position of the spindle is detected by an external encoder and send to the MTX.

The information from the comparison between the model spindle data and the returned data extend the functional scope of the external spindle. The following functions are supported:

- Motion functions (M3, M4, M5)
- Speed values (S, S1= ... S8=)
- Path movement and spindle functions (SpindleWait and SpindleWait-Mode)
- Dwell time and revolutions (G4)

Drives (axes, spindles)

- Thread cutting (G33)
- Feed in mm/rev (G95)

Spindle interface qSp

The signals

qSp_Reset
 qSp_JogPlus
 qSp_JogMinus
 qSp_TurnCW
 qSp_TurnCCW
 qSp_Stop
 qSp_ManSpeed_00..02
 qSp_SpeedLimit
 qSp_Override100
 qSp_Override_00..15
 qSp_DrvLock

are processed.

Spindle interface iSp

The signals

iSp_TurnCmd
 iSp_TurnDirM4
 iSp_Reset
 iSp_ProgSpReach
 iSp_SpLim
 iSp_Stop
 iSp_Override0
 iSp_Override100
 iSp_DryRun

are set.

Connecting the position encoder

An IndraDrive (from MPx17) with a free external encoder connection can be used to transfer encoder data.

Analog spindle with encoder The MTX controls this spindle variant directly via a DA converter controlled via Profibus.

All spindle functions that can operate without position control, can be used.

Spindle interface qSp

All signals without reference to Sercos specific functions are processed.

Spindle interface iSp

All signals without reference to Sercos specific functions are set.

Connecting the position encoder

An IndraDrive (from MPx17) with a free external encoder connection can be used to transfer encoder data.

Special features and restrictions

- External spindle**
- The MTX has no information on the drive used. It only forwards the programmed auxiliary functions to the PLC.

- Analog spindle**
 - Analog outputs controlled by the MTX have to be available.
- External spindle with encoder**
 - System spindle programming (SSpMove, SSpSpeed...) not possible for this spindle.
 - Encoder data has to be provided to the MTX by a drive amplifier. This data is position data in the modulo format in preference scaling (0.0001 degrees).
- Analog spindle with encoder**
 - Analog outputs controlled by the MTX have to be available.
 - Encoder data has to be provided to the MTX by a drive amplifier. This data is position data in the modulo format in preference scaling (0.0001 degrees).

Relevant machine parameters (MP)

External spindle

1001 00001	Drive function type
1040 00001	Spindle type

Tab. 4-13: Relevant machine parameters

Analog spindle (without and with encoder)

1001 00001	Drive function type
1040 00001	Spindle type
1040 00011	Minimum speed of the gear stage
1040 00012	Maximum speed of the gear stage
1040 00015	Jog speeds
1040 00020	Speed window absolute
1040 00021	Speed window in percent
1040 00031	Max. acceleration
1040 00042	Maximum spindle override
1040 00401	System spindle name
4075 00104	Assignment of the analog outputs
Additionally for the analog spindle with encoder	
1040 00022	Standstill window
1040 00081	Drive number of the external spindle encoder
1040 00082	Sercos parameter of the external spindle encoder

Tab. 4-14: Relevant machine parameters

External spindle with encoder

1001 00001	Drive function type
1040 00001	Spindle type
1040 00011	Minimum speed of the gear stage
1040 00012	Maximum speed of the gear stage
1040 00015	Jog speeds
1040 00020	Speed window absolute
1040 00021	Speed window in percent
1040 00022	Standstill window
1040 00031	Max. acceleration

1040 00042	Maximum spindle override
1040 00081	Drive number of the external spindle encoder
1040 00082	Sercos parameter of the external spindle encoder

Tab. 4-15: Relevant machine parameters

Relevant Sercos parameters

S-0-1050.1.6	Configuration list AT of the drive entered in 1040 00081
S- or P-	Position data of the external encoder
S- or P-	Position encoder type: Direction of motion of the external encoder

Tab. 4-16: Relevant Sercos parameters

4.5.2 Applying

External spindle

IW Engineering/configuration: Editing parameters

- Creating a spindle in the NC basic configuration:
 - Right mouse key on the node **MAIN** "NC Basic Configuration"
 - "New"
 - SpFunc[<i>]</i>** (for spindle)

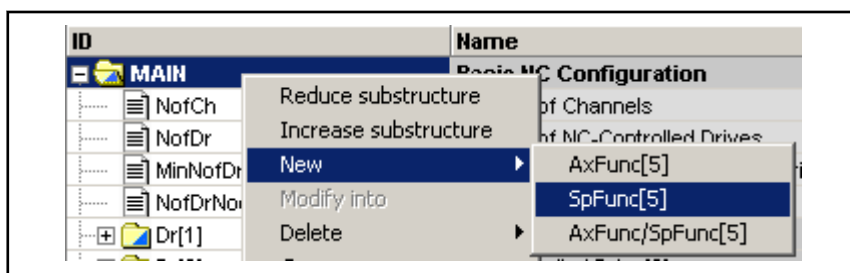


Fig. 4-3: Create spindle

- SpType** "Spindle type" (1040 00001):
Determining the spindle type "External spindle".

The syntax might still have to be set:

- SpCW** "Clockwise spindle rotation" (1040 00101):
Syntax for the "clockwise rotation" function
- SpCWCool** "Clockwise spindle rotation - coolant on" (1040 00102):
Syntax for the "clockwise rotation – coolant on" function
- SpCCW** "Anticlockwise spindle rotation" (1040 00103):
Syntax for the "counterclockwise rotation" function
- SpCCWCool** "Anticlockwise spindle rotation - coolant on" (1040 00104):
Syntax for the "counterclockwise rotation – coolant on" function
- SpStop** "Spindle stop" (1040 00105):
Syntax for the "spindle stop" function

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

- qSp_Reset** "Spindle reset"

Analog spindle

IW Engineering/configuration: Editing parameters

- Creating a spindle in the NC basic configuration:
 - Right mouse key on the node **MAIN** "NC Basic Configuration"
 - "New"
 - **SpFunc[<i>]** (for spindle)

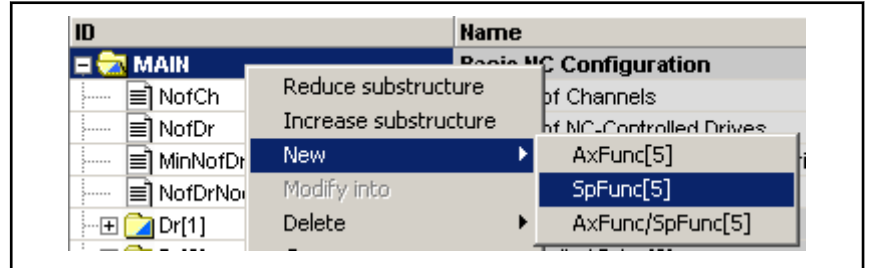


Fig. 4-4: Create spindle

- **SysSpName** "Physical spindle name" (1040 00401):
If no name is entered, MTX generates the name SSPxx (e.g. SSP05 for the 5th system spindle) from the running system spindle number.
- **SpType** "Spindle type" (1040 00001):
Determining the spindle type "Analog spindle".
- For the first gear stage, the following parameters have to be adjusted:
 - **MinSpSpeed** "Minimum speed" (1040 00011)
 - **MaxSpSpeed** "Maximum speed" (1040 00012)
 - **MaxSpAcc** "Maximum acceleration" (1040 00031)
- **JogSpeed[<i>]** "Jog speed[<i>]" (1040 00015):
Spindle jog speed in rpm.
- **SpOvrStep[<i>]** "Override value for step [<i>]" (1040 00041)
- **MaxSpOvr** "Max. spindle override" (1040 00042):
Maximum spindle override.
- **AnaloAllocOut** "Assignment of analog outputs" (4075 00104):
Assignment of system spindle number to analog output (401 ... 432).

The syntax might still have to be set:

- **SpCW** "Clockwise spindle rotation" (1040 00101):
Syntax for the "clockwise rotation" function
- **SpCWCool** "Clockwise spindle rotation - coolant on" (1040 00102):
Syntax for the "clockwise rotation – coolant on" function
- **SpCCW** "Anticlockwise spindle rotation" (1040 00103):
Syntax for the "counterclockwise rotation" function
- **SpCCWCool** "Anticlockwise spindle rotation - coolant on" (1040 00104):
Syntax for the "counterclockwise rotation – coolant on" function
- **SpStop** "Spindle stop" (1040 00105):
Syntax for the "spindle stop" function

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

- **qSp_DrvOn** "Drive Enable"
- **qSp_DrvLock** "Spindle inhibit"
- **qSp_Reset** "Spindle reset"

Drives (axes, spindles)

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

- **iSp_TurnCmd** "Rotation command"
- **iSp_TurnDirM4** "Direction of rotation M4"
- **iSp_Reset** "Spindle reset"
- **iSp_DryRun** "Test mode"

External spindle with encoder

IW Engineering/configuration: Editing parameters

- Creating a spindle in the NC basic configuration:
 - Right mouse key on the node **MAIN** "NC Basic Configuration"
 - "New"
 - **SpFunc[<i>]** (for spindle)

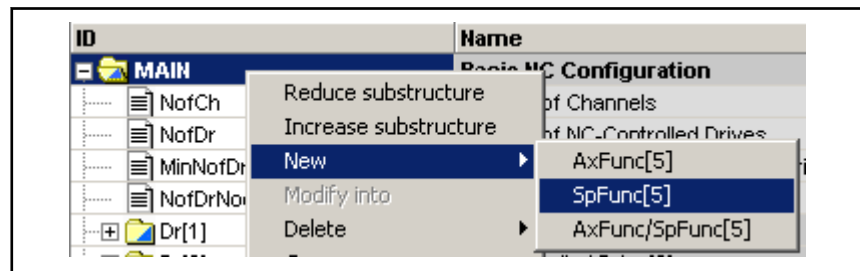


Fig. 4-5: Create spindle

- **SpType** "Spindle type" (1040 00001):
Determining the spindle type "External spindle with encoder".
- The speed limit values and the acceleration of the model spindle have to be set. For the first gear stage, the following parameters have to be adjusted:
 - **MinSpSpeed** "Minimum speed" (1040 00011)
 - **MaxSpSpeed** "Maximum speed" (1040 00012)
 - **MaxSpAcc** "Maximum acceleration" (1040 00031)
- **JogSpeed[<i>]** "Jog speed[<i>]" (1040 00015):
Spindle jog speed in rpm.
- **AbsWin** "Speed window absolute" (1040 00020):
Spindle speed window in rpm.
- **PercWin** "Percentage of speed window" (1040 00021):
Spindle speed window in %.
- **StandstillWin** "Standstill window" (1040 00022):
Spindle stands in window in rpm.
- **SpOvrStep[<i>]** "Override value for step [<i>]" (1040 00041)
- **MaxSpOvr** "Max. spindle override" (1040 00042):
Maximum spindle override.
- **DrIndSpEnco** "Drive number of external spindle encoder" (1040 00081):
Number of the drive that provides data of the external encoder to the MTX.
- **ScsParSpEnco** "Sercos parameter of the external spindle encoder" (1040 00082):
Sercos parameter name

The syntax might still have to be set:

- **SpCW** "Clockwise spindle rotation" (1040 00101):
Syntax for the "clockwise rotation" function
- **SpCWCool** "Clockwise spindle rotation - coolant on" (1040 00102):
Syntax for the "clockwise rotation – coolant on" function
- **SpCCW** "Anticlockwise spindle rotation" (1040 00103):
Syntax for the "counterclockwise rotation" function
- **SpCCWCool** "Anticlockwise spindle rotation - coolant on" (1040 00104):
Syntax for the "counterclockwise rotation – coolant on" function
- **SpStop** "Spindle stop" (1040 00105):
Syntax for the "spindle stop" function

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

- **qSp_DrvLock** "Spindle inhibit"
- **qSp_Reset** "Spindle reset"

If manual mode is required:

- **qSp_TurnCW, qSp_TurnCCW** "Spindle manual"
 - The PLC sets qSp_TurnCW or qSp_TurnCCW.
 - The NC starts the model spindle by using latest programmed S-word.
 - As soon as the NC set the iSp_TurnCmd signal, the PLC starts the spindle motor using the latest programmed S-auxiliary function.
- **qSp_Stop** "Spindle M5 manual"
 - The PLC sets qSp_Stop and stops the spindle motor.
 - The NC stops the model spindle.

If "Jog spindle" is required:

- **qSp_JogPlus, qSp_JogMinus** "Jog spindle"
 - The PLC selects a speed level via qSp_ManSpeed_00, qSp_ManSpeed_01 and qSp_ManSpeed_02.
 - The PLC sets qSp_JogPlus or qSp_JogMinus.
 - The NC starts the model spindle using the specified speed level.
 - As soon as the NC set the iSp_TurnCmd signal, the PLC starts the spindle motor using the selected speed level.
 - The PLC resets qSp_JogPlus or qSp_JogMinus and stops the spindle motor.
 - The NC stops the model spindle.

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

- **iSp_TurnCmd** "Rotation command"
- **iSp_TurnDirM4** "Direction of rotation M4"
- **iSp_Reset** "Spindle reset"
- **iSp_ProgSpReach** "Speed reached"
- **iSp_Stop** "Spindle static"
- **iSp_DryRun** "Test mode"

IW Operation/program: SCS file Adapt phase 2

Drives (axes, spindles)

The drive telegram of the drive entered in "DrIndSpEnco" has to be supplemented by the parameter from "ScsParSpEnco".

- **S-0-1050.1.6** "Configuration list of the drive telegram":
Enter content of ScsParSpEnco in the drive telegram.
- **S or P** "direction of motion of the external encoder":
 - At spindle clockwise rotation (M3), the encoder pulses have to be counted negatively.
 - At spindle anticlockwise rotation (M4), the encoder pulses have to be counted positively.

Analog spindle with encoder

IW Engineering/configuration: Editing parameters

- Creating a spindle in the NC basic configuration:
 - Right mouse key on the node **MAIN** "NC Basic Configuration"
 - "New"
 - **SpFunc[<i>]** (for spindle)

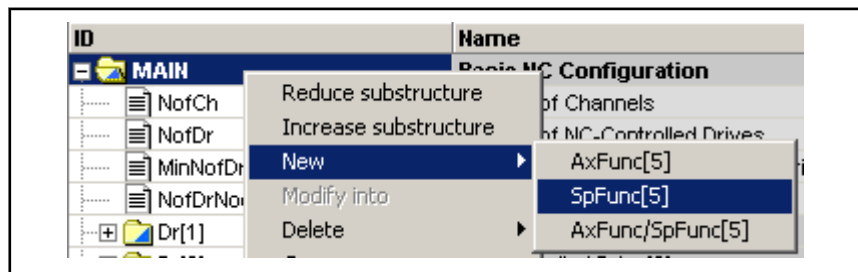


Fig. 4-6: Create spindle

- **SysSpName** "Physical spindle name" (1040 00401):
If no name is entered, MTX generates the name SSPxx (e.g. SSP05 for the 5th system spindle) from the running system spindle number.
- **SpType** "Spindle type" (1040 00001):
Determining the spindle type "Analog spindle with encoder".
- For the first gear stage, the following parameters have to be adjusted:
 - **MinSpSpeed** "Minimum speed" (1040 00011)
 - **MaxSpSpeed** "Maximum speed" (1040 00012)
 - **MaxSpAcc** "Maximum acceleration" (1040 00031)
- **JogSpeed[<i>]** "Jog speed[<i>]" (1040 00015):
Spindle jog speed in rpm.
- **AbsWin** "Speed window absolute" (1040 00020):
Spindle speed window in rpm.
- **PercWin** "Percentage of speed window" (1040 00021):
Spindle speed window in %.
- **StandstillWin** "Standstill window" (1040 00022):
Spindle stands in window in rpm.
- **SpOvrStep[<i>]** "Override value for step [<i>]" (1040 00041)
- **MaxSpOvr** "Max. spindle override" (1040 00042):
Maximum spindle override.
- **DrIndSpEnco** "Drive number of external spindle encoder" (1040 00081):

Number of the drive that provides data of the external encoder to the MTX.

- **ScsParEnco** "Sercos parameter of the external spindle encoder" (1040 00082):
Sercos parameter name
- **AnaIOAllocOut** "Assignment of analog outputs" (4075 00104):
Assignment of system spindle number to analog output (401 ... 432).

The syntax might still have to be set:

- **SpCW** "Clockwise spindle rotation" (1040 00101):
Syntax for the "clockwise rotation" function
- **SpCWCool** "Clockwise spindle rotation - coolant on" (1040 00102):
Syntax for the "clockwise rotation – coolant on" function
- **SpCCW** "Anticlockwise spindle rotation" (1040 00103):
Syntax for the "counterclockwise rotation" function
- **SpCCWCool** "Anticlockwise spindle rotation - coolant on" (1040 00104):
Syntax for the "counterclockwise rotation – coolant on" function
- **SpStop** "Spindle stop" (1040 00105):
Syntax for the "spindle stop" function

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

- **qSp_DrvOn** "Drive Enable"
- **qSp_DrvLock** "Spindle inhibit"
- **qSp_Reset** "Spindle reset"

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

- **iSp_TurnCmd** "Rotation command"
- **iSp_TurnDirM4** "Direction of rotation M4"
- **iSp_Reset** "Spindle reset"
- **iSp_ProgSpReach** "Speed reached"
- **iSp_Stop** "Spindle static"
- **iSp_DryRun** "Test mode"

IW Operation/program: SCS file Adapt phase 2

The drive telegram of the drive entered in "DrIndSpEnco" has to be supplemented by the parameter from "ScsParSpEnco".

- **S-0-1050.1.6** "Configuration list of the drive telegram":
Enter content of ScsParSpEnco in the drive telegram.
- **S or P** "direction of motion of the external encoder":
 - At spindle clockwise rotation (M3), the encoder pulses have to be counted negatively.
 - At spindle anticlockwise rotation (M4), the encoder pulses have to be counted positively.

4.6 Spindle with Sercos interface

4.6.1 Description

Function

Applies a connected Sercos drive as spindle.

A spindle is a machine axis which:

- Performs only rotary movements;
- usually operates in velocity mode;
- drives the tool.



If the drive is to run in position-synchronous mode with other drives (spindle coupling), apply it as Spindle/c-axis (see [chapter 4.7 "Spindle/C-axis " on page 41](#)) with special boundary conditions.

For detailed information on spindle coupling, see [chapter 21](#).

If the "Face transformation" function is also needed for the drive, apply it as "Spindle/C-axis" (see [chapter 4.7 "Spindle/C-axis " on page 41](#)).

With drives of the "Spindle" type, "Spindle orient" (M19) is also possible. Within the drive, the system switches to position-controlled spindle operation until another spindle command (e.g. M3, M4, M5) becomes effective.

See the relevant NC functions in the "Rexroth IndraMotion MTX Programming Manual" for more detailed information.

Relevant Sercos parameters

S-0-1050.1.6	Configuration list AT 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 speed value) and S-0-0144 (signal status word).
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-0-0032	Primary mode Has to be set to "Velocity control" mode.
S-0-0044 and S-0-0160	Scaling type "Rotary scaling" has to be set as scaling type.
S-0-0076	Scaling type of the position data. "Rotary scaling in the modulo format" has to be set as scaling type.

S-x-0103	Modulo value Parameterize 360 degrees as modulo value.
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.

Tab. 4-17: Relevant Sercos parameters



For detailed information on spindles, see the Programming Manual of the control.

Relevant machine parameters (MP)

1001 00001	Drive function type
1040 00001	Selection of spindle type
1040 00010	Number of gear ranges
1040 00011	Minimum spindle speed in each gear stage
1040 00012	Max. spindle speed of the gear
1040 00015	Spindle jog speed in rpm
1040 00020	Spindle speed window in rpm
1040 00021	Spindle speed window in %
1040 00031	Max. spindle acceleration rate
1040 00042	Maximum spindle override
1040 00051	Position interface: 1. acceleration in rad/s^2
1040 00052	Position interface: 2. acceleration in rad/s^2
1040 00053	Position interface: Switching speed for second acceleration
1040 00401	System spindle name

Tab. 4-18: Relevant machine parameters

Relevant system data

/SysSpCmdData[s]/ReverseRotation	
0	Standard use case
1	The direction of rotation of the spindle is inverted for M3/M4 and G63. - M3 generates a positive speed and M4 generates a negative speed - G63(M3,) generates increasing position values and G63(M4,) generates decreasing position values
Note: This system data acts like a machine parameter and is only applied upon control startup. An SDDat file is required to set the value.	

Tab. 4-19: Relevant system data

4.6.2 Handling instruction: Applying a Sercos Drive as spindle

A spindle is a machine axis which performs only rotary movements, is usually operated in velocity mode and drives the tool.

Drive: Drive commissioning

- Commissioning the drive. When doing so, parameterize the drive according to its requirements in respects of velocity, dynamics, etc.
- and -
- Ensure that the axis is correctly incorporated in the PLC program in respect of its interface signals.
- and -
- Sercos parameters have been correctly parameterized in the drive or are transmitted by the NC to the drive in the course of Sercos initializing (addressing, timing, operation mode, weightings).

	Documentation	
Documentation:	MTX Functional Description	Drive commissioning

IW Engineering/configuration: Editing parameters

- Creating a spindle in the NC basic configuration:
 - Right mouse key on the node **MAIN** "NC Basic Configuration"
 - "New"
 - **SpFunc[<i>]** (for spindle)

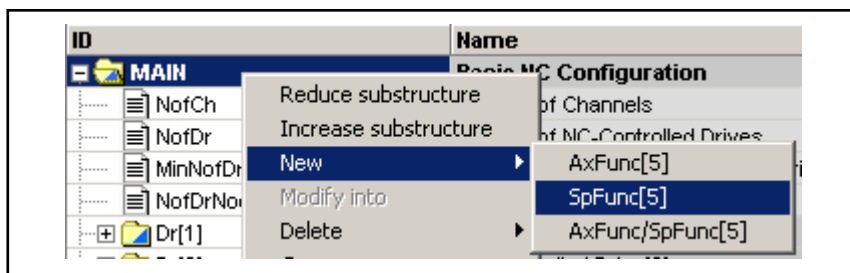


Fig. 4-7: Create spindle

- **SysDrName**
"Physical axis name" (1003 00001): Has to be empty string.
- **EnablDr**
"Enable drive" (1001 00001): Has to be set to **"Yes"**.
- **SysSpName**
"Physical spindle name " (1040 00401): If no name is entered, MTX generates the name SSPxx (e.g. SSP05 for the 5th system spindle) from the running system spindle number.
- **SpType** "Spindle type" (1040 00001):
Determining the spindle type Sercos spindle.
- **DrType** "Drive Type" (1050 00001):
Specify the type for the drive used (e.g. IndraDrive).
- **EnablScsCom** "Activate drive (in the ring)" (1050 00002)
- **ScsAddr** "Sercos address" (1050 00004)
- **SpGrInd** "Group" (1040 00002):
Assignment of a physical spindle to a spindle group.
- **GearChgMethod** "Gear change method" (1040 00008)

- **NofGears** "Number of gears" (1040 00010)
- For each gear, the following parameters have to be adjusted:
 - **MinSpSpeed** "Minimum speed" (1040 00011)
 - **MaxSpSpeed** "Maximum speed" (1040 00012)
 - **MaxSpAcc** "Maximum acceleration" (1040 00031)
 - **MaxSpAccPosCtrl[1]** "Acceleration in the position controller range[1]" (1040 0051)
 - **MaxSpAccPosCtrl[2]** "Acceleration in the position controller range[2]" (1040 0052)
 - **SwiSpeed** "Switching speed to the second acceleration" (1040 00053):
- **JogSpeed[<i>]</i>** "Jog speed[<i>]" (1040 00015):
Spindle jog speed in rpm.
- **AbsWin** "Speed window absolute" (1040 00020):
Spindle speed window in rpm.
- **PercWin** "Percentage of speed window" (1040 00021):
Spindle speed window in %.
- **SpOvrStep[<i>]</i>** "Override value for step [<i>]" (1040 00041)
- **MaxSpOvr** "Max. spindle override" (1040 00042):
Maximum spindle override.
- **SpCW** "Clockwise spindle rotation" (1040 00101):
Syntax for the "clockwise rotation" function
- **SpCWCool** "Clockwise spindle rotation - coolant on" (1040 00102):
Syntax for the "clockwise rotation – coolant on" function
- **SpCCW** "Anticlockwise spindle rotation" (1040 00103):
Syntax for the "counterclockwise rotation" function
- **SpCCWCool** "Anticlockwise spindle rotation - coolant on" (1040 00104):
Syntax for the "counterclockwise rotation – coolant on" function
- **SpStop** "Spindle stop" (1040 00105):
Syntax for the "spindle stop" function
- **SpPos** "Spindle positioning" (1040 00106):
Syntax for the "spindle orient" function

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

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

- **qSp_DrvOn** "Drive Enable"
- **qSp_DrvLock** "Spindle inhibit"
- **qSp_Reset** "Spindle reset"
- **qSp_Stop** "Spindle M5 manual"
- **qSp_Orientate** "Spindle M19 manual"

	Documentation	Instruction
Instruction:		Editing PLC signals
Documentation:	MTX PLC Interface	Spindle signals

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

- **iSp_TurnCmd** "Rotation command"
- **iSp_InPos** "Spindle in position"
- **iSp_PosCtrl** "Position control active"
- **iSp_Reset** "Spindle reset"
- **iSp_ProgSpReach** "Speed reached"
- **iSp_Stop** "Spindle static"
- **iSp_OrientateFinish** "Spindle oriented"
- **iSp_OrientateAct** "Spindle orientation active"
- **iSp_DryRun** "Test mode"
- **iSp_DrvPower** "Enabled for power activation"
- **iSp_DrvReady** "Drive operative"
- **iSp_DrvAct** "Drive active"

	Documentation	Instruction
Instruction:		Editing PLC signals
Documentation:	MTX PLC Interface	Spindle signals

IW Operation/program: SCS file Adapt phase 2

- **S-0-1050.1.6** "Configuration list of the drive telegram":
Adapt the telegram for linear axis.
- **S-0-1050.0.6** "Configuration list MDT":
Adapt the telegram for linear axis.
- **S-0-0044** "Scaling type for velocity data" = **rotary**
- **S-0-0076** "Scaling type for position data" = **rotary in modulo format**
- **S-0-0160** "Scaling type for acceleration data" = **rotary**

	Documentation	Instruction
Instruction:		Editing Sercos Parameters
Documentation:	IndraDrive Parameter Description	Spindle

IW Engineering/configuration: Use of gear steps

If gears are used, additional parameters have to be adjusted. Therefore, follow the **Instruction** "Gear shifting".

		Instruction chapter 16 "Spindle functions " on page 251
Instruction:		Using gears

4.6.3 Handling instruction: Spindle reset

Activating Using the spindle reset, the current movement input or an active command (e.g. spindle orientation) can be canceled for the respective spindle. The spindle acknowledges the receipt of the reset instruction by setting the **iSp_Reset** "Spindle reset" interface signal.

IW Engineering/IndraLogic: Triggering spindle reset per interface signal

Setting the interface signal **qSp_Reset** "spindle reset" causes the cancelation of the current movement of a spindle. The spindle is decelerated to standstill. Moreover, error states are reset.



The interface signal **iSp_Reset** "spindle reset" shows that the spindle has accepted the reset command and that the spindle is ready again to accept new movement specifications.

	Documentation	Instruction
Instruction:		Editing PLC signal
Documentation:	MTX PLC Interface	Trigger spindle reset

4.7 Spindle/C-axis

4.7.1 Description

Function

Applies a connected Sercos drive as "spindle/C-axis".

This type of application is interpreted as a machine axis which

- performs only rotary movements
- Is optionally used as a spindle (spindle mode) or as a rotary/endless axis (C-axis mode).



In C-axis mode, the axis can be operated as an asynchronous or synchronous axis. To use it as a synchronous axis, it is simply integrated into a channel by means of the "Axis transfer" function.

Restrictions

- On activation of complete lock, the NC will save the current drive status (spindle/C-axis operation). To deactivate the complete lock once more, the current drive status has to be identical with the saved status.
- When the main spindle (⇒ MP 7020 00010) is switched to C-axis mode, G95 (feed programming in mm/U) is not switched off automatically.

Relevant NC functions

SpindleToAxis (STA)	Switches from spindle operation to C-axis operation. Refer to chapter 4.7.2 "Switching from spindle operation to C-Axis operation" on page 44.
AxisToSpindle (ATS)	Switches from C-axis operation to spindle operation. Refer to chapter 4.7.3 "Switching from c-axis operation to spindle operation" on page 46.

SpindleToAxisWait (STAW)	Switches from spindle mode to C-axis mode and waits for the execution.
AxisToSpindleWait (ATSW)	Switches from C-axis mode to spindle mode and waits for the execution.

Tab. 4-20: Relevant NC functions



It is recommended to use STAW and ATSW in the part program. The C-axis or the spindle are directly available.



STA or ATS can be entered in MP 7060 00010 (switched-on state after startup) and/or 7060 00020 (on state after reset). Please note:

- After control startup, spindle mode is always active.
- No NC functions should be entered in MP 7060 00010 (switched-on state after startup).
- It is recommended to use ATS in MP 7060 00020 (switched-on state after reset).

Please also note the information on switching between C-axis and spindle mode in [chapter 4.7.2 "Switching from spindle operation to C-Axis operation"](#) on page 44 and [chapter 4.7.3 "Switching from c-axis operation to spindle operation"](#) on page 46.

Relevant macros

Macros and respective subroutines are available for "automatic switching"

STC (<phys. axis > <phys. axis index>)	
	Macro for automatic switching from spindle mode to C-axis mode
STCL (<phys. axis > <phys. axis index>,<log. axis name>)	
	Macro for automatic switching from spindle mode to C-axis mode with assignment of a logical axis name
CTS (<phys. axis > <phys. axis index>)	
	Macro for automatic switching from C-axis mode to spindle mode

Tab. 4-21: Relevant macros



- The macro definitions of the command are located in the file system under **/feprom/macro/MacroDefMtb.xml**.
- The respective subroutines **SwitchToCax** and **SwitchToSpl** are located in the file system under **/feprom/cycles**.
- The macros and/or the subroutines can be adapted on the own requirements is necessary.

Relevant Sercos parameters

S-0-1050.1.6	Configuration list AT 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 speed value) and S-0-0144 (signal status word).
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-0-0032	Primary mode Has to be set to "Velocity control" mode.
S-0-0033	Secondary mode 1 Has to be set to the "position control without following errors" mode.
S-0-0034	Secondary operation mode 2 Has to be set to "position control" mode.
S-0-0076	Scaling type "Rotary scaling in the modulo format" has to be set as scaling type.
S-0-0103	Modulo value 360 degrees is to be set as Modulo value.
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.

Tab. 4-22: Relevant Sercos parameters

Relevant spindle IF signals

qSp_CAxOnC axis ON	
	Switches from spindle operation to C-axis operation.
qSp_CAxOffC axis OFF	
	Switches from C-axis operation to spindle operation.
iSp_CAxAct	C-axis is active
iSp_CAxSwitch	C-axis switching is active

Tab. 4-23: Relevant spindle IF signals

Relevant machine parameters (MP)

For spindle operation:	
1001 00001	Drive function type
1040 00001	Selection of spindle type
1040 00010	Number of gear ranges
1040 00011	Minimum spindle speed in each gear stage
1040 00012	Max. spindle speed of the gear

1040 00015	Spindle jog speed in rpm
1040 00020	Spindle speed window in rpm
1040 00021	Spindle speed window in %
1040 00031	Max. spindle acceleration rate
1040 00401	System spindle name

Tab. 4-24: Relevant machine parameters - for spindle operation



For detailed information on spindles, see the Programming Manual of the control.

Additionally for C-axis operation:	
1003 00001	Names of the system axis
1003 00004	Axis movement type
1003 00005	Positioning logic
1003 00050	Switching the positioning logic of endless axes
1040 00060	C-axis: tool turret
1040 00009	Drive parameter blocks per G-range (only for IndraDrive)

Tab. 4-25: Relevant machine parameters - for C-axis operation

4.7.2 Switching from spindle operation to C-Axis operation

- Program STA or STAW in the part program or via MDI.
For the syntax, refer to "IndraMotion MTX Programming Manual".
In special cases, the PLC can alternatively trigger the change from spindle to C-axis mode via the spindle interface signal qSp_CAxOn "C-axis on".



Spindle reset is not effective in C-axis mode.

After switching, the axis stands at any position between 0 and 359.9999 degrees. At this time, the axis is not yet assigned to a channel (asynchronous axis) and is thus available for each channel as an auxiliary axis.

If you trigger the switching in the part program, we recommend to use STAW and ATSW or WAIT commands (see examples). This ensures that the axis will really be available in the course of the program at the time the block is processed.

Example:

The C-Axis is to be Used as an Asynchronous Axis.

:

N100 STAW(WA) The spindle with the axis name WA is switched as asynchronous axis to C-axis mode and waits till the axis is available.

- If the asynchronous axis is to be integrated into the active channel, WAX is to be programmed.
For the syntax, refer to the "MTX Programming Manual".

Example:

Variant 1: The C-axis should be in parallel connection to other activities, e.g. CPL instructions and should subsequently be implemented into the channel.

:

N100 STA(WA) The spindle which, as an axis, has the name WA is switched to C-axis mode as an asynchronous axis.

:

:

:

N250 WAIT Integrate C-axis (named WA) into the channel.

N260 WAX(WA)

Example:

Variant 2: The C-axis should be directly available.

:

N100 STAW(WA) The spindle with the axis name WA is switched as asynchronous axis to C-axis mode and waits till the axis is available.

N110 WAX(WA) Integrate C-axis (named WA) into the channel.

- An automatic switching from spindle mode to C-axis mode can be carried out via the macros STC or STCL

Example:

Variant 1: The C-axis should be directly switched and should be integrated in the channel.

:

N100 STC(WA) The spindle which, as an axis, has the name WA is directly switched to C-axis mode as an asynchronous axis.

Example:

Variant 2: The C-axis should be switched directly, integrated in the channel and assigned to a logical axis name.

:

N100 STCL(WA,C) The spindle which, as an axis, has the name WA is directly switched to C-axis mode as asynchronous axis with the logical name C.

C-axis switching process:

- Switching while spindle is rotating (M3/M4):

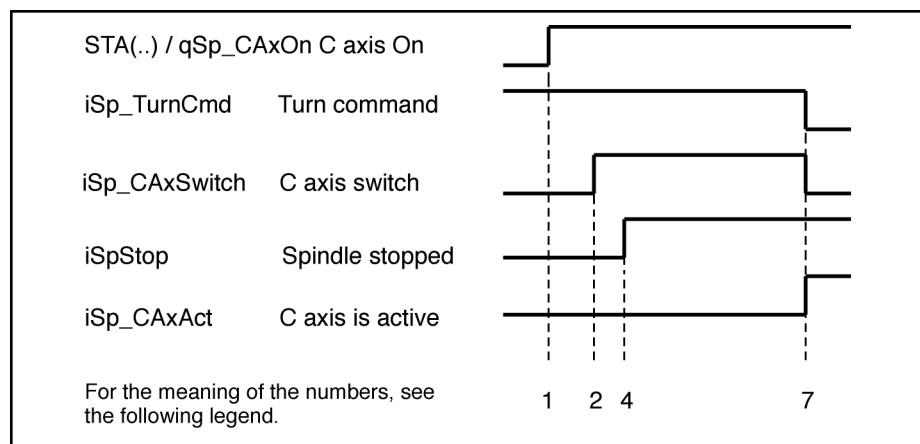
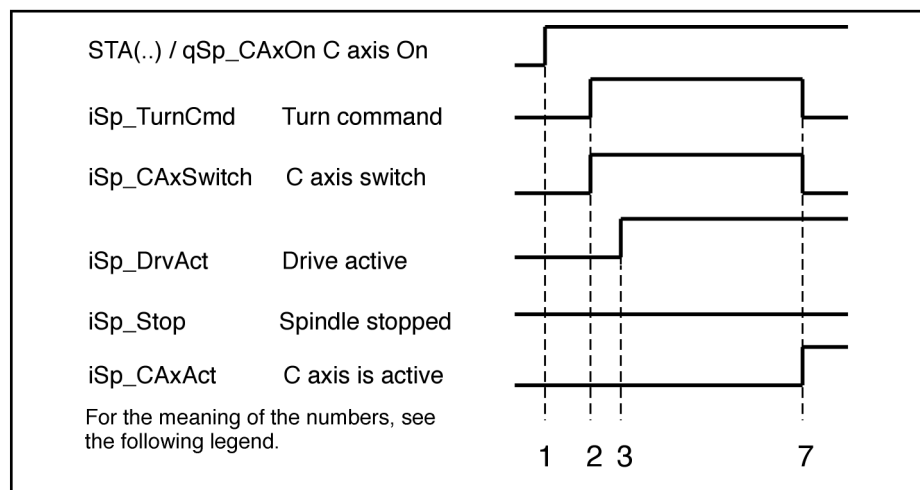


Fig. 4-8: Switching while spindle is rotating (M3/M4)

- Switching from standstill (M5):



1. Activation of the C-axis with STA(..) or qSp_CAxOn "C-axis ON".
2. The MTX sets iSp_TurnCmd "Rotation command" and iSp_CAxSwitch "C-axis switching".
3. The MTX waits for iSp_DrvAct "Drive in operation".
4. The MTX waits for iSp_Stop "spindle in standstill".
5. The MTX changes the operation mode of the drive (->S-0-34) and waits for feedback. (to diagnose, use S-0-135 "Drive status").
6. The MTX reads S-0-403 "Status actual position values" (iAx_RefKnow "Reference point known").
7. The MTX sets iSp_CAxAct "C-axis is active" and deletes iSp_TurnCmd "Rotation command" and iSp_CAxSwitch "C-axis switching".

Fig. 4-9: Switching from standstill (M5)

4.7.3 Switching from c-axis operation to spindle operation

- Ensure that the axis is not assigned to a channel.

If the axis is still assigned to a channel, program RAX. This releases the axis from the active channel; now, it is once more available for all channels in the system as an asynchronous axis.

For the syntax, refer to "IndraMotion MTX Programming Manual".

- Program ATS or ATSW. Thus, switch back to spindle mode. ATS and ATSW must not be programmed in one block after RAX.
See the programming instructions for the syntax.



Only after completed processing of the ATS block, the spindle can be addressed by all channels or through the spindle interface.

Example:

Switching from C-axis to spindle operation

N100 RAX (C) Remove the C-axis from the active channel.
N110 ATSW (C) Switch back to spindle mode.

- An automatic switching from C-axis mode to spindle mode can be carried out via the CTS macro.

Example:

The C-axis should be directly removed from the channel and switched to spindle mode.

:

N100 CTS(WA) The axis named WA is removed from the channel and switched to spindle mode.

4.7.4

Handling instruction: Applies a Sercos drive as c-axis (spindle and rotary axis functionality)

Applies a Sercos drive as C-axis (spindle and rotary axis functionality)
NC: Creating spindle
Before a C-axis (spindle plus rotary axis function) is created, a spindle has to be created.

		Instruction chapter 4 "Drives (axes, spindles)" on page 15
Instruction:		Create spindle

IW Engineering/configuration: Editing parameters

- Create a C-axis (spindle plus rotary axis function) in the axis parameters:
 - Right mouse key on the node **Dr[i]** "NC-controlled drive[i]"
 - "Change to"
 - **AxFunc/SpFunc** (for C-axis)

Drives (axes, spindles)

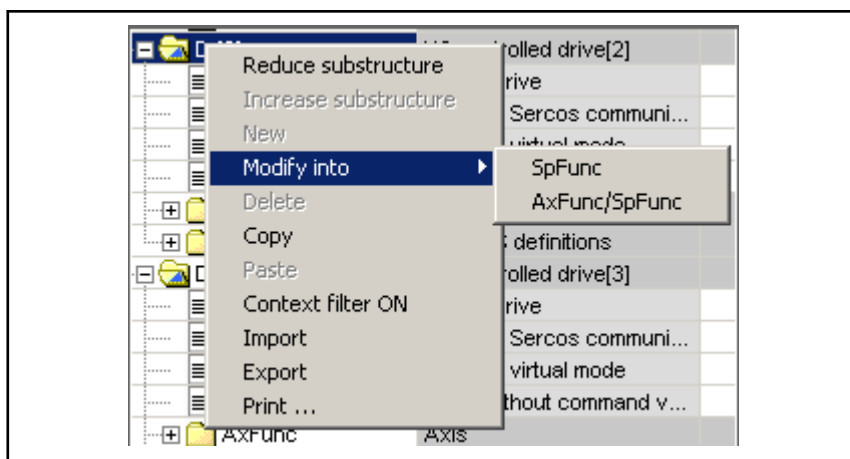


Fig. 4-10: Create a c-axis

- **SysDrName**
"Physical axis name" (1003 00001): Enter the system axis name.
- **SysAxType** "Axis Type" (1003 00004):
Defines the axis motion type, here rotary axis.
- **EnablModCalc** "Enable modulo calculation" (1003 00004):
Enable the modulo calculation if required.
- **SysSpName**
"Physical spindle name " (1040 00401): If no name is entered, MTX generates the name SSPxx (e.g. SSP05 for the 5th system spindle) from the running system spindle number.
- **SpType** "Spindle type" (1040 00001):
Determining the spindle type Sercos spindle.
- **DrType** "Drive Type" (1050 00001):
Specify the type for the drive used (e.g. IndraDrive).
- **ModePosLog** "Positioning logic" (1003 00005)
- **MeasSysSpTurrAx** "Turret mode and measurement system" (1040 00060)
Set to 0 (no tool turret).

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

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

- **qSp_CAxOn** "C-axis ON"
- **qSp_CAxOff** "C-axis OFF"

	Documentation	Instruction
Instruction:		Editing PLC signals
Documentation:	MTX Machine Parameters	C-axis signals

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

- **iSp_CAxAct** "C-axis is active"
- **iSp_CAxSwitch** "C-axis switching"

	Documentation	Instruction
Instruction:		Editing PLC signals
Documentation:	MTX Machine Parameters	C-axis signals

IW Operation/program: SCS file Adapt phase 2

- **S-0-1050.1.6** "Config. list master data telegram":
Enter at least S-0-0040, S-0-0051 or S-0-0053 and S-0-0144.
- **S-0-1050.0.6** "Configuration list MDT":
Enter at least S-0-0036 and S-0-0047.
- **S-0-0026** "Configuration list signal status word".
Enter at least S-0-0331. The entry of S-0-0336 and S-0-0403 is additionally recommended.
(In S-0-0328 "Assignment list of the signal status word" select always bit 0.)
- **S-0-0032** "Main operation mode" = **velocity control**
- **S-0-0033** "secondary operation mode 1" = **position control without following errors**
- **S-0-0033** "secondary operation mode 2" = **position control**
- **S-0-0044** "Scaling type for velocity data" = **rotary**
- **S-0-0076** "Scaling type for position data" = **rotary in modulo format**
- **S-0-0103** "Modulo value":
Parameterize the value 360 degrees
- **S-0-0160** "Scaling type for acceleration data" = **rotary**

	Documentation	Instruction
Instruction:		Editing SCS parameters
Documentation:	MTX Machine Parameters	C-axis settings

4.8 Spindle/turret axis with motor encoder

4.8.1 Description

Function

Applies a connected Sercos as "spindle/turret axis". The drive has to be applied as "spindle/C-axis" (see [chapter 4.7 "Spindle/C-axis" on page 41](#)).

A "spindle/turret axis" is a machine axis which

- Performs only rotary movements;
- Is optionally used as a spindle (spindle operation) or as a rotary/endless axis (C-axis mode; here turret mode).
- Drives the tool in spindle mode and the tool turret (a lathe) in C-axis mode.

As a precondition for activation of a tool turret, the system has to have been switched to turret operation from the NC as well as from the mechanic system, the turret having a gear ratio of its own.

In spindle mode, the drive moves a tool in the form of a milling cutter. To change tools, the drive is switched to C-axis operation; simultaneously, the PLC mechanically switches the motor to the turret axis.

In this state, the NC/PLC can perform a tool exchange via the C-axis.

After switching back to spindle mode (NC and mechanic system), machining of the workpiece can continue.



In C-axis mode, the turret axis can be operated as an asynchronous or synchronous axis. To use it as a synchronous axis, it is simply integrated into a channel by means of the "Axis transfer" function.

Restrictions

- The functionality is currently only available in connection with Indra-Drives.
- The drive must support the command S-0-0197 (support system of coordinates).
- As the contents of the CPL variable @_C_AXISxy_POS are overwritten when an archive (file extension: *.tar) is read, the turret axis has to be referenced after the data have been read in.



Risk of damage to the machine!

Random manipulation of the CPL variables @_C_AXISxy_POS is not permissible. Reference the turret axis.

- For spindle and turret axis, the motor encoder is used as measuring system.
IndraDrive:
There is generally one motor encoder. The absolute evaluation has to be generally be deactivated through S-0-0277. The turret has a cam for homing; its homing point switch is on grid location 1.
- The "spindle orient" position is the turret couple position, i.e. the position in which the mechanical system can be switched over.
- The transmission ratios have to be integer numbers:
 - Spindle gear ratio = 1 : 1
 - Turret gear transmission = N : 1
- From the point of numbering, the drive parameter block of the turret is the first after the spindle parameter blocks.
- On activation of complete lock, the NC will save the current drive status (spindle/C-axis operation). To deactivate the complete lock once more, the current drive status has to be identical with the saved status.
- When the main spindle (MP 7020 00010) is switched to c-axis mode, G95 (feed programming in mm/rev) is not switched off automatically.

Relevant NC functions

STA(..) STAW(..)	Switches from spindle mode to C-axis mode (here: turret mode). In connection with the required parameterization of MP 1040 00060 (setting: 2), the following sequence of action results: • Spindle is decelerated to standstill • The last approached "spindle orient" position is stored internally. • The turret parameter block is activated in the drive (see above). • Turret operation is activated. • The axis position from the CPL variable @_C_AXISxy_POS is applied. STAW(..) stops the part program additionally till the switch-over is executed completely.
ATS(..) ATSW(..)	Switches from C-axis mode to spindle mode (here: turret mode). In connection with the required parameterization of MP 1040 00060 (setting: 2), the following sequence of action results: • The current position of the turret axis is written into the CPL variable @_C_AXISxy_POS. • Spindle mode is activated. • In the drive, the parameter block (gear) last active in spindle mode is activated once more. • As a spindle position, the last "spindle orient" position that has been saved internally is used. ATSW(..) stops the part program additionally till the switch-over is executed completely.
SPHOME	Optional additional function for "Orient spindle" If programmed together with "Orient spindle", SPHOME forces to redetermine the reference point of the spindle. Example: M219 SPHOME

Tab. 4-26: Relevant NC functions



Switching between spindle and turret mode always coincides with mechanical switching. Do not cancel this connection by performing a reset (channel or system reset).

Therefore, it is not allowed to enter STA, STAW or ATS, ATSQ in MP 7060 00010 (switched-on state after startup) or 7060 00020 (switched-on state after reset).

Relevant Sercos parameters



IndraDrive:

In the drive, spindle mode is configured with the parameter blocks 0 to n (1st to 4th gear). For turret mode, the first parameter set following the highest spindle parameter set used is relevant.

S-0-1050.1.6	Configuration list AT The Sercos parameter S-0-0051 (actual position value 1, motor encoder) or S-0-0053 (actual position value 2, external encoder), S-0-40 (actual speed value) und S-0-144 (signal status word) must be entered.
S-0-1050.0.6	Configuration list MDT There, the Sercos parameters S-0-0047 (position command value) and S-0-36 (command speed value) must be entered.
S-0-0026	Configuration list signal status word S-0-331 (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	Secondary mode 1 Has to be set to the "position control without following errors" mode.
S-0-0034	Secondary operation mode 2 Has to be set to "position control..." mode.
S-0-0044, S-0-0076 and S-0-0160	Scaling type... Has to be set to the scaling type "rotary", data reference "on the load" and processing format "modulo format".
S-x-0103	Modulo value If "modulo format" is set as processing format in S-0-0076 (bit 7=1), parameterize 360 degrees as modulo value.
S-0-0147	Homing parameter S-0-147 has to be in Sercos file of Phase 2. S-0-147 has to be set for the spindle operation. (Bit 5 set = Do not evaluate reference switch!)
Only for IndraDrive:	
S-0-0277	Position encoder type 1 Absolute evaluation has to be disabled via bit 6,7.
P-0-2216	Parameter block switching group selection Via bit 15 = 1, the loss of reference has to be suppressed during gear shifting.

Tab. 4-27: Relevant Sercos parameters

Relevant spindle IF signals

qSp_CAxOn	Axis ON Switches from spindle mode to C-axis mode (here: turret mode).
qSp_CAxOff	C-axis OFF Switches from C-axis mode to spindle mode (here: turret mode).

iSp_CAxAct	C-axis is active
iSp_CAxSwitch	C-axis switching active

Tab. 4-28: Relevant IF signals

Relevant machine parameters (MP)

For spindle operation:	
1001 00001	Drive function type
1040 00001	Selection of spindle type
1040 00002	Assignment of a physical spindle to a spindle group
1040 00010	Number of gear ranges
1040 00011	Minimum spindle speed in each gear stage
1040 00012	Max. spindle speed of the gear
1040 00015	Spindle jog speed in rpm
1040 00020	Spindle speed window in rpm
1040 00021	Spindle speed window in %
1040 00031	Max. spindle acceleration rate
1040 00042	Maximum spindle override
1040 00101 to 1040 00112	Spindle syntax
1040 00401	System spindle name

Tab. 4-29: Relevant machine parameters for spindle operation



For detailed information on spindles, see the Programming Manual of the control.

Additionally for C-axis operation:	
1003 00001	System axis designation
1003 00004	Axis movement type
1003 00005	Positioning logic
1003 00050	Switching the positioning logic of endless axes
1040 00060	C-axis: tool turret

Tab. 4-30: Relevant machine parameters for C-axis operation

4.8.2 Application

Prerequisites:

- Sercos parameters were correctly parameterized in the drive or are transmitted by the NC to the drive during Sercos initialization (addressing, timing, operation mode, scalings, etc.. Information on Sercos, refer to the manual "MTX Functional Description 14VRS Special Functions", chapter "Functions for drive parameterization").
- For a spindle/C-axis, the scaling type for example has to be set to "rotary" and the data reference to "on the load" in the following Sercos parameters:
 - S-0-0044 (type of scaling for velocity data)

- S-0-0076 (Scaling type for position data)
- S-0-0160 (type of scaling for acceleration data)
- The mechanical switching between spindle and C-axis mode (here: turret mode) requires a suitable PLC program called by an auxiliary function requiring acknowledgment.

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

Do not reset the control during machining.

1. Assign the value "3" to the relevant drive in MP 1001 00001.
2. Enter the system axis designation for the C-axis in MP 1003 00001.
3. According to your requirements, specify the value "2" (endless) or "3" (rotary) as axis movement type in MP 1003 00004.
Set S-0-0076 always to "rotary in modulo format" in the drive and enter 360 (degrees) in S-0-0103.
4. Set the desired positioning logic with the help of MP 1003 00005.
5. Parameterize the axis according to your requirements in respects of velocity, dynamics etc.
6. Ensure that the axis is correctly implemented in the PLC program in respect of its IF signals. The PLC has to operate the relevant axis interface in C-axis mode. The PLC receives the information whether C-axis operation is active via the corresponding spindle interface signal iSp_CAxAct "C-axis is active".
7. Enter the spindle type "1" (Sercos spindle) in MP 1040 00001.
8. Set MP 1040 00060 (C-axis: Tool turret) to value "2".
9. Parameterize the spindle according to your requirements in respect of gears, velocity, dynamics, syntax, etc. (see above under "relevant machine parameters").
10. Ensure that the spindle is correctly incorporated in the PLC program in respect of its IF signals.
11. If necessary, specify the current position of the turret axis by direct writing (CPL) of the variables @_C_AXISxy_POS (in degrees).
12. Trigger control reset.

4.8.3 Synchronization after control startup

After control startup, the NC is still in spindle mode. As it cannot be supposed that the mechanical switching of the motor (to spindle or turret axis) is likewise in spindle mode at that moment in time, both components have to be synchronized by the PLC.

To this end, the PLC should first identify the status of the mechanical switching (spindle/turret mode) and perform suitable actions for synchronization.

The following **mechanical statuses** may be present:

1. Spindle operation
The motor is mechanically switched to the spindle.
2. Turret mode
The turret axis is **on a grid location**.
The motor is mechanically switched to the turret axis.

3. Turret mode

The turret axis is **between 2 grid locations**.

The motor is mechanically switched to the turret axis.

4. Intermediate position

The mechanical system is neither in spindle nor in turret mode.

Actions for synchronization:

- For 1.:** No action required as the NC is always in spindle mode after control startup.
- for 2.:** NC has to be adapted to the status of the mechanical system:
- The PLC switches to turret mode via spindle IF signal qSp_CAxOn "C-axis ON".
This triggers the following actions in the NC:
 - The parameter block is switched in the drive.
 - Turret operation (C-axis mode) is activated.
 - The axis position from the CPL variable @_C_AXISxy_POS is applied.
 - For reasons of safety, the PLC should always reference the turret axis.
- For 3.:** NC has to be adapted to the status of the mechanical system:
- The PLC switches to turret mode via spindle IF signal qSp_CAxOn "C-axis ON".
This triggers the following actions in the NC:
 - The parameter block is switched in the drive.
 - Turret operation (C-axis mode) is activated.
 - The axis position from the CPL variable @_C_AXISxy_POS is applied.
 - When the NC sets the spindle interface signal iSp_CAx-Act "C-axis is active", the PLC has to start referencing the turret axis.
Simultaneously with the output of the iAx_RefKnow signal "reference point known" at the axis interface of the turret axis, the NC integrates the position of the turret axis into CPL variable @_C_AXISxy_POS.
- For 4.:** The PLC has to set the mechanical system either to spindle or to turret mode. Depending on the resulting mechanical position, the suitable action is performed then (see par. 1. to 3.).

NOTICE**Improper turret positions possible!**

If the position of the turret axis is changed after the NC has last been switched from turret to spindle mode for any reasons, the contents of the variable `@_C_AXISxy_POS` will no longer correspond to the current position of the turret axis!

This may cause severe damage to the machine or the tools.

If the contents of the variable `@_C_AXISxy_POS` no longer correspond to the actual position of the turret axis, the turret axis has to be referenced!

We recommend to reference the turret generally after each control startup for reasons of safety.

4.8.4 Determining reference point of turret axis

Prerequisites: NC and mechanical system have to be in turret operation.

To identify the reference point, proceed as with "normal" axes.

At the end of homing, the reference point position is automatically integrated into the CPL variable `@_C_AXISxy_POS`.

4.8.5 Examples: Switching between spindle and turret mode

Usually, tool change and thus switching between spindle and turret mode is executed in the frame of a tool change program in the following sequence:

1. Stop the spindle with the "spindle orient" function. The auxiliary function (M219) used in the following example programs should require acknowledgment. Alternatively, wait for the spindle IF signal `iSp_OrientateFinish` "spindle oriented".
2. Via STAW, switch the NC to turret operation (C-axis mode).
3. Wait for completion of switching by the NC.
4. By means of an auxiliary function requiring acknowledgment, give the PLC the instruction to mechanically switch to turret operation.
5. Wait for completion of switching of mechanical system.
6. Perform the tool change (including activation of tool correction).
7. By means of an auxiliary function requiring acknowledgment, give the PLC the instruction to mechanically switch to spindle operation.
8. Wait for completion of switching of mechanical system.
9. Via ATSW, switch the NC to spindle operation. `iSp_OrientateFinish`

Example:

1: Turret axis is an asynchronous axis (auxiliary axis)

:

N020 M219

Orient milling spindle.

;N020 M219 SPHOME

Optional: Orienting milling spindle by redetermining the reference point

N030 ...

If necessary, wait for `iSp_OrientateFinish` "spindle oriented".

Do not forget the logics for a potential error.

N040 STAW (REV)	Switch on NC-side from spindle to turret mode and wait till the C-axis becomes active.
	Name of the turret axis: REV
N050 Mxxx	Instruct the PLC to perform the mechanical switch to turret mode.
:	
N160 ...	Perform tool change.
:	
N200 Mxxx	Instruct the PLC to perform the mechanical switch to spindle mode.
N210 ATSW (REV)	Switch on NC-side from turret to spindle mode and wait till the spindle is active..
M30	End of program.

Example:

2: Turret axis is a synchronous axis

:	
N020 M219	Orient milling spindle.
;N020 M219 SPHOME	Optional: Orienting milling spindle by redetermining the reference point
N030 ...	If necessary, wait for iSp_OrientateFinish "spindle oriented". Do not forget the logics for a potential error.
N040 STAW (REV)	Switch on NC-side from spindle to turret mode and wait till the C-axis becomes active.
	Name of the turret axis: REV
N050 Mxxx	Instruct the PLC to perform the mechanical switch to turret mode.
N060 WAX (REV)	Integrate the turret axis REV into the active channel.
:	
N160 ...	Perform tool change.
:	
N200 Mxxx	Instruct the PLC to perform the mechanical switch to spindle mode.
N210 RAX (REV)	Turret axis REV released from channel.
N220 ATSW (REV)	On the NC side, switch from turret to spindle mode and wait for the execution.
M30	End of program.

4.9 Virtual drive

4.9.1 Description

Function

Virtual drives are not assigned to any actually existing drive. They exist exclusively within the NC as objects realized through the software; there, they can be defined as axes, spindles, or c-axes.

By means of certain NC functions, e.g. "axis coupling" or "selective additive coordinates coupling", virtual drives can be used as a source for special sequences of movement which the NC transfers through other axes into the actual movement space at the machine.

Properties - Virtual drive:

- Operating state:
A virtual drive works internally in the NC like a real drive switched to test mode.
- Positioning:
The NC function VirtAxisPos (VAP) assigns an axis position (ACS) to a synchronous virtual axis (channel axis).



The command VAP (VirtAxisPos) is only effective if the axis was previously referenced.

- Homing:
 - Move to reference point (asynchronous and via G74(HOME)):
The "move to reference point" function uses the position stored in the machine parameter 101500110.
 - Move to reference point position (G74):
The "move to reference position" function approaches the position stored in the machine parameters.
 - Limit switch check:
The limit switches of the virtual drive are checked when the reference point position is set with the "move to reference point" function.

Restrictions

- Gantry axes:
Within a gantry group, mixing of virtual and real drives is not permissible.
- Laser power control:
The velocity of a virtual axis is calculated in the laser power control from the NC-internal specified axis velocity.
- Traverse to touch probe (G75), Flying measurement (FlyMeas or FME), measuring at fixed stop (FsProbe (FSP)) and travel to fixed stop (FsMove or FSM):
These functions are not available in connection with virtual drives (axes).

Relevant NC functions

VirtAxisPos (VAP)	The axis position of one or several virtual channel axes is set to the respective value. A local incremental programming [e.g. axis name = IC(value)] is not permitted.

Tab. 4-31: Relevant NC functions

Relevant CPL functions

APOS	"Actual position of an axis": It provides the command position of the virtual axis.
SD(5,3,3) SD(5,4,3) SD(205,3,x)	"Actual spindle speed": It provides the command speed of the spindle.
CPROBE	CPROBE cannot be used when there is a virtual axis in the channel.

Tab. 4-32: Relevant CPL functions

Relevant IF signals

iAx_DryRun	Test mode (axis interface).
iSp_DryRun	Test mode (spindle interface).

Tab. 4-33: Relevant IF signals



All axis and spindle signals based on Sercos signals are generally not activated.

Relevant machine parameters (MP)

1001 00001	Drive type
1001 00010	Virtual drive
1015 00110	Reference point position of virtual axes in mm or degree.

Tab. 4-34: Relevant machine parameters

4.9.2 Handling instruction: Applying a virtual drive

Virtual drives are not assigned to any actually existing drive. They exist exclusively within the NC as objects realized through the software; there, they can be defined as axes, spindles, or c-axes.

NC: Creating axis

Create axis/spindle type

	Documentation	
Documentation:	MTX Functional Description	Create an axis/spindle

IW Engineering/configuration: Editing parameters

- **EnabVirtMode** "Virtual drive" (1001 00010)
- **RefPos** "Reference point position" (1015 00110)

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

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

- **iAx_DryRun** "Test mode"
- **iSp_DryRun** "Test mode"

	Documentation	Instruction
Instruction:		Editing PLC Signals
Documentation:	MTX Machine Parameters	Virtual axis signals

IW Operation/NC programming:

- **VirtAxisPos (VAP)**
The axis position of one or several virtual channel axes is set to the respective value. Local incremental programming (e.g. axis name = IC(value)) is not permitted.
- **APOS** "Actual position of an axis":
It provides the command position of the virtual axis.
- **CPROBE** cannot be used when there is a virtual axis in the channel.



- Within a gantry group, mixing of virtual and real drives is not permissible.
- Laser power control:
The velocity of a virtual axis is calculated in the laser power control from the NC-internal specified axis velocity.
- Touch probe (**G75**), Flying measurement (**InitMeas (IME)**), **FlyMeas (FME)**, Measuring at fixed stop (**FsProbe (FSP)**) and Move to fixed stop (**FsMove (FSM)**) are not possible in connection with virtual drives (axes).

	Documentation	
Documentation:	MTX Programming Manual	Virtual axis

5 Operating functions (manual mode)

5.1 Override

5.1.1 Description

Function

Affects specified...

1. Path velocities in one channel
Velocities of machining axes in setup mode
2. Velocities of auxiliary axes
3. Spindle speeds

The absolute (e.g. programmed) specified values are constantly converted control-internally with regard to the separate, currently set percentages. These values are then applied at the machine.

The currently valid percentage results from signals present at the...

- Channel-specific digital interface
(for velocities according to point 1 above)
- Axis-specific digital interface
(for velocities according to point 2 above)
- Spindle-specific digital interface
(for velocities according to point 3. above).

The control interprets the applied bit pattern directly as binary number (width: 16 bit) and converts it with a scaling of 0.01% into a percentage with 2 decimal places.

Thus, override values ranging from 0.00 to 655.35% can be specified with a resolution of 0.01%.

Restrictions

The response time on a change of override values depends on the PLC revolution period (average: ca. 20 ms)

Relevant NC functions

OVD	Feed 100% Disables the override function in the current channel.
OVE	Feed 100% off Enables the override function in the current channel.

Tab. 5-1: Relevant NC functions

Relevant CPL functions

SD(2,...)	Read-only access to potentiometer data in the active channel.
MCODS(..)	See "IndraMotion MTX Programming Manual, MCODES, Potentiometer"

Tab. 5-2: Relevant CPL functions

Relevant Sercos parameters

S-0-0024	Configuration list of the master data telegram Enter the override value if required. This value is transferred cyclically to the drive.
S-0-0108	Feed rate override Acts on the referencing velocity S-0-0041

Tab. 5-3: Relevant Sercos parameters

Relevant IF signals

At the channel interface	
qCh_Override100	Override 100%
qCh_Override_00 to qCh_Override_15	Override bit 0 to 15
iCh_Override0	Override 0%
iCh_Override100	Override 100%
at the axis interface	
qAx_Override100	Axis override 100%
qAx_Override_00 to qAx_Override_15	Axis override bit 0 to 15
qAx_Override0	Axis override 0%
iAx_Override100	Axis override 100%
At the spindle interface	
qSp_Override100	Spindle override 100%
qSp_Override_00 to qCh_Override_15	Spindle override bit 0 to 15
iSp_Override0	Spindle override 0%
iSp_Override100	Spindle override 100%

Tab. 5-4: Relevant IF signals

Relevant machine parameters

7030 00012	Maximum feed override Defines the maximum valid override value for a specific channel. Under channel "0", the corresponding values are parameterized globally for all auxiliary axes in the system.
1003 00100	Override for auxiliary axes Determines whether the override function is to be enabled or disabled for auxiliary axes.
1040 00042	Maximum spindle override Defines the maximum valid override value for a specific spindle.

Tab. 5-5: Relevant machine parameters

5.1.2 Handling instruction: Override for auxiliary axes (asynchronous axes)

Applying IW Engineering/configuration: Specify max. override

- Enter the maximum override value in % into the axis-specific parameter **MaxAxOvr** "Max. axis-specific override" (7030 00012).
- Continuous override (without steps):
Set all 32 values to 0 in the axis-specific parameter **AxOvrStep**.
- Override in 32 steps:
Specify the percentage of each step in the axis-specific parameter **AxOvrStep** "Override value for step[1..32]".
- Enable the override function for the auxiliary axis in **EnblAxAsynchOvr** "Enable override" (1003 00100).

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

Activating/deactivating

IW Engineering/IndraLogic: Changing axis override interface signal

The interface signal **qAx_Override100** "Override 100%" acts on the desired axis:

qAx_Override100 = 1:

The axis override is switched off temporarily.

qAx_Override100 = 0:

Axis override is active.



The response time on a change of override values depends on the PLC revolution period (average: ca. 20 ms).

	Documentation	Instruction
Instruction:		Editing PLC Signals
Documentation:	MTX PLC Interface	Override

5.1.3 Handling instruction: Channel feed override

Applying IW Engineering/configuration: Specify max. override

- Enter the maximal override value into the channel-specific parameter **MaxChOvr** "Max. channel-specific override" (7030 00012) in percent.
- Continuous override (without steps):
Set all 32 values to 0 in the channel-specific parameter **ChOvrStep** (7030 00010).
- Override in 32 steps:
Specify the percentage of each step in the channel-specific parameter **ChOvrStep** "Override value for step[1..32]" (7030 00010).
- If a separate override value is to be used for rapid traverse motions, it can be enabled by setting **EnablRapTravOvr** "Enable rapid traverse override" (7030 00014).

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

Activating/deactivating IW Engineering/IndraLogic: Change channel feed override interface signal

The interface signal **qCh_Override100** "Override 100%" is applied to all axes defined in the channel:

qCh_Override100 = 1:

Cancels the override function. The channel traverses with the programmed/preselected velocity.

qCh_Override100 = 0:

The Override function is effective.

IW Engineering/IndraLogic: Change rapid traverse override interface signal

Prerequisite:

Separate rapid traverse override is enabled by setting **EnablRapTravOvr**.

The interface signal **qCh_Rap100** "Rapid traverse 100%" is applied to all axes defined in the channel:

qCh_Rap100 = 1:

Cancels the override function. The channel traverses with the programmed/preselected velocity.

qCh_Rap100 = 0:

The Override function is effective.



The response time on a change of override values depends on the PLC revolution period (average: ca. 20 ms).

	Documentation	Instruction
Instruction:		Editing PLC signals
Documentation:	MTX PLC Interface	Override

IW Operation/NC programming: Activating/deactivating feed override

- To activate the feed potentiometer, program **OvrEna** or **OVE**. The feed depends on the position of the feed potentiometer.
- To deactivate the feed potentiometer, program **OvrDis** or **OVD**. The feed is set to 100% of the programmed value irrespective of the feed potentiometer.

	Documentation	
Documentation:	MTX Programming Manual	Activating/deactivating override

5.1.4 Handling instruction: Override for spindles

Applying IW Engineering/configuration: Specify max. override

- Enter the maximum override value into the parameter **MaxSpOvr** "Max. spindle override" (1040 00042) in percent.
- Continuous override (without steps):
Set all 32 values to 0 in the spindle-specific parameter **SpOvrStep** (1040 00041).
- Override in 32 steps:
Specify the percentage value of each step in the spindle-specific parameter **SpOvrStep** "Override value for step[1..32]" (1040 00041).

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

Activating/deactivating

IW Engineering/IndraLogic: Changing spindle override interface signal

The interface signal **qSp_Override100** "Override 100%" acts on the desired spindle

qSp_Override100 = 1:

The spindle override is switched off temporarily.

qSp_Override100 = 0:

Spindle override is active.



The response time on a change of override values depends on the PLC revolution period (average: ca. 20 ms).

	Documentation	Instruction
Instruction:		Editing PLC Signals
Documentation:	MTX PLC Interface	Override

5.2 Approaching reference point

5.2.1 Description

Basics

The NC always has to be informed about the absolute position of all connected drives. Otherwise, an intended and safe controlling of traversing motions is impossible with regard to the maximum traversing range of the machine and the application of safety-relevant functions (position/range monitoring, software limit switch, etc.).

The drive has to inform the NC whether its actual position values relate to the axis reference point:

- If this is the case, either an absolute value encoder is used or the reference point has already been approached when using an incremental encoder. "Approach reference point" is then not required.
- If this is not the case, "approach reference point" has not yet been triggered when using an incremental encoder.

In this case, "approach reference point" is required.

The drive always has the latest version of this information available in S-0-0403 (bit 0).

The NC-integrated logic to approach the reference point (searching the flag, etc.) is not part of the NC anymore (including the measuring system input), but part of the Sercos drive.

Thus, the drive - and not the NC - has to be configured for the respective measuring system via Sercos parameter.



If the automatic parameter download is enabled for the drives, adjust the Sercos parameters in the Sercos files of the NC.

Refer to the manual "MTX Functional Description 14VRS Special Functions", chapter "Functions for the Drive Parameterization".

Sercos drives can be commanded to perform a "drive-controlled referencing" on its own.

To do so, the drive - after receiving the "drive-controlled referencing" command (S-0-0148) - decouples from the position command values of the NC and independently generates its own position specifications for referencing.

For doing so, it uses the parameters S-0-0147 (referencing parameters), S-0-0041 (referencing velocity) and S-0-0042 (referencing acceleration).



To apply the axis potentiometer with these drive-controlled movements as well, it has to be transmitted to the drive with the Sercos parameter S-00108 (feed rate override):

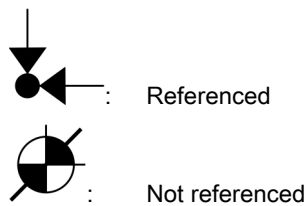
- From the PLC to the service channel in the range of 500 ms - or -
- in the cyclic telegram (MDT) Therefore, S-0-0108 has to be entered into the configuration list MDT (S-0-0024).

Simultaneously, it informs the NC that its actual position value does not refer anymore to the reference point of the axis (S-0-0403: Bit 0=0).

Once the drive successfully completed the "drive-controlled referencing", bit 0 in S-0-0403 switches back to logic "1".

The NC acknowledges this information by setting the axis interface signal "iAx_RefKnow" at the corresponding axis interface.

In the axis display, this status is displayed with an extra character to the position value:



Function

The NC can use different methods to trigger a required "approach to reference point":

- Case 1: Manually via the standard user interface
- Case 2: With PLC via digital interface
- Case 3: With NC syntax via the NC function G74 (HOME)
- (Case 4: With NC syntax via the NC function G74)

In the cases 1 to 3,

- the axis interface signal "iAx_RefReached" is reset at the respective axis interface
- the "drive-controlled referencing" command (S-0-0148) is transmitted to the relevant drives. The reactions of the drives on this command depends on the drive configuration
- The system waits for a positive acknowledgement of the involved drives (bit 0=1 in S-0-0403).

When it is received, the NC will set the interface signals "iAx_RefReached" and "iAx_RefKnow" at the respective axis interface.



If a drive does not issue a positive acknowledgement after a reasonable period of time, "approach reference point" has to be aborted with a reset.

In this case, the interface signals

"iAx_RefKnow" and

"iAx_RefReached"

remain reset at the respective axis interface.

Case 4(G74)

is, compared to the cases 1 to 3 no real "approach to reference point" but exclusively a linear traversing motion in rapid traverse to the reference point coordinates of the programmed axes.

The NC directly determines the positions to be approached from the connected drives at startup.

Depending on the measuring system, the following Sercos parameters are read out:

Motor encoder, incremental: S-0-0052 (Reference dimension actual position value 1)

Ext. encoder, incremental: S-0-0054 (Reference dimension actual position 2)

Motor encoder, absolute: As for the motor encoder, incremental

Ext. encoder, absolute: As for the ext encoder, incremental



If the automatic parameter download is enabled for the drives, adjust the Sercos parameters in the Sercos files of the NC.
Refer to the manual "MTX Functional Description 14VRS Special Functions", chapter "Functions for the Drive Parameterization".

Relevant NC functions

NOTICE Workpiece or machine can be damaged!

During the positioning via G74, any active corrections are not considered. Furthermore, G74 is executed for non-referenced axes as well.

G74	Approach reference point coordinates In block G74, program the axes to be traversed. Example: N.. G74 X0 Y0 Z0 B0 Numerical values behind the axis addresses have to be programmed, but are not significant for the actual traversing motion. Auxiliary and additional functions can be programmed in the same block.
G74(HOME)	Approach reference point In the G74(HOME) block, program the axes for which "approach reference point" is to be triggered. If multiple axes are programmed in the G74(HOME) block, these axes approach their reference points independently of each other (no path mode). Example: N.. G74(HOME) X0 Y0 Z0 B0 Note that the numerical values behind the axis addresses have to be programmed, but are not significant. Auxiliary and additional functions can be programmed in the same block.

Tab. 5-6: Relevant NC functions

Relevant IF signals

iAx_RefReached "Reference point reached" (axis interface)	
	Is effective only for "approach to reference point" via GUI, PLC and G74(HOME). The signal is not set if • "approach reference point" not yet took place • a reference point approach is still active, - or - • the last reference point approach was aborted. It is only set again if "approach reference point" was completed successfully.
iAx_RefKnow	"Reference point known" (axis interface). Is effective for "approach reference point" via GUI, PLC and G74(HOME). The signal is not set if the drive is not informed about the absolute axis position. It is set if the drive is informed about the absolute axis position.

Tab. 5-7: Relevant IF signals

Relevant machine parameters (MP)

1005 00002	Rapid traverse velocities of axes. Is only relevant for "approaching reference point" via G74.

Tab. 5-8: Relevant machine parameters

5.2.2 Handling instruction: Reference point coordinates via NC syntax using the function G74

Approach reference point coordinates via NC the function G74.

IW Operation/program: Adapting SCS file to phase 2

Depending on the measuring system used, it could be possible that the following parameters have to be adjusted, since the command positions of the axes programmed in the G74 block have to be determined by the parameters:

IW Operation/NC programming: Editing program

Programming G74 <AxisCoordinates>

Example: N.. G74 X1 Y1 Z1



The traversing velocity depends on G00 (rapid traverse)/G01 (feed).

Active corrections are not considered during positioning

It is only possible if the axes are already referenced. This has to be checked via the status of the following interface signals:

- **iAx_RefReached = 1** "Reference point reached" and
- **iAx_RefKnow = 1** "Reference point known".

	Documentation	Instruction
Documentation:	MTX Programming Manual	Reference point traveling

5.2.3 Handling instruction: Traveling to reference point via NC syntax using the function G74 (Home)

Applying: Referencing Sercos drives with incremental encoder via the NC function G74 (Home)

IW Operation/program: Adapting SCS file to phase 2

Refer to respective drive documentation (establishing reference - drive-controlled) for the relevant Sercos parameters. The following parameters are generally valid:

- To apply the axis potentiometer for drive-controlled motions as well, it is recommended to continuously transfer the active override value to the drive. Extend the master data telegram - configured via S-0-0024 or S-0-1050.0.6 - by the entry **S-0-0108** "Feedrate override"
- **S-0-0026** "Configuration list; signal status word".
Has to contain the Sercos parameter **S-0-0403** "Status of actual position values".
- **S-0-0041** "Referencing velocity" in m/min
- **S-0-0042** "Referencing acceleration" in m/s²
- **S-0-0147** "Referencing parameter".

For the necessary coding of this parameter, refer to the Sercos parameter description



The listed Sercos parameters can also be directly configured in the drive. In this case, adjust the Sercos parameters in the drive.

	Documentation	Instruction
Instruction:		Editing Sercos parameters
Documentation:	IndraDrive Parameter Description	Reference dimension

Activating: IW Operation/NC programming: Programming approach to reference point
Program **G74 (HOME) <AxisCoordinates>**

Example: N.. G74(HOME) X1 Y1 Z1



The Sercos command **S-0-0148** "C0600 Drive-controlled referencing" is set for the relevant drives.

The synchronous axes of the channel as well as the asynchronous axes can be programmed.

		Documentation
Documentation:	MTX Programming Manual	Reference point traveling

IW Engineering/IndraLogic: Interface signals for checking

The "approach reference point" is successfully completed if:

- **iAx_RefReached = 1**
"Reference point reached" and

- **iAx_RefKnow = 1**
"Reference point is known"

	Documentation	Instruction
Instruction:		Editing PLC Signals
Documentation:	MTX PLC Interface	Reference point traveling

5.2.4 Handling instruction: Reference point traveling with PLC via digital interface

Applying: Referencing Sercos drives with incremental encoder via the NC function G74 (Home)

IW Operation/program: Adapting SCS file to phase 2

Refer to respective drive documentation (establishing reference - drive-controlled) for the relevant Sercos parameters. The following parameters are generally valid:

- To apply the axis potentiometer for drive-controlled motions as well, it is recommended to continuously transfer the active override value to the drive. Extend the master data telegram - configured via S-0-0024 or S-0-1050.0.6 - by the entry **S-0-0108** "Feedrate override"
- **S-0-0026** "Configuration list; signal status word".
Has to contain the Sercos parameter **S-0-0403** "Status of actual position values".
- **S-0-0041** "Referencing velocity" in m/min
- **S-0-0042** "Referencing acceleration" in m/s²
- **S-0-0147** "Referencing parameter".

For the necessary coding of this parameter, refer to the Sercos parameter description



The listed Sercos parameters can also be directly configured in the drive. In this case, the Sercos parameters have to be adjusted in the drive.

	Documentation	Instruction
Instruction:		Editing Sercos Parameters
Documentation:	IndraDrive Parameter Description	Reference dimension

Activating: IW Engineering/IndraLogic: Necessary setting of axis interface signals

- **qAx_HandwSel_00 = 0** "handwheel selection bit 0"
- **qAx_HandwSel_01 = 0** "handwheel selection bit 1"
- **qAx_DrvLock = 0** "feed inhibit"
- At the respective channel interface:
qCh_OpModePlc "PLC operation mode" = 1
- Configure "Setup approach reference point" mode for auxiliary axes via the following interface signals:
qAx_OpModeSel_00 = 0 (axis mode bit 0)

Operating functions (manual mode)

qAx_OpModeSel_01 = 1 (axis mode bit 1)

qAx_OpModeSel_02 = 0 (axis mode bit 2)

qAx_OpModeSel_03 = 0 (axis mode bit 3)



Axis for which "approach reference point" is to be triggered has to be at standstill.

Approaching reference point: Referencing synchronous axes

For synchronous axes, the channel mode "Setup approach reference point" has to be selected via the interface signals **qCh_OpModeSel_00...qCh_OpModeSel_03**.

Operating mode	Operation mode selection			
	Bit 3 (qCh_OpModeSel_03)	Bit 2 (qCh_OpModeSel_02)	Bit 1 (qCh_OpModeSel_01)	Bit 0 (qCh_OpModeSel_00)
Setup approach reference point	0	0	1	0

Approach reference point: Referencing auxiliary axes

For asynchronous axes, the axis mode "Setup approach reference point" has to be selected via the interface signals **qAx_OpModeSel_00** and **qAx_OpModeSel_01**.

Operating mode	Operation mode selection	
	Bit 1 (qAx_OpModeSel_01)	Bit 0 (qAx_OpModeSel_00)
Setup approach reference point	1	0

	Documentation	Instruction
Instruction:		Editing PLC Signals
Documentation:	MTX PLC Interface	Operation mode selection

IW Engineering/IndraLogic: Resetting signals

- **qAx_HandwSelWcs_00** "handwheel selection " bit 0 = 0
- **qAx_HandwSelWcs_01** "handwheel selection " bit 1 = 0
- **qAx_DrvLock** "feed inhibit" = 0

	Documentation	Instruction
Instruction:		Editing PLC Signals
Documentation:	MTX PLC Interface	Resetting signals

IW Engineering/IndraLogic: Approaching to reference point

Set the corresponding axis interface signal at the desired axis:

- **qAx_JogPlus = 1** Manual + or
- **qAx_JogMinus = 1** "Manual - "



The Sercos command **S-0-0148** "C0600 Drive-controlled referencing" is set for the relevant drives.

	Documentation	Instruction
Instruction:		Editing PLC Signals
Documentation:	MTX PLC Interface	Reference point traveling

IW Engineering/IndraLogic: Interface signals for checking

The "approach reference point" is successfully completed if:

- **iAx_RefReached = 1** "Reference point reached" and
- **iAx_RefKnow = 1** "Reference point known"

	Documentation	Instruction
Instruction:		Editing PLC Signals
Documentation:	MTX PLC Interface	Reference point traveling

5.3 Jogging axes

5.3.1 Description

Function

With "Jogging axes", axes can be manually traversed in positive or negative direction.

The distance to be traversed can be influenced by

- The period of the set interface signal qAx_JogPlus "Manual+" or qAx_JogMinus "Manual-"
- four incremental steps stored in the system (1000, 100, 10, 1 increments)
- a user-definable incremental step (x increments)



The operation mode "Setup mode, jogging" can move all axes in the NC. For synchronous axes, this operation mode directly affects the axes of the axis coordinate system (ACS). Thus, the actual axes and not the coordinates are traversed if an axis transformation is active.

First, the individual options are selected via the axis interface signals qAx_ManFeed_00 to qAx_ManFeed_03 "Manual feed bit 0 to 3".

The traversing motion is then triggered via the axis interface signals qAx_JogPlus "Manual+" or qAx_JogMinus "Manual-".

For each of the three options mentioned above, the traversing velocities can be stored separately via a configuration parameter.

The value to be accelerated to the command velocity can be set for all options together via a configuration parameter.

Prerequisites:

- The Sercos communication to the corresponding drive is established correctly.

Refer to the descriptions of the following signals:	
qAx_DrvOn	"Drive On" (axis interface)
iAx_DryRun	"Dry run" (test mode) (axis interface)

Operating functions (manual mode)

iAx_DrvPower	"Enabled for power activation" (axis interface)
iAx_DrvReady	"Drive ready for operation" (axis interface)
iAx_DrvAct	"Drive in operation" (axis interface)

- "Manual setup" ("jog mode") mode has to be enabled.

In contrast to the auxiliary axes, the operation mode of synchronous axes is not influenced by its axis interface, but by the channel interface.

The operation mode of a channel can be set via the PLC (binary-coded via channel interface). The operation mode of auxiliary axes can only be specified via axis interface.

Refer to the descriptions of the following signals:	
qCh_OpModeSel_00	"Sel. operation mode bit 0" (channel interface)
qCh_OpModeSel_01	"Sel. operation mode bit 1" (channel interface)
qCh_OpModeSel_02	"Sel. operation mode bit 2" (channel interface)
qCh_OpModeSel_03	"Sel. operating mode bit 3" (channel interface)
qCh_OpModePlc	"PLC operation mode" (channel interface)
For auxiliary axes :	
qAx_OpModeSel_00	"Axis mode bit 0" (axis interface)
qAx_OpModeSel_01	"Axis mode bit 1" (axis interface)

- The axis interface signals qAx_HandwSel_00 "handwheel section bit 0" and qAx_HandwSel_01 "handwheel selection bit 1" must not be set.
- Axis interface signal qAx_DrvLock "Feed inhibit" must not be set.

Special features



- If the axis does not follow a specification anymore during the jog motion, the jog motion is aborted and a warning is issued.
The following drive warnings are possible causes: E8034 (E-stop enabled), E8042 (both travel range limit switches actuated), E8043 (travel range limit switch positively actuated) and E8044 (travel range limit switch negatively actuated).
- This abort does not include any **start inhibit**! Another jog motion can be started any time.

Relevant CPL functions

MCODS(...)	Refer to the "IndraMotion MTX Programming Manual", MCODES(27,...)

Tab. 5-9: Relevant CPL functions

Relevant IF signals

qAx_JogPlus	"Manual+" (axis interface)
qAx_JogMinus	"Manual-" (axis interface)
qAx_JogInch	"Incremental step in inch" (axis interface)
qAx_JogDia	"Incremental step diameter" (axis interface)
qAx_ManFeed_00	"Manual feed bit 0" (axis interface)

qAx_ManFeed_01	"Manual feed bit 1" (axis interface)
qAx_ManFeed_02	"Manual feed bit 2" (axis interface)
qAx_ManFeed_03	"Manual feed bit 3" (axis interface)

Tab. 5-10: Relevant IF signals

Relevant machine parameters

1005 00003	Manual feed, slow
1005 00004	Manual feed, medium
1005 00005	Manual feed, fast
1005 00006	Manual feed, rapid traverse
1005 00007	Manual feed for defined steps
1005 00008	Manual feed for variable steps
1010 00002	Jog acceleration
1015 00001	Axis resolution
1015 00002	Variable increment step

Tab. 5-11: Relevant machine parameters

5.3.2 Handling instruction: Jogging in axis coordinates

Applying **IW Engineering/configuration: Editing parameters**

Use the following table to set the relevant configuration parameter:

- **AccAxJog**
"Jog acceleration" (1010 00002)
- **IncrAxJog**
"Scaling a jog increment" (1015 00001)
- **VarDistAxJog**
"Variable of jog distance" (1015 00002)
- **SlowAxVel**
"Jog velocity slow" (1005 00003)
- **MediumAxVel**
"Jog velocity medium" (1005 00004)
- **FastAxVel**
"Jog velocity fast" (1005 00005)
- **RapidAxVel**
"Jog velocity rapid traverse" (1005 00006)
- **DefStepAxVel**
"Jog velocity defined step width" (1005 00007)
- **VarDistAxVel**
"Jog velocity variable step width" (1005 00008)

The resulting jog variants are:

Operating functions (manual mode)

Jog mode ⁽¹⁾	Length of the distance to go ⁽²⁾	Effective velocity	Effective acceleration
Rapid traverse	depending on the period of the set axis interface signal "Manual -" or "Manual +"	RapidAxVel	AccAxJog (if assigned with "0", the value from FeedAxAcc applies)
Fast		FastAxVel	
Medium		MediumAxVel	
Slow		SlowAxVel	
x increments	per L/H edge to "Manual +" or "Manual -": VarDistAxJog * IncrAxJog	VarDistAxVel	
1000 increments	per L/H edge to "Manual +" or "Manual -": 1000 * IncrAxJog	DefStepAxVel	
100 increments	per L/H edge to "Manual +" or "Manual -": 100 * IncrAxJog		
10 increments	per L/H edge to "Manual +" or "Manual -": 10 * IncrAxJog		
1 increments	per L/H edge to "Manual +" or "Manual -": 1 * IncrAxJog		

- (1) Selection via the axis interface signals from qAx_ManFeed_00 to qAx_ManFeed_03 "Manual feed Bit0 to Bit3".
- (2) If the axis signal qAx_JogInch "Incremental step in inch" is set at the axis interface of a linear axis, the NC multiplies the value entered into "IncrAxJog" by the factor 2.54. If the qAx_JogDia "Incremental step as diameter" signal is set at the axis interface of a linear axis, the NC divides the value entered into "IncrAxJog" by 2.

Tab. 5-12: Overview on parameters in ACS

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

Activating: Jogging: Jogging synchronous axes

To jog synchronous axes, the channel mode "Manual setup" has to be selected via the interface signals qCh_OpModeSel_00...qCh_OpModeSel_03.

Operating 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	0	0	0	1

Jogging: Jogging auxiliary axes

To jog an asynchronous axis, the axis mode "Manual setup" has to be selected via the interface signals qAx_OpModeSel_00 and qAx_OpModeSel_01

Operating mode	Operation mode selection	
	Bit 1 (qAx_OpModeSel_01)	Bit 0 (qAx_OpModeSel_00)
Manual setup	0	1

	Documentation	Instruction
Instruction:		Editing PLC Signals
Documentation:	MTX PLC Interface	Selecting operation mode

IW Engineering/IndraLogic: Resetting signals

- **qAx_HandwSelWcs_00** "handwheel selection " bit 0 = 0
- **qAx_HandwSelWcs_01** "handwheel selection " bit 1 = 0
- **qAx_DrvLock** "feed inhibit" = 0

	Documentation	Instruction
Instruction:		Editing PLC Signals
Documentation:	MTX Machine Parameters	Resetting signals

IW Engineering/IndraLogic: Selecting jog mode

Select the desired jog mode. Select via the axis interface signals from **qAx_ManFeed_00** "Manual feed bit 0" to **qAx_ManFeed_03** "Manual feed bit 3".

	Documentation	Instruction
Instruction:		Editing PLC Signals
Documentation:	MTX PLC Interface	Selecting jog mode

Jogging: Triggering jog motion

Trigger the traversing motions of the relevant axes via the axis interface signals **qAx_JogPlus** "Manual+" or **qAx_JogMinus** "Manual-".



The interface signals "qAx_JogInch" "Incremental step in inch" and **qAx_JogDia** "Incremental step as diameter" affect the length of the distance to be traversed in the increment modes!

If **qAx_JogInch** is set, the NC multiplies the value entered into the parameter **IncrAxJog** "Scaling a jog increment" (1015 00001) by the factor 2.54. If **qAx_JogDia** is set, the NC divides the value entered into the parameter **IncrAxJog** "Scaling of a jog increment" (1015 00001) by 2.

	Documentation	Instruction
Instruction:		Editing PLC Signals
Documentation:	MTX PLC Interface	Trigger jog motion

Deactivating: IW Operation/IndraLogic: Exiting jogging

- Exiting the channel mode "Manual setup" completes all jog motions of synchronous axes.
- Channel reset and system reset abort an active jog motion of a synchronous axis.

- Exiting the axis mode "Manual setup" completes all jog motions of an auxiliary axis.

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

5.4 Handwheel for Axes

5.4.1 Description

Function

Up to two digital handwheels can be coupled to traverse axes manually in setup mode.

Communication takes place via Profibus-DP.

The distance to be traversed can be influenced by:

- four incremental steps stored in the system (1000, 100, 10, 1 increments)
- a user-definable incremental step (x increments).

The incremental steps are always interpreted as evaluation factor for the graduation of the handwheel. If the handwheel is for example adjusted by 1 scale gradation, the corresponding axis is moved by the corresponding number of increments.

Switch the incremental step via the axis interface signals

from qAx_ManFeed_00 to qAx_ManFeed_03 "Manual feed bit 0" to "Manual feed bit 3".

The traversing velocity is approximately derived from the rotating velocity of the handwheel and is limited to the rapid traverse velocity of the axis.

Prerequisites:

- The number of pulses per handwheel revolution (number of marks of the handwheel * multiplication of the evaluation electronics) is correctly set in MP 9060 00001.
- Handwheels directly connected via Profibus-DP without machine control panel have to transmit values to the NC representing the count of the counter with its signs. Incremental values are not permitted!

NOTICE

Electric/electronic equipment can be damaged!

Switch off the system before removing/connecting lines or installing a plug-in card!

- The Sercos communication to the corresponding drive is established correctly.

Refer to the description of the following signals:	
qAx_DrvOn	"Drive On" (axis interface)
iAx_DrvPower	"Enabled for power activation" (axis interface)
iAx_DrvReady	"Drive ready for operation" (axis interface)

iAx_DrvAct	"Drive in operation" (axis interface)
iAx_DryRun	"Dry run" (test mode) (axis interface)

- The "manual setup" ("jog mode") mode has to be enabled.
In contrast to the auxiliary axes, the operation mode of synchronous axes is not influenced by its axis interface, but by the channel interface.
The operation mode of a channel is set by the PLC (via channel interface). The operation mode of auxiliary axes can only be specified via axis interface.

Refer to the description of the following signals:	
qCh_OpModeSel_00	"Sel. operation mode bit 0" (channel interface)
qCh_OpModeSel_01	"Sel. operation mode bit 1" (channel interface)
qCh_OpModeSel_02	"Sel. operation mode bit 2" (channel interface)
qCh_OpModeSel_03	"Sel. operating mode bit 3" (channel interface)
qCh_OpModePlc	"PLC operation mode" (channel interface)
For auxiliary axes :	
qAx_OpModeSel_00	"Drive mode bit 0" (axis interface)
qAx_OpModeSel_01	"Drive mode bit 1" (axis interface)

- Axis interface signal qAx_DrvLock "Feed inhibit" must not be set.

Special features



- If the axis does not follow a specification anymore when the handwheel is deselected, the handwheel is deselected and a warning is issued.
The following drive warnings are possible causes: E8034 (E-stop enabled), E8042 (both travel range limit switches actuated), E8043 (travel range limit switch positively actuated) and E8044 (travel range limit switch negatively actuated).
- This deselection does not include any **start inhibit**! Another handwheel selection can be executed any time.

Relevant IF signals

qAx_JogInch	"Incremental step in inch" (axis interface)
qAx_JogDia	"Incremental step diameter" (axis interface)
qAx_HandwDir	"Handwheel direction" (axis interface)
qAx_ManFeed_00	"Manual feed bit 0" (axis interface)
qAx_ManFeed_01	"Manual feed bit 1" (axis interface)
qAx_ManFeed_02	"Manual feed bit 2" (axis interface)
qAx_ManFeed_03	"Manual feed bit 3" (axis interface)
qAx_HandwPosMode	"Position handwheel"
qAx_HandwSel_00	"Handwheel selection bit 0" (axis interface)
qAx_HandwSel_01	"Handwheel selection bit 0" (axis interface)

Tab. 5-13: Relevant IF signals

Relevant machine parameters

1005 00002	Rapid traverse velocity of axes
1010 00002	Jog acceleration
1015 00001	Axis resolution
1015 00002	Variable increment step
9060 00001	Number of increments per revolution.

Tab. 5-14: Relevant machine parameters

5.4.2 Handling instruction: Activating handwheel in axis coordinates

Applying: IW Engineering/configuration: Editing parameters

- **AccAxJog**
"Jog acceleration" (1010 00002)
- **IncrAxJog**
"Scaling a jog increment" (1015 00001)
- **VarDistAxJog**
"Variable of jog distance" (1015 00002)
- **IncrPerRev**
"Number of increments/revolutions" (9060 00001)



The configuration parameter **IncrPerRev** "Number of increments/revolutions" (9060 00001) has to be checked, as in case of an incorrect parameterization, the motion valence is not correct. The axis does not move with the number of increments as set.

Incremental step ⁽¹⁾	Length of the distance to go ⁽²⁾	Effective velocity
x increments	Per scale division at the hand-wheel: $\text{VarDistAxJog} * \text{IncrAxJog}$	Depending on the rotary velocity of the handwheel
1000 increments	Per scale division at the hand-wheel: $1000 * \text{IncrAxJog}$	
100 increments	Per scale division at the hand-wheel: $100 * \text{IncrAxJog}$	
10 increments	Per scale division at the hand-wheel: $10 * \text{IncrAxJog}$	
1 increments	Per scale division at the hand-wheel: $1 * \text{IncrAxJog}$	

(1) Selection via axis interface signals from qAx_ManFeed_00 to qAx_ManFeed_03 "Manual feed bit 0 to bit 3".

(2) If the axis signal qAx_JogInch "Incremental step in inch" is set at the axis interface of a linear axis, the NC multiplies the value entered into "IncrAxJog" by the factor 2.54. If the qAx_JogDia "Incremental step as diameter" signal is set at the axis interface of a linear axis, the NC divides the value entered in "IncrAxJog" by 2.

Tab. 5-15: Overview on parameters in ACS

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

Activating: Handwheel: Traversing synchronous axes

To traverse synchronous axes using the handwheel, the channel mode "Manual setup" has to be selected via the interface signals **qCh_OpModeSel_00...qCh_AxModeSel_03**.

Operating 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	0	0	0	1

Handwheel: Traversing asynchronous axes

To traverse asynchronous axes using the handwheel, the axis mode "Manual setup" has to be selected via the interface signals **qAx_OpModeSel_00...qAh_AxModeSel_01**.

Operating mode	Operation mode selection	
	Bit 1 (qAx_OpModeSel_01)	Bit 0 (qAx_OpModeSel_00)
Manual setup	0	1

	Documentation	Instruction
Instruction:		Editing PLC Signals
Documentation:	MTX PLC Interface	Traversing axes with handwheel

IW Engineering/IndraLogic: Configuring incremental step

Selection via the axis interface signals from **qAx_ManFeed_00** "Manual feed bit 0" to **qAx_ManFeed_03** "Manual feed bit 3"

	Documentation	Instruction
Instruction:		Editing PLC Signals
Documentation:	MTX PLC Interface	Configure incremental step

IW Engineering/IndraLogic: Selecting handwheel

Selection via axis interface signals **qAx_HandwSel_00** "Handwheel selection bit 0" or **qAx_HandwSel_01** "Handwheel selection bit 1"



- Select between the handwheel modes "Position-exact" or "Motion-exact" via the interface signal **qAx_HandwPosMode**
- The axis interface signals **qAx_JogPlus** "Manual+" or **qAx_JogMinus** "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 is to interpret the direction of rotation of the handwheel via the axis interface signal **qAx_HandwDir** "Handwheel direction of rotation".

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

Handwheel: Triggering traversing motion

Trigger the traversing motion by rotating the handwheel.



- Note that the signals **qAx_JogInch** "Incremental step in inch" and **qAx_JogDia** "Incremental step as diameter" affect the length of the distance to be traversed.
- If **G70 (Inch)** is active, the handwheel increments are to be multiplied by the factor 2.54 (0.001 mm → 0.0001 inch) for linear coordinates.

Deactivating: **Handwheel: Deactivating**

Reset the respective axis interface signal **qAx_HandwSel_00** "Handwheel selection bit 0" or **qAx_HandwSel_01** "Handwheel selection bit 1".

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

6 Channel control

6.1 Reset

6.1.1 Description

Function

This function is available by default and sets the NC to a defined state. It can be triggered via interface signal for example.

There are two "types of control reset":

- "Reset":
Only effective for the channel in which the function is triggered.
- "System reset":
active system-comprehensively.

"Reset" triggers the following actions:

- Deceleration of all machining axes of the channel to standstill.
- Deselection of the active part program in the channel (also refer to the "automatic program selection" function from [chapter 6.4 "Automatic program reselection"](#) on page 99).
- Reset of error states in the channel.
- Call of the "Switched-on state" function and interpretation of the init string in the machine parameter **CHAN/Ch[k]/Ini/ChResetState** (7060 00020).

All functions programmed in front of the "#SysRes" text are considered.

- Calling the channel-specific init program (only program sequences for the main program end with M30 and channel reset) located in the parameter **CHAN/Ch[k]/Ini/ChIniProgFile** "File name channel initialization program" (7060 00030). See [chapter 6.2 "Channel initialization program \(Init program\)"](#) on page 89.

"System reset" additionally triggers further actions:

- Deceleration of all auxiliary axes and spindles in the system to standstill.
- Also calling the functions programmed in the machine parameter **ChResetState** (7060 00020) behind the text "#SysRes".
- Calling the init programs for all active machining channels (program sequences for the main program end with M30; channel reset with system reset) entered into the parameter **CHAN/Ch[k]/Ini/ChIniProgFile** "File name for main program end with M30; channel reset and system reset" (7060 00030). See [chapter 6.2 "Channel initialization program \(Init program\)"](#) on page 89.
- Application of changed configuration parameter values.

If individual parameters can only be applied by the control startup, the NC informs accordingly.

Relevant IF signals

qGen_Reset	"System reset" (global interface) L/H signal edge triggers the function
qCh_CtrlReset	"Reset" (channel interface) L/H signal edge triggers the function
iCh_Reset	"Channel reset" (channel interface) The signal is not set if the channel is not in the switched-on state and a program is selected or active It is set as long as the channel is in the switched-on state and no program is selected If an L/H signal edge is present, the control performed a reset. The PLC has to execute qCh_CtrlReset "Reset" subsequently

Tab. 6-1: Relevant IF signals

6.1.2 Handling Instructions: System reset

Applying IW Engineering/configuration: Editing parameters

- Use the parameter **DelZpo** "Delete SetPos offset with reset" (7050 00510) to delete the SetPos offset with the reset
- Use the parameter **AutoProgSel** "Automatic program reselection" (7060 00410) to automatically reselect the latest selected program.
- To determine the init string in the parameter **ChResetState** "State after channel reset" (7060 00020)

	Documentation	Instruction
Instruction:		Editing machine parameters
Documentation:	MTX Machine Parameters	System reset

Activating

NOTICE

Workpiece or machine can be damaged!

During machining, do not trigger "system reset"!

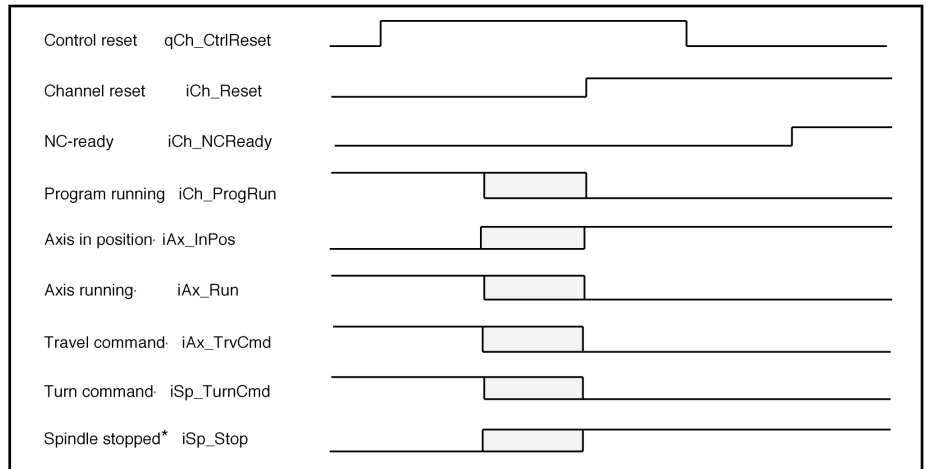
This signal is applied to all NC channels. The effect of a system reset corresponds to a simultaneous reset of all channels, axes and spindles. Furthermore, modifications for a majority of the machine parameters are applied after a system reset.

IW Engineering/IndraLogic: Triggering a system reset via interface signal

Setting the global interface signal **qGen_Reset** "System reset" causes:

- Deceleration of all machining axes, auxiliary axes and spindles to the system to standstill.
- Deselection of the active part program in the channel.
- Reset of error states in the channel.
- Call of the function "switched-on state" and interpretation of the Init string contained in the parameter **ChResetState** State after channel reset (7060 00020). All functions are considered - even those that are programmed behind the "#SysRes" text.

- Application of changed configuration parameter values. If individual parameters can only be applied by the control startup, the NC informs accordingly.



*) Only for auxiliary axes and spindles in case of a complete reset

Fig. 6-1: System reset

	Documentation	Instruction
Instruction:		Editing PLC signal
Documentation:	MTX PLC Interface	Trigger system reset

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

	Instruction
Instruction:	Assigning the M-key

IW Operation/machine: Triggering a system control reset via GUI

- Select the OP-key "Machine" (OP3)



- Use the F-key ">>" (F8) to switch the F-keypad



- Trigger the reset using the F-key "System reset" (F2)



- Acknowledge the following message

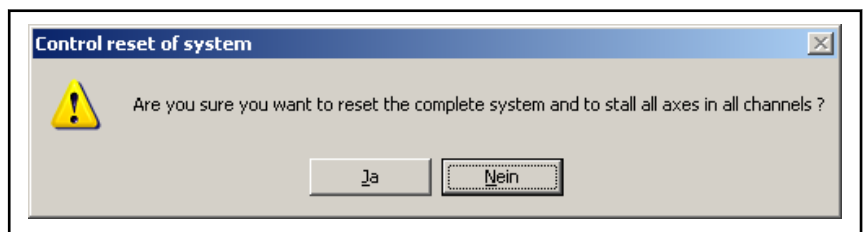


Fig. 6-2: System reset

	Documentation	Instruction
Documentation:	IndraWorks HMI	Applying the axis distance control

6.1.3 Handling instruction: Channel reset

Applying IW Engineering/configuration: Editing parameters

- Use the parameter **DelZpo** "Delete SetPos offset with reset" (7050 00510) to delete the SetPos offset with the reset.
- Use the parameter **AutoProgSel** "Automatic program reselection" (7060 00410) to automatically reselect the latest selected program.
- Determine the init string in the parameter **ChResetState** "State after channel reset" (7060 00020).

	Documentation	Instruction
Instruction:		Editing machine parameters
Documentation:	MTX Machine Parameters	Channel reset

Activating

NOTICE

Workpiece or machine can be damaged!

During machining, do not trigger "reset"!

The channel reset is applied to the axes interpolating in a channel as well as to the part program running in this channel.

IW Engineering/IndraLogic: Triggering a control reset via interface signal

Setting the channel-specific interface signal **qCh_CtrlReset** "Control reset" causes:

- Deceleration of all machining axes of the channel to standstill.
- Deselection of the active part program in the channel.
- Reset of error states in the channel.
- Call of the function "switched-on state" and interpretation of the init string contained in the parameter **ChResetState** "State after channel reset" (7060 00020). All functions programmed in front of the "#SysRes" text are considered.

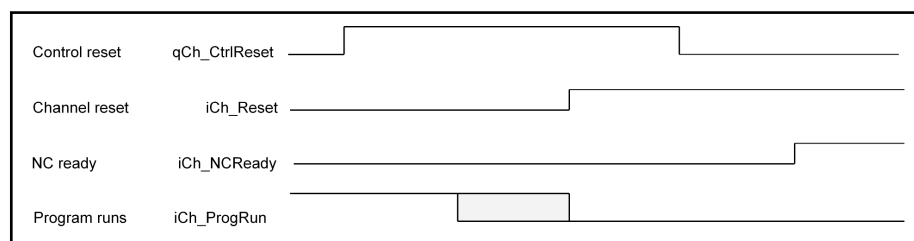


Fig. 6-3: Channel reset

	Documentation	Instruction
Instruction:		Editing PLC signal
Documentation:	MTX PLC Interface	Triggering Channel Reset

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

		Instruction
Instruction:		Assigning the M-key

IW Operation/machine: Triggering a reset via GUI

- Select the OP-key "Machine" (OP3)



- Use the F-key ">>" <F8> to switch the F-keypad



- Trigger the reset using the F-key "Reset channel" <F2>



	Documentation	
Documentation:	IndraWorks HMI	Triggering Channel Reset

6.2 Channel initialization program (Init program)

6.2.1 Description

General information

Channel initialization programs (called init programs in the following) supplement the channel-specific init strings. These are the settings in the parameter **ChResetState** "State after channel reset" (7060 00020). The entries in the init strings are limited to 240 characters and processed as one NC block. These restrictions do not apply to the init programs.

After the processing of the init strings, the control executes the init programs entered into the parameter **ChIniProgFile** "File name channel initialization program" (7060 00030). This is performed after the following events

- Program end of a main program with M30
- Channel reset
- System reset

The init programs are executed in the current operation mode. Depending on the system event, the corresponding program sections for the main program end with M30, channel reset or system reset are processed.

Init programs are usually contained in the NC file system in the directories of the system search paths (/ → /usrfep → /feprom). To change or create init programs, an init program can be copied from the "/feprom" directory to the root directory and edited there. Subsequently, store them in the "/usrfep" directory.

Pre-assignments

The following init programs are applied in the /feprom directory by default:

- Ch01Ini.prg (for channel 1)
- Ch02Ini.prg (for channel 2)

Channel control

- ...
- Ch12Ini.prg (for channel 12)

The default init program for channel 1 contains multiple instructions that were part of the init string pre-assignment of channel 1 for versions older than IndraMotion MTX 13VRS (refer to the following program code). The default init programs for the channels 1 to 12 are equally structured, but they are neutral, since they contain only "empty" instructions except for the CLRALL-WARN call.

Default init program for channel 1 (/feprom/Ch01Ini.prg):*Program:*

```

0010 GOTO .NOCOM
;*****
; Channel specific initialisation program to be executed *
; after the following conditions: *
; SD(30)=1 : after program end (M30) of main program *
; SD(30)=2 : after channel resetting *
; SD(30)=3 : after system resetting *
; SD(30)=0 : not called as Init program but as normal *
; NC program *
;*****
;
;*** Which condition caused this program call? ***
0020 .NOCOM
0030 INICOND%=SD(30)
0040 CASE INICOND% OF
0050 LABEL 0
; *****
; * Sequence for call as normal NC program *
; *****
N060 LP NcProg
0100 LABEL 1
; *****
; * Sequence after program end (M30) of main program *
; *****
N110 LP ProgEndProg
0200 LABEL 2
; *****
; * Sequence after channel resetting *
; *****
N210 LP ProgEndProg
N220 LP ChResetProg
0300 LABEL 3
; *****
; * Sequence after system resetting *
; *****
N310 LP ProgEndProg
N320 LP ChResetProg
N330 LP SysResetProg
0400 ENDCASE

; End of Init Program, local subprograms follow
;
LPS ProgEndProg
;*****
; Subprogram after program end (M30) of main program *
;*****
N1000 G1 G17 G90 G71 G94 G8 IPS1 G79 CLN(CollErr2) G43 G45 OVE ZOS(1)
# DCS(1) KVP() FFW() AAC() PAC() PMD() F0
PEND
;
LPS ChResetProg
;*****
; Subprogram after channel resetting *
;*****
N2000 P /feprom/cycles/CLRALLWARN
PEND
;
LPS SysResetProg
;*****
; Subprogram after system resetting *
;*****
; N3000 ...
PEND
;
LPS NcProg
;*****

```

```

;* Subprogram for call as normal NC program *
;*****
; N4000 ...
PEND
;

```

The file names "Ch01Ini.prg" to "Ch12Ini.prg" are specified in the parameter **ChIniProgFile** "File name channel initialization program" (7060 00030) and can be changed or deleted if required.



When deleting, no init program is executed in the corresponding channel.

Init program instructions

Any CPL and NC functions can be programmed in init programs. They act like normal part programs. No axis and spindle motions may be programmed. Use the CPL function SD(30) to query within an init program the event that caused the call:

- SD(30) = 1: After program end of a main program with M30
- SD(30) = 2: After channel reset
- SD(30) = 3: After system reset
- SD(30) = 0: None of the previous events occurred. The init program was called as normal NC program

Some functions have to remain in the init string. When defining which NC functions belong to the init string and which become part of the init program, the following is significant:

Each modal NC function is provided with one of the following

- a "switch-off property" or
- an "init property" or
- both these properties.

Local NC functions are generally provided with an "init property".

The DAX command always has to be entered into the init string behind the keyword "#SysReset:". In this case, the corresponding channels are synchronized at system reset.

All functions with "switch-off property" have to be programmed earlier (in the init string). All functions with "init character" have to be programmed afterwards (in the init program).

For functions provided with both properties, it can be sufficient if these are exclusively programmed in the init string. This is for example the case if a general channel state is created. This channel state has an axis/coordinate reference, but its axis reference is only of minor significance for this function (switching off an axis transformation, an inclined plane and a scaling).

However, if the axis reference has a superordinate significance, this function has to be entered into the init string as well as into the init program. Examples are all functions configuring axis properties directly affecting the drive (such as FFW or KVP).

The following table shows the properties of these functions and the assignment to init string or init program for the selection of NC functions:

Channel control

Assigning NC functions (selection) to init string or init program

NC function	Switch-off property	Init property	Init string	Init program
AAC()		x		x
ATR()	x		x	
ATS(7)	x		x	
BCR()	x		x	
CLN		x		x
CLN(CollErr2)		x		x
CRD()	x		x	
D0	x		x	
DAX	x		x	
DCS(1)		x		x
F0		x		x
FFW()	x	x	x	x
G1		x		x
G8		x		x
G17		x		x
G40	x		x	
G43		x		x
G45		x		x
G48	x		x	
G53	x		x	
G62	x	x	x	
G71		x		x
G79	x	x		x
G80	x		x	
G90		x		x
G94		x		x
G140	x		x	
G153	x		x	
IPS1		x		x
KVP()	x	x	x	x
MIR()	x		x	
OVE		x		x
PAC()		x		x
PLS()	x		x	

NC function	Switch-off property	Init property	Init string	Init program
PMD()		x		x
RND()	x		x	
ROT()	x		x	
SCL()	x		x	
SHT()	x		x	
TRS()	x		x	
ZOS(1)		x		x

Tab. 6-2: Assigning NC functions to init string or init program

Executing init programs

The init programs are executed in the current operation mode. The "block transfer inhibit" signal and the operation in "single block" or "single step" are ignored.

Syntax or runtime errors in the init program abort the blocks and block the init program to impede a restart after the next reset. Recovering the error is possible instead. The init program is not blocked anymore if one of the following init program characteristics changed:

- File name
- Storage location in the NC file system
- Time stamp
- Call deselected via parameter



If an automatic program reselection follows a system or channel reset and the processing of init string and init program, the signal iCh_Reset ("channel reset") remains present as long as the machining channel is in the "ReadyForProgStart" state. Thus, the end of the reset can be surely detected.

If the init program is not to be executed, it can be disabled following the handling instruction (see below).

Restrictions

There are the following restrictions when using init programs:

- No axis and spindle motions must be programmed
- Synchronous traversing motions are ignored and cause a runtime error
- Calling asynchronous subroutines (ASUP) is prevented (→ runtime error)
- Calling external modal subroutines is prevented (→ runtime error)
- Calling external non-modal subroutines is prevented (→ runtime error)

Exception: Non-modal subroutines can be called if they are stored in one of the "/" (root directory), "/usrfep", "/usrfep/cycles" or "/feprom/cycles" paths.

It is recommended to call a valid, non-modal subroutine with path specification. In case of a call without path specification, the non-modal subroutine is always searched for in the directory of the calling init program

first (except for /feprom) and subsequently in the subroutine search path (if required, add "/usrfep/cycles" or "/feprom/cycles")

- Some invalid commands (e.g. RCB, RCBDel) cause a runtime error

Archiving and restoring

Due to its close relation with the init strings, init programs are considered as part of the machine parameter configuration. Thus, when creating a machine parameter archive, the init programs parameterized for the active machining channels are automatically backed up if available in the system search path (without /feprom).

An init program is also automatically backed up if the keyword "#NoIniPrg" is entered into the init string of the corresponding channel.

If init programs use the option limited call of external non-modal subroutines, the subroutines are not automatically backed up! In this case, the subroutines have to be backed up separately (without /feprom/cycles).

Analogously to the creation of the machine parameter archive, the init programs of such an archive are also restored when reading in. Previously existing init programs are deleted or overwritten.

If init programs use the option limited call of external non-modal subroutines, the subroutines are not automatically restored! In this case, these subroutines have to be restored separately.

After reading in archives created before the IndraMotion MTX 13V04, the execution of the default init program of channel 1 can be impeded due to compatibility reasons. This is performed by superimposing an "empty" init program in the system search path.

Relevant machine parameters (MP)

CHAN/Ch[1..60]/Ini/ ChResetState (7060 00020)	State after channel reset ("Init strings")
CHAN/Ch[1..60]/Ini/ ChIniProgFile (7060 00030)	File name of the channel initialization program ("Init programs")

Tab. 6-3: Relevant machine parameters

6.2.2 Handling instruction: Init programs

Applying: The init programs are active in the presetting. To adjust an existing init program, proceed as follows:

- Copy the init program to be adjusted from the "/feprom" directory or an already adjusted init program from the "/usrfep" directory to the root directory
- Make the desired extensions or changes using the table [chapter "Init program instructions" on page 91](#)
- To test the changes, reset the system. Correct existing error and perform system reset again
- Copy correct init program to the "/usrfep" directory
- Delete init program in root directory

Activating: The init programs are active in the presetting. If deactivated with one of the steps described under "Deactivate", the respective step has to be undone.

Deactivating: If the init program is not to be executed, it can be disabled as follows:

- Delete the corresponding entry in the parameter **ChIniProgFile** "File name channel initialization program" (7060 00030)
- Insert the keyword **"#NoIniPrg"** into the init string (between beginning and keyword **"#Reset:"**) in the parameter **ChResetState** "State after channel reset" (7060 00020)
- Superimposing another (e.g. "empty") init program at a superordinate location in the system search path (/ → /usrfep → /feprom)

If a previously active init program located in the root directory or "/usrfep" directory is to be deleted, but the deletion is not possible, proceed as follows:

- Renaming the program
- Reset channel in the corresponding channel or system reset
- Deleting the program

6.3 Program selection

6.3.1 Description

General information

Before processing a part program in a channel, the program has to be selected in this channel. All axes to be traversed synchronously by the part program (unless these axes are not transferred by the part program using the axis transfer commands) have to belong to the corresponding channel. Thus, a program can only be selected if the correct channel is selected.

All NC programs stored in the internal file system of the control or in a file system mounted by the control can be selected.

There are three options to select a program:

- Manual program selection via user interface
- Program selection via PLC
- Program selection from a part program (CPL)

Program selection via user interface:

In the operating area "Machine", the "Program selection" softkey selects a program in the directory using the "Confirm selection" softkey. More selection options are located under the softkey "NC-Prg. functions"

Program selection via PLC:

The program block "MT_ProgSel" is available for the PLC.

Thus, the following is allowed for the PLC

- Program selection
- Program selection with an automatic deselection of the old program
- Program change
- Program deselection

Program selection from a part program (CPL):

The CPL command MCOPS(4,...) can be used to select - from one part program - another part program in a different channel. Enter an optional specification of a starting and an end block to execute only one program section.

A selected program is indicated by the "ready icon"  in the status bar of the user interface.

Press <Start> at the machine operating panel to start the program. The

"ready icon" changes to the  "running icon" .

DIN/CPL programs:

A program has to be preprocessed before it is selected. This includes a syntax check and the creation of jump tables for subroutines as well as management information for CPL variables used in the part program. Preprocessing, also called linking, is indicated in the status bar by a flashing "Ready" symbol if "Linking at program selection" is enabled (MP 7070 00030). If linking is enabled at program runtime, the flashing "Ready" symbol is not visible or only for a very short moment. The program can be started immediately.

By linking, the NC generates a managing file (link table) stored in the "/usr/lnk" directory by default [setting via the machine parameter **NCO/FileOrg/SrchPathLinkTab** (MP 3080 00004) "Directory name for link tables"].

The name of the link table consists of the program name and the file extension ".l" (for "link").

If a corresponding link table already exists at the program selection and the programs in the link table are unchanged, the program is directly selected without linking it again.

If a part program does not contain any CPL sections but consists only of default NC commands (DIN) and subroutine calls, a program can also be called without being pre-processed. If a subroutine call is subsequently detected in the regular block preparation during program processing, this subroutine is "postlinked" if necessary.

If "Linking at program selection" is enabled, linking at program selection can be started or suppressed by:

- the user interface via the "Link NC Prg" toggle softkeys
- the programming in the part program
programs starting with "(DIN)" in their first program line are not linked. This also applies to subroutines identified accordingly.
- the parameterization for the MT-ProgSel program function block
- Via program parameters for the CPL command MCOPS(4,...)



For more detailed information on the subject of "linking", refer to the chapter "Linking NC programs" in the "IndraMotion MTX Programming Manual".

Prerequisites:

A part program can only be selected from a channel, if one of the following automatic NC operation modes has been selected from this channel:

- Automatic continuous block
- Automatic single block
- Automatic single step
- Automatic program block

Relevant IF signals

qCh_OpModeSel_00	Select operating mode, bit 0
qCh_OpModeSel_01	Select operating mode, bit 1
qCh_OpModeSel_02	Select operating mode, bit 2
qCh_OpModeSel_03	Select operating mode, bit 3
qCh_OpModePlc	PLC operating mode
qCh_Restart	Automatic restart

qCh_NCStart	NC start
qChReSelOff	Automatic reselection off
iCh_OpMode_00	Selected operating mode, bit 0
iCh_OpMode_01	Selected operating mode, bit 1
iCh_OpMode_02	Selected operating mode, bit 2
iCh_OpMode_03	Selected operating mode, bit 3
iCh_NCReady	NC ready
iCh_ProgRun	Program running
iCh_State_00	Program state, bit 0
iCh_State_01	Program state, bit 1
iCh_State_02	Program state, bit 2
iCh_State_03	Program state, bit 3
iCh_State_04	Program state, bit 4
iCh_State_05	Program state, bit 5

Tab. 6-4: Relevant IF signals

Relevant machine parameters (MP)

NCO/FileOrg/ SrchPath-SubProg (3080 00001)	Search path for subroutines
NCO/Mem/ SysMem/ BuffSizeNfs (3080 00002)	Buffer size for processing via NFS
NCO/FileOrg/ SrchPathLinkTab (3080 00004)	Directory name for link tables
NCO/FileOrg/ ChFileOrg/Ch[k]/ MaxOpenFile (7060 00210)	Number of open files per channel
CHAN/Ch[k]/Ini/ AutoProgSel (7060 00410)	Automatic program reselection
CHAN/Ch[k]/ (7070 00030)	Linking in the background (presetting)/linking at program selection

Tab. 6-5: Relevant machine parameters

Relevant PLC program blocks

MT_ProgSel	Program (de-)selection
MCOPS(4,...)	Program selection via CPL

Tab. 6-6: Relevant PLC program blocks

6.3.2 Handling Instruction: NC part program selection via PLC

Example of a program selection using the function block **MT_ProgramSelect**. Within the library MT_MTX.lib, the function block MT_ProgSel is available.

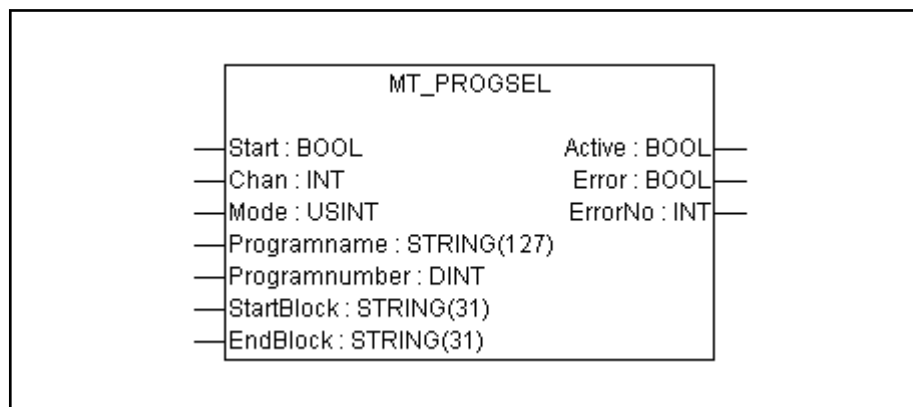


Fig. 6-4: Function block MT_ProgramSelect

Thus, a NC program can easily be selected from the PLC.

IW-Engineering/IndraLogic: Editing sequence

The following program sequence shows an exemplary call whose parameterization applies to the majority of applications:

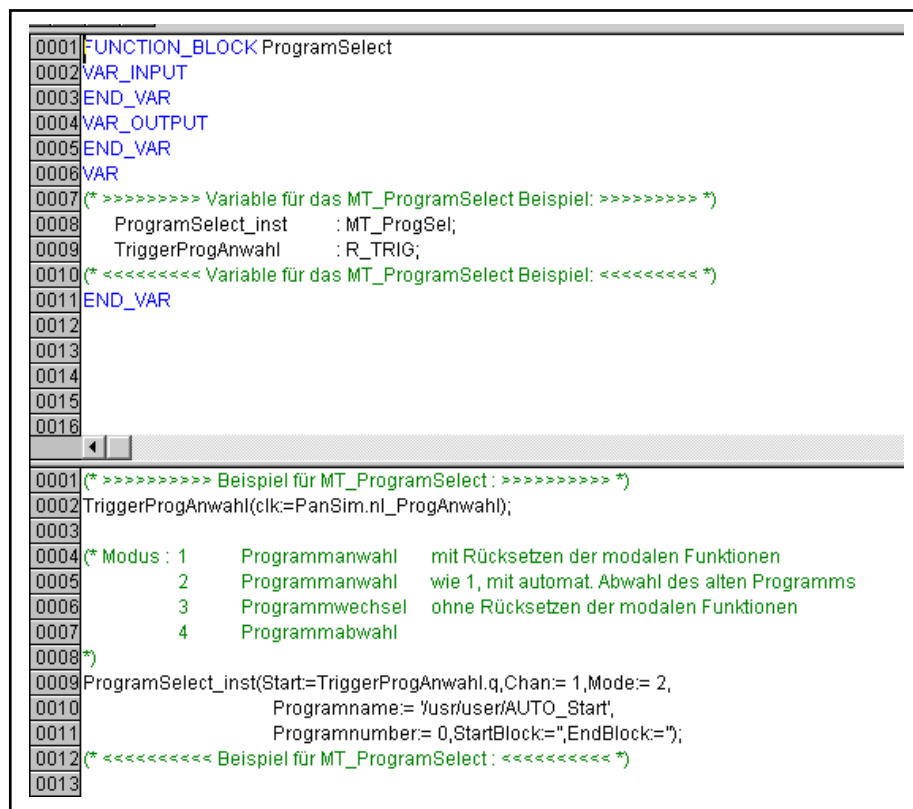


Fig. 6-5: Program sequence of MT_ProgramSelect

In channel 1, the example selects the program **"Auto_Start"** that is stored in the file system of the MTX in the directory **usr\user**.

An "old" program that might possibly still be active is deselected here, the new program **"Auto_Start"** is completely processed after the NC start at the interface.

Example	Documentation	Instruction
Example:		FB "ProgrammSelect.exp"
Instruction:		Edit PLC program
Documentation:	MTX PLC Interface	Integrate functional block

6.4 Automatic program reselection

6.4.1 Description

Function

After a program has been selected in a channel for the first time after the control startup, one program remains selected in this channel under the "Automatic" mode.

The following applies:

- A program aborted in "automatic" mode, is reselected immediately unless the abort has been caused by the selection of another program (e.g. after reset in the channel).
- If a program is aborted by changing the operation mode in the channel, the control selects the aborted program again when switching back to "automatic" mode in this channel.

The selected program can then be started by means of the channel-specific interface signal **qCh_NCStart** "NC-Start", if the **iCh_NCRReady** "NC ready" signal has been set at the channel interface.

Relevant IF signals

At the channel interface:	
qCh_ReSelOff	Autom. reselection off
qCh_NCStart	NC start
iCh_NCRReady	NC ready

Tab. 6-7: Relevant IF signals

Relevant machine parameters (MP)

CHAN/Ch[k]/Ini/ AutoProgSel (7060 00410)	Automatic program reselection.

Tab. 6-8: Relevant machine parameters

6.4.2 Handling instruction: Automatic program reselection

Applying

Automatic program reselection means that a program selected in "automatic" mode is reselected after a channel reset or a channel change has been completed in the "automatic" mode.

IW Engineering/configuration: Configuring parameters

- Set the parameter **AutoProgSel** (MP 706000410) "Automatic program reselection" accordingly:
"0"- automatic program reselection off.
"1"- automatic program reselection on.
- Trigger control reset.

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

Do not reset the control during machining.

	Documentation	Instruction
Instruction:		Editing machine parameters
Documentation:	MTX Machine Parameters	Automatic program reselection

Activating/deactivating:

It is also possible to activate/deactivate the automatic program reselection via the interface signal **qCh_ReSelOff** "Automatic reselection off":

- **qCh_ReSelOff = 1:**
Automatic program reselection is suppressed.
- **qCh_ReSelOff = 0:**
Automatic reselection permitted.



When this function is active, an error detected by the control during program selection cannot be reset via control reset (because of the automatic reselection, the error will re-occur again and again).

Remedy this status by

- Selection of an error-free program,
- Correction of the program containing the error or
- Correction of the faulty program selection, if applicable.

6.4.3 Automatic program selection after system startup

If the automatic program reselection is applied, the automatic program selection can additionally be enabled after system startup. Path and name of the program to be selected have to be comprised in the system data **SD.SysAPrg.LastSelProg**. If this system data contains the entry "NoAutoProgSel", no program is selected after control startup. Any other string starts a search for a program with this name and its selection. With each subsequent program selection, the **SysAPrg.LastSelProg** data is updated with the path and name of the selected program. The program selected last is selected again after the next control startup as soon as a program selection is possible again.

The following states can occur:

SD.SysAPrg.LastSelProg = "NoAutoProgSel"	
	No program selection after startup Entry is not updated.
SD.SysAPrg.LastSelProg = "invalid name"	

	No program selection, runtime warning Entry is updated with each program selection. This can be used as initial state. At runtime, the state only results if the program selected last has not been found.
SD.SysAPrg.LastSelProg = "correct name"	
	Program selection completed Entry is updated with each program selection.

Tab. 6-9: States after program selection

6.5 Canceling distance to go

6.5.1 Description

Function

"Cancel distance to go" discards all blocks already prepared when the function was triggered. It prepares them again including the block in which the machining was stopped. The channel goes into the "READY" state if the interface signal qiCh_NCRReady is set.

To ensure that the changes of a signal or a value in a program sequence is already applied to prepared blocks, the PLC can

- stop the program sequence at a suitable position, execute "Cancel distance to go" using the interface signal qCh_CancDist and
- continue the processing via the interface signal qCh_NCStart "NC start".



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

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

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



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

Restrictions

- "Cancel distance to go" always initiates an axis standstill
- "Cancel distance to go" is only effective in the automatic modes "continuous block", "single block", "single step" and "program block"
- "Cancel distance to go" is not possible in retracing operation (qCh_Retrace = 1)

Relevant IF signals

At the channel interface	
qCh_CancDist	Canceling distance to go L/H signal edge triggers the function
qiCh_NCRReady	NC ready Is set when "Cancel distance to go" was executed and the channel is again in the "READY" state "NC ready" is the precondition for the re-start via "NC start"
qCh_NCStart	NC start The selected program is continued at the end of the current block Prerequisite: The qiCh_NCRReady "NC ready" interface signal is set at the channel interface
qCh_Retrace	"Retrace mode" If active, "Cancel distance to go" is not possible

Tab. 6-10: Relevant IF signals

6.5.2 Handling instruction: Cancel distance to go

Activating:

1. Activate "Feed hold".
2. Trigger "Cancel distance to go" with the L/H signal edge to qCh_CancDist of the corresponding channel interface.
3. Continue the program via "NC start".

"Cancel distance to go" deletes all blocks already prepared by the block preparation when the function was triggered and prepares the blocks - including the block in which machining was stopped again - after an NC start.

IW Engineering/IndraLogic: Setting interface signals

- Optionally, trigger "Feed hold" (NC stop) by setting the interface signal **qCh_FeedHold = 1** "Feed hold".
- Set the interface signal to trigger **qCh_CancDist = 1**, trigger "Cancel distance to go"
If no "feed hold" was set, the control now triggers an implicit feed inhibit. Subsequently, the function "cancel distance to go" is executed.
- Program can be continued via "NC start"



- "Cancel distance to go" is only applied in the automatic modes "continuous block", "single block", "single step" and "program block" and in "manual data input NC block".
- If thread cutting is active, the interface signal **qCh_CancDist** must not be set.
- In the display, the end point is set to the current position and the indicated distance to go is deleted at the same time.
- If an NC program is continued, new compensation values are taken into consideration.

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

6.6 Asynchronous subroutines

6.6.1 Description

Function

Subroutines can interrupt an active part program under event control. These subroutines work asynchronously. After processing the asynchronous subroutine, there is a traversing motion towards a point of interruption (default) or a defined point of return (starting or end position) of the interrupted (frozen) block. The interrupted active part program continues starting with the interrupted block. All programmed instructions of the freeze block are executed again.

An asynchronous subroutine always triggers a "Cancel distance to go" internally and thus an axis standstill. Subsequently, all blocks - including the interrupted block - already prepared when triggering the asynchronous subroutine are discarded.

All discarded blocks - including the interrupted block - have to be prepared again after returning to the asynchronous subroutine. The changes that resulted for the active correction values are considered.



After returning from the asynchronous subroutine, the interrupted part program is prepared and processed. It is started with the interrupted block. All instructions of the freeze block are executed again.



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. These changed values also apply in the asynchronous subroutine.

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

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



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

Control functions for asynchronous subroutines in the part program:

- Log in:
Per channel, up to eight asynchronous subroutines can be logged in. These subroutines overwrite any previously logged in asynchronous subroutines.
- Deactivation:

Channel control

An asynchronous subroutine becomes inactive and cannot be started.

- **Reactivation:**

An asynchronous subroutine switched to inactive is logged in again. It can be started again.

- **Delete:**

Logs off an asynchronous subroutine.

- **Start:**

Only logged in asynchronous subroutines may be started.

In the following modes, a start is allowed:

Channel state	Automatic modes			Setup operation modes			
	Automatic	Single step	Single block	DebugCont	Approach reference point	Jog	Manual data input
Inactive	■	■	■	■	■	■	
Ready for program start	■	■	■	■	■	■	■
Ready	■	■	■	■	■	■	■
Running	■	■	■	■	■	■	■

Tab. 6-11: Operation modes

- **Restrictions under manual data input:**

At the end of Asup, manual data input is aborted and the channel goes to the INACTIVE state. This also occurs if other NC blocks have already been specified under manual data input or if a program has just been processed.

- **CPL functions for asynchronous subroutine:**

All the above mentioned NC control functions can also be programmed via CPL command.

- **Ncs functions for asynchronous subroutine:**

All the above mentioned NC control functions can also be implemented via Ncs functionality.

Return to contour functions after interruption of the part program:

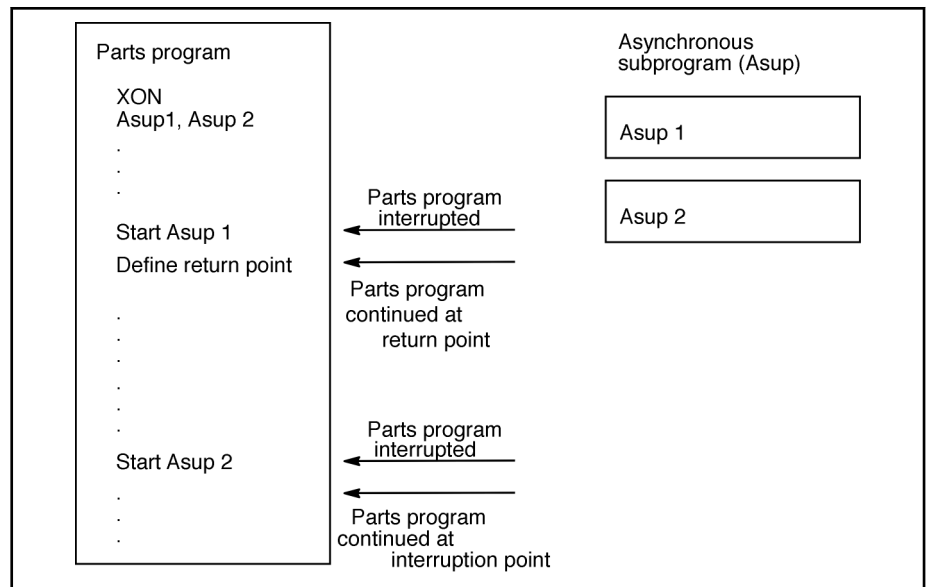


Fig. 6-6: Return to contour functions after interruption of the part program

After the asynchronous subroutine, a part program can either be continued at the breakpoint or as follows:

- Determination of a point of return to contour:
For each asynchronous subroutine, a point of return to contour is defined.
- Locally override point of return to contour:
An approach point defined in the asynchronous subroutine (locally) can be set which overwrites the point of interruption or a point to return to contour in the part program,
- Repositioning individual coordinates:
On a straight line, selected coordinates are traveled at a given feed to a defined point of return to contour or the breakpoint. After the asynchronous subroutine, all coordinates of the part program not yet selected are automatically traveled to the point of return to contour or the breakpoint.



If no point of return to contour is defined, the part program is continued at the breakpoint.



Damage to machine, workpiece, and personal injury possible if the point of return to contour is not defined as collision-free.

Check the course of the part program and the asynchronous subroutine!

An asynchronous subroutine (Asup) can be additionally unlocked and locked to interrupt active Asups:

- For unlocking, the switch-on function of the Asup is used (e.g. ASPENA(1,2) - Subroutine 1 may interrupt subroutine 2). The specified subroutines have to be logged in.
The start aborts the active Asup and the execution is continued in the new Asup.
- For locking, the switch-off function of the subroutine is used (e.g. ASPDIS(1,2) - Subroutine 1 may not interrupt subroutine 2).

Channel control

Thus, the subroutine cannot be started while the asynchronous subroutine is processed.

- Logging out an Asup also deletes the unlocking of its aborts.

Only the first Asup interrupting the part program creates a backup for re-entry. Returning to an uninterrupted Asup is not possible. A regularly exited Asup returns the control to the interrupted part program.

Restrictions

The following list provides an overview on the present restrictions:

Function	While leaving	While approaching	Remarks
Axis transfer, coordinate switching	Can be used	Can be used	The axis/coordinate configuration in the channel has to be the same before and after the asynchronous subroutine.
Tensor orientation	Can be used	Cannot be used	Approach is possible if the asynchronous subroutine was not triggered in a singularity block.
3D radius correction (G141/G142)	Can be used	Can be used	Approaching an inner corner with a larger tool radius can cause a contour violation.
Tapping (G63)	Cannot be used	Cannot be used	
Thread cutting (G33)	Can be used	Cannot be used	Leaving only if thread cutting with retraction is configured.
Coordinate coupling (SCC)	Cannot be used	Cannot be used	
Block switching at HS (HSB)	Can be used	Cannot be used	
Retracing on the contour	Cannot be used	Cannot be used	"Asynchronous subroutines" are not possible in retrace mode (qCh_Retrace = 1).

Tab. 6-12: Restrictions

Note the following for the CPL variable:

- In the asynchronous subroutine, CPL variables of the subroutine may only be accessed via permanent variables. No other access is possible.
- Asynchronous subroutines in the CPL code require twice the memory capacity:

MP **NCO/Mem/ChMem/Ch[k]/CplStack** (707000010) data areas for CPL programs: 12288 bytes per channel (default setting)

Relevant NC functions

The NC functions are valid for "asynchronous subroutines" in the individual channel:

ASPSET	Login
ASPDIS	Switch to inactive
ASPENA	Activate again after subroutine was switched to inactive.
ASPCLR	Clear (logging off)
ASPSTA	Start

ASPRTP	Definition of up to eight points to return to contour (one per asynchronous subroutine)
REPOSTP	Locally overrides a point to return to contour defined by ASPRTP with a point to return to contour defined in the asynchronous subroutine.
REPOS	The coordinates specified in REPOS travel on a straight line with the specified feed to the point of return to contour (break-point or specification of ASPRTP or REPOSTP).
REPOSDEF	- Locally (within an asynchronous subroutine) defines the returning feed. - Locally overrides a point of return to contour defined by ASPRTP with a point of return to contour defined in the asynchronous subroutine (same function as REPOSTP).

Tab. 6-13: Relevant NC functions

Relevant CPL functions

MCOPS(..)	With MCOPS, the NC functions ASPSET, ASPSET, ASPDIS, ASPENA, ASPCLR and ASPSTA can be executed for any channel
SDR(77,..)	In asynchronous subroutines, the start, interruption or end point of a machine coordinate can be queried in the freeze block of the part program (e.g. repositioning the coordinates close to the desired approach point with G78)
SD(77)	Query of the current approach point with regard to starting point, end point or interruption point

Tab. 6-14: Relevant CPL functions

Relevant IF signals

qCh_ASubx	Asynchronous subroutines x Starting a channel-related "asynchronous subroutine x"
iCh_ASubx	Asynchronous subroutines x Displaying an active channel-related "asynchronous subroutine x"

Tab. 6-15: Relevant IF signals

Relevant machine parameters (MP)

NCO/Mem/ChMem/ Ch[k]/CplStack (707000010)	Data areas for CPL programs

Tab. 6-16: Relevant machine parameters

6.6.2 Handling instruction: Asynchronous subroutines

Applying: Adjusting CPL memory for asynchronous subroutines with CPL code.

IW Engineering/configuration: Adjusting memories for CPL programs

Asynchronous subroutines with CPL code require double the memory capacity that is defined in the channel-specific parameter **CplStack** "Memories for CPL programs (SRs, variable, ...)" (7070 00010).



For CPL variables, observe the following:

In the asynchronous subroutine, CPL variables of the subroutine may only be accessed via permanent variables. No other access is possible.

	Documentation	Instruction
Instruction:		Editing machine parameters
Documentation:	MTX Machine Parameters	Adjusting memories for CPL programs

Set up asynchronous subroutines for use in various operation modes:

Linking interface signals

Activating: **Prerequisites:**

- All necessary asynchronous subroutines exist and are correspondingly incorporated in the part program.
- Point of return to contour or breakpoints are defined as **collision-free**.

Automatic mode:

- Selecting one of the above mentioned automatic modes.
- Starting the part program (asynchronous subroutines are for example started directly in the NC code or by the CPL code).

Setup mode:

- Selecting one of the above mentioned setup modes.
- Logging in to an asynchronous subroutine.
- Starting the logged-in asynchronous subroutine.
- Select point of return to contour if required.

Logging in to the asynchronous subroutine in the current channel

IW Operation/NC programming: Logging in via NC function

- **Asynchronous subroutine in the NC main program**
ASPSET(<SRNo.>,<SR-Name>{,<Flags>})
 - **<SR no.>**
Number of the subroutine 1..8.
 - **<SR name>**
Name of the subroutine; with or without path information. If the path is missing, the file is searched according to the parameter **SrchPathSubProg** "Search path" (3080 00001).
 - **<Flags>**
Switch to affect the behavior after the call:
 - 00 neither linking nor automatic start.
 - 10 linking, but no automatic start.
 - 01 no linking, but automatic start.
 - 11 linking and automatic start.

	Documentation	
Documentation:	MTX Programming Manual	Logging In to Asynchronous Subroutine

Activate the asynchronous subroutine in the current channel. Only activated subroutines can be called when the relevant event occurs.

IW Operation/NC programming: Activating via NC function

- **ASPENA(<SR no.>)**
<SRNo> Number of the subroutine 1..8.



The asynchronous subroutine has to be logged in.

	Documentation	Instruction
Documentation:	MTX Programming Manual	Activating asynchronous subroutine

Tab. 6-17: IW Operation/NC programming: Activating via NC function

Defining the point at which machining is continued after the asynchronous subroutine.

IW Operation/NC programming: Defining via NC function

- **ASPRTP(<SR no.>,<Point>)**
<SR no.>
 Number of the subroutine 1..8.
 - or -
<SR no.> = -1:
 Define the desired point of return to contour for all asynchronous subroutines of the current channel.
<Point>
 Desired point of return to contour:
 - 1: Starting point
 - 2: End point
 - 3: Interruption point
- If the point of return to contour is not defined by the NC function **ASPRTP(<SRNo.>,<Point>)**, the point of return to contour at the end of an asynchronous subroutine defined using G77 is used.



- The point of return to contour is always approached on a straight line.
- The set desired point of return to contour is cleared by a re-set or M30.
- Any correction changes made in an asynchronous subroutine are automatically included in the internal calculation of the required point of return to contour.
- Using the NC function **REPOSTP(<Point>)** "Defining point of return to contour in the asynchronous SR", the defined point of return to contour can temporarily be overridden in the asynchronous subroutine.
- The NC function **REPOSDEF({F<Value>} {,TP<Point>})** can be used to specify the feed rate F. The point of return to the contour can be specified the same way (cf. REPOSTP).

Return to contour functions after interruption of the part program:

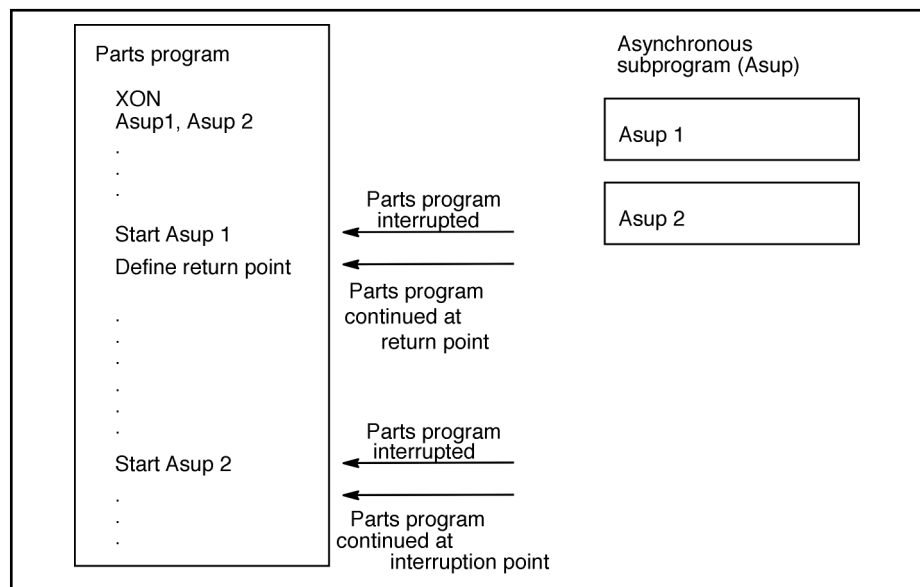


Fig. 6-7: Point of return to contour in the program code

	Documentation	Instruction
Documentation:	MTX Programming Manual	Defining point of return to contour

Starting the asynchronous subroutine:

The start of an asynchronous subroutine can be triggered via PLC, NC function or an exception handling.

IW Engineering/IndraLogic: Starting via interface signal

Use the interface signal **qCh_ASub1..8** "Asynchronous subroutine 1 ... 8" to start one out of eight asynchronous subroutines



- When asynchronous subroutines are active, the following NC functions may not be active.
- Only one active asynchronous subroutine is permitted at a time and subroutines may not be nested.

	Documentation	Instruction
Instruction:		Editing PLC signal
Documentation:	MTX PLC Interface	Starting asynchronous subrou- tine

IW Operation/NC programming: Starting via NC function (for a non-automatic start)

- **ASPSTA(<SR no.>{,<channel no.>})**
 - **<SR no.>**
Number of the subroutine 1..8.
 - **<Channel no.>**
Target channel in which <SR no.> is to be called.
If not programmed, <SR no.> is called in the current channel.



- Only one active asynchronous subroutine is permitted at a time and subroutines may not be nested.
- The following CPL functions can be used:
 - **MCOPS(..)**
With MCOPS, the NC functions ASPSET, ASPSET, ASPDIS, ASPENA, ASPCLR and ASPSTA can be executed for any channel.
 - **SDR(77,..)**
In asynchronous subroutines, the start, interruption or end point of a machine coordinate can be queried in the interruption block of the part program (e.g. repositioning the coordinates close to the desired approach point with G78).
 - **SD(77)**
Query of the current approach point with regard to starting point, end point or interruption point.
- When asynchronous subroutines are active, the following NC functions may not be active:
 - **G6** "Splines / nurbs"
 - **G33** "Thread cutting"
 - **G41, G42** "Radius correction"
 - **G141, G142** "3D radius correction"
 - **PUN** "Punching"
 - **NIB** "Nibbling"
 - **AXC(...)** "Axis coupling"
 - **COORD(..)** "Axis transformations with automatic by-pass of singularities"
 - **G63** "Tapping without compensating chuck "

	Documentation	Instruction
Documentation:	MTX Programming Manual	Starting asynchronous subrou- tine

IW Operation/NC programming: Starting via exception handling

The asynchronous subroutine is explicitly logged in and automatically started when the event occurs:

Measuring on a contour

- **MCO(...,RE 2,AS <SrNo>,...)**
 - **RE 2**
Reaction to measuring result: Start asynchronous subroutine in the current channel
 - **<SR no.>**
Number of the subroutine 1..8.

	Documentation	Instruction
Documentation:	MTX Programming Manual	Measuring on a contour

Error reaction at system axis coupling

- **SD.SysAxCoupleCmd[<axis>].AbortAsupNo=<Sr no.>**
SD.SysAxCoupleCmd[<axis>].AbortAsupChan=<Channel no.>
 Start asynchronous subroutine <Sr no.> in the channel <Channel no.> if an error occurs in the system axis coupling.

	Documentation	Instruction
Documentation:	MTX Functional Description	System axis coupling/exception handling

Deactivating: Switching off and logging off an asynchronous subroutine.

IW Operation/NC programming: Switching off via NC function

- **ASPDIS(<Sr no.>)**
<SRNo> Number of the subroutine 1..8.



- The asynchronous subroutine has to be logged in.
- An asynchronous subroutine is no longer effective if there is a reset and if an active asynchronous subroutine is interrupted. Defined points of return to contour are deleted.
- After **M30**, all defined points of return to contour are deleted.

	Documentation	Instruction
Documentation:	MTX Programming Manual	Switching off asynchronous subroutine

Logging off an asynchronous subroutine.

IW Operation/NC programming: Logging off via NC function

- **ASPCLR(<Sr no.>)**
<SRNo> Number of the subroutine 1..8.



The asynchronous subroutine has to be logged in and deactivated.

	Documentation	Instruction
Documentation:	MTX Programming Manual	Logging Off Asynchronous Sub-routine

6.7 Creating channels

6.7.1 Description

The following handling instructions describe creating the first and/or further channels in IndraWorks Engineering and in IndraLogic.

6.7.2 Handling instruction: Creating a channel

The goal of these handling instructions is to insert a new channel within the overall system IndraMotion MTX.

IW Engineering/channel interface: Creating a channel interface

		Instruction chapter 6.7 "Creating channels" on page 113
Instruction:		Creating new channel interface

IW Engineering/configuration: Editing parameters

Increase the value of the parameter **NofCh** "Channel number (9040 00001)" by "1".



- The parameter **ActCh** "Active channel in the display after activation (6001 00010)" determines which channel is active after activating IndraWorks operation.
- Adapt the parameter **SupprChDisp** "Hide channel within the interface (6005 00010)" to hide a channel in IndraWorks Operation

	Documentation	Instruction
Instruction:		Editing machine parameters
Documentation:	MTX Programming Manual	Specifying number of channels

6.7.3 Handling instruction: Creating a channel interface

The goal of this handling instruction is to add a new channel to the PLC.

IW Engineering/channel interface: Creating new channel interface

Right click on the node "Channel Interfaces" and left click to select "New Channel Interface".



The node number for channel interfaces always has to be unambiguous. Thus, each node number may occur only once within the channel interface.

Channel control

	Documentation	Instruction chapter 6.7 "Creating channels" on page 113
Instruction:		Creating new channel interface
Documentation:	MTX Commissioning Manual	Creating new channel interface

7 Dynamics and velocity control

7.1 Axis jump capability

7.1.1 Description

Function

At contour knees, the axis jump ability is active. It is determined by the axis acceleration reserve and in case of an active axis-specific jerk limitation (JerkControl(1)) also by the axis jerk reserve.

The axis acceleration reserve in [m/s²] determines the maximum allowed acceleration (acceleration jump) at block transitions for machining axes which may occur between two successive interpolator cycles. The axis jerk reserve [m/s³] determines accordingly the maximum permitted jerk (acceleration jump). The control internally calculates an accordingly reduced path velocity. If the axis-specific jerk limitation (i.e. JerkControl(0)) is deactivated, the axis jerk reserve is not significant.

Thus, it is possible to:

- prevent the control - during the processing of part programs in "continuous block" mode - from issuing velocity command values at block transitions dynamically overcharging a connected drive.
- shorten the machining time for a workpiece if G8 (path slope, see [chapter 7.4 "Path slope" on page 120](#)) is active.

Relevant NC functions

G8	Activate path slope The control tries to generate - even at block transitions - a velocity as constant as possible and as high as the programmed feed. Refer to chapter 7.4 "Path slope" on page 120 for more details. During this process, the control monitors the available braking distance.
JerkControl	Enabling axis-specific jerk limitation, that means JerkControl(1) The control automatically guarantees that each path-relevant axis maintains its corresponding valid maximum value for velocity, acceleration and jerk (see chapter 7.5 "Basics of the axis-specific jerk limitation" on page 124). In case of an active, axis-specific jerk limitation, the subsequent functions DefTangTrans and PrecProc are ignored, i.e. they do not have any effects.
DefTangTrans	Block transition without reduction of velocity Ensures that path velocity at block transition is not reduced below a defined contour knee angle due to the axis jump capability. The basic setting of the minimum angle specified in the machine parameter 7030 00310 can be influenced under program control.
PrecProg	Accuracy programming (see chapter 9 "Accuracy" on page 141) Reduction of path velocity at block transition.

Tab. 7-1: Relevant NC functions

Relevant machine parameters (MP)

1010 00012	Axis acceleration reserve
1010 00013	Axis jerk reserve
7030 00310	Minimum angle to consider the axis jump capability

Tab. 7-2: Relevant machine parameters

7.1.2 Applying

NOTICE

Workpiece or machine can be damaged!

Do not reset the system during machining.

1. Enter the maximum permissible axis jump accelerations for your application into the machine parameter 1010 00012. During the calculation, take the mechanical data limits of the machine as well as the expected average load into account.
2. Adapt the machine parameter 7030 00310 (minimum angle to consider the axis jump ability).
3. Trigger system reset.

7.1.3 Activating

Programming G8 and activating JerkControl (JKC(1)).

When G8 is active, the axis jump capability is automatically taken into account when part programs are processed in "continuous block" mode.

7.1.4 Deactivating

-

7.2 Limiting the path acceleration

7.2.1 Description

Function

The function ensures that a parameterizable positive/negative acceleration is not exceeded on the programmed path.

Thus, the processing time of a part program can be insignificantly longer, but a creeping dynamic overload of the drives is avoided. This is especially advantageous when processing often takes place in the limit range of load and dynamics.

Relevant machine parameters (MP)

7030 00210	Path acceleration Max. permissible positive path acceleration on the programmed contour.
7030 00220	Path braking acceleration Max. permissible negative path acceleration on the programmed contour.

Tab. 7-3: Relevant machine parameters

7.2.2 Applying

NOTICE

Workpiece or machine can be damaged!

Do not reset the system during machining.

1. Assign the desired acceleration values to MP 7030 00210 and MP 7030 00220.
2. Trigger system reset.



Both parameters have to be assigned with values > 0.

7.3 Look-ahead

7.3.1 Description

Function

The "look-ahead" function enables the NC to check the programmed path across several single machining blocks already before machining.

Thus, problematic path characteristics can be detected and it can be reacted accordingly.

Look-ahead is the basis for all functions which

- perform a contour-dependent modification of the path velocity and
- influence a programmed contour, e.g. by automatically inserted additional contour segments.

For correct functioning of the look head, it is generally required that a sufficient number of blocks remain in the block preparation. Thus, the control is able to access block information it absolutely requires for the "advance calculation" of different strategies.

A minimum of ten blocks should be reserved for machining channels. If a sequence of many very small contour segments is to be processed, it can be required to increase the number of blocks.

For CPL channels, three blocks are usually sufficient for the look-ahead function, as no contour machining is performed in the CPL channel.

Define the number of blocks in the block preparation per applied channel using the configuration parameter 7060 00110. Via the "BlkNmb" function, the block number can temporarily be restricted per the part program.

By default, 30 blocks each are preset for the machining channels 1 and 2, and three blocks for the CPL channel. Reserve a minimum of 30 blocks for additional machining channels.

Using the configuration parameters 7060 00120 and 7060 00130, the total number of the blocks available for the look-ahead is divided between various functional areas within the control, e.g. velocity profile computation, splines, etc.

If the presetting of the machine parameter 7060 00110 can be critical or limited for functions using the look-ahead function, we draw your attention to this fact within the respective functional description.



Note that the look-ahead function requires approximately 8k class 2 system memory per block!

Before **planning an increase** in block numbers, check whether there is sufficient memory.

A control reset is required to apply modifications in MP 7060 00110.

Relevant machine parameters (MP)

7060 00110	Total number of blocks in block preparation
7060 00120	Maximum number of blocks for block look-ahead
7060 00130	Number of blocks for block look-ahead in percentages

Tab. 7-4: Relevant machine parameters

Parameter <AdditionalNCBlockMem> in the "startup.xml" file

The CNC control requires memory especially for the NC block data. The memory requirement is estimated using the configuration and is reserved during control startup for this purpose. In some use cases, this evaluation cannot be sufficient so that the waiting state of the channel 20 ("waiting for resources") or the channel error 1927 is generated in case of lack of memory.

If these effects result, the memory area for the NC block data has to be increased using the parameter AdditionalNCBlockMem in the "startup.xml" file. The parameter is specified in bytes and added to the internally determined memory size.

Due to the variety of applications and configurations of the CNC control, it is not possible to specify a value valid for all applications. As a rule of thumb, 10000 bytes per configured NC block in the channel (Macoda parameter 7060 00110) can be assumed.

For example, if 300 blocks were configured for channel 1 and 500 for channel 2, a good value is:

$$(300+500)*10000 = 8000000$$

It is required that NC programs with such a high number of blocks are processed simultaneously in both channels.

Example:

Entry for the "startup.xml" file:

```
<AdditionalNCBlockMem>8000000</AdditionalNCBlockMem>
```

The template of the "startup.xml" file is contained in the "/feprom" directory. The changed file has to be stored in the directories "/usrfep" or "/" (root directory) of the control.

The option to increase the memory size by creating another unused channel with a high number of blocks is not recommended. The memory area is increased by the additional channel but the memory requirement and the administrative expense is also increased by the additional channel. The parameter in the "startup.xml" is significantly more memory-efficient.

Relevant special functions

BlkNmb	Limits the maximum number of blocks prepared in the block preparation. For more details, refer to the "IndraMotion MTX Programming Manual".

Tab. 7-5: Relevant special functions

7.3.2 Handling instruction: Look-ahead - Configuring look-ahead

Configuring the look-ahead function.

Applying: **IW Engineering/configuration: Configuration**

NofBIPrep "Total number of blocks for block preparation and interpolation" (7060 00110)

Maximum number of function-related blocks:

- **MaxBIVelCalc**
"Maximum number of blocks for jerk-limited velocity control" (7060 00120)
- **MaxBISpline**
"Maximum number of blocks for spline functions" (7060 00120)
- **MaxBITICorr3D**
"Maximum number of blocks for 3D milling cutter path correction" (7060 00120)

Distribution of the entire number of blocks in percentages:

- **PercVelCalc**
"Number of blocks for jerk-limited velocity control" in percent (7060 00130)
- **PercSpline**
"Number of blocks for spline functions" in percent (7060 00130)
- **PercTICorr3D**
"Number of blocks for 3D milling cutter path correction" in percent (7060 00130)



- Per block, the block look-ahead function requires approximately 8K system memory.
- Note that in addition to the set number of blocks for the block look-ahead, the block demand of the interpolator increases when SHAPE is active (up to the SHAPE order). These blocks also have to be considered in the parameter **NofBIPrep** "Total number of block for block preparation and interpolation" (7060 00110)!
- The parameter **NofBIPrep** "Total number of blocks for block preparation and interpolation" (7060 00110) has to be assigned at least with a value of "3".
- Before planning an increase of block numbers, check whether there is sufficient memory.

	Documentation	Instruction
Instruction:		Editing machine parameters
Documentation:	MTX Machine Parameters	Look-ahead

IW Operation/machine: Triggering system reset

To apply the changed machine parameters, reset the system.

IW Operation/NC programming: Temporary limitation of the look-ahead function

- Using the NC function **BlkNmb(<Number>)**, the maximum number of blocks in the block preparation can be limited.
- To switch off the NC function, program **BlkNmb(0)**.

	Documentation	
Documentation:	MTX Programming Manual	Look-ahead

7.4 Path slope

7.4.1 Description

Function

This function tries to keep the programmed feed velocity as constant as possible - even at block transitions - while machining in feed mode.

It observes the programmed contour and the available braking distance using the look-ahead function. It is always checked whether the distance from the end point of the current block to the end point of the next block is long enough to decelerate the current path velocity to zero. If this is not the case, the next blocks are also included in the feed calculation and limited by the maximum number of blocks defined in the machine parameters. Thus, impermissibly high accelerations on block change are avoided.

If the function detects an unsteady contour transition (corner), feed velocity - taking into account the axis jump capability (see [chapter 7.1 "Axis jump capability" on page 115](#)) - is reduced only as far as required to travel around the corner due to the available drive dynamics.

Thus,

- a better quality of contour transitions and above all
- shorter processing times are achieved,

as it is no longer necessary to decelerate to path velocity $v=0$ at each contour transition.



At the end of the programmed path, it is always decelerated to $v=0$.

Optionally, the calculated velocity profile can be smoothed further using the axis-specific jerk limitation or the SHAPE functionality, as any jumps of the path acceleration are distributed to several interpolation cycles. The number of cycles can be programmed and is limited to 100 interpolation cycles.

Restrictions

- "Path slope" acts only on machining axes (synchronous axes).
- After a G0 block, velocity is always decelerated to $v=0$ if G0 (NIPS) was not programmed.
- "3" has to be assigned as minimum value to the machine parameter MP 7060 00110 (number of blocks in block preparation).

- A look-ahead function set too low can cause an unnecessary reduction of path velocity.
- The function "BlkNmb", if activated in the part program, limits the jerk-limited velocity control so far that fewer blocks than defined are available in the relevant machine parameters.
- If "path slope" is active, the execution time of a block for machining has to be greater than the execution time of auxiliary functions (incl. acknowledgement) programmed in the same block.

Execution time is primarily defined by the programmed distance to be traversed as well as by the command feed.

Relevant NC functions

G8	Activate "path slope" JerkControl(1): The parameterizes axis jerk limitation "AxJerk" is applied. JerkControl(0): Optionally, a SHAPE parameter can be programmed: G8 (SHAPE<value>). If G8 is programmed without the optional SHAPE parameter, the order defined in MP 7050 00320 is applied.
G9	Disable "path slope" At block transitions, it is always decelerated to v=0.

Tab. 7-6: Relevant NC functions



The G8 and G9 NC functions form one group and thus deselect each other.

Relevant machine parameters (MP)

1010 00020	Maximum jerk in feed mode	
	0:	Presetting
	0...1000000:	Maximum jerk in m/s ³
	Important for the active axis-specific jerk limitation: Maximum jerk in feed mode	
1010 00022	Maximum jerk in rapid traverse mode	
	0:	The value from 1010 00020 is effective.
	0...1000000:	Maximum jerk in m/s ³
	Important for the active axis-specific jerk limitation: Maximum jerk in rapid traverse mode	
7050 00320	Only important if the axis-specific jerk limitation is disabled. Default order for SHAPE if G8 is programmed without SHAPE parameter.	
	0:	Presetting
	0...100:	Number of interpolation cycles

7050 00330	Only important if the axis-specific jerk limitation is disabled. Minimum SHAPE order for path jerk limitation. Limits the maximum permissible path jerk (acceleration change) for G8 and G9.	
7060 00110	Required total number of blocks for the look-ahead function including the interpolation in the channel.	
	0:	Presetting
	1 ...1000:	Number of blocks
	On the side of the interpolator, the maximum number of blocks required is the number set or programmed as SHAPE order (number of interpolation cycles) (1 cycle = 1 block). If the interpolation NC blocks are longer, the block demand of the interpolator is reduced accordingly. All NC blocks not required by the interpolation are available to the "look-ahead" functionality.	
7060 00120	Maximum number of blocks for look-ahead (Look ahead). This maximum value is only relevant if MP 7060 00110 was configured sufficiently high and if a corresponding large part is assigned to jerk-limited velocity control in the parameter 7060 00130.	
	1:	Presetting
	0...1000:	Number of blocks
7060 00130	Number of blocks for look-ahead in percentages. (Look ahead). Applied for the groups "Velocity profile", "Buffered blocks", "Spline" and "Others".	
	100:	Presetting
	0 .. 100:	Percentage
	The individual parameter [1] of this parameter specifies the percentage of the "Total number of blocks for look-ahead" (7060 00110) available for jerk-limited velocity control.	

Tab. 7-7: Relevant machine parameters

Relevant special functions

BlkNmb	Limits the maximum number of blocks prepared in the block preparation. Higher priority compared to the set values of the relevant machine parameters. For more details, refer to the "IndraMotion MTX Programming Manual".

Tab. 7-8: Relevant special functions

7.4.2 Handling instruction: Path slope

Activating/deactivating the path slope. This function tries to keep the programmed feed velocity - even at block transitions - as constant as possible while machining in feed mode.

Applying: Prerequisite:

Note that in addition to the set number of blocks for the block look-ahead, the block demand of the interpolator increases when SHAPE is active (up to the SHAPE order). These blocks also have to be considered in the parameter **NofBlkPrep** (7060 00110) "Number of blocks in block preparation"!



Before planning an increase of block numbers, check whether there is sufficient memory.

IW Engineering/configuration: Determining SHAPE order

The parameter **ChShapeG8** (7050 00320) determines on how many interpolation steps a jump of the path acceleration is distributed (this parameter is not important at the active NC function "Jerk control").

NC: Configure SHAPE

		Instruction
Instruction:		Configure SHAPE
		NC: Configuring SHAPE (alternative)

IW Engineering/configuration: Determining total number of blocks

The number of NC blocks a channel has to provide in total for the block preparation and interpolation is set in the parameter **NofBlkPrep** (7060 00110).

IW Engineering/configuration: Determining jerk-limited velocity control

The parameter **MaxBlkVelCalc** (7060 00120) indicates the maximum number of the velocity profile generation.

IW Engineering/configuration: Determining number of blocks for jerk-limited velocity control in percent

The maximum number of blocks **NofBlkPrep** (7060 00110) in percentages the velocity profile generation can be assigned to at maximum is defined in the parameter **PercVelCalc** (7060 00130).



When specifying the number of blocks, also consider other functions if applicable (e.g. spline online calculation).

If the number of interpolation cycles is not sufficient, change the ratio in **PercVelCalc** (7060 00130) or the total number in **NofBlkPrep** (7060 00110).

NC: Configuring look-ahead

		Instruction chapter 7.3.2 "Handling instruction: Look-ahead - Configuring look-ahead" on page 119
Instruction:		Configuring look-ahead

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 Operation/NC programming: Activating/deactivating the path slope

- For the activation, program **G8**

	Documentation	
Documentation:	MTX Programming Manual	Path slope

Program **G8**.

Deactivating:

IW Operation/NC programming: Activating/deactivating the path slope

- For the deactivation, program **G9**

	Documentation	
Documentation:	MTX Programming Manual	Path slope

7.5 Basics of the axis-specific jerk limitation

7.5.1 Description

Function

For a following error-free machining of a smooth motion sequence free of jerks, the specified maximum axis jerks must not be exceeded. For this purpose, the resulting axis velocities, axis accelerations and axis jerks are limited to the correspondingly stored maximum axis values. The rough interpolation guarantees that all valid maximum values are maintained. Thus, the first, second and third derivation of the path is monitored. This reduces the peak load of the mechanical system and drive components and can avoid servo errors in borderline cases.

Furthermore, the target of the axis-specific torque limitation is to commission machine tools easier. The maximum value for velocity, acceleration and jerk per machine parameter is defined for each axis. All necessary path parameters are then automatically determined and used in the control, i.e. no SHAPE parameters are necessary on user level.

The axis-specific jerk limitation is implemented internally via SHAPE. In case of an active axis-specific jerk limitation, the control determines automatically an appropriate SHAPE order from the path-relevant axes using axis jerk and axis acceleration.

The axis-specific jerk limitation can be switched on/off. It is switched off by default. It is activated by the NC function using the command:

JerkControl or JerkControl(1)	Enable jerk limitation
JerkControl() or JerkControl(0)	Disable jerk limitation
Short form: JKC(. .)	

Tab. 7-9: Syntax JerkControl, JKC

If the jerk limitation is activated, G8 and G9 are always called without parameter list. If, nevertheless, G8 and G9 are programmed with the present SHAPE parameters (e.g. G8(SHAPE<n>), G9(SHAPE<n>), G9(ASHAPE), ASO etc.), a runtime error occurs.

In case of active, axis-specific jerk limitation, all axes of the channel (i.e. all axes which are channel members or become channel members as a result of axis transfer) are considered by default for the determination of the internal SHAPE order. Each path jerk is intended for the respectively weakest path-relevant axis. The list of path-relevant axes can be programmed using the following command:

JerkAxlst (...) JAL (...) or	Set list of path-relevant axes.

The command JerkAxlst can be used to change the list of path-relevant axes. This command can be used for example to remove an individual axis, which is in the channel but does not participate in the actual machining, from the SHAPE calculation. If this axis only tolerates a small jerk, it otherwise slows down the whole machining cycle unnecessarily.

The list can be programmed absolutely or relatively (see "MTX Programming Manual")

JerkAxlst (<SysAxisList>)	
	Absolute list of system axis names, can, for example, be stored in the NC reset string (MP 7060 00010)
JerkAxlst (DEF)	Absolute list (e.g. from NC reset string)
JerkAxlst (Def, -, <SysAxisList >)	
	Relative list of system axis names

In the first two cases, specify the list of axes considered in the internal SHAPE calculation. In the third case, individual axes are removed from the absolute list.

For the active jerk limitation, the machine parameter "Maximal jerk in feed mode [m/s³]" (MP 1010 00020) has to be set $\neq 0$ for each axis, since otherwise a warning occurs at runtime when the jerk limitation is activated. Optionally, the maximum jerk value for the rapid traverse mode has to be set via the individual machine parameter "Maximum jerk in rapid traverse mode [m/s³]" (MP 1010 00022). Analogously to the acceleration, the jerk values (FeedAxJerk and RapidAxJerk) stored in the machine parameters can be modified using the NC function. The following command is available to change the jerk values:

AxJerk(<Val>)	Modify maximum axis jerks Axis name and jerk value Data for several axes are separated by a comma.

The axis-specific jerk limitation always has an effect for C² constant block transitions. If the block transition is not C² constant, the path velocity has to be reduced to (almost) zero to avoid dynamic jumps (unsteadiness). The axis jump capability (see [chapter 7.1 "Axis jump capability" on page 115](#)) avoids in these cases that the velocity is reduced too much. The machine parameter for the "axis jerk reserve" reduces the path velocity more than the axis acceleration reserve.

Asynchronous axes In case of asynchronous axes, the internal SHAPE order is derived from the axis jerk. If an axis jerk $\neq 0$ is entered into the machine parameters, asynchronous axes automatically travel in a jerk-limited way. If the machine parameter for axis jerk = 0, the response does not change.

There is no AxJerk programming for asynchronous axes, but only for channel axes. For asynchronous axes, the machine parameter is always applied.

Jog mode When jogging the axes, the value set in "Jog jerk" (MP 1010 00021) is applied.

Relevant NC functions

JerkControl	Switches on or off the jerk limitation.
JerkAxList	Modifies or determines the list of the possible system axes considered when calculating the SHAPE order of the active jerk limitation.
AxJerk	Modifies the maximum axis jerk of the channel axes: Blends machine parameter 1010 00020 and/or machine parameter 1010 00022.

Tab. 7-10: Relevant NC functions

Relevant machine parameters (MP)

1010 00012	Axis acceleration reserve
1010 00013	Axis jerk reserve
1010 00020	Maximum jerk in feed mode
1010 00022	Maximum jerk in rapid traverse mode
1040 00055	Maximum spindle jerk for active positioning interface (if necessary)

Tab. 7-11: Relevant machine parameters

7.5.2 Applying

1. Set maximum jerk values specific for the manufacturer in the machine parameter AX/Dr[i]/Jerk/FeedAxJerk (MP 1010 00020) for each axis. If required, suitable values can also be entered into AX/Dr[i]/Jerk/RapidAxJerk (MP 1010 00022).
2. If necessary, store an absolute list of system axes via the NC reset string (MP 7060 00010) which can be considered when calculating the SHAPE order in case of an active jerk limitation.
3. Set acceleration margin (MP 1010 00012) and Set jerk reserve (MP 1010 00013) axis by axis.
4. In the machine parameters
SP/SpFunc/Sp[j]/Gear/GearStep[k]/MaxSpJerkPosCtrl
(MP 1040 00055), assign a maximum jerk to all spindle/c-axes without compensating chuck
5. Trigger system reset
6. Activate jerk limitation (JerkControl(1)).

7.5.3 Activating

Program JerkControl(1)

7.5.4 Deactivating

Program JerkControl (0)

7.6 Parameters of the axis-specific jerk limitation "JerkControl, JKC"

7.6.1 Description

Function

The desired path velocity (programmed feed, F-value) and the desired path acceleration (NC function, PathAcc) for accelerations and decelerations in the channel are specified by the NC program. These values can partially be preset by the machine parameters 7030 00120, -210 and -220. They are maximum values the control intends to reach. If the axes are overcharged by this, the control reduces the maximum values based on the desired ones so that the axis limit values - that means the maximum axis velocities, axis accelerations and axis jerks are not exceeded. The control autonomously calculates the best SHAPE order.

The reduction consists of three steps:

1. The control reduces the maximum path acceleration so that at least the axis limit values are not exceeded if the path velocity is zero.
2. The control reduces the maximum path velocity so that at least the axis limit values are not exceeded if the path acceleration is zero. The maximum axis velocities are not exceeded in any case. That applies to all path accelerations.

Whenever there is a linear relation between the distance which the tool tip (TCP) covered since the starting point of the NC block on its path and the positions of the participating axes, steps 1 and 2 are sufficient to keep the axis limit values and step 3 described in the following does not have any effect. In most of the cases, this applies to linear blocks.

3. The control reduces the maximum path velocity and the maximum path acceleration together so that even the maximum axis accelerations and jerks are not exceeded; especially if the maximum path velocity and acceleration are present at the same time. The user can specify what has a higher reduction using the parameter "KINLIMWEIGHT" or "KLW" (kinematic limitation scaling):
 - The higher the value set, the more the maximum path acceleration is reduced and the higher the resulting maximum path velocity.
 - The smaller the value set, the more the maximum path velocity is reduced and the higher the resulting maximum path acceleration

The parameter is provided with a value range from 0.001 to 1000. The parameter can be preset by the machine parameter 7030 00 420. In most of the cases, a value of 1 is recommended.

It is an extreme example that the maximum path velocity and path acceleration are present at the same time. This causes a reduction that is too strong in many cases. Thus, the user can reduce step 3 via the parameter "KINLIMLEVEL" or "KLL" (kinematic limitation level):

- KLL = 0:
Step 3 is executed as described.
- KLL = 1:

Step 3 to maintain the maximum axis accelerations is executed as described. A less strong variant is used for the maximum axis jerks.

- KLL = 2:

Only the less strong variant for maximum axis jerks is executed in step 3.

- KLL = 3:

Step 3 is completely omitted. Thus, the parameter KLLW remains ineffective.

The parameter KLL can be preset by the machine parameter 7030 00 410.

The maximum axis accelerations are only safely kept for KLL = 0.1 and the maximum axis jerks only for KLL = 0. Exceedance can occur for all other values. Its frequency increases with KLL, but remains in most of the cases - even for KLL = 3 - very low.

The processing time of an NC program is affected by the following factors:

1. KINLIMLEVEL, KLL

The execution time reduces when the value increases.

2. KINLIMWEIGHT, KLLW

The influence on the processing time depends on the NC program. The more the locations at which has to be accelerated and decelerated, the better a value smaller than 1. For free-form surfaces with a few or no edges, a value greater than 1 is often more suitable.

3. Desired path velocity and acceleration

When specifying a low path acceleration for example, the reduction in step 3 becomes smaller. This means a smaller reduction in the maximum path velocity.

If the reduction steps 2 and/or 3 are not required, the control determines the maximum possible path velocity the axis limit values are still maintained with and provides them as the maximum possible test feed after executing step 1.

Relevant NC functions

The long form instead of the short form can be used in the following table. That is "JerkControl" instead of "JKC", "KINLIMLEVEL" instead of "KLL" and "KINLIMWEIGHT" instead of "KLLW",

JKC - or - JKC (1)	Enable jerk limitation using the presettings from the machine parameters 7030 00410 and 7030 00420.
JKC () - or - JKC (0)	Disable jerk limitation.
JKC (KLL=<KLL>)	Enable jerk limitation with the setting programmed for KLL and the presetting from the machine parameter 7030 00420.
JKC (KLLW=<KLLW>)	Enable jerk limitation with the presetting from the machine parameter 7030 00410 and the setting programmed for KLLW .

JKC (KLL=<KLL>, KLW=<KLW>)	
	Enable jerk limitation with the settings programmed for KLL and KLW.

Tab. 7-12: Relevant NC Function

Relevant machine parameters (MP)

1005 00002	Maximum axis velocity
1010 00001	Maximum axis acceleration
1010 00020	Maximum axis jerk
7030 00120	Maximum path velocity
7030 00210	Maximum path acceleration
7030 00220	Maximum path deceleration
7030 00410	Kinematic limitation level
7030 00420	Kinematic limitation scaling

Tab. 7-13: Relevant machine parameters

Example: Velocity optimization of an NC program

Axis limit values:

System axis number	1005 00002	1010 00001	1010 00020
1	20000.00	1.00	200.00
2	20000.00	1.00	200.00

Tab. 7-14: Example of the velocity optimization of an NC program; axis limit values

NC program:

```

N10 G0 X0 Y0
N20 G8 G1 F10000
N30 JKC(KLL=0,KLW=1)
N40 VREC_START(VrecRes,1)
N50 G1 X10
N60 G3 Y10 R20
N70 G1 Y0
N80 VREC_STOP
M30

```

If the "VrecRes" file is opened after this NC program has been executed, the following is displayed in excerpts:

F100:	Fmax:	AccUp:	AccDown:	NC source:
10000.00	20000	1.00000	1.00000	G1 X10
10000.00	5045.38	0.70711	0.70711	G3 X0 Y10 R-10
10000.00	20000	1.00000	1.00000	G1 Y0

Tab. 7-15: Example of the velocity optimization of an NC program; initial position

As it can be seen, the maximum path velocity drops to 5045.38 in the circular block N60 while a maximum path velocity of 20,000 is possible in straight line blocks N50 and N70, i.e., the programmed feed of 10,000 is to be maintained.

Circle and straight lines are to be traversed with the same path velocity. F5045.38 is thus programmed instead of F10000, so that block N20 is changed as follows:

N20 G8 G1 F5045.38

The result is as follows:

F100:	Fmax:	AccUp:	AccDown:	NC source:
5045.38	20000	1.00000	1.00000	G1 X10
5045.38	4119.54	0.85188	0.85188	G3 X0 Y10 R-10
5045.38	20000	1.00000	1.00000	G1 Y0

Tab. 7-16: Example of the velocity optimization of an NC program, F-value reduction

As it can be seen, the maximum path velocity now even drops to 4119.54 in the circular block N60. This is caused, since the maximum path velocity based on the third reduction step is now lower than for F10000 due to the lower programmed feed of F5045.38. Since the programmed path acceleration was not changed and the maximum path velocity and the maximum path acceleration are reduced together in the third reduction step, this results in a lower maximum path velocity of 4119.54, but in a higher maximum path acceleration of 0.85188.

To solve this problem, the control has to reduce only the maximum path acceleration in the third reduction step and not the maximum path velocity. Thus, set the parameter KLW to the highest possible value. NC block N30 is therefore changed as follows:

N30 JKC(KLL=0,KLW=1000)

The result is as follows:

F100:	Fmax:	AccUp:	AccDown:	NC source:
5045.38	20000	1.00000	1.00000	G1 X10
5045.38	5043.90	0.70752	0.70752	G3 X0 Y10 R-10
5045.38	20000	1.00000	1.00000	G1 Y0

Tab. 7-17: Example of velocity optimization of an NC program, F-value reduction and KLW = 1000

The goal was reached.

The result obtained with KLL = 3 is as follows (for comparison reasons):

N20 G8 G1 F10000

N30 JKC(KLL=3)

F100:	Fmax:	AccUp:	AccDown:	NC source:
10000.00	20000	1.00000	1.00000	G1 X10
10000.00	6000	1.00000	1.00000	G3 X0 Y10 R-10
10000.00	20000	1.00000	1.00000	G1 Y0

Tab. 7-18: Example of the velocity optimization of an NC program, KLL = 3

7.6.2 Applying

Refer to the previous section.

7.6.3 Activating

Program JerkControl or JKC with the parameters described.

Examples:

JKC(1), JKC(KLL=1,KLW=2), JKC(KLL=3), JKC(KLW=3),
 JKC(KINLIMLEVEL=2,KLW=0.5), JKC(KLL=0,KLW=4)

7.6.4 Deactivating

Program JerkControl(0), JerkControl(), JKC(0) or JKC().

7.7 Dynamic limit values

7.7.1 Description

Function

Is used by the control to limit velocity and acceleration of coupled drives.

Thus, situations caused by the NC and resulting in dynamic overload of individual drives can be prevented in advance. Thus, the NC automatically reduces the path velocity and acceleration so that the defined limits of all participating axes are not exceeded.

Potential causes for dynamic overload:

- Excessive path feed
With the given path feed, the necessary velocities of individual axes participating in the path can increase significantly.
- Tightly dimensioned drive
Peak loads can cause a dynamic overcharge of the drives.
- Incorrect drive parameterization
The limits in the drive are not parameterized correctly (via Sercos parameter).
- Unfavorable programming
e.g. many short machining blocks with up slope and down slope.

Potential effects of dynamic overloads:

- Contour distortions
- Overtemperature of the drive
- Increased machine wear
- Servo error.

Restrictions

All command specifications, independently generated by the drive, cannot be influenced by the limits defined on the control side (e.g. for active "interpolation in the drive", for "drive-controlled referencing" or when setting the drive to "standstill" after an error)! In such cases, suitable parameterizing on the drive side is required.

Relevant NC functions

AxAcc	Acceleration programming ON Overrides the acceleration values defined in 1010 00001 and/or 1010 00003 with programmed or previously stored values.
AxAcc(0)	Acceleration programming OFF The values stored in the machine parameters apply again.
AxAccSave	Saves the currently active acceleration values of all axes.

Tab. 7-19: Relevant NC functions

Relevant Sercos parameters

S-X-0091	Bipolar velocity limit value
S-0-0138	Bipolar acceleration

Tab. 7-20: Relevant Sercos parameters

Relevant machine parameters (MP)

1005 00002	Maximum axis velocity
1010 00001	Maximum acceleration in feed mode
1010 00003	Maximum acceleration in rapid traverse mode
1010 00012	Axis acceleration reserve (relevant at contour knees)
1040 00012	Maximum spindle speed of each gear (if necessary)
1040 00051	Acceleration in position controller range[1] (if necessary)

Tab. 7-21: Relevant machine parameters

7.7.2 Handling instruction: Changing the axis acceleration

Applying:

NOTICE

Workpiece or machine can be damaged!

Do not reset the system during machining.

IW Engineering/configuration: Parameterizing maximum values

1. Ensure that the relevant Sercos parameters are correctly parameterized.

The Sercos parameter values are normally set according to the maximum dynamic performance of the respective drive. Ensure that the coupled axis mechanic as well as the maximum occurring load are considered!

If changes were made, back up all Sercos parameters to the EEPROM of the drive or enter their values into the relevant Sercos files of the NC. The Sercos files are transferred from the control to the drive during the

Sercos initialization (for information on Sercos, refer to the manual "MTX Functional Description 14VRS Special Functions", chapter "Functions for Drive Parameterization").

2. For axes:

Configure **MaxVel** (MP 1005 00002), **FeedAxAcc** (MP 1010 00001) and if required **RapidAxAcc** (MP1010 00003). The limit values set on the drive side must not be exceeded.

For all relevant spindles/c-axes for tapping without compensating chuck:

Configure **MaxSpSpeed** (MP 1040 00012) and **MaxSpAccPosCtrl[1]** (MP 1040 00051).

The limit values set on the drive side must not be exceeded. The gear ratio has to be considered.

3. Configure **AxAccReserve** (MP1010 00012).

The acceleration values set under point 2 may not be exceeded.

4. Restart control and drives.



All command value specifications independently generated by the drive cannot be influenced by the limit values defined on the control side.

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

Activating IW Operation/NC programming: Programming the dynamic limit value

Using the AxAcc or AAC command, the acceleration of channel axes can be changed during program operation.

Program **AxAcc(<AxisName><AccelerationValue>{,Validity})**. Using **AxAcc**, only values smaller than the value set in the parameter **FeedAxAcc** Maximum acceleration in feed mode (1010 00001) or in the parameter **RapidAxAcc** "Maximum acceleration in rapid traverse mode" (1010 00003) can be programmed.

Short form **AAC(...)**

	Documentation	
Documentation:	MTX Programming Manual	Axis acceleration

Deactivating IW Operation/NC programming: Deactivating AxAcc(0) or AAC(0)

	Documentation	
Documentation:	MTX Programming Manual	Axis acceleration

7.8 Feed Group

7.8.1 Description

Function

Unless particularly specified, the path feed programmed with address F refers to all axes or, if axis transformation is active, to the spatial coordinates. This preset reference can be changed using the "feed group" (FeedGrp, FDG) function.

FeedGrp defines the coordinates referring to the feed F. All coordinates existing in the feed group are carried along.

If no other coordinates than coordinates not belonging to the feed group are moved, the programmed feed refers to these coordinates.

If members and non-members of the feed group are programmed together, the "feed group type" (FeedGrpType, FDGT) function affects the feed course at a block limit.

The "Hide coordinates" (FeedAd, FAD) function allows removing coordinates from the feed group.

Variants

1. No programmed feed group
 - 1.1 No axis transformation active
All axes of the channel form the feed group.
 - 1.2 One axis transformation is active
The spatial coordinates (x, y, z) form the feed group.
2. Default feed group
FeedGrp(1) activates the default feed group. The coordinates with the meaning x-, y- and z-direction form the feed group.
If extended plane switching (G17(...), G18(...), G19(...)) is used, this feed group is automatically adjusted accordingly.
3. Feed group by name
FeedGrp (<Coordinate1>{,<Coordinate2>, ..) assigns all programmed coordinates to the feed group.
If an axis transformation capable of vector orientation is active, the orientation vector O can also be assigned to the feed group.

Relevant NC functions

FeedGrp	Define feed group
FeedGrpType	Define feed group type.
FeedAd	Hide coordinates
G17(...), G18(...), G19(...)	Extended plane switching

Tab. 7-22: Relevant NC functions

Relevant CPL functions

SD(594,...)	SD(594,<Coordinate>) or SD(594,<Coordinate>,0) 2: The coordinate is an effective member of the active feed group (FeedGrp + FeedAd) 1: Coordinate follows 0: Coordinate is a slave
	SD(594,<Coordinate>,1) 1: Coordinate is a programmed member of the feed group (FeedGrp)
	SD(594,<Coordinate>,2) > 0: Coordinate is excluded from the feed group (FeedAd)

Tab. 7-23: Relevant CPL functions

Relevant machine parameters (MP)

701000030	Axis meaning The entries with the values 1 (main axis X), 2 (main axis Y) and 3 (main axis Z) are used for the default feed group.

Tab. 7-24: Relevant machine parameters

Example

A full circle arranged in XY and consisting of four quadrants is to be machined in the plane with 2,000 mm/min despite a feed motion in plane Z. The velocity course is the same for both program examples.

Without FeedGrp

Program:

```
N100 G17 G94 F2000
N110 G0 X0 Y0 Z0
N120 FeedGrp()
N130 G2 X5 Y5 I5 F2000
N140 G2 X10 Y0 J-5 Z5 F2370.9
N150 G2 X5 Y-5 I-5 F2000
N160 G2 X0 Y0 J5 Z10 F2370.9
```

With FeedGrp

Program:

```
N100 G17 G94 F2000
N110 G0 X0 Y0 Z0
N120 FeedGrp(X,Y)
N130 G2 X5 Y5 I5
N140 G2 X10 Y0 J-5 Z5
N150 G2 X5 Y-5 I-5
N160 G2 X0 Y0 J5 Z10
```

7.8.2 Activating

- FeedGrp(1) or FeedGrp(<Coordinatelist>) activates the feed group.
FeedGrpType(...) can be used to affect the feed course at block transition.
The presetting at control startup is FeedGrpType(SMOOTH). Normally, this presetting does not have to be changed.
- FeedAd(1) or FeedAd(<Coordinatelist>) excludes coordinates from feed calculation, even if the feed group is inactive.

7.8.3 Disabling

- FeedGrp() disables the feed group.

- FeedAd() stops hiding coordinates from feed calculation.

8 Safety/monitoring

8.1 Electronic limit switches (software limit switches)

8.1.1 Description

Function

Prevents that an axis leaves a defined physical traversing range due to position specifications.

Features:

- Two physical traversing ranges (limit switch ranges) can be defined for each axis. The ranges may overlap.
- The desired traversing range can be selected at the respective axis interface via two interface signals.
- The limit switch check can be switched off (deactivated) dynamically via the interface signal.

Restrictions

- As long as the control does not know the absolute axis position (see interface signal iAx_RefKnow "Reference point known"), the electronic limit switches are not applied to this axis (reference point has to be known).
- The range limits must not exceed the position limit values (S-0-0049, S-0-0050) set in the drive.
- Any violation of the selected active limit switch range results in a runtime error.

Relevant IF signals

qAx_TrLim_00	Activate "limit switch range bit 0" (axis interface)		
qAx_TrLim_01	Activate "limit switch range bit 1" (axis interface)		
	Select the desired limit switch range:		
	Area	Required signal state	
		qAx_TrLim_01	qAx_TrLim_01
	1	0	0
	2	0	1
qAx_SwLimOff	Hide "software limit switch" (axis interface) High signal: Hide selected range		
iAx_RefKnow	"Reference point known" (axis interface). Low signal: Limit switches not effective for the axis!		

Tab. 8-1: Relevant IF signals

Relevant machine parameters

1020 00001	Software limit switch, positive
1020 00002	Software limit switch, negative

1020 00003	2. Software limit switch, positive
1020 00004	2. Software limit switch, negative

Tab. 8-2: Relevant machine parameters

8.1.2 Handling instruction: Electronic limit switches - Software limit switches

Applying: Engineering/configuration: Configuring software limit switches

For the software limit switch [1] or [2]

- **EnablTrvLim** "Activate software limit switches" (1020 00001)
- **PoTrvLim** "Positive travel range limit" (1020 00001 or 1020 00003)
- **NeTrvLim** "Negative travel range limit" (1020 00002 or 1020 00004)



Software limit switches are not active at the respective axis if no reference point is known.

The travel range limits must not exceed the position limit values (**S-0-0049**, **S-0-0050**) set in the drive.

	Documentation	Instruction
Instruction:		Editing machine parameters
Documentation:	MTX Machine Parameters	Software limit switches

Activating: IW Engineering/IndraLogic: Selecting software limit switch range

To activate the desired software limit switch range, the following interface signals are to be set as follows:

- **qAx_TrvLim_00** "Limit switch bit 0 "
- **qAx_TrvLim_01** "Limit switch bit 1"

Area	Required signal state	
	qAx_TrvLim_01	qAx_TrvLim_00
1	0	0
2	0	1

Tab. 8-3: Interface signals



Without being controlled by the PLC, qAx_TrvLim_00, qAx_TrvLim_01 and qAx_SwLimOff are always selected and activated in range 1 if the limit switch is not hidden in the machine parameters.

	Documentation	Instruction
Instruction:		Editing PLC Signals
Documentation:	MTX PLC Interface	Software limit switch range

Deactivating: IW Engineering/IndraLogic: Deactivating software limit switch

Set the interface signal **qAx_SwLimOff** "Hide limit switch" and the software limit switch can be **temporarily** deactivated.

	Documentation	Instruction
Instruction:		Editing PLC Signals
Documentation:	MTX PLC Interface	Software limit switch range

The software limit switch allows a **permanent** deactivation by deleting **Enabl-TravLim** "Enable software limit switch" (1020 00001).

	Documentation	Instruction
Instruction:		Editing machine parameters
Documentation:	MTX Machine Parameters	Software limit switch range

9 Accuracy

9.1 Exact stop

9.1.1 Description

Function

With active exact stop, the target position programmed in the NC block is approached in a defined range. At active exact stop, the control decelerates to 0 at each block end and checks subsequently whether all axes of the channel report "InPos". The control continues machining by interpolating the following block only after all axes had reported that they reached their positioning window.

Exact stop is recommended whenever positions have to be approached with high accuracy such as tool change positions or machining sharp contour corners, etc. A coarser exact stop, however, reduces the attained accuracy but is more beneficial regarding the machining time. If an exact stop is required, a coarser exact stop window should be used to facilitate the cycle time reduction.

The positioning windows are defined axis-specifically via Sercos ident numbers. It is possible to define a fine positioning window (S-0-0057) and a rough positioning window (S-0-0261) for each axis. The program of the control provides the option to move from one positioning window to the other and vice versa or to completely ignore the following error (positioning window: v=0).

Internally, the control processes the drive status messages "InPos Exact Stop Window 1" (S-0-0336) and "InPos Exact Stop Window 2" (S-0-0341) via the status real-time bit 2 (S-0-0308) to which the particular relevant status signal is assigned.

In the control, the real-time bit 2 is linked to the travel command of the axis (signal iAx_TrVCmd) and transferred to the PLC as iAx_InPos signal at the axis interface.

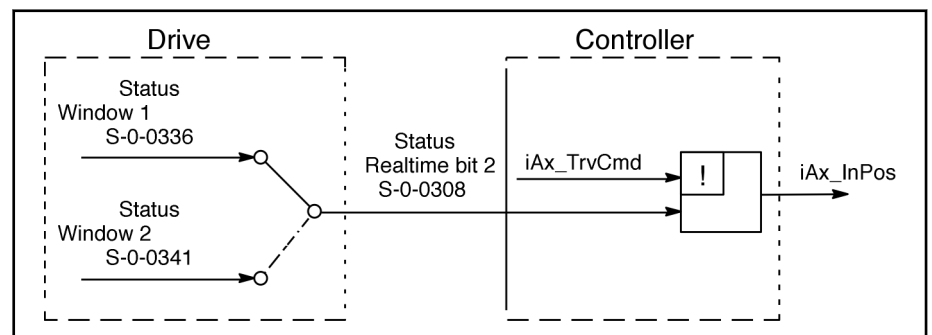


Fig. 9-1: Real-time bit 2

With regard to programming, the control distinguishes between the exact stop function in feed mode and the exact stop function in rapid traverse. Both functions can be activated and deactivated separately.

According to the standard, the exact stop function is activated and deactivated with G61 and G62 in feed mode (G1, G2, G3), whereas the rapid traverse mode (G0) always requires traversing with exact stop.

More programming options are available in the MTX. For example, the exact stop can be disabled during rapid traverse.

Moreover, the positioning window to be checked can be defined separately for the feed and rapid traverse modes (1: fine, 2: rough, 3: v=0). The control

automatically configures the drive to the corresponding window whenever the feed mode is changed to rapid traverse mode and vice versa.

The following table provides an overview on all syntax elements referring to the exact stop function.

	Feed mode	Rapid traverse mode
Selection Exact stop function	yes: G61 modal G1(IPS) local-modal G1(IPS1) local-modal G1(IPS2) local-modal G1(IPS3) local-modal No: G62 modal	yes: G0 modal G0(IPS1) modal G0(IPS2) modal G0(IPS3) modal No: G0(NIPS) local-modal
Selection/Preselection Exact stop window	IPS1 modal IPS2 modal IPS3 modal G61(IPS1) modal G61(IPS2) modal G61(IPS3) modal	IPS1 modal IPS2 modal IPS3 modal G0(IPS1) modal G0(IPS2) modal G0(IPS3) modal

Tab. 9-1: Syntax elements

The effect of the non-standardized functions G1 with exact stop (G1(IPS) or G1(IPS_x) with x = 1,2,3) and G0 without exact stop (G0(NIPS)) is always local-modal with respect to the exact-stop on/off-functionality.

Local-modal means that the exact-stop on/off-function is only applied until the next programming in the modal group G0,G1,G2,....

Modal means that the state is valid until it is explicitly deselected.

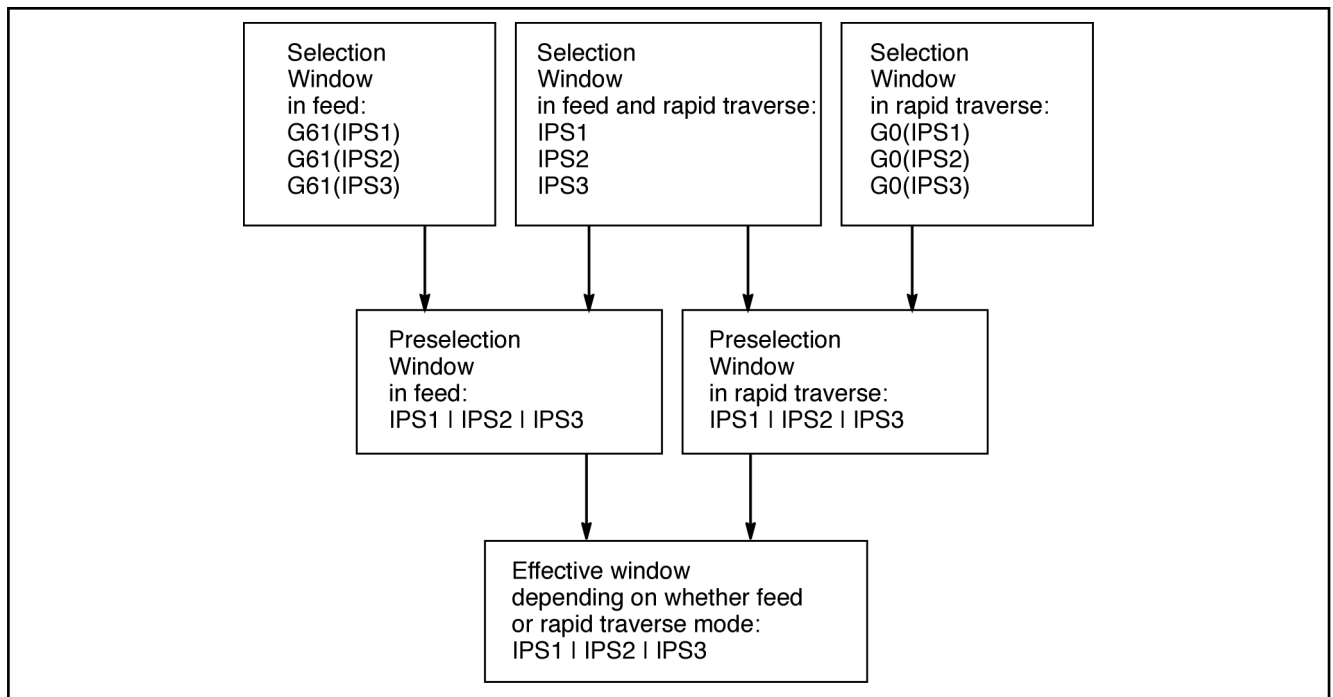


Fig. 9-2: Exact stop

Prerequisites:

- The Sercos parameters S-0-0057 and S-0-0261 for the fine or rough exact stop windows were configured properly.
- The corresponding drives were compensated with respect to closed-loop control so that the fine exact stop window can be reached in a suitable response time.

Restrictions

The exact stop can only be activated control-internally for channel axes (synchronous axes). The exact stop mode is not possible during asynchronous axis mode.

Relevant NC functions

G61	Enable exact stop function during feed (modal until deselected by G62).
G61(IPS1)	As for G61, however, additionally preselect and activate the exact stop window 1 modally for the feed mode.
G61(IPS2)	As for G61, however, additionally preselect and activate the exact stop window 2 modally for the feed mode.
G61(IPS3)	As for G61, however, additionally preselect and activate the exact stop window 3 modally for the feed mode.
G62	Disable (modally) the exact stop function in the feed mode.
IPS1	Modally preselect the exact stop window 1 for the feed mode and the rapid traverse mode.
IPS2	Modally preselect the exact stop window 2 for the feed mode and the rapid traverse mode.
IPS3s	Modally preselect the exact stop window 3 for the feed mode and the rapid traverse mode.

G0	Activate standard G0 (with exact stop function). The exact stop window previously selected for the rapid traverse mode is applied automatically.
G0(NIPS)	Activate G0 without exact stop function. NIPS is acting locally-modally (active until the next programming in the G0, G1, G2 group, etc.).
G0(IPS1)	Activate standard G0 (with exact stop function). Modally preselect and activate the exact stop window 1 for the rapid traverse mode.
G0(IPS2)	Activate standard G0 (with exact stop function). Modally preselect and activate the exact stop window 2 for the rapid traverse mode.
G0(IPS3)	Activate standard G0 (with exact stop function). Modally preselect and activate the exact stop window 3 for the rapid traverse mode.
G1	Activate standard G1. Exact stop function in the feed is modally controlled via the function G61/G62.
G1(IPS)	Activate G1 with exact stop function. IPS is acting locally-modally (active until the next programming in the G0, G1, G2 group, etc.). The exact stop window previously selected for the feed mode is applied automatically.
G1(IPS1)	Activate G1 with exact stop function. The exact stop is acting locally-modally (active until the next programming in the G0, G1, G2 group, etc.). Modally preselect and activate the exact stop window 1 additionally for the feed mode.
G1(IPS2)	Activate G1 with exact stop function. The exact stop is acting locally-modally (active until the next programming in the G0, G1, G2 group, etc.). Modally preselect and activate the exact stop window 2 additionally for the feed mode.
G1(IPS3)	Activate G1 with exact stop function. The exact stop is acting locally-modally (active until the next programming in the G0, G1, G2 group, etc.). Modally preselect and activate the exact stop window 3 additionally for the feed mode.

Tab. 9-2: Relevant NC functions

Relevant CPL functions

NCF(...)	Query of active modal NC functions: NCF(G0) → G0, G1, G2, G3, G5, G6 NCF(NIPS) → NIPS, IPS, SPLCO, SPLC1, SPLC2, NURBS NCF(G61) → G61, G62 NCF(IPS1) → IPS1, IPS2, IPS3 Also refer to the description of MP 6005 00040.
MCODS(6,...)	Provides the InPos status information for the system axes
MCODS(47,...)	Provides the status information on whether there is a travel command for the system axes.

Tab. 9-3: Relevant CPL functions

Relevant Sercos parameters

S-0-0057	Value for exact stop window 1 (fine)
S-0-0261	Value for exact stop window 2 (rough)
S-0-0336	Status "InPos exact stop window 1"
S-0-0341	Status "InPos exact stop window 2"
S-0-0307	Status "real-time bit 2". The MTX uses the status real-time bit 2 to transfer the InPos status from the drive to the control. Depending on the activated exact stop window, the status real-time bit 2 is linked to ident number S-0-0336 or S-0-0341.

Tab. 9-4: Relevant Sercos parameters

Relevant IF signals

iCh_InPosAct	Indicates that the positioning window 2 (rough) is activated for the channel axes of this channel.
iAx_InPos	Indicates that there is no travel job for the axis in question and that it is positioned in the parameterized exact stop window. The signal is formed for all system axes including asynchronous axes.

Tab. 9-5: Relevant IF signals

Relevant machine parameters

6005 00040	Display filter for modal NC functions This parameter defines channel-specifically which modal NC functions are to be displayed on the interface. The following modal groups are relevant for the exact stop functionality: G0, G1, G2, G3, G5, G6: Interpolation type, e.g. rapid traverse and linear NIPS, IPS, SPLCO, SPLC1, SPLC2, NURBS: This group provides additional information on the interpolation type. The following can be displayed: For G0: "non-standardized design": NIPS For G1: "non-standardized design": IPS For G6: Spline type: SPLC, SPLC1, SPLC2, NURBS G61, G62: exact stop in feed on/off IPS1, IPS2, IPS3: effective exact stop window (depending on rapid traverse mode and feed mode)
7060 00020	Switched-on state after reset Specifies the functions to be active after M30, channel or system reset.

Tab. 9-6: Relevant machine parameters

Example:

		Exact stop in		Preselection InPos window		Effective InPos window	Additional info
		Rapid traverse	Feed	In rapid traverse	In feed		
G61 IPS2	Exact stop in feed IPS2 preselection (feed + rapid traverse)	-	■	IPS2	IPS2	IPS2	
G1		-	■	IPS2	IPS2	IPS2	
G0		■	(■)	IPS2	IPS2	IPS2	
G0(NIPS)	Rapid traverse without exact stop	-	(■)	IPS2	IPS2	(IPS2)	NIPS
G0		■	(■)	IPS2	IPS2	IPS2	
G0(IPS3)	IPS3 preselection in rapid traverse mode	■	(■)	IPS3s	IPS2	IPS3s	
G0		■	(■)	IPS3s	IPS2	IPS3s	
G1(IPS1)	G1 with exact stop IPS1 preselection in feed	-	■	IPS3s	IPS1	IPS1	IPS
G1		-	■	IPS3s	IPS1	IPS1	

		Exact stop in		Preselection InPos window		Effective InPos window	Additional info
		Rapid traverse	Feed	In rapid traverse	In feed		
G62	Exact stop off in feed	-	-	IPS3s	IPS1	(IPS1)	
G0		■	-	IPS3s	IPS1	IPS3s	
G0(NIPS)	Rapid traverse without exact stop	-	-	IPS3s	IPS1	(IPS3)	NIPS
G0		■	-	IPS3s	IPS1	IPS3s	
G1(IPS)	G1 with exact stop	-	■	IPS3s	IPS1	IPS1	IPS
G1		-	-	IPS3s	IPS1	(IPS1)	
IPS2	IPS2 preselection (feed + rapid traverse)	-	-	IPS2	IPS2	(IPS2)	
G1(IPS)	G1 with exact stop	-	■	IPS2	IPS2	IPS2	IPS
G1		-	-	IPS2	IPS2	(IPS2)	
G61(IPS1)	Exact stop in feed IPS1 preselection in feed	-	■	IPS2	IPS2	IPS1	
G0		■	(■)	IPS2	IPS2	IPS2	

Tab. 9-7: Example

9.1.2 Handling instruction: Exact stop in feed mode

- Applying:**
1. Parameterize positioning windows 1 and 2 in the drive (S-0-0057 and S-0-0261).
 2. Configure desired modal NC functions to be displayed in MP 6005 00040.
 3. Configure switched-on state after reset (MP 7060 00020).

In feed mode, the function "Exact Stop On" (G61) minimizes the following error at the end of block. G61 acts on all path motions (G1, G2, G3, G5, G6). Using the optional parameter IPS, the desired positioning window for the feed mode (G61(IPSx)) can be switched modally.

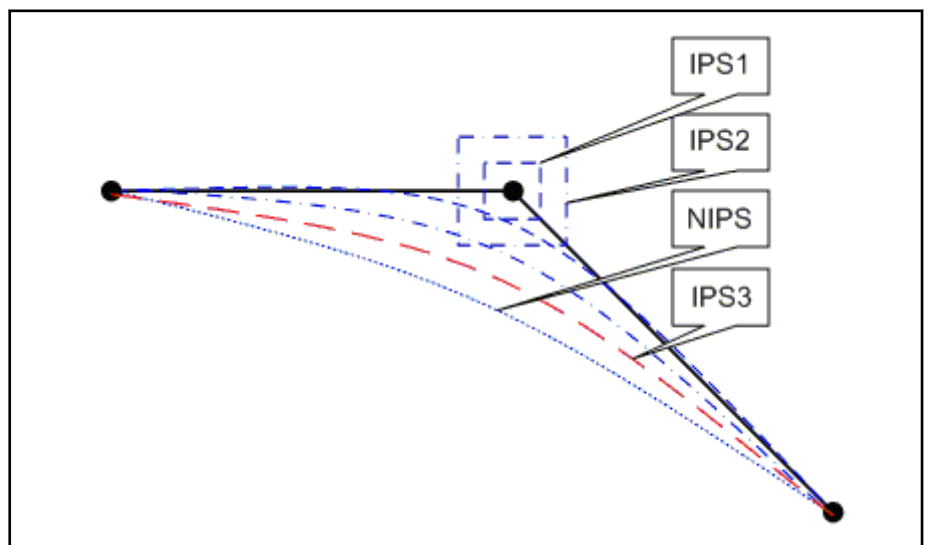


Fig. 9-3: Example: Point-to-point

IW Operation/NC programming: Activating exact stop

- Programming **G61**
 - or -
- program with positioning window **G61(IPS1|IPS2|IPS3)**
 - The positioning windows "**IPSx**" are adjusted in the Sercos files for phase 3.
 - **IPS1** Exact stop ON.
Wait for fine positioning window. At the end of the block, the control first reduces the path velocity to $v=0$. Only when this positioning window was reached for all axes involved, the next block is traversed.
 - **IPS2** Exact stop ON.
Wait for rough positioning window. "Inpos range 2 active" is displayed at the channel interface. At the end of the block, the control first reduces the path velocity to $v=0$. Only when this positioning window was reached for all axes involved, the next block is traversed.
 - **IPS3** Exact stop ON.
Decelerating to $v=0$ at the end of the block. At the end of the block, the control reduces the path velocity to $v=0$. The next block is then traversed to a positioning window without being checked.



If only "exact stop" G61 or G1(IPS) or G0 is programmed, the latest activated positioning window is still applied for the path mode or the rapid traverse mode. The default setting is "fine positioning window".

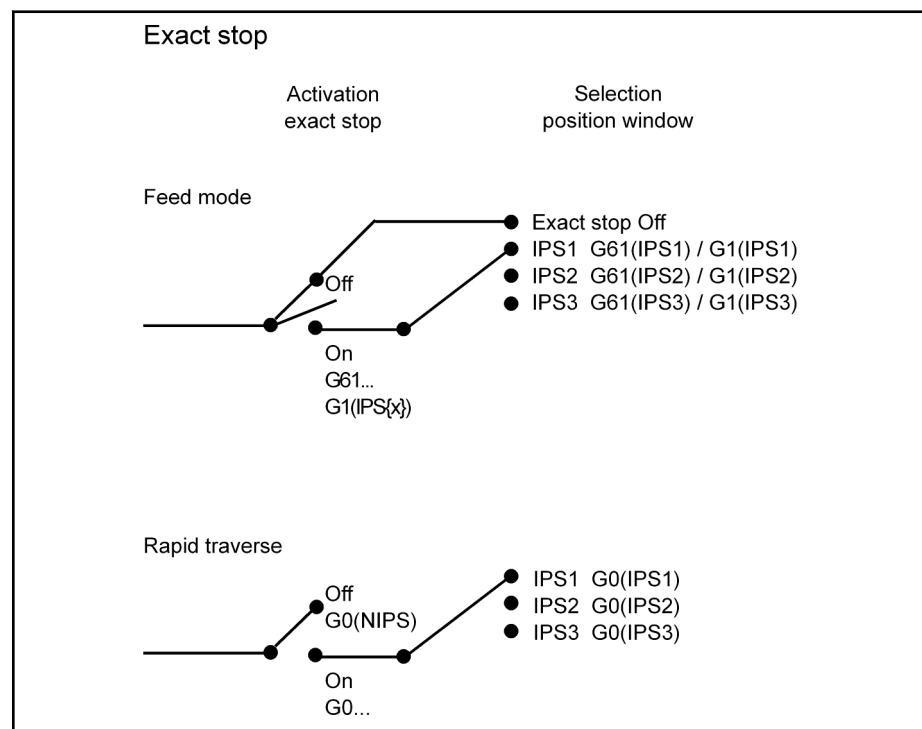


Fig. 9-4: Different exact stop variants

	Documentation	
Documentation:	MTX Programming Manual	Activating Exact Stop

IW Operation/NC programming: Deactivating exact stop

Programming G62

	Documentation	
Documentation:	MTX Programming Manual	Deactivating Exact Stop

9.2 Radius adjustment for circles

9.2.1 Description

Function

For circular path segments (G2/G3), the programmed circle parameters can cause an inconsistent geometry due to faulty entries or insufficient accuracy (number of decimal digits).

The control is able to automatically compensate these specifications without runtime error. The input values center point or radius are always corrected in such a way that the deviation between the programmed and the corrected value is mathematically reduced to a minimum.

This refers to the programming of a circle by means of radius and end point (radius programming) as well as programming by center and end point (center point programming).

For radius programming:

The starting point of the circular motion is the current position. The position of the end point and the radius are programmed for the radius programming with G2/G3.

Due to mathematical reasons, the circle diameter has to be at least of half the size as the distance between the starting and the end point.

If this is not the case due to inaccurate entries for example (can result easily when programming a semicircle), the MTX increases the radius up to half the distance between the starting point and the end point.

To ensure that there are no programming errors, the change in radius has to be smaller than the value set in the machine parameter MP 705000020. Otherwise, the MTX reports the error "Invalid radius programming".

For point programming:

The starting point of the circular motion is the current position. The positions of center point and end point are programmed for the center point programming with G2/G3.

Due to mathematical reasons, the distance of the center point from the starting point has to be equal to the distance of the center point to the end point.

If this is not the case due to inaccurate entries for example, the MTX displaces the center point and uses the shortest distance possible to achieve equality.

The following conditions have to be met to ensure that there are no programming errors.

- The difference of both the distances has to be smaller than the value set in the machine parameter MP 705000020.
- The displacement of the center point has to be smaller than double the value set in the machine parameter MP 705000020.

Both conditions have to be met. Otherwise, the MTX reports the error “Invalid circle center point”.

Especially if the programmed center point is very far from the starting and the end point, it is useful to allow a bigger offset of the center point.

Create the channel-specific permanent SD variable system date SD.SysCtTolCtPrg for this case. Enter for example the following text between both the lines labeled with “..” into the file SDDefMTB.xml and trigger a system start-up.

Program:

```
<? xml version="1.0" encoding="UTF-8" ?>
<SDDef>
  ..
  <Variable Dimension="Channel" Storage="permanent">
    <Name>SysCtTolCtPrg</Name>
    <Type>Float_t</Type>
  </Variable>
  ..
</SDDef>
```

In addition to the error message “Invalid circle center point”, the distance to offset the center point to obtain a valid circle is output. If this error message is issued, check the programmed coordinates for the starting point, the end point and the center point of the arc first. If you want to tolerate the center point offset, set SD.SysCtTolCtPrg to a value that is higher than the distance to exclude rounding errors and trigger the channel reset.

Instead of “permanent“, SD.SysCtTolCtPrg can also be defined as “volatile“. In the first case, the value entered into SD.SysCtTolCtPrg remains when the control is shut down. In the second case, the value is lost. To avoid this in the second case, enter the current value of SD.SysCtTolCtPrg into an SD initialization file before, e.g. SDDatMTB.xml.

To apply the value 130000.0 for channel 1 for example and the value 70000.0 for channel 2, enter the following:

Program:

```
<? xml version="1.0" encoding="UTF-8" ?>
<SDDef>
  ..
  <SysCtTolCtPrg Dimension='1'>130000.0</SysCtTolCtPrg>
  <SysCtTolCtPrg Dimension='2'>70000.0</SysCtTolCtPrg>
  ..
</SDDef>
```

Values smaller than or equal to 0 are invalid. In these cases, the double of the value set in the machine parameter MP 705000020 applies.

Relevant NC functions

G2/G3	Circle interpolation/helical interpolation on.

Tab. 9-8: Relevant NC functions

Relevant machine parameters

7050 00020	Radius tolerance for center point programming.
7050 00030	Radius tolerance for radius programming.

Tab. 9-9: Relevant machine parameters

9.2.2 Handling instruction: Radius adjustment of circles for center point programming

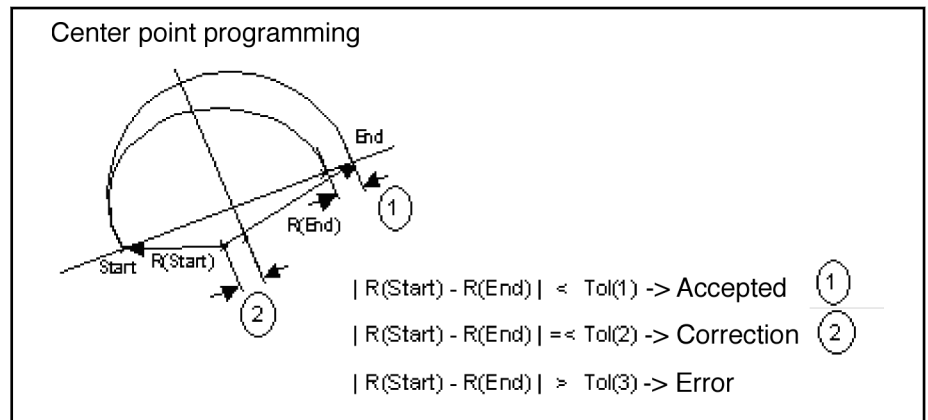


Fig. 9-5: Overview for the center point programming

IW Engineering/configuration: Adjusting tolerances

Enter a value for each channel into the machine parameter **RadTolCentProg** "Tolerance from which a runtime error is generated" (MP 7050 00020) and trigger a system reset to apply the entries. When programming arcs with a center point far from the starting and end point, create the system date SD.SysCtTolCtPrg additionally and enter a sufficiently high maximum center point correction into the system date. Trigger system reset to apply the system date.

	Documentation	Instruction
Instruction:		Editing machine parameters
Documentation:	MTX Machine Parameters	Center point compensation

9.2.3 Handling instruction: Radius adjustment of circles for radius programming

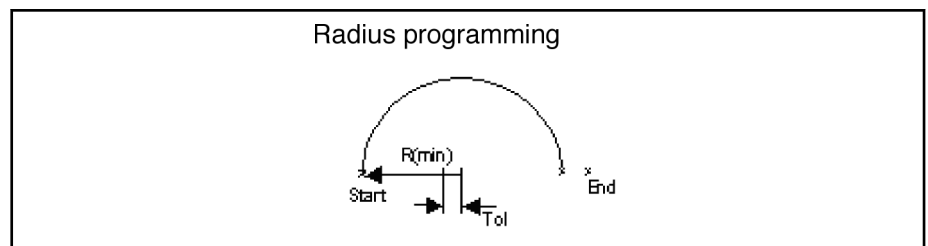


Fig. 9-6: Overview on radius programming

IW Engineering/configuration: Adjusting tolerance

Enter a value for each channel into the machine parameter **RadTolRadProg** "Tolerance from which a runtime error is generated" (MP 7050 00030) and trigger a system reset to apply the entries.

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

10 Axes, coordinates, coordinate systems

10.1 Overview

In the MTX, axes, coordinates and coordinate systems determine the basic system to use certain NC functions specified for the machine. The names given to axes, the reference to coordinate systems and the use of axis and coordinates transformations is of special significance.

In the following, system axes, channel axes, coordinates and coordinate systems are introduced.

10.2 System axes and system coordinates

The **drives** integrated in a machine belong to a "system axis pool", used to define **system axes** and **system coordinates** to solve various tasks (NC functions...).

Drives The drives (Sercos) of a machine tool can be classified into two **drive function types**:

- **Axes:**
All drives controlled by position command values are described as axes.
- **Spindle:**
However, spindle drives operate with speed interface. Spindle drives which can be activated in spindle mode (speed interface) and in C-axis mode (positioning interface) are called spindle/C-axis.

The corresponding drive function type is defined in MP 1001 00001.

System axes Each system axis (max. 99) corresponds to a real Sercos drive (unless hidden via machine parameters). Via MP 1003 00001 "System axis name", a name, unambiguous in the system, is assigned to each system axis.

System axes with an interpolatory interrelation to an axis group within one a channel are also called **synchronous axes**.

System axes not assigned to any channel are called **asynchronous axes** or **auxiliary axes**. They can be controlled from any channel, but do not have any interpolatory interrelation with other axes. Auxiliary axes can for example be used to drive tool or pallet changers.

System axis index System axes contain a system axis index / drive index (1...99) that correspond to the respective index of the drive parameters and axis parameters within the machine parameters. That means that it corresponds to the MP 1001 00001 and it is system-comprehensively unique.

Channel and axis groups: Use MP 1003 00002 **Channel assignment** to assign each system axis to exactly one machining channel. Within a channel, all axes interrelated in terms of machining are combined. The total of all system axes of a channel is called axis group (within the channel, these system axes are seen as "channel axes"). As different axis groups are assigned to different channels, several axis groups can perform various machining processes simultaneously and independently of each other. Each channel can only control the axis group assigned to it.

System Coordinates: The system coordinates are for an unambiguous assignment of coordinates when axis transformations are used at the points of action AT1 and/or AT2. System coordinates are input variables of an active (backward) axis transformation.

Thus, a total of twelve names is reserved for system coordinates:

- Six workpiece coordinates (x, y, z, phi, theta, psi)
- Six machine coordinates (X, Y, Z, PHI, THETA, PSI)

For each of these twelve coordinates, there is one unambiguous **system coordinates index** defined as follows:

System coordinate	System coordinate index
Linear workpiece coordinate x	-12
Linear workpiece coordinate y	-11
Linear workpiece coordinate z	-10
Rotary workpiece coordinate phi	-9
Rotary workpiece coordinate theta	-8
Rotary workpiece coordinate psi	-7
Linear machine coordinate X	-6
Linear machine coordinate Y	-5
Linear machine coordinate Z	-4
Rotary machine coordinate PHI	-3
Rotary machine coordinate THETA	-2
Rotary machine coordinate PSI	-1

Tab. 10-1: System coordinate index

The system coordinate index is set with negative signs to differentiate it from the system axis indices (1 to 99).

The total of all indices is called **SACI** (System Axis and Coordinate Index) and consists of the system coordinate indices and the system axis indices:

SACI = {-12,-11,...,-1;1,2,...,99}

The axis transformation used (e.g. Bipod with X and Y: -6 corresponds to the system coordinate X and -5 corresponds to the system coordinate Y) determines the system coordinates to be used. In the channel, specific axis or coordinate names can be assigned to the system coordinates via machine parameters.

Various coordinate names can thus be assigned to the same coordinate in different channels.

Example:

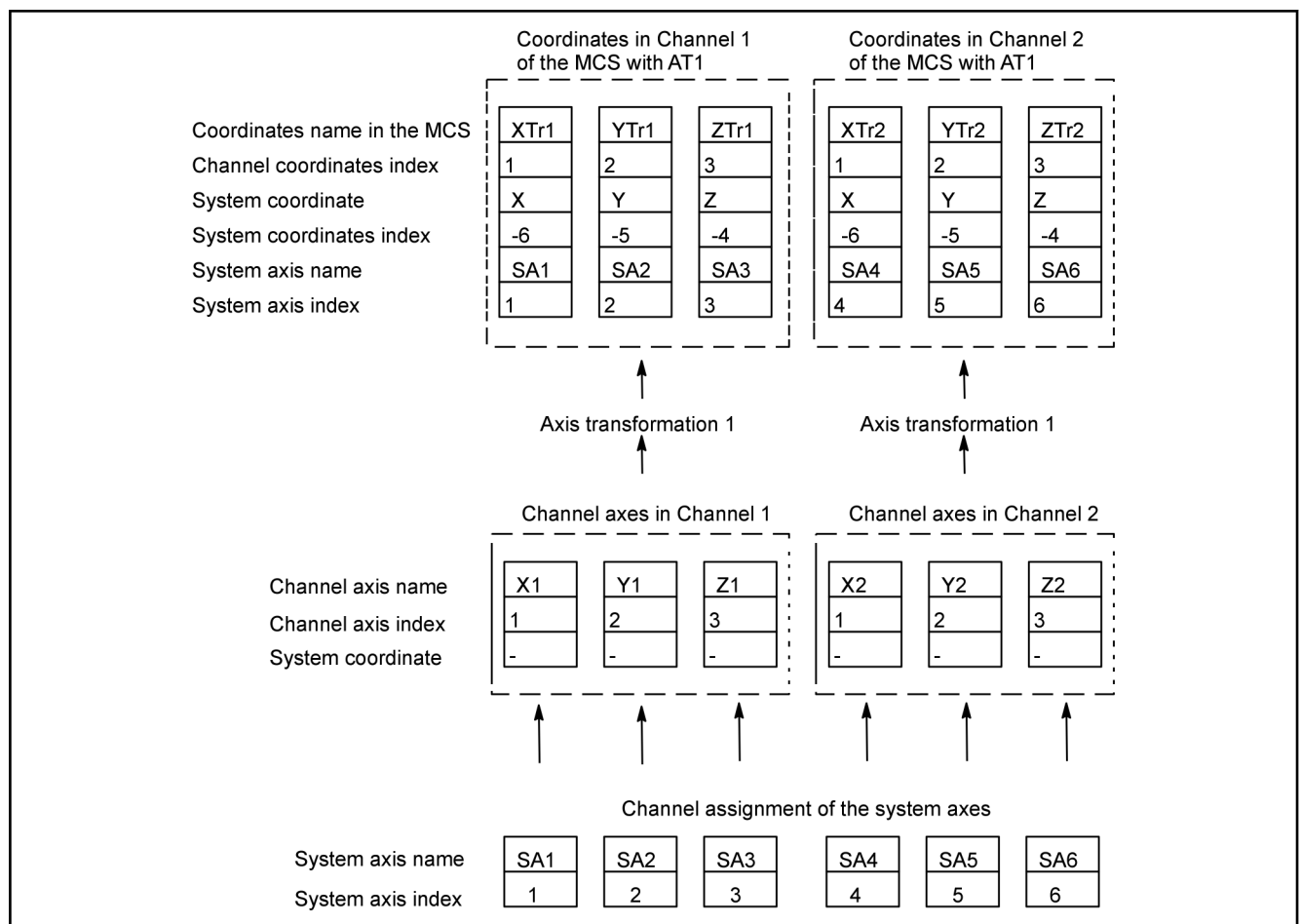


Fig. 10-1: Example - Assigning coordinate names for system coordinates

Special features of the system coordinates:

- As for the channel axis names, the names assigned to the system coordinates can be set channel-specifically via machine parameters with the coordinate names (also see [chapter 10.5 "Axes and coordinates in the channel"](#) on page 160).
- Compared to the system axes, system coordinates do not have any other characters beyond the attributes: name, type (linear, rotary, linear Cartesian).
- A system coordinate can be present in several channels simultaneously, a system axis only in one channel (see figure).

10.3 Coordinate systems in the channel

10.3.1 General information

Coordinate systems are used to determine direction and position of individual "machining axes" at the machine. The directions of the machining axes do not necessarily coincide with those of the machine axes but can be oriented in completely different direction due to the transformation.

Usually, the axes are assigned in a Cartesian manner, i.e. at right angles to each other. The positive directions of the coordinate system axes is determined by a right-handed, right-angled coordinate system.

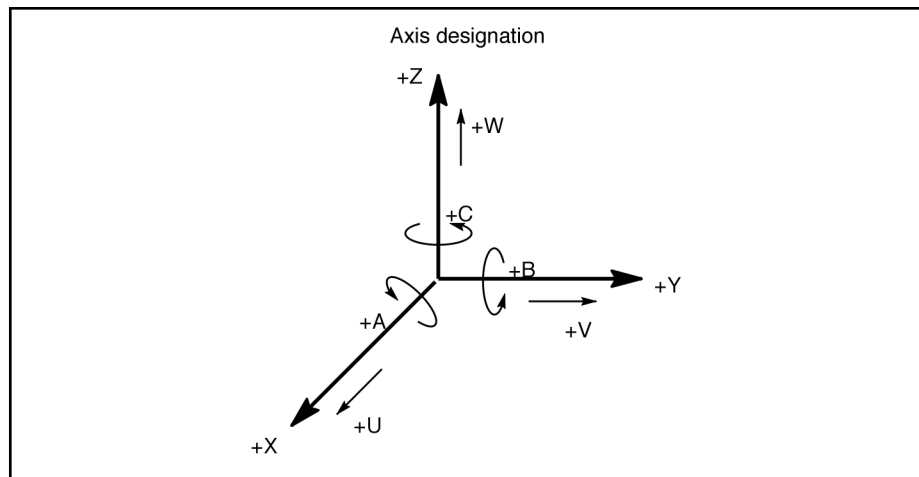


Fig. 10-2: Coordinate system

The machine axes can also be arranged in a "non-Cartesian" manner. Thus, a non-Cartesian coordinate system is created.

10.3.2 Overview on coordinate systems

Purpose: To execute part programs with different tool dimensions or on different machines, **various coordinate systems** are defined within the control. These coordinate systems are partially fixed and machine-dependent, but mainly freely definable and machine-independent.

The coordinate systems (also see [fig. 10-3 "Correlation between machine, workpiece and program coordinate system" on page 158](#)) of the MTX refer to

- Axis coordinates
- Machine coordinates
- Workpiece coordinates



For a detailed description of the individual coordinate systems, refer to the following chapters

10.3.3 Axis coordinate system ACS

The axis coordinate system ACS (ACS = **A**xis **C**oordinate **S**ystem) is spanned by all axes defined in a channel.

The zero points of the individual axes are defined by the respective zero points of the position measuring systems:

- The reference points of the individual axes (with **cyclically absolute measuring systems**) are defined axis-by-axis in relation to the respective axis zero position. Frequently, they are in the marginal area of the axes and can be approached automatically.
- Reference points are required to establish reference between an axis equipped with an **incremental measuring system** and the respective axis zero point. For these axes, the reference point has to be approached first after machine activation or power failure.

The reference point R represents the positions resulting when all axes are on their reference position.

- For axes with absolute measuring systems, no reference point is needed as the measuring system directly provides the axis positions after the activation.

Designation and symbol:

Reference point R 

10.3.4 Machine coordinate system MCS

Generally, the machine coordinate system MCS (**M**achine **C**oordinate **S**ystem) is a system of coordinates which is solidly connected to the machine frame in a Cartesian manner and which is defined in a channel by the total of the machine coordinates. If the machine kinematics does not have a Cartesian structure (e.g. machines with parallel kinematics), the relation between the axis (ACS) and the Cartesian machine coordinates (MCS) is established via a special axis transformation at point of action 1 (AT1).

Without AT1, MCS and ACS are identical.

The machine position - at which all machine coordinates are on their respective zero position - is also called machine zero point. It is defined by MP 1030 000130 "Axis / machine position of reference position".

Zero point name and symbol:

Machine zero point M 

Additional $MCS^{(i=\max)}$ machine coordinate systems are created by zero offset (ZO). The first machine coordinate system is $MCS^{(0)}$, the last machine coordinate system is described as the "Local machine coordinate system" LCS ($=MCS^{(i=\max)}$).

10.3.5 Workpiece coordinate system WCS

Workpiece coordinate systems (WCS) are linked to the workpiece and thus fixedly linked to the respective workpiece holder.

The BCS (see [chapter 10.3.6 "Basic workpiece coordinate system BCS" on page 157](#)) is the first workpiece coordinate system.

With so-called placements (see [chapter 12 "Coordinate transformations" on page 175](#) for coordinates transformation), the workpiece coordinate systems are randomly shifted and rotated in space. Since multiple placements can be connected in series, each placement defines a new local workpiece coordinate system $W^{(i)}$ ($i=1\dots n$).

Example:

A workpiece coordinate system describes a local coordinate zero point $W^{(1)}$ regarding the absolute workpiece zero point $W^{(0)}$, another workpiece coordinate system describes the relation between two local workpiece zero points (e.g. $W^{(1)}$, $W^{(2)}$).

10.3.6 Basic workpiece coordinate system BCS

The first workpiece coordinate system in a potential sequence of several local workpiece coordinate systems connected in series is the basic workpiece coordinate system BCS (**B**asic **W**orkpiece **C**oordinate **S**ystem)

The zero point of the basic workpiece coordinate system is the first workpiece zero point $W^{(0)}$ in a series of n-possible local workpiece zero points $W^{(n)}$ which can be generated by placements (see [chapter 12 "Coordinate transformations" on page 175](#) for more details).

Zero point name and symbol:

Workpiece zero point W 

An axis transformation at point of action 2 (AT2) establishes the relation between the basic workpiece coordinate system and the machine coordinate system LCS. In this axis transformation, the "zero position" of the machine coordinates involved is defined via machine parameters to get the workpiece zero point $W^{(0)}$ of the BCS. Therefore, the machine parameter 1030 00130 "Axis machine position of reference position" is used.

If there is no AT2 axis transformation, since the machine coordinates run parallelly to the workpiece coordinates, the workpiece zero point $W^{(0)}$ of the BCS corresponds to the zero point of the local machine coordinate system (LCS).

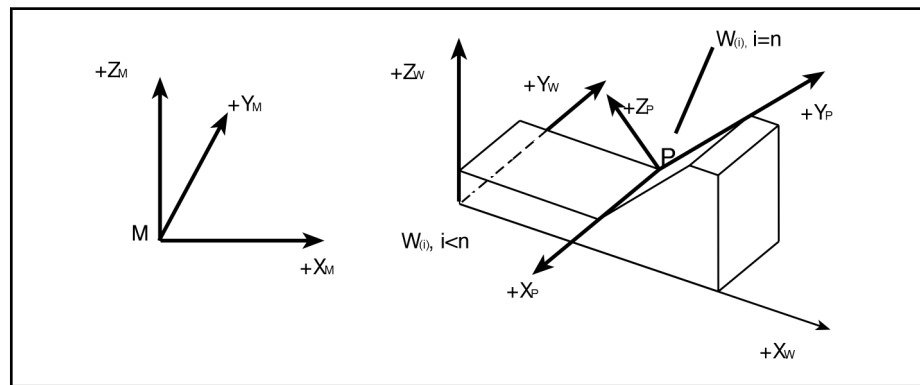


Fig. 10-3: Correlation between machine, workpiece and program coordinate system

10.3.7 Program coordinate system (PCS)

The nth workpiece coordinate system provided with the currently highest index in each case ($W(i)$, with $i=n$) is described as program coordinate system PCS (= Program Coordinate System).

All programmed coordinate values of the part program refer to the program coordinate system. According to the definition of the workpiece coordinate systems, the program coordinate system can be randomly shifted and rotated in relation to any workpiece zero point $W(i)$.

The program zero point is identical with the zero point of the program coordinate system.

Zero point name and symbol:

Program zero point P 

10.4 Transformations between coordinate systems

Axis transformations

The transition between coordinate systems, i.e. between axis and machine coordinates (ACS - MCS) or between machine and workpiece coordinates (MCS - WCS) is realized using **axis transformations** at certain **points of action**.

Usually, an axis transformation at point of action AT1 transforms from a non-Cartesian axis coordinate system (ACS) into a usually Cartesian coordinate system solidly connected with the machine frame (MCS).

On this basis, a second axis transformation (at point of action AT2) can transform from the machine coordinate system to the workpiece coordinate system. At AT2, there are usually transformations based on a **Cartesian** machine coordinate system.



For a detailed description on "transformations between coordinate systems", refer to the documentation "MTX Functional Description 14VRS Extension", chapter "Axis transformation".

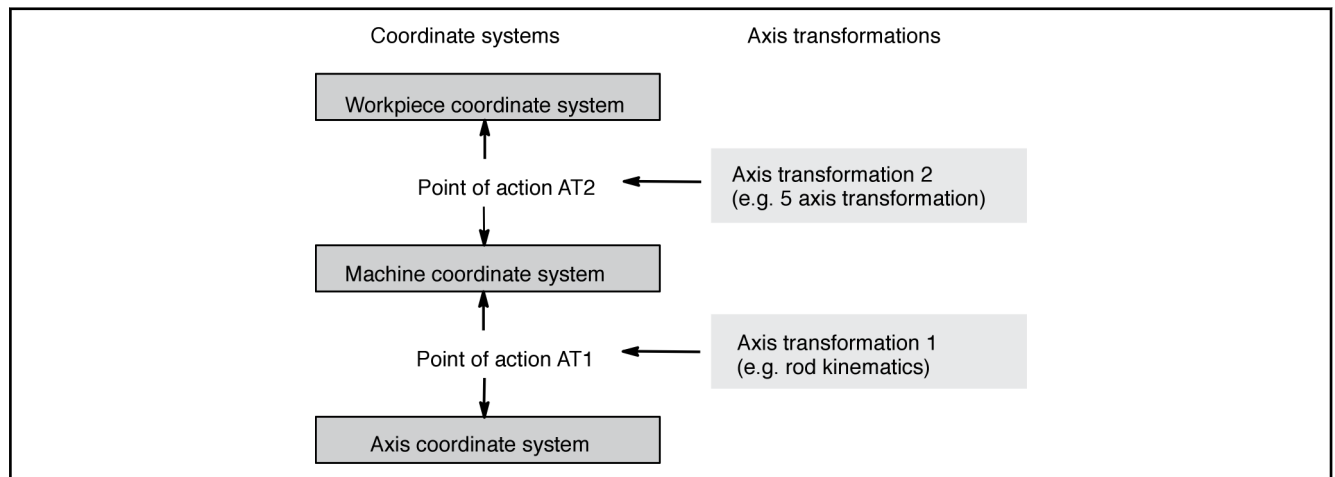


Fig. 10-4: Axis transformation

Zero offsets

The coordinate system MCS is divided into further coordinate systems $MCS(i=1..n)$ when **zero offsets** (see [chapter 11 "Zero offsets \(ZO\)" on page 169](#)) are considered.

Coordinate transformations

The workpiece coordinate system (WCS) is divided into further coordinate systems $WCS(i=1,..,n)$ if the following **linear coordinate transformations** are considered:

- Placement: Workpiece position correction BcsCorr (BCR)
(refer to [chapter 12.2 "Placement: Workpiece position correction BcsCorr \(BCR\)" on page 175](#))
- Placement: Inclined plane G154.1 .. G159.5
(refer to [chapter 12.3 "Placement: Inclined Plane G152.1...G159.5" on page 177](#))
- Program coordinate offset Trans (TRS), ATrans (ATS)
(refer to [chapter 12.4 "Program Coordinate Offset Trans, \(TRS\); ATrans, \(ATR\)" on page 185](#))

Additionally, all corrections and compensations that can be effective in the various coordinate systems have to be mentioned. Including:

- Tool corrections in the WCS
(refer to [chapter 14 "Tool corrections" on page 189](#))
- Axis compensations, such as LSEC, cross compensation in the ACS

The following figure shows the axis transformations as well as the linear coordinate transformations responsible for the transitions of the coordinate systems:

Axes, coordinates, coordinate systems

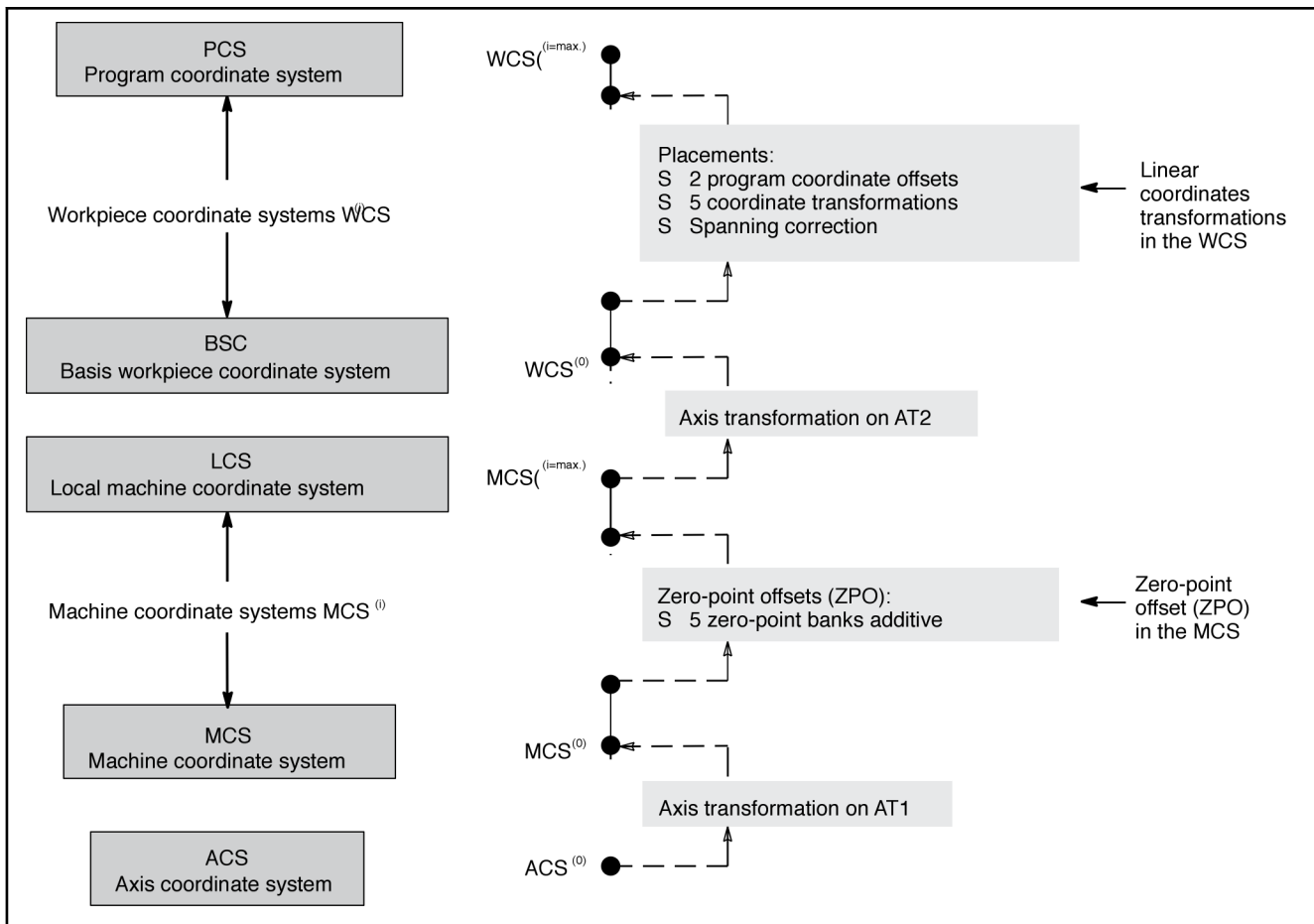


Fig. 10-5: Coordinate transformation

10.5 Axes and coordinates in the channel

Axis names:(channel axis names)

The term "coordinates identifiers" describes axis names (channel axis names) and coordinates names.

For each axis of a channel, a **channel axis name** can be defined in the MP 7010 00010 "channel axis name". Within the respective channel, i.e. within the axis group, this channel axis name has to be unique.

An axis can be controlled via its channel axis name or its system axis name.

The system axis name is defined in MP 1003 00001. This name has to be unique for all system axes.

If no channel axis name is defined for a channel axis, it implicitly assumes the system axis name as channel axis name.

If a name is assigned as channel axis name as well as system axis name, the channel axis name is always resolved while programming the name. In this case, a channel axis name covers a system axis name.

A channel axis index is assigned to each axis name.



In case of identical names, a channel axis name always has a higher priority than a system axis name!

Coordinate names:

When axis transformations are used, the system coordinates can be identified for each channel with **coordinate names** in MP 7080 00010 "Name of

workpiece coordinates" and in MP 7080 00020 "Name of machine coordinates":

- Default settings in MP 7080 00010:
x, y, z, phi, theta, psi
- Default settings in MP 7080 00020:
X, Y, Z, PHI, THETA, PSI

The default settings correspond to the internal definitions of the system coordinates (see [chapter 10.2 "System axes and system coordinates" on page 153](#)).

Having unambiguous system coordinate indices in the range of -12 to -1, all twelve coordinates can be uniquely identified across all channels.

If required, coordinate names can be changed. In case of suitable machines, coordinates and axes can be named equally.

Channel axis index:

Within one channel, all channel axes are numbered consecutively starting with the index 1. The channel axis index results from the sequence of the system axis indices, i.e. within a channel, the axis with the lowest system axis index is provided with channel axis index 1. Accordingly, the axis with the highest system axis index is assigned to the highest channel axis index.

During the axis transfer of a new axis in the channel and during the removal of an axis from a channel, the channel axis indices are renumbered.

Channel axes can only be controlled from the channel they were assigned to via machine parameter or NC function (see "Channel axis names and axis group").

The assignment of axes to channels set in the machine parameter can be changed in the part program via the "Axis transfer" functionality (GetAxis, WaitAxis, RemAxis, DefAxis).

Axis identifiers in the channel:

System axis index /drive index = Machine parameter index	1	2	3	4	5	6	7	8
System axis name (1003 00001)	X1	Y1	X2	Y2	-	Z1	Z2	E
Channel assignment (1003 00002)	1	1	2	2	-	1	2	0
Drive function type (1001 00001)	Axis	Axis	Axis	Axis	Spindle	Axis	Axis	Axis
Channel axis index, channel 1	1	2	-	-	-	3	-	-
Channel axis name, channel 1 (7010 00010)	X	Y	-	-	-	Z	-	-
Channel axis index channel 2	-	-	1	2	-	-	3	-
Channel axis name channel 2 (7010 00010)	-	-	X	Y	-	-	Z	-
Axis type	Synchro- nous Axis	Synchro- nous Axis	Synchro- nous Axis	Synchro- nous Axis		Synchro- nous Axis	Synchro- nous Axis	Asyn- chronous Axis

Tab. 10-2: Axis identifiers in the channel

Axes, coordinates, coordinate systems

Example:

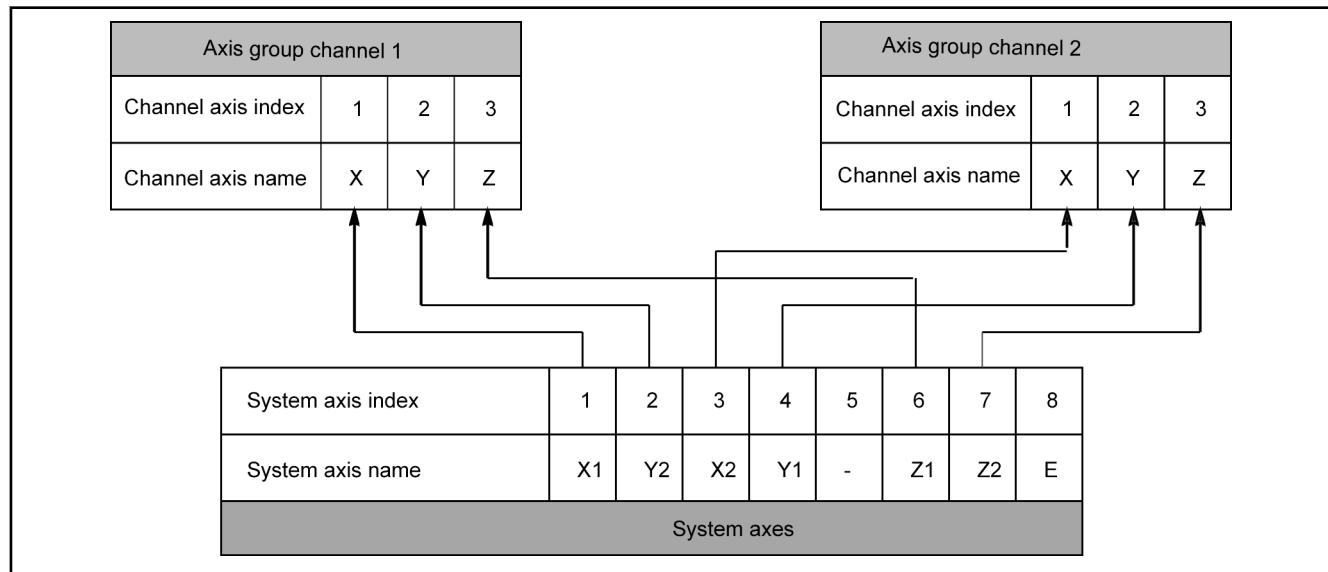


Fig. 10-6: Example - Axis identifiers in the channel

Coordinates identifier depending on the coordinate system:

In the coordinate systems ACS, MCS and WCS, various coordinate identifiers (coordinate names and axis names) are used with regard to the respective axis transformation (also see the following illustration):

- There are only channel axis names (MP 7010 00010) in the **ACS**.
- In the **MCS**, there can be the machine coordinate names (MP 7080 00020) in addition to the channel axis names.
- In the **WCS**, there can be channel axis names, machine coordinate names and workpiece coordinate names (MP 7080 00010).

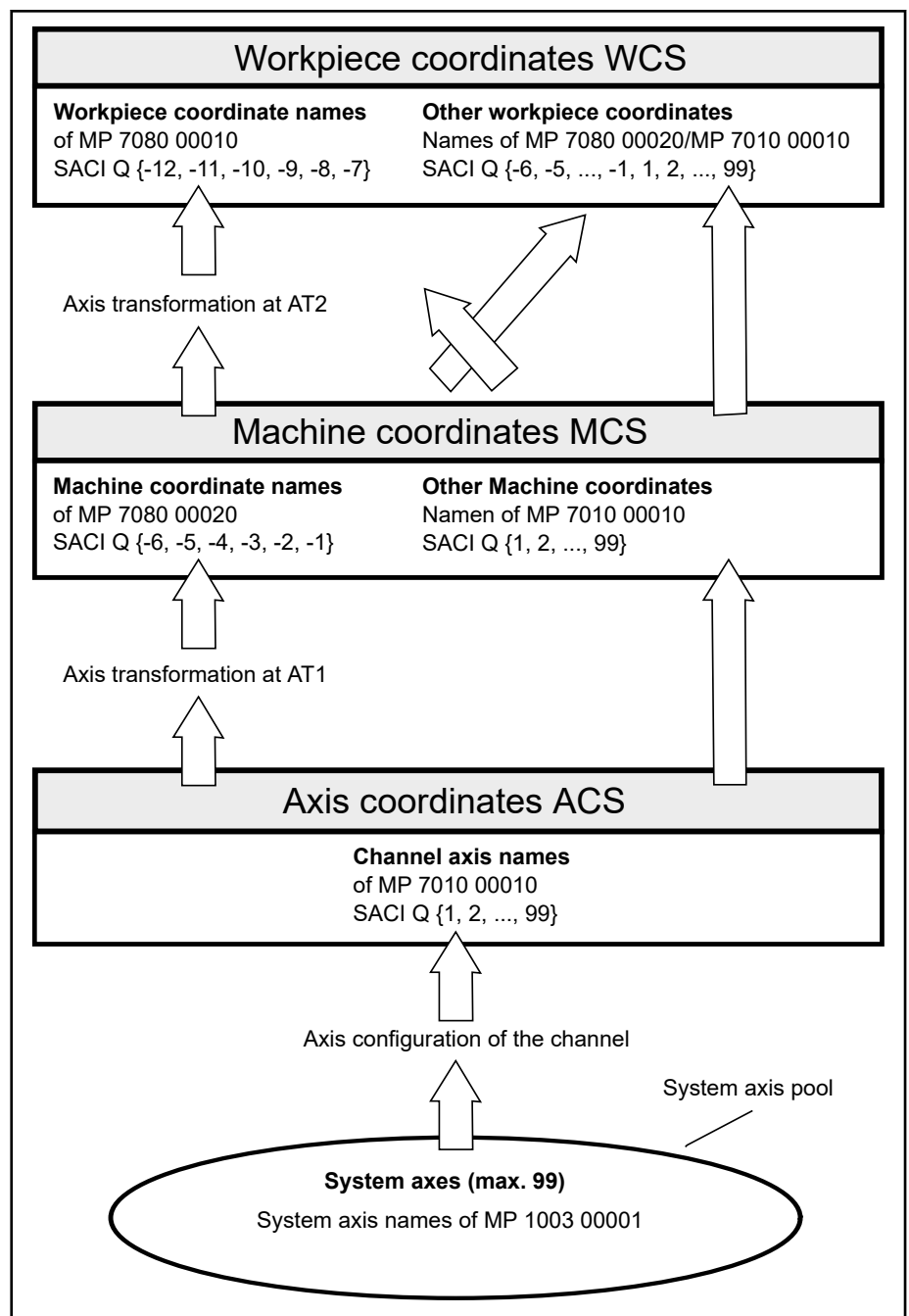


Fig. 10-7: Coordinates identifiers depending on the coordinate system

Rules for coordinate identifiers:

- The name of machine coordinates linked with an axis transformation AT1 is always identical to the identifier of the respective Cartesian coordinate (MP 7080 00020).
- The name of machine coordinates not linked to any axis transformation is always identical to the identifier (channel axis name) of the respective channel axis (MP 7010 00010). This program coordinate can alternatively be addressed via the respective system axis name (MP 1003 00001) (precondition: The system axis name has to be unambiguous).
- The name of program coordinates linked with an axis transformation AT2 is always identical to the identifier of the corresponding spatial coordinate (MP 7080 00010).

- The name of program coordinates not linked with an axis transformation AT2, but with an axis transformation AT1, is always identical to the identifier of the corresponding Cartesian coordinate (MP 7080 00020).
- The name of program coordinates not linked to any axis transformation is always identical to the identifier (channel axis name) of the corresponding channel axis (MP 7010 00010). This program coordinate can alternatively be addressed via the respective system axis name (MP 1003 00001) (precondition: The system axis name has to be unambiguous).

Example 1:

Machine with 17 system axes: No axis transformation active

As no axis transformation is active, the coordinate identifiers in all coordinate systems are identical to the names of the channel axes.

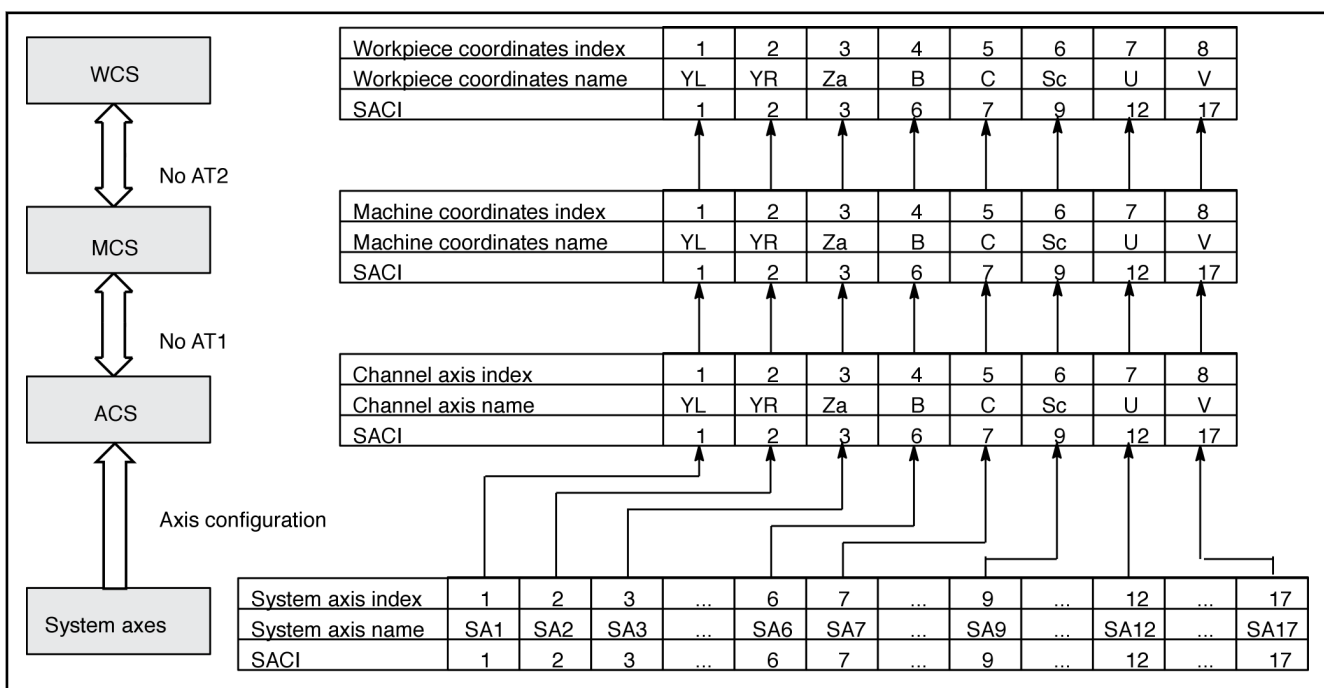


Fig. 10-8: Example 1

Example 2:

Machine with 17 system axes: Bipod axis transformation is active at point of action AT1

The Bipod axis transformation transforms between the channel axes:

YL, YR, Za

and the coordinates

X(-6), Y(-5), Z(-4).

The WCS and MCS coordinates B, C, and Sc are directly assigned to the axes in the ACS.

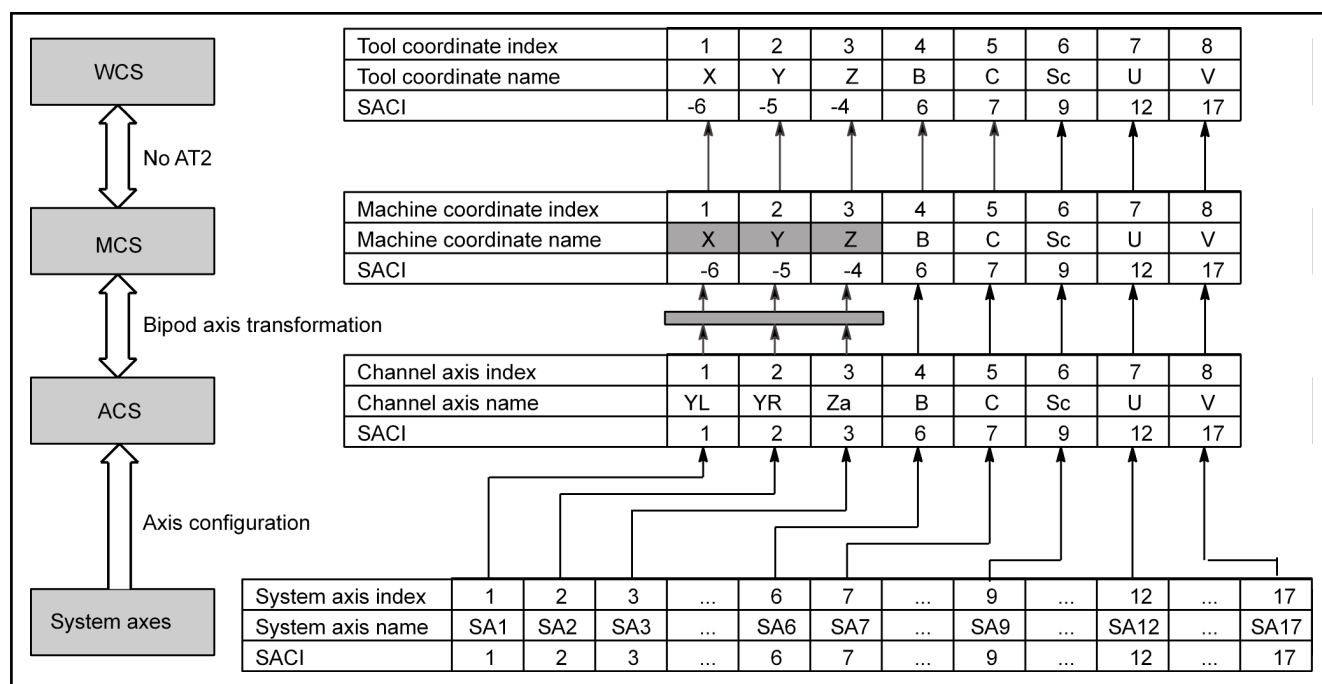


Fig. 10-9: Example 2

Example 3:

Machine with 17 system axes: AT1 = Bipod, AT2 = 5-axis transformation

The Bipod axis transformation is followed by a 5-axes transformation (e.g. LLLRR, type 3232201). It transforms between the MCS machine coordinates: X(-6), Y(-5), Z(-4), B(6), C(7)

and the WCS coordinates:

x(-12), y(-11), z(-10), phi(-9), theta(-8).

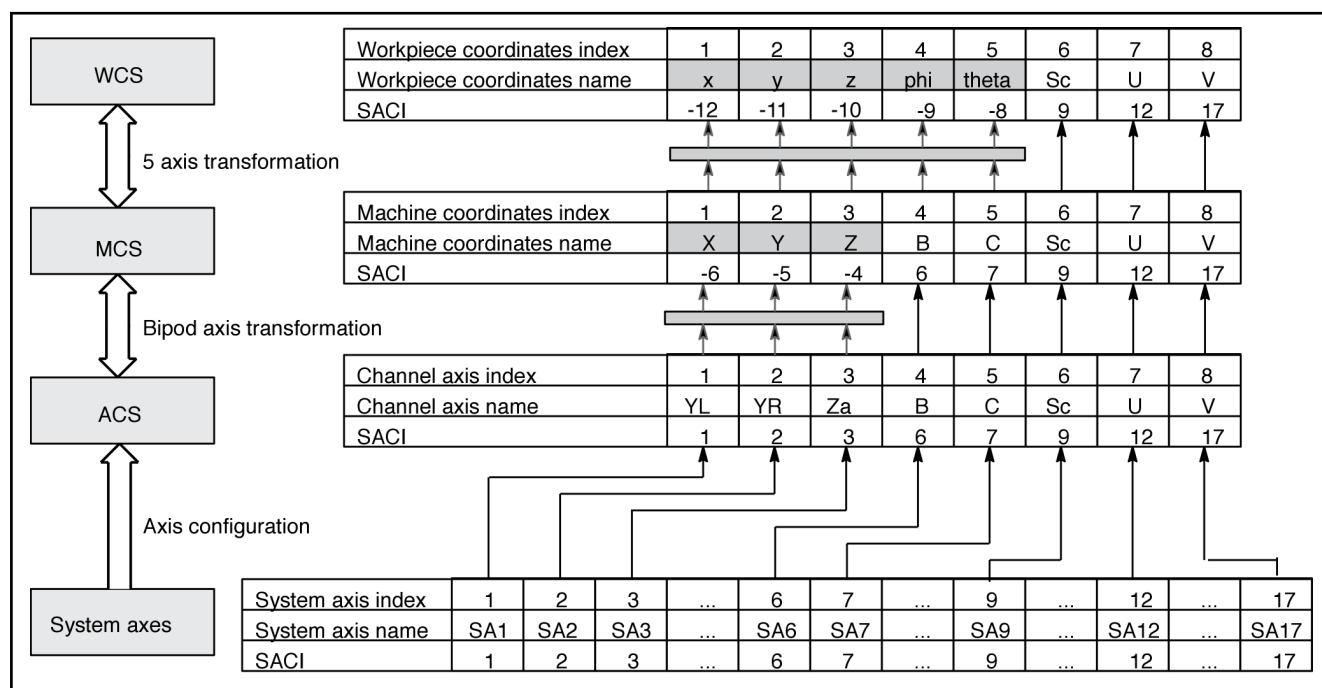


Fig. 10-10: Example 3

Example 4:

Machine with 17 system axes:

AT1 = no axis transformation,

AT2 = 5-axis transformation

The 5-axis transformation transforms between the machine coordinates:

Xa(1), Ya(2), Za(3), B(6), C(7)

and the WCS coordinates:

x(-12), y(-11), z(-10), phi(-9), theta(-8).

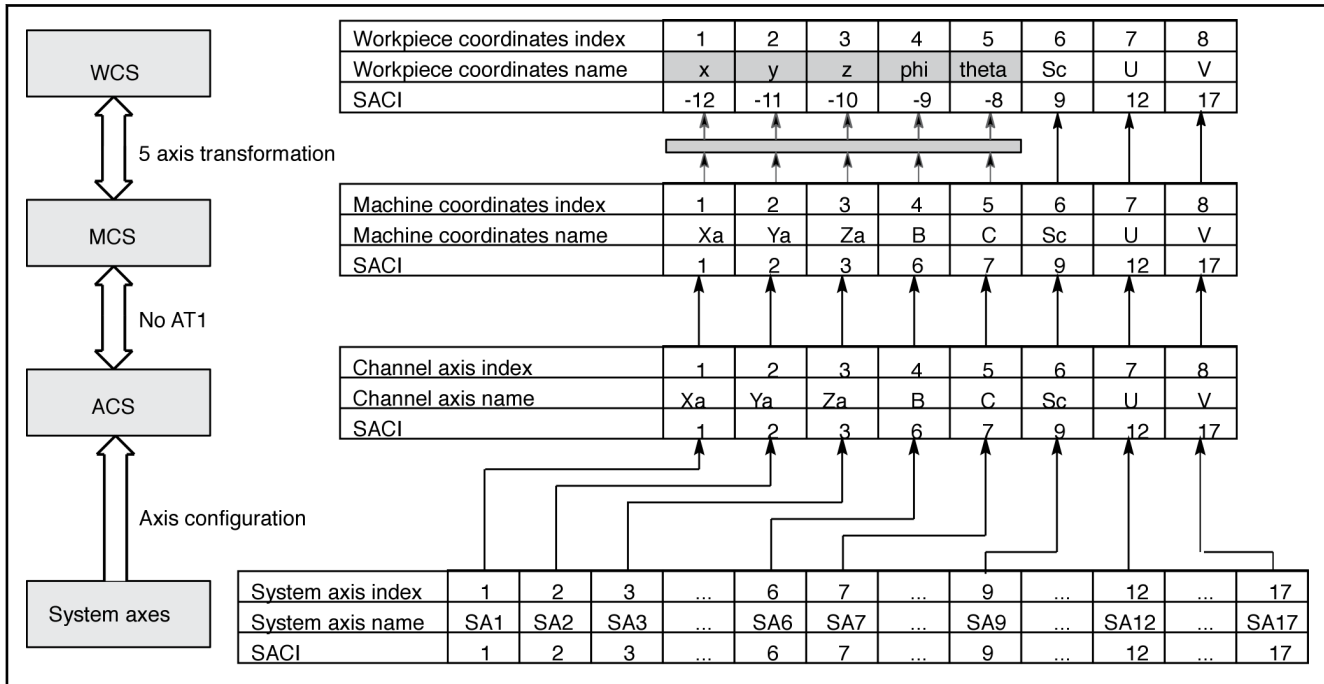


Fig. 10-11: Example 4

Rules for the coordinate index:

Within the coordinate system, the coordinates are assigned to the coordinate index starting consistently from coordinate 1 according to an ascending SACI (System Axis and Coordinate Index). This is to say:

- Coordinate index 1 is provided with the coordinate of the smallest SACI
- Coordinate index 2 is provided with the coordinate of the next higher SACI
- etc.

A re-assignment of the coordinate to the coordinate index takes place if:

- a new axis is integrated into to the channel
- an axis is removed from the channel
- an axis transformation is switched on / activated in the channel
- an axis transformation is switched off in the channel

10.6 NC functions and their relation to coordinate systems

Many NC functions are programmed with axes or coordinates. During execution, some NC functions refer to different coordinate systems. Thus, the reference to these coordinate systems also determines the axis and coordinate names assigned to axes and coordinates.

The following lists include NC functions classified according to their action in ACS, MCS, or WCS.

NC functions in the ACS:

Syntax	NC function
AxAcc	Maximum axis acceleration
G9(X...,Y...,Z...)	Axis-specific SHAPE
KvProg	CO programming
G74	Approach reference point
FeedForward	Feed forward
PosMode	Positioning type for endless axes
RedTorque	Torque reduction
AxCouple	Axis coupling
FeedAd	Hiding axes in the feed calculation
Writeld	Sercos telegrams
PtBlkEnd, PtlInpos	Stroke release times
TangToolTangentiale	Tool guidance
TangToolOriTangentiale	Tool orientation
GetAxis, ...	Axis transfer
AssLogName, RemLog-Name	Assigning channel axis names

Tab. 10-3: NC functions in the ACS

NC functions in the MCS:

Syntax	NC function
G54.x - G59.x	Zero offset
G76	Approach fixed machine position

Tab. 10-4: NC functions in the MCS

NC functions in the WCS:

Syntax	NC function
PolarPole	Polar coordinates
G17, G18, G19, G20	Plane selection
Scale, Mirror	Rotation and mirroring
Shift	Contour offset
G47	Tool correction
G78, G79	Switching drill axis
SetPos	Set program position
G1, G2, G3, etc.	Path interpolation functions
BcsCorr	Workpiece position correction
G152.x - G159.x	Inclined plane
Trans, ATrans	Programmable coordinate offset

Tab. 10-5: NC functions in the WCS

11 Zero offsets (ZO)

11.1 Basics

11.1.1 Description

Function

The zero offsets are used to displace the machine coordinate system (MCS) in direction of the machine coordinates (i.e. transformed). The resulting offset coordinate system is called a "Local Machine Coordinate System" (LCS).

If no coordinate transformations and no axis transformations are active when the ZO is called, the offset machine zero point corresponds to the workpiece zero point.

The offset values of the machine coordinates are stored in zero offset tables.

Each zero offset table comprises five zero offset banks (groups) with six zero offsets each:

- G54.1 .. G59.1
- G54.2 .. G59.2
- G54.3 .. G59.3
- G54.4 .. G59.4
- G54.5 .. G59.5



For the structure of zero offset tables, refer to [chapter 11.2 "Structure of zero offset tables" on page 172](#).

Activating an Offset

To activate the offset, first select the desired ZO table using the ZoTSel (ZOS). Subsequently, select the offset values of the respective machine coordinates using the ZO functions "G54.1 .. G59.5".

Example:

Only zero offsets of the "ZO1.zot" table are active!

Zero offsets (ZO)

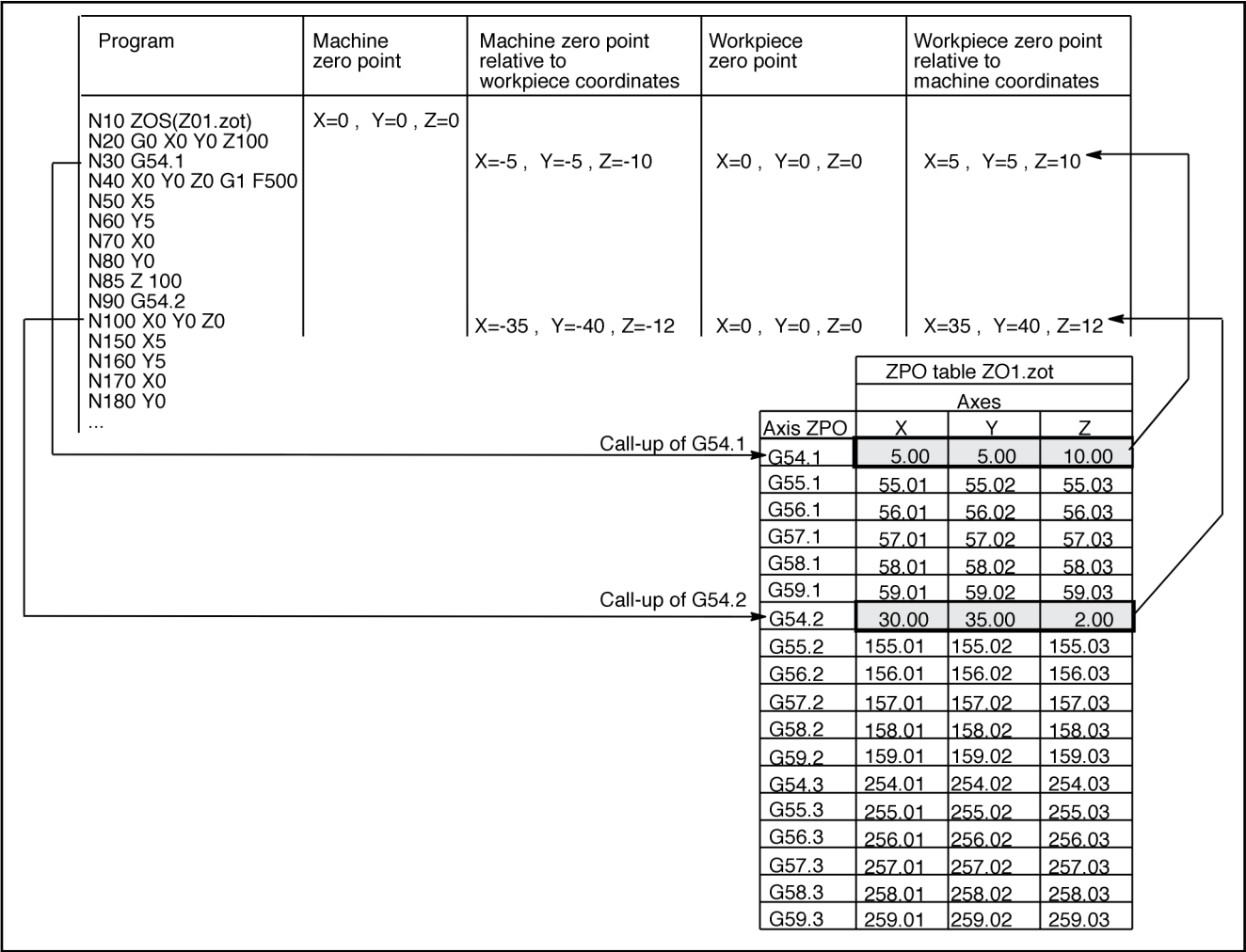


Fig. 11-1: Zero offsets (ZO)

Since an activation of a zero offset does not mandatorily require that offset values are added to all available axes, the following special feature can result together with **additively effective zero offsets**:

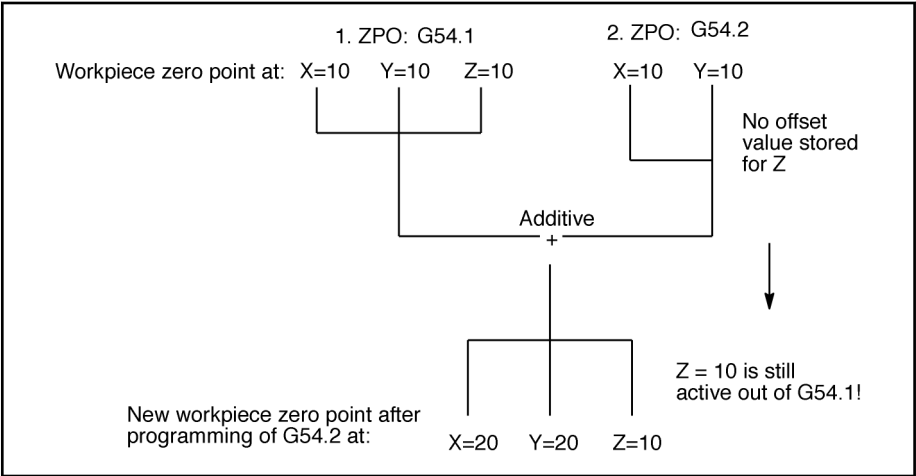


Fig. 11-2: Special feature - Zero offset (ZO)

Restrictions

- Zero offsets of a zero point bank (group) act modally and neutralize each other within their own group.

This applies to:

- G54.1 .. G59.1: 1. Zero offset
 - G54.2 .. G59.2: 2. Zero offset
 - G54.3 .. G59.3: 3. Zero offset
 - G54.4 .. G59.4: 4. Zero offset
 - G54.5 .. G59.5: 5. Zero offset
- Active zero offsets from different banks (groups) act **additively**.
- If no axis transformation is active, **linear coordinates transformations** affect the zero offsets additively.

The following functions are coordinates transformations:

- Placement: Workpiece position correction BcsCorr (BCR)
- Placement: Inclined plane G154.1 .. G159.5
- Program coordinates offset Trans (TRS), ATrans (ATR)

Relevant NC functions

ZoTSEL(<{Path}/Table name{.Extension})	Selection and activation of a ZO table (short syntax ZOS)
G54{.1} .. G59{.1}	1. Activate zero offset
G54.2 .. G59.2	2. Activate zero offset
G54.3 .. G59.3	3. Activate zero offset
G54.4 .. G59.4	4. Activate zero offset
G54.5 .. G59.5	5. Activate zero offset
G53	Disable all active zero offsets
G53.1	1. Disable zero offset
G53.2	2. Disable zero offset
G53.3	3. Disable zero offset
G53.4	4. Disable zero offset
G53.5	5. Disable zero offset

Tab. 11-1: Relevant NC Functions

Relevant CPL functions

Access to ZO tables	
ZOT(...)	Write and read-only access to any zero offset table.
ZOTCR(...)	Creates a new ZO table without correction blocks
ZOCDEL(...)	Deleting a columns in a ZO table
ZOCINS(...)	Inserting a columns into an existing ZO table
ZOV(...)	Provides the effective ZO values for an axis/machine coordinate

Zero offsets (ZO)

General access to XML tables	
XTAB(...)	Write and read-only access to any XML table.
XTABCR(...)	Creates an XML table with no data within the MTX file system

Tab. 11-2: Relevant CPL functions

Relevant MCODS (Motion Control data services of the NCS per CPL)

Corrections	
MCODS(14,...)	Name of the active axis/ZO table
MCODS(15,...)	Active axis/ZO values (provides values for ZO blocks 1-3 only)
MCODS(52,...)	Active external axis/ZO values
System structure	
MCODS(99,...)	Active data block; zero offset
MCODS(100,...)	Sum of zero offset

Tab. 11-3: Relevant MCODS functions

Relevant machine parameters (MP)

User interface (GUI)	
ZotDisp/Ch(x)/ NofZotSets (6005 00061)	Number of zero offset blocks to be displayed channel-specifically
TabDisp/ PrecLin(Rot)Metr (6020 00021)	Defines the number of decimal positions in the zero offset table to be entered and displayed for linear axes or rotary axes (editor with metric input).
TabDisp/ PrecLin(Rot)Inch (6020 00022)	Defines the number of decimal positions in the zero offset table to be entered and displayed for linear axes or rotary axes (editor with inch entry).
NC optimization (NCO)	
SrchPathSubProg (3080 00001)	Search paths
CorrUnit/SysCorrUnit (9020 00010)	Indicates in which unit (mm or Inch) table values are stored internally.

Tab. 11-4: Relevant machine parameters



Any modifications to MP NCO/CorrUnit/SysCorrUnit (9020 00010) also affect D-correction tables, placement tables (inclined plane) and the interpretation of the geometry correction values transmitted by the PLC!

11.2 Structure of zero offset tables

11.2.1 Description

Structure

The zero offset values are provided via the zero offset tables. In a ZO table, 30 offsets can be stored per applied machine coordinate. A table contains information for up to 99 coordinates.

Within a ZO table, there is one entry for each relevant machine coordinate consisting of:

- Axis/coordinate name
- Axis/coordinate type (linear or round)
- Offset values for the functions G54.1-G59.6
- Channel assignment



Offsets with channel number 0 become effective in every channel in which they are activated. However, offsets with a channel number > 0 can only become effective in the specified channel.

Do not to use the channel number 0 in case of multiple entries of the same machine coordinates with different channel details, as all following channel-specific offsets of this coordinate are not considered in this case.

Furthermore, there is global information, e.g.:

- Unit in which the offset values for the linear machine coordinates are stored: mm or inch in machine parameter NCO/CorrUnit/SysCorrUnit (9020 00010)
- ID "strict assignment"

The control saves such tables as XML files with an exactly defined structure.



Use the ZO editor to edit these tables easily (see Standard NC Operation).

Table name

The table name of an ZO table consists of a freely selectable file name and the file name extension ".zot" (e.g. ZO1.zot).

MTX default name for ZO tables: ZO<integer number>.zot

Table access

When a ZO table is selected, the complete table is read in and stored control-internally. When a zero offset is selected, the current machine coordinates of the channel are internally assigned to the table columns via the respective column names.

If one assignment to the same machine coordinate is to be possible for several columns, the column with the lowest column index is linked.

After each modification of the current machine coordination configuration of the channel (axis exchange, switching the axis transformation AT 1), the assignment is redefined.

If the ZO table active in one channel is edited via the MTX table editor or the CPL (ZOT command), the table is automatically read in once more with the next NC block to be prepared.

11.2.2 Handling Instruction: Zero Offsets

- Applying:
- Tool must no longer be active.
 - A zero offset table for the machining program exists and contains the correct offset values.
 - Any other still active offsets have to be considered when a zero offset is programmed as they can act additively or overwrite each other.

Zero offsets (ZO)

- Offsets from different ZO banks are always additive.
- Offsets within a ZO bank mutually overwrite each other.

Activating: The following NC program line activates a zero offset table in the channel:

ZoTSel({<Path/>Table name>{.zot}) or

ZOS({<Path/>Table name>{.zot}) or

With MTX standard names according to naming convention (**ZO**<IntValue>.zot):

ZoTSel(IntValue) or **ZOS**(IntValue)

Example: For the zero offset table 5

ZoTSel(ZO5.zot) or **ZoTSel**(5)

Activate the desired zero offset with the respective NC function **G54.1...G59.5**. The corresponding offset of the ZO base 1 is activated using the short variant of the NC function **G54...G59**.

Deactivating: Deactivating the ZO from the NC program.

IW Operation / NC programming: Deactivating zero offsets

Enter the following commands:

- using the NC function **G53**, all zero offsets are deactivated for all correction banks
- or -
- using the NC function **G53.x**, the active zero offset of a ZO base is deactivated.



Tool must no longer be active.

12 Coordinate transformations

12.1 Overview

Coordinate transformations transform workpiece coordinates. This changes the reference system for processing.

WCS(0 till i=max.); e.g. BCS and PCS.

Linear coordinate transformations comprise the following functions:

- Placement: Workpiece position correction BcsCorr (BCR)
- Placement: Inclined plane G152.1 .. G159.5
- Program coordinate offset Trans (TRS), ATrans (ATS)

12.2 Placement: Workpiece position correction BcsCorr (BCR)

12.2.1 Description

Function

With the placement **Workpiece position correction**, the workpiece coordinate system (WCS) is freely moved and oriented in space. The workpiece position correction affects the Cartesian axes/coordinates (e.g. axes X, Y and Z or with active axis transformation AT2 on the system coordinates x, y and z) in the corresponding channel.

Application:

The workpiece position correction is a spanning correction used to measure the workpiece position after the spanning. The workpiece position correction reduces the setup effort.

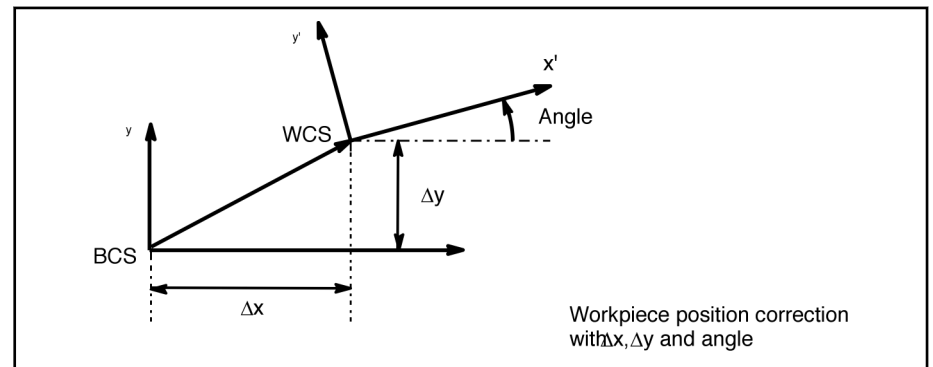


Fig. 12-1: Placement: Workpiece position correction

Restriction

- Additional placements (e.g. inclined plane) have an additive effect. The workpiece position correction is located before inclined plane in the "sequence of calculation".
- Activating and deactivating the workpiece position correction interrupts the block look-ahead. Thus, it may not be programmed when the milling cutter path correction is active.

Relevant NC functions

BcsCorr(...)	Workpiece position correction ON
BcsCorr()	BcsCorr(0) workpiece position correction OFF
BCR(..)	Short form of the workpiece position correction

Tab. 12-1: Relevant NC Functions

Relevant CPL functions

DPC(..)	Returns for the current channel (here: channel in which the program is runs with the DPC command) the latest programmed parameters of the workpiece position correction "BcsCorr" of an axis (offset values and angles of rotation).
ROT2EUL	Calculates the Euler angle from a maximum of three successive rotations along the axes x, y and/or z.
ABSROT2EUL	Calculates the Euler angle from a maximum of three rotations along the spatially-fixed axes x, y and/or z.
PT2EUL	Calculates Euler angles so that the three points are located in the xy-level of a rotated coordinate system.
VEC2EUL	Calculates Euler angles for a rotated coordinate system for which the vectors for the z- and x-direction were applied.

Tab. 12-2: Relevant CPL Functions

12.2.2 Handling Instruction: Workpiece Position Correction BcsCorr (BCR)

Application: The placement "workpiece position correction" can shift and orientate the workpiece coordinate system anywhere in space. The workpiece position correction is effective in the respective channel and is used as setting-up correction.

BcsCorr({<XW offset>},{<YW offset>},{<ZW offset>},{<angle1>},{<angle2>},{<angle3>}})

<XW offset>:

Offset value in the direction of the primary coordinate

<YW offset>:

Offset value in the direction of the secondary coordinate

<ZW offset>:

Offset value in direction of the normal coordinate

<Angle1>:

Angle for turning by the z-coordinate.

Value range: 0 = <Angle1> < 360 degrees

<Angle2>:

Angle for turning by the new Y' coordinate.

Value range: 0 = <Angle2> < 180 degrees

<Angle3>:

Angle for turning by the new Z'' coordinate.

Value range: 0 = <Angle3> < 360 degrees

Activate: The workpiece position correction is activated with **BcsCorr(...)**. To this end, the respective information for offsets and rotation of the coordinates of the workpiece coordinate system has to be supplied.



Additional placements (e.g. Inclined plane) have an additive effect. The workpiece position correction is at first place in the calculation chain.

Activating and deactivating workpiece position correction interrupts block look-ahead; therefore, it may not be programmed when milling cutter path correction is active.

N70 G40	Milling cutter path correction OFF
N80 BCR (50,300,10,1.23)	Workpiece position correction ON
:	
:	
:	The zero point of the new workpiece coordinate system is located in the BCS on X50 Y300 and Z10. With regard to the BCS, the X and y-coordinate axes of the new workpiece coordinate system rotated counter-clockwise by 1.23 degrees around the new z-coordinate axis.
:	
:	
:	
:	
N200 BCR()	workpiece position correction

Deactivate: Program **BcsCorr(0)**

12.3 Placement: Inclined Plane G152.1...G159.5

12.3.1 Description

Function

The "inclined plane" function causes an offset and/or rotation of several coordinates of the workpiece (WCS) or program coordinate system (PCS) in space.

Coordinate transformations

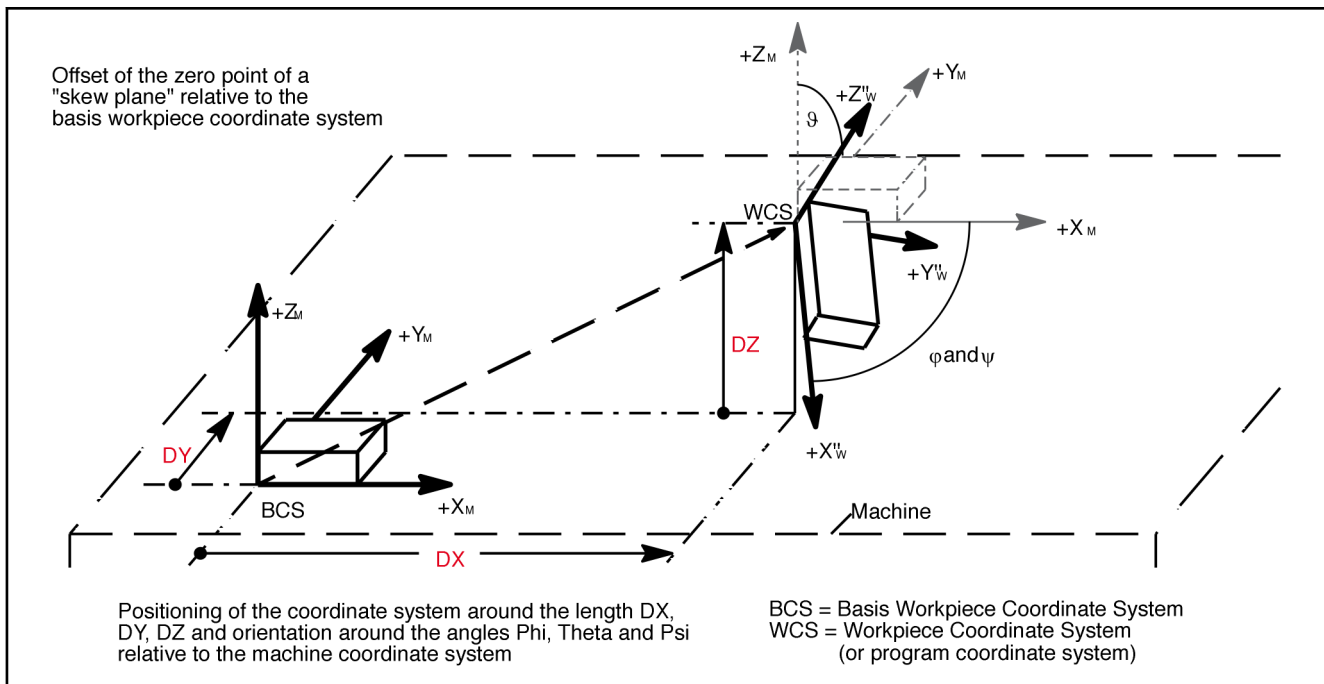


Fig. 12-2: Placement: Inclined Plane G152.1...G159.5

Thus, circular motions in any planes in space are possible and part programs to machine monopitch-shaped workpieces for example can be considerably simplified.

Another application of the inclined plane is drilling holes with a tool freely oriented in space. The NC function "Inclined plane vertical to tool" supports this application.

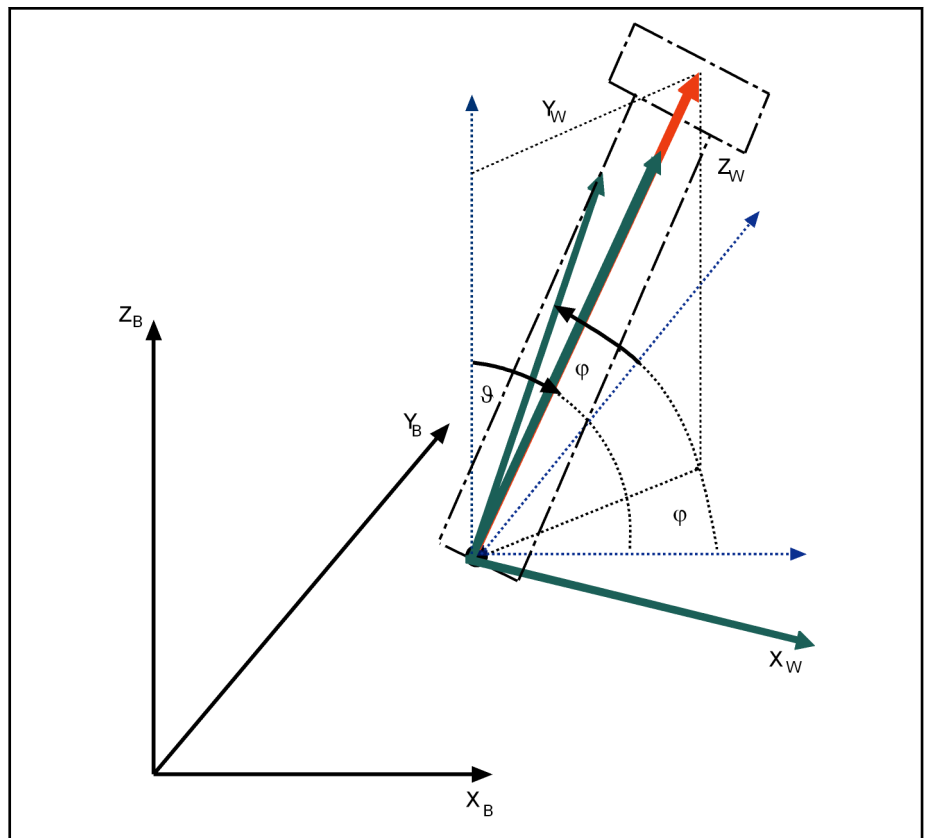


Fig. 12-3: Inclined plane vertical to tool

The end position of the inclined plane is defined by

- Basic rotations with the Euler angles phi, theta and/or psi
- Offsets in the coordinates around the paths DX, DY, DZ

Required offset values and rotary angles can either be

- stored in tables or
- can be programmed **directly** in the G152.x block or
- can be activated implicitly according to the active TCP and the current tool orientation by programming G151.x.

An inclined plane is activated via a placement table. The offset values and orientation angles are preset with the placement functions "G154.1..G159.5".

Relevant NC Functions

PmTSel(...)	Activate "inclined planes" table via NC block.
PMS(..)	Activate short form for "inclined planes" table via NC block.
G152.x(...)	Program "inclined planes" with all required parameters in the G152.x block and switch on. In this case, no "inclined plane" table is required.
G153	Switch off all active "inclined planes".
G153.x	Deselect the inclined plane of the respective group.
G154.x... G159.x	Activate one of up to six "inclined planes" of the bank <x> of an "inclined plane" table (x = 1, .. ,5).

G151.x(...)	"Inclined plane vertical to tool". The shifting and rotating parameters are implicitly determined from the position of the TCP and the tool orientation. This requires an active orientable axis transformation.
Coord(...)	Selection/deselection of an axis transformation

Tab. 12-3: Relevant NC Functions

Relevant CPL Functions

PMT(..)	Write and read access to placement tables.
ROT2EUL	Calculates the Euler angle from a maximum of three successive rotations along the axes x, y and/or z.
ABSROT2EUL	Calculates the Euler angle from a maximum of three rotations along the spatially-fixed axes x, y and/or z.
PT2EUL	Calculates Euler angles so that the three points are located in the xy-level of a rotated coordinate system.
VEC2EUL	Calculates Euler angles for a rotated coordinate system for which the vectors for the z- and x-direction were applied.
BCS2WCS	Transforms a point from BCS to WCS. Active placements are included in that point for calculation purposes.
WCS2BCS	Transforms a point from WCS to BCS. Active placements are withdrawn from the point for calculation purposes.

Tab. 12-4: Relevant CPL Functions

Relevant MP Functions

3080 00001	Search path for the file
9020 00010	Define the unit (mm or inch) the table values are stored internally.

Tab. 12-5: Relevant MP Functions



Any modifications to the machine parameter 9020 00010 also affect D-correction / zero point offset tables and the interpretation of the correction and offset values transmitted by the PLC.

Placement Tables

The offset values and rotation angles for the inclined plane can be preset via placement tables.

Each placement table comprises five placement banks (groups) with six placements each:

- G154.1 .. G159.1
- G154.2 .. G159.2
- G154.3 .. G159.3
- G154.4 .. G159.4
- G154.5 .. G159.5

Table structure: The structure of an "inclined plane" table consists of six lines for each of the five placement banks for the offset values D1= DX, D2= DY and D3= DZ and the Euler angles phi, theta and psi.

All six lines/placement banks have the same structure:

<G..><:><D1> <D2> <D3> <Phi> <Theta> <Psi>

With

- G154.x...G159.x:
G-function to activate the xth inclined plane (x = 1.. 5)

The control saves the tables as XML file with an exactly defined configuration.



Use the placement editor to easily edit these tables (see MTX Operating Instructions).

- Properties:**
- The inclined plane refers to the respective current workpiece coordinates
 - The "inclined plane" functions G152.x to G159.x are modally effective and deselect each other
 - The "inclined plane" functions **G151.x** to G159.x are modally effective and deselect each other.
 - Values for the "inclined plane" **can be provided in tables**. They contain information on the unit of the stored offset values (also see MP 9020 00010)

12.3.2 Handling Instruction: Placement (coordinate transformation)

Application: The placement "Inclined Plane" can shift and orientate the workpiece coordinate system anywhere in space. Up to five additively acting placements can be programmed.



The individual placements are additive. Workpiece position correction (BCR) is before the inclined plane in the calculation chain.

Offsets within a placement bank mutually overwrite each other.

Activating and deactivating workpiece position correction interrupts block look-ahead; therefore, it may not be programmed when milling cutter path correction is active.

Tool must no longer be active.

Coordinate transformations

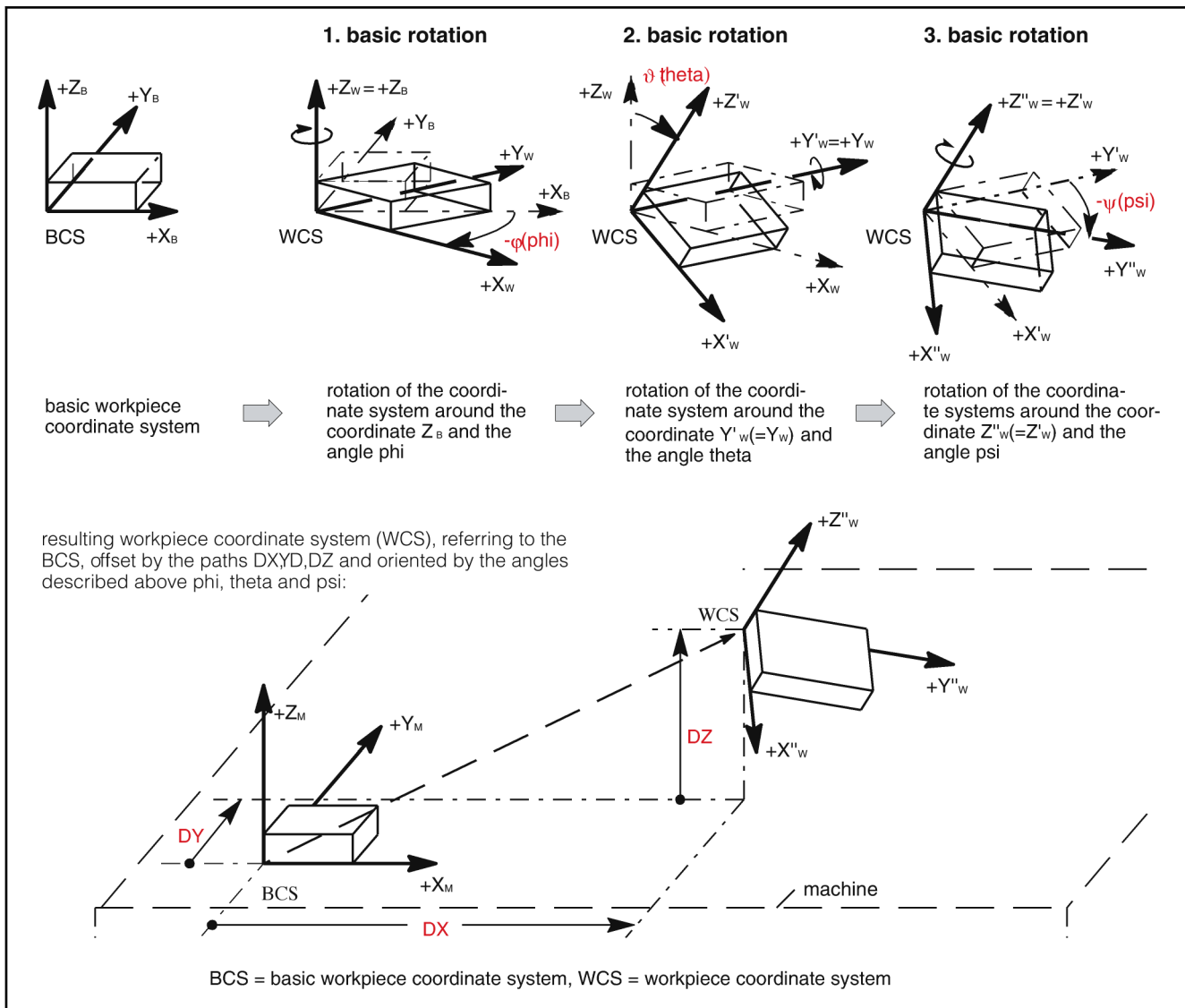


Fig. 12-4: Placement: Inclined plane

Programming can be alternatively executed via

1. Tables via **G154.<placement> .. G159.<Bank>**
 2. direct value specification via **G154.<placement> .. G152.<Bank>**
- with placement = 1,...5.

Explanation to 1:

To activate a Placement table via NC block, use the PmTSEL command (PMS).

Syntax: PmTSEL({<Path>/}<TableName>{.pmt})

If <Path> is not specified, the NC uses the search path (MP 3080 00001) to search for the <table name>.

IW Operation/Program: Create a placement file

- Select the "Program" mode using the OP key "Program" (OP4)



- Navigate to the desired directory or create a new one if necessary.



Fig. 12-5: Root directory

- Select the F-key "New" <F2>



- Select "Placements" <F8>
- The "Create new placement table" dialog appears.

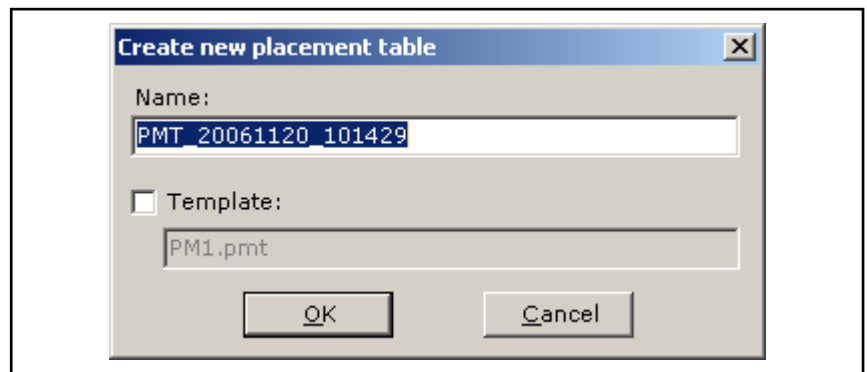


Fig. 12-6: Dialog "Create new placement table"

- Assign a name for the new file.
- The placement editor opens.

IW Operation / Program / Placement editor: Create/delete correction bank

Use the F-key "Add correction bank" <F2> to add a new placement bank/group or

use the F-key "Delete correction bank" <F3> to delete an old placement bank/group by selecting the desired bank.

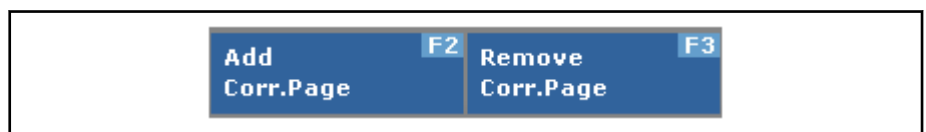


Fig. 12-7: Adding/Deleting correction bank

IW Operation / Program / Placement editor: Enter values

The desired placement can be entered/modified by selecting the desired block.

Placements - /PMT_20061120_101814.pmt							
Correction	X [mm]	Y [mm]	Z [mm]	Phi [°]	Theta [°]	Psi [°]	Comment
G154.1	0.000	0.000	0.000	0.000	0.000	0.000	
G155.1	0.000	0.000	0.000	0.000	0.000	0.000	
G156.1	0.000	0.000	0.000	0.000	0.000	0.000	
G157.1	0.000	0.000	0.000	0.000	0.000	0.000	
G158.1	0.000	0.000	0.000	0.000	0.000	0.000	
G159.1	0.000	0.000	0.000	0.000	0.000	0.000	

Fig. 12-8: Overview placement tables in the placement editor

Explanation to 2:

The placement values can directly be programmed via G152:

G152.<Bank>({<XW offset>},{<YW offset>},{<ZW offset>} {,<Angle1>} {,<Angle2>},{<Angle3>}}))

<Bank>: Number of the desired bank/block (1..5).

<XW offset>:

Offset value in the direction of the primary coordinate

<YW offset>:

Offset value in the direction of the secondary coordinate

<ZW offset>

Offset value in direction of the normal coordinate

<Angle1>:

Angle for turning by the z-coordinate.

Value range: 0 = <Angle1> < 360 degrees

<Angle2>:

Angle for turning by the new Y' coordinate.

Value range: 0 = <Angle2> < 180 degrees

<Angle3>:

Angle for turning by the new Z" coordinate.

Value range: 0 = <Angle3> < 360 degrees

Activate: Select the desired placement table using the NC function **PmTSEL({<Path>/}<Table name>{.pmt})**.

The table must exist and contain the correct offset values.

Activate with the G-code **G154.<Bank> .. G159.<Bank>** the desired placement of the placement group/bank.

– or –

The G-code **G152.<Bank>({<XW offset>},{<YW offset>},{<ZW offset>} {,<Angle1>},{<Angle2>},{<Angle3>}}))** can directly program and enable the desired placement of the bank (1...5).

Deactivate: If you program **G153**, all banks are switched off (switch off inclined plane completely).

– or –

If you program **G153.<Bank>**, all placements of a bank are switched off.

12.4 Program Coordinate Offset Trans, (TRS); ATrans, (ATR)

12.4.1 Description

Function

The "Program coordinate offset Trans" function shifts the active program coordinate system parallel to the coordinate axes of the current workpiece coordinate system, i.e. the program zero point is defined again.

The additive "Program coordinate offset ATrans" permits additional parallel and always additive offsets of the program coordinate system.

The program coordinate offset always acts additively on the active zero point offset and on the "Inclined plane".

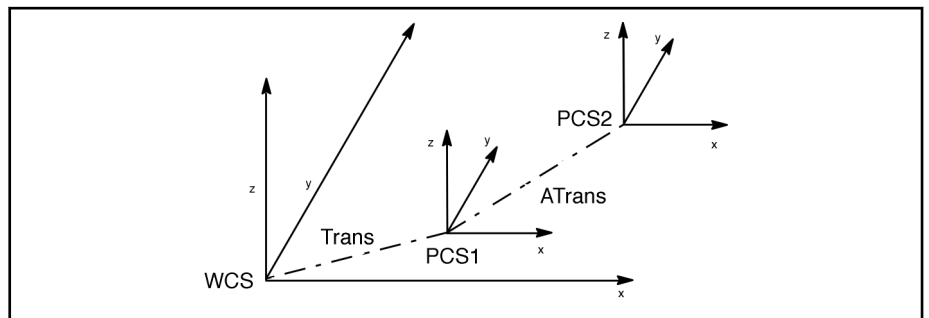


Fig. 12-9: Program coordinate offset

Restrictions

- Trans() switches off "Trans" as well as "ATrans" if active.
- It is not allowed to use "Trans" and "ATrans" in one traversing motion.

Relevant NC Functions

Trans(..)	Program coordinate offset ON
ATrans(..)	Additive program coordinate offset ON
Trans(), Trans(0)	Program coordinate offset OFF Short form TRS(..)
ATrans(), ATrans(0)	Additive program coordinate offset OFF Short form ATR(..)

Tab. 12-6: Relevant NC Functions

12.4.2 Handling Instruction: Program Coordinate Offset "Trans""ATrans"

Application: "Trans" offsets the active program coordinate system parallel to the coordinate axes of the current workpiece coordinate system.

"ATrans" permits additional parallel and always additive offsets of the program coordinate system after the command "Trans" (TRS).

Coordinate transformations

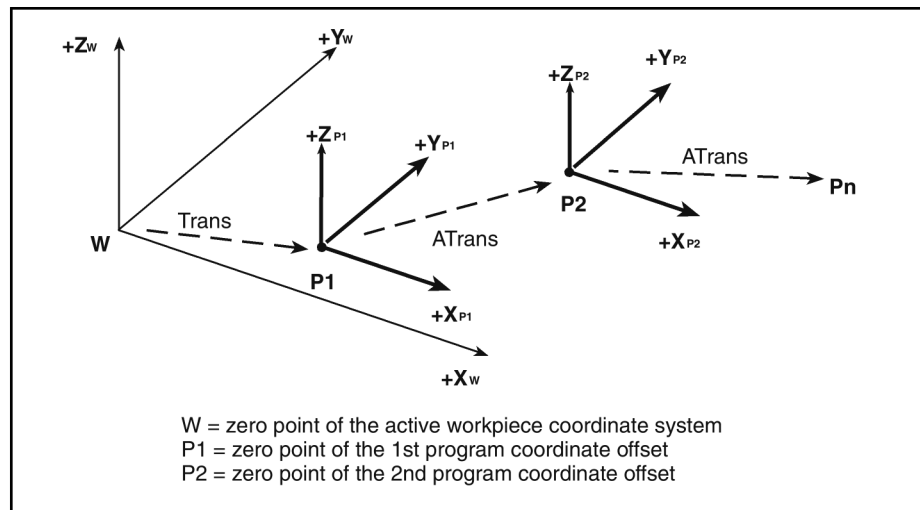


Fig. 12-10: Program coordinate offset

Tool must no longer be active.

Activation/deactivation of the program coordinate offset `Trans(..)` or `ATrans(..)` does not trigger a traveling movement.

Input tools that are based on the active program coordinate system (e.g. programmed contour offset, scaling, mirroring, rotation) are independent of a program coordinate offset and therefore need not be adapted.

Activate: Program `Trans(<Coordinates>)` or `TRS(<Coordinates>)`.

or

Program `ATrans(<Coordinates>)` or `ATR(<Coordinates>)`.

The zero point of the resulting program coordinate system is set to the programmed <coordinates> of the active program coordinate system.

Deactivate: Program `Trans()` or `Trans(0)`

or

Program `ATrans()` or `ATrans(0)`

13 Input helps

13.1 Mirroring, scaling, rotating, shifting

The input helps mentioned above are described in detail without interactions in the programming instructions.

Refer to G37-G39 and G60 for the syntax of the respective functions.

14 Tool corrections

14.1 Overview

Dimensions and position of a tool can be described by means of the following data:

- Tool reference points
- Tool coordinate system
- Geometrical dimensions (corrections) and tool type

14.2 Basics

14.2.1 Tool reference points

A tool has various reference points:

The tool reference point T (Tool Center Point TCP) is the origin of the coordinates of the Tool Coordinates System TCS. Usually, it is situated on the symmetry axis of the tool on the level of the tool action point .

To define the position of the tool reference point TCP, the tool dimensions "tool radius R" and the "tool length L" have to be known. They indicate the position of the TCP in respect of the position of the tool zero point E.

If the tool is in tool holder, tool zero point E and tool holder point N are identical.

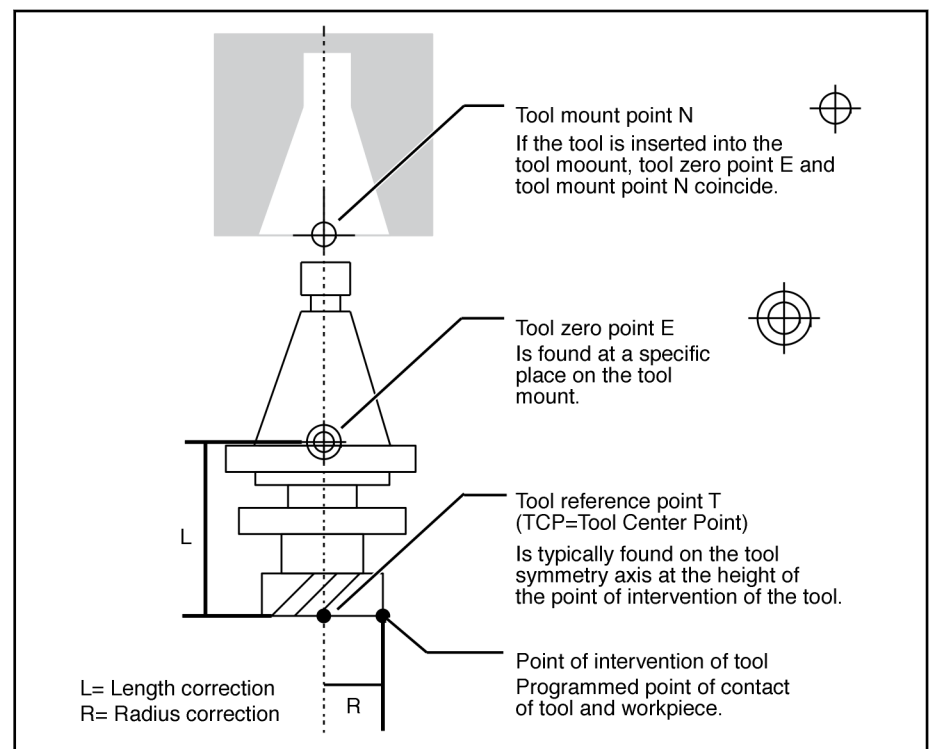


Fig. 14-1: Tool reference points

Example:

For 3-axis machines, the dimensions of the tools are based on a fixed tool setting point (tool holder point, tool zero point).

14.2.2 Tool coordinate system

The Tool Coordinate System TCS (= Tool Coordinate System) is unlike the other coordinate systems (as e.g. PCS, WCS, BCS, MCS, etc.) in that it is not defined when assuming reference position but is specified by means of the programmed movement.

Exception: position definition with active axis transformation, see [chapter 14.2.3 "Tool coordinate system with active axis transformation"](#) on page 190.

If the movement is programmed in the part program, the path of the tool in respect of the workpiece is specified in the program coordinate system PCS.

These path coordinates are the coordinates of the tool engaging point. Under consideration of the lengths and radius corrections of the active tool, the control will use this data to compute the actual path of the TCP which is the zero point of the TCS.

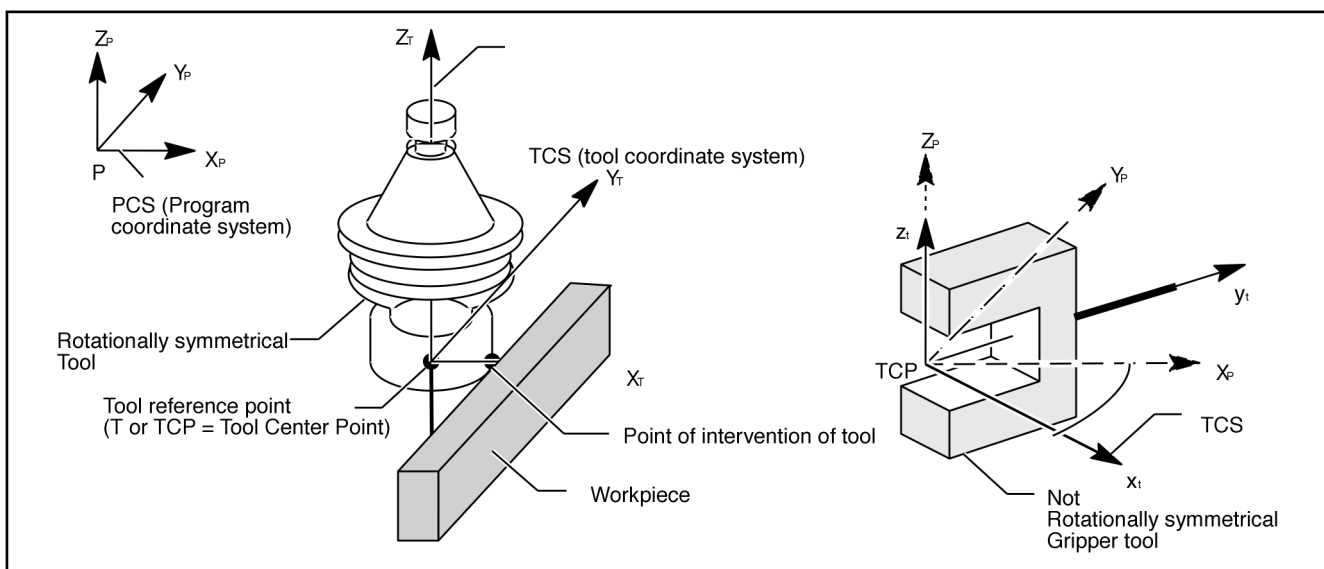


Fig. 14-2: Tool coordinate system

If, in case of a rotation-symmetric tool, the TCS lies parallel to the coordinates of the PCS, the position of the TCS in relation to a tool coordinate system is irrelevant. In this case, **no** space coordinate programming is active.

14.2.3 Tool coordinate system with active axis transformation

With **space coordinate programming** while **axis transformation** is active (TCP programming), the relative position of the tool engaging point is programmed in relation to the program coordinate system PCS.

The TCP position is specified on the basis of the tool corrections L and R in respect of the tool zero point E, the current tool orientation and the programmed coordinates.

With non-rotation-symmetrical tools, the position of the TCS can be defined indirectly via an NC function (Rotax(..), O(..)).

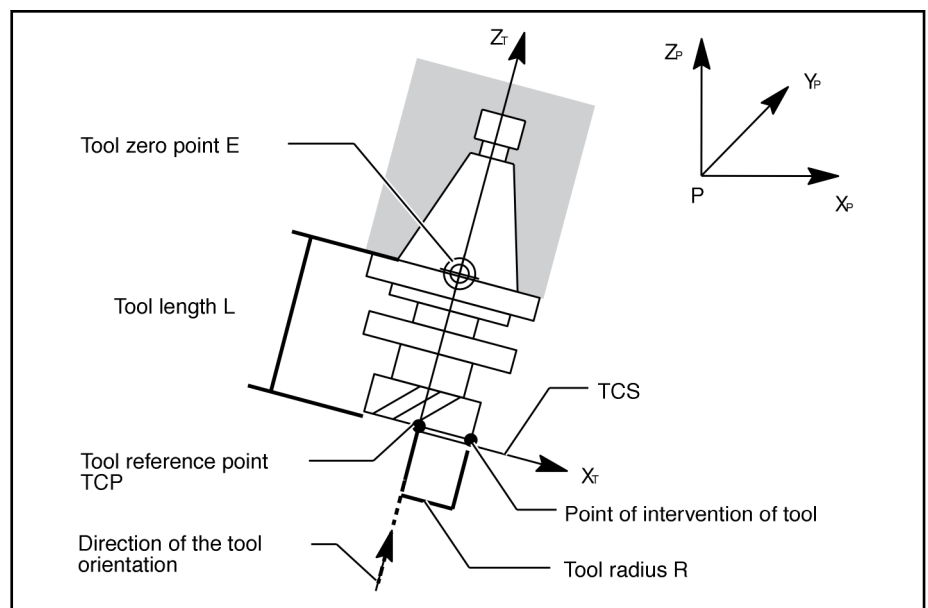


Fig. 14-3: Tool coordinate system with active axis transformation

While **axis transformation is active**, frequently machining with spatial changes of **tool orientation** is executed. Then, tool correction takes place within the axis transformation along the axes of the tool coordinate system TCS.

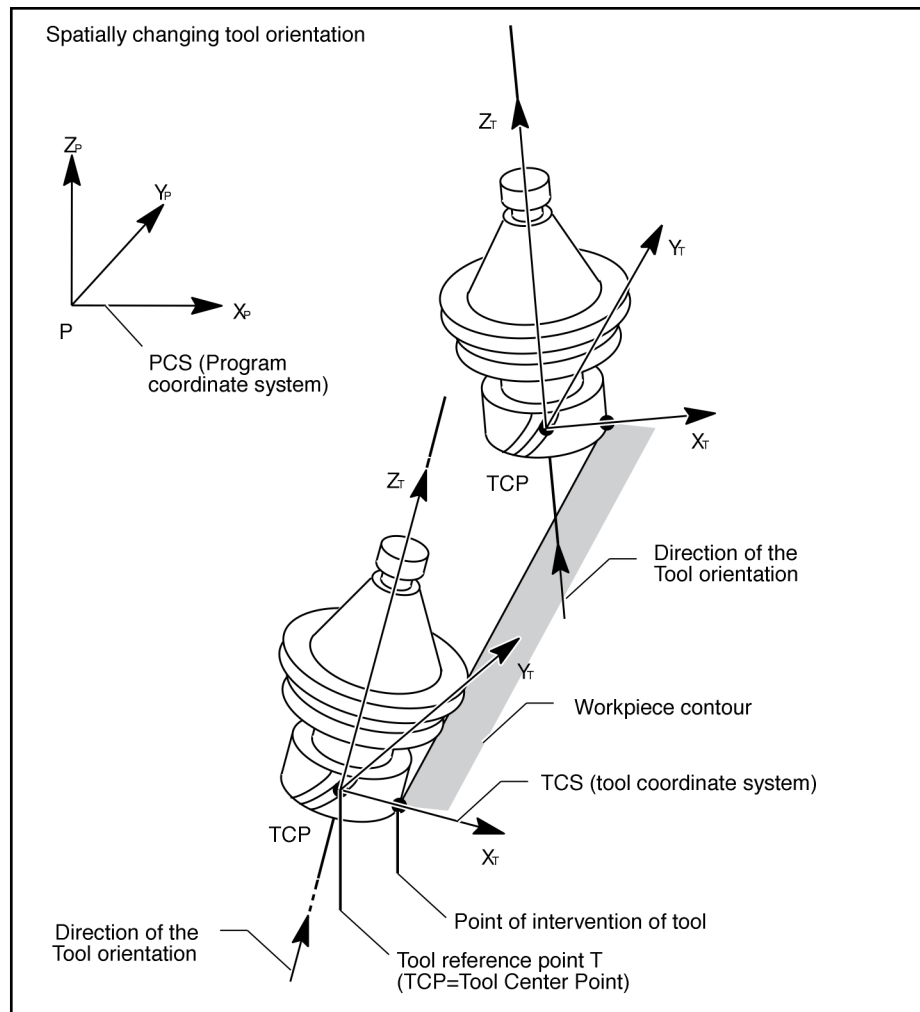


Fig. 14-4: Spatial changes of tool orientation

Reference position: With 5- and 6-axis transformation, the tool coordinate system TCS takes a **reference position** first, which is arranged axis parallel in respect of the BCS and which allows for 24 different orientations (see the following example). In its reference position, the tool coordinate system has the designation **TCS₁** (for all orientations). For the tool, the TCS₁ can be defined with the axis transformation parameter MP 1030 00150.

Example:

Three of 24 potential orientations of the TCS in respect of the BCS in the reference position of the axis transformation.

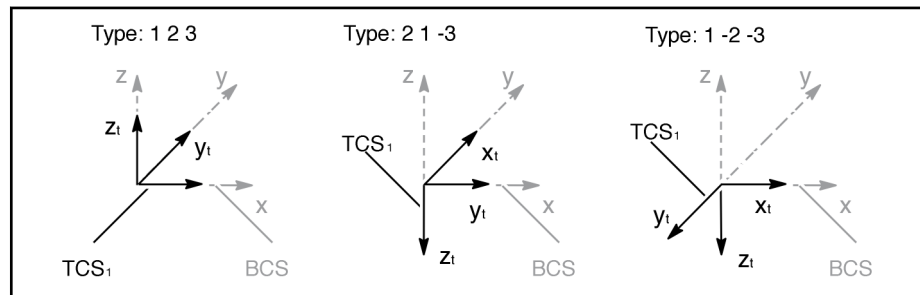


Fig. 14-5: Example

Relevant machine parameters (MP)

1030 00150	Reference position of tool coordinate system

Tab. 14-1: Relevant machine parameter

Position of the tool coordinate system:

The position of the TCP and the connected TCS can be affected during axis transformation. By special corrections and definitions, the TCS_1 is offset and rotated if applicable.

The position of the TCS_1 can be modified as follows:

- By **shifting** the TCS_1 into a tool coordinate system "TCS_c" with an explicit tool correction (e.g. CPL command "TC")
- By **shifting and optionally rotating** the TCS_1 into a tool coordinate system "TCS_c" with an explicit tool correction (e.g. CPL command "DCT")
- By **specifying spatial coordinates** for the tool coordinate system TCS_p with the "TCS definition in program coordinates" function (TCSDEF command).

Explicit tool correction:**Prerequisite:**

Applicable only with an active 5- or 6-axis transformation.

Function:

By means of an explicitly specified tool correction, a TCS_c is created which can be offset and rotated in relation to TCS_0 or TCS_1 .

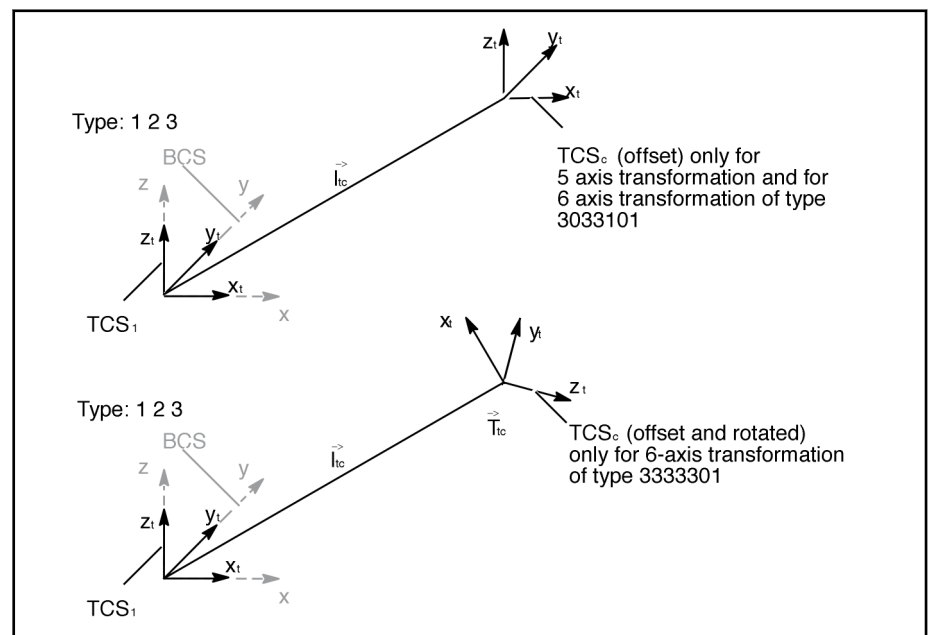


Fig. 14-6: Explicit tool correction

Restrictions:

- Shifting **and** rotating the tool coordinate system is only possible for the 6-axis transformation of type 333301.
- Shifting the tool coordinate system is only possible for 5-axis and 6-axis transformation of type 3033101.

TCS definition in program coordinates:**Prerequisite:**

Applicable only with an active 6-axis transformation.

Function:

The "TCS definition in program coordinates" function generates a tool coordinates system TCS_p which overrides the current TCS_c or TCS_1 . The coordinate values specified in the TCSDEF command for the TCS_p are converted NC-internally in $\vec{T}_t$ and $\vec{T}_t^p$ and stored in the tool correction memory. The TCS_c or - if no explicit tool correction is active - the TCS_1 is activated again by switching off with TCSUNDEF.

The figure below shows the active TCS_p and the other inactive tool coordinate systems:

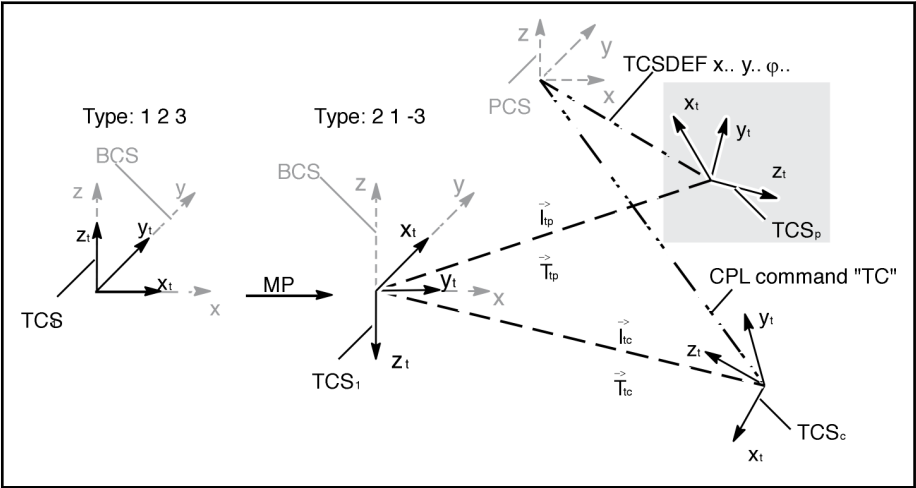


Fig. 14-7: Active TCS and other inactive tool coordinate systems

Restrictions:

The TCSDEF command can only be used if an axis transformation with orientation identification 3 (tensor orientation) is active.

Relevant NC functions and addresses

TCSDEF[<LinearCoordinates >] [<OrientationCoordinates >]	
	Define the position of the tool coordinate system TCS_p Linear coordinates: Coordinates with reference to the current PCS. Orientation coordinates: Coordinates with reference to the current PCS or all alternative syntax options of tensor orientation (see chapter 4 "Drives (axes, spindles)" on page 15).
TCSUNDEF	Reset to the previously active tool coordinate system (e.g.: TCS ₀ or TCS ₁) An automatic reset is triggered with each coordinate switching "Coord(..)".

Tab. 14-2: Relevant NC functions and addresses

Example:

N10 Coord(1)	Axis transformation on
N20 x300 y100 phi315	
	Motion of the TCS ₁ to the starting position
N30 TCSDEF x100 y150 phi0	

	Tool correction to
N40 phi45	Rotation of the TCS _p
N45 ...	Position and orientation data refer to TCS _p
...	
N50 TCSUNDEF	Programming again refers to TCS ₁

Tab. 14-3: Example

In the figure below, the TCS₁ fixedly connected with the tool is located on the position $x=300$, $y=100$ and is provided with the following orientation: $\varphi = 315^\circ$. By programming **TCSDEF x100 y150 phi0**, the TCS_p is defined:

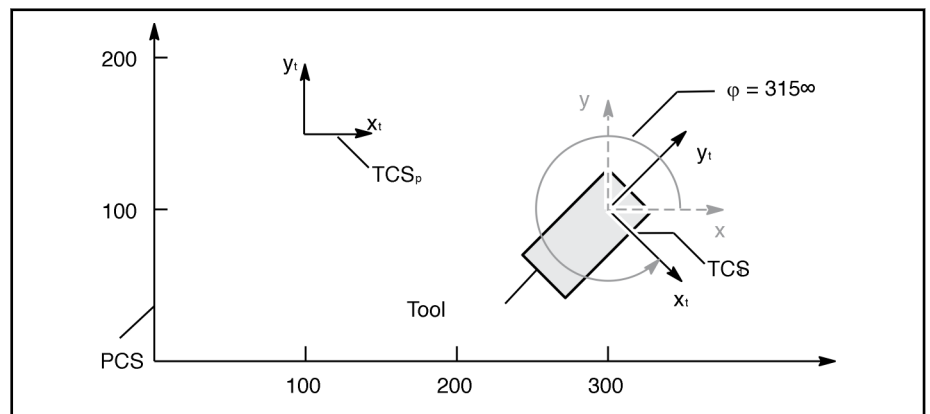


Fig. 14-8: Example

The position and orientation data following after TCSDEF in the NC program refer to the TCS_p motion (NC program section: N45 phi45):

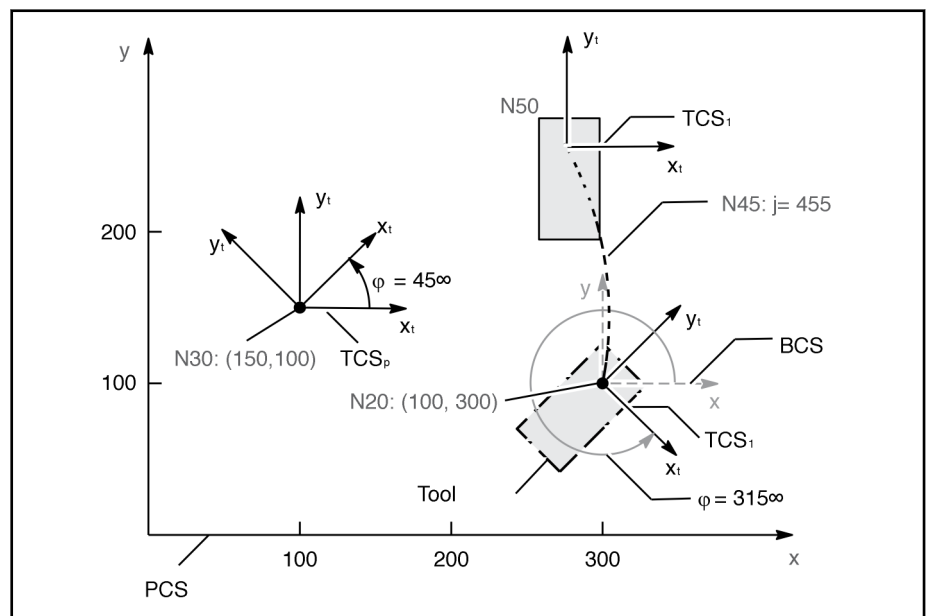


Fig. 14-9: Example

Exemplary use case: Sheet machining system

Two coupled robots (also see "Channel-comprehensive coordinates coupling") grip a sheet at two positions and move it in space.

The tool coordinate systems TCS_{1a} and TCS_{1b} of the two gripper tools are corrected to a common TCS_p of the sheet. If the sheet is to be moved in space, only position and orientation of the common TCS_p have still to be programmed.

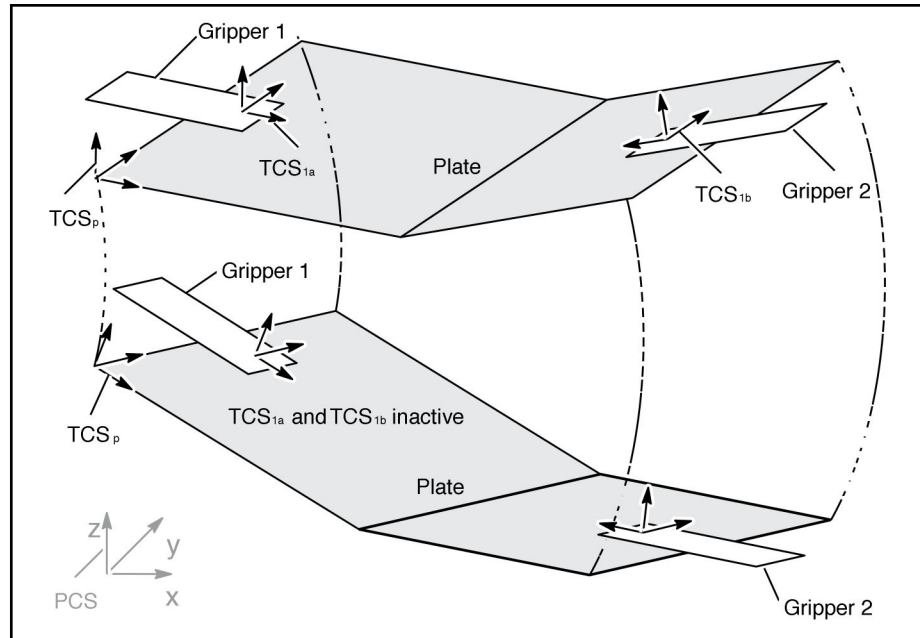


Fig. 14-10: Example - Sheet machining system

14.2.4 Applying

Explicit tool correction

1. Make all settings for your application.
Refer to [chapter 4 "Drives \(axes, spindles\)"](#) on page 15.
2. Note the selection of the type of transformation. Shifting and rotating the TCS is only possible for certain types (see restrictions).

TCS- definition in program coordinates

1. Make all settings for your application.
Refer to [chapter 4 "Drives \(axes, spindles\)"](#) on page 15.
2. Note the selection of the type of transformation. The TCS definition can only be used for types with tensor orientation (see restrictions).

14.2.5 Activating

Explicit tool correction

Make all settings for your application.

Refer to [chapter 4 "Drives \(axes, spindles\)"](#) on page 15.

TCS- definition in program coordinates

1. Make all settings for your application.
Refer to [chapter 4 "Drives \(axes, spindles\)"](#) on page 15.
2. The function is activated with TCSDEF...

14.2.6 Deactivating

Explicit tool correction

1. Make all settings for your application.

Refer to [chapter 4 "Drives \(axes, spindles\)"](#) on page 15.

2. The function is reset with each coordinate switching Coord(n).

TCS- definition in program coordinates

1. Make all settings for your application.

Refer to [chapter 4 "Drives \(axes, spindles\)"](#) on page 15.

2. *The function is disabled via:*

- TCSUNDEF or
- Coordinate switching Coord (n)

14.3 Specifying tool correction values

14.3.1 Description

Overview

By using the current tool data (e.g. lengths, radius, edge position), the NC can automatically compute the actually required

- Path characteristics
- Path depths (also for bores)

for a programmed workpiece contour depending on the tool used.

Thus, the dimensions in technical drawings can be used directly as basis to create part programs. With regard to the dimensions specified, your part program becomes "independent of tools".

To use tool corrections, the relevant tool data has to be known to the control.

In the MTX, the tool correction data can be provided to the NC kernel as follows (two options):

- Specifying correction values via **D-correction tables**. These tables are XML-based. The D-correction is used to select the current correction block from the active D-correction table.
- Specifying the correction values via the total of **16 correction registers** per machining channel of the tool correction interface. In the correction registers, the correction data of 16 correction blocks can be stored. The active correction block is selected via the ED-correction (external tool correction). The ED-correction is also referred to as edge correction (ED: abbr. for edge).



Incorrect contour due to incorrectly specified tool dimensions!

As tool wear occurs during machining, tool lengths and radii change depending on the tool type. This effect also occurs when tools are sharpened.

When using geometry corrections, ensure that all tool data always corresponds to the current measurements of the tools used!

Measure the tools used in certain intervals again (depending e.g. on edge life, tool action time) and update the provided tool data to the current status.

For more details on the individual geometry corrections, see the following pages:

- D-correction from [chapter 14.3.2 "D-correction"](#) on page 198
- External tool correction (ED-correction) from [chapter 14.3.6 "External tool correction \(ED\)"](#) on page 202

D-correction and external tool correction (ED) can be active simultaneously and overlapping each other additively.

The provided tool corrections are processed by the following functions:

- **Path correction G41/G42 (2.5D-tool radius correction)**

For a description, see from [chapter "2.5D-tool radius correction \(G41/G42\)" on page 213](#).

- **Path correction G141/G142 (3D-tool radius correction)**

For a description, see from [chapter "3D-tool radius correction" on page 236](#).

14.3.2 D-correction Function

Any number of D-correction tables can be stored in the file system of the MTX. Every table can consist of up to 99 tool correction blocks with each correction block consisting of three tool lengths "L1", "L2", "L3", the tool radius "Rad" as well as the tool edge "Ori". The D-correction tables are XML-based and are thus open any time for subsequent extensions. The D-correction tables of the MTX have the file extension ".dct".

In the MTX user interface, the individual table elements can be comfortably edited with a table editor.

Alternatively, the CPL command "DCT" offers the option to write or read individual table elements directly from the part program.

By means of the "DcTsel (DCS)" NC command, the desired D-correction table is activated. It is sufficient to specify the name (without the ".dct" file extension). If the desired table not be in the configured search path of the control, the corresponding path has to be specified as well. If the name given to a D-correction table corresponds to the MTX naming convention ("DC<Index>.dct") and is located in the search path, only the index has to be specified, e.g. DcTsel(4).

After selection of a table, the respective correction set can be selected via "D" address: D<CorrectionBlockNo.>

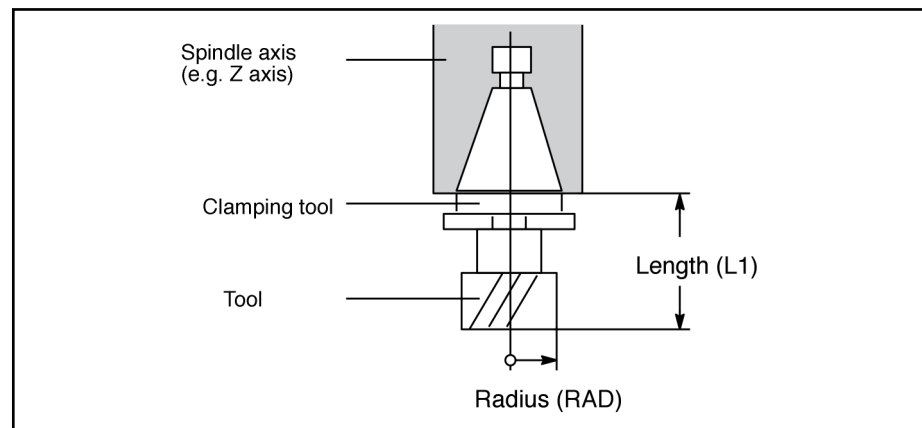


Fig. 14-11: D-correction

Restrictions

If, while tool correction is active (G41/G42, G44, G141/G142), the tool data of the currently selected D-correction is changed externally (e.g. via the table editor), the changed correction is only considered for the first time for the next block to be prepared. At that time, the previous correction cannot only be

used for the currently processed block, but also for multiple NC blocks prepared with the previously valid correction.

The maximum number of NC blocks to be prepared is channel-specifically set in the machine parameter 7060 00110.

Relevant NC functions and addresses

DcTSel(<{Path/}TableName> <TableIndex>)	
	Activate tool correction table via NC block
DCS	Short form for DcTSel
D1... D99	Select D-correction (1 .. 99)
D0	Deselect the D-correction

Tab. 14-4: Relevant NC functions and addresses

Relevant CPL functions

DCT(..)	Write (absolute/additive) and read access to external tool correction values stored within the NC
MCODS(13,..)	Name of the current tool correction table
MCODS(54,...)	Active D-correction values

Tab. 14-5: Relevant CPL functions

Relevant machine parameters (MP)

3080 00001	Search path for subroutines For a table in the search path, the path does not have to be explicitly programmed in the table selection
7060 00110	Total number of blocks for block look-ahead
9020 00010	Indicates in which unit the values of the geometry correction tables are stored within the NC

Tab. 14-6: Relevant machine parameters (MP)

14.3.3 Handling instruction: Activating the D-correction

D-corrections are equally suitable for drilling, milling, lathing and angle-head tools. With a total of three offset values (L1, L2 and L3), constant three-dimensional tool offsets for a tool as well as parallel length corrections for up to three different tools can be performed.

IW Operation/program: Creating D-correction table

		Instruction chapter 14.3.2 "D-correction" on page 198
Instruction:		D-correction table

IW Operation/NC programming: Loading D-correction table

- **DcTSel({<Path>}<FileName>)** or **DCS({<Path>}<FileName>)**
 - **<Path>** Optional path specification to the directory the **file name** is stored in. If there is no specification, it is searched in the /database path.

If file name cannot be found, the control uses the search path for subroutines to search for <FileName> in other directories.

- <File name> File name of the D-correction table, incl. file extension. Tables with standard names (DC <Number>.dct) can directly be activated with this number, e.g. DcTSel(7) activates the table DC7.dct.

	Documentation	Example
Example:		Program example
Documentation:	MTX Programming Manual	Command "DCS"

IW Operation/NC programming: Preselecting a D-correction set

- Use the command **D<NoToolBlock>** to preselect a tool correction block from an active D-correction table.
 - **<NoToolBlock>** Number of the correction data block (1...99).



- A D-correction can be programmed in the same block as other path conditions, traversing motions and auxiliary functions.
- The tool correction is only settled after the corresponding NC function was activated:
G47, G41, G42, G141, G142.

	Documentation	Example
Example:		Program example
Documentation:	MTX Programming Manual	Command "DCS"

14.3.4 Handling instruction: Deactivating the D-correction

Deselection of a D-correction set.

IW Operation/NC programming: Deselecting a D-correction set

Using the command **D0**, deselect a tool correction block without preselection a new one.

	Documentation	Example
Example:		
Documentation:	MTX Programming Manual	Deselect a D-correction set

14.3.5 Handling instruction: Creating a D-correction table

D-corrections are equally suitable for drilling, milling, lathing and angle-head tools. With a total of three offset values (L1, L2 and L3), constant three-dimensional tool offsets for a tool as well as parallel length corrections for up to three different tools can be performed.

IW Operation/program: Create a D-correction table

- Select the "Program" mode using the OP key "Program" (OP4)



- Navigate to the desired directory or create a new one if necessary.



Fig. 14-12: Root directory

- Select the F-key "New" <F2>



- Select the "D-corrections" <F6> from the F-key-pad.
- The "Create new D-correction table" dialog is displayed

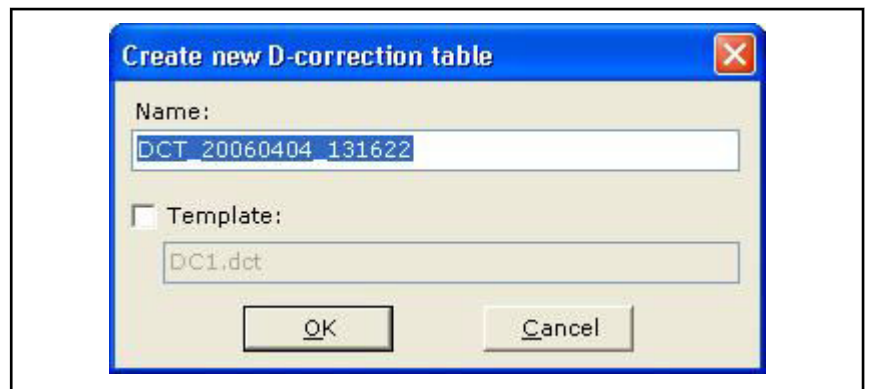


Fig. 14-13: "Create new D-correction table" dialog

- Assign a name for the new file
- The D-correction editor opens

	Documentation	
Documentation:	IndraWorks HMI	D-correction table

IW Operation/program/D-correction editor: Create/delete correction bank

- Use the F-key "Add correction bank" <F2> to add a new block (D-correction bank/group)

- or -

- use the F-key "Delete correction bank" <F3> to delete an old block (D-correction bank/group) by selecting the desired block.

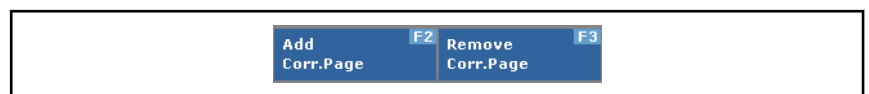


Fig. 14-14: Adding/Deleting correction bank

	Documentation	
Documentation:	IndraWorks HMI	D-correction table

IW Operation/program/D-correction editor: Enter values

- The desired D-correction can be entered/modified by selecting the desired block.

D-corrections - /database/DC1.dct						
Correction	L1 [mm]	L2 [mm]	L3 [mm]	R [mm]	Edge position	Comment
D1	0.0000	0.0000	0.0000	0.0000	0	Korrektur Bohrung
D2	0.0000	0.0000	0.0000	0.0000	0	

Fig. 14-15: Overview on D-correction tables in the D-correction editor



D-corrections overlap additively.

A correction table can contain a maximum of 99 data blocks.

Each data block contains the following correction values:

- Three tool lengths L1, L2, L3,
- Tool edge radius RAD,
- Tool edge position ORI.

A D-correction can be programmed in the same block as other path conditions, traversing motions and auxiliary functions.

The tool correction is only settled after the corresponding NC function was activated: G47, G41, G42, G141, G142.

	Documentation	
Documentation:	IndraWorks HMI	D-correction table

14.3.6 External tool correction (ED)

Function

In addition to the tool correction tables of the D-correction, the MTX also provides the special channel-specific correction registers of the tool correction interface on which can directly be written from the PLC. To this end, the PLC program component "MT_Tcorr" is used.

Usually, the utilization of these correction registers is to be seen in connection with the MTX tool management. After a new tool selection, it is searched in the tool management and its corrections are transferred directly or with special adjustments (e.g. offset of a tool adapter) to the correction register of the corresponding machining channel.

Alternatively, the contents of the correction registers can be read and written from the CPL. This is done analogously to the D-correction with the "DCT" command; in this case, the value "0" is transmitted as a parameter instead of the table name: DCT(.....,0 {...}).

16 **correction blocks** are available for each channel.

Each correction block consists of:

-
- $L_1, L_2,$ Length correction or offset values
 L_3 :
 - RAD: Tool radius
 - ORI: Edge position
(see [Explanations of the edge position \(ORI\)](#), page 226)

The option to provide several correction block for one channel takes the fact into account that one single tool can have several machining edges, each with its own specific tool corrections.

Use the NC command "ED" to select the desired correction block: E<CorrectionBlockNo.>.

Restrictions

- If, while tool correction is active (G41/G42, G44, G141/G142), the tool data of the currently selected ED-correction is changed externally (e.g. by the PLC), the changed correction is only considered for the first time for the next block to be prepared. At that time, the previous correction cannot only be used for the currently processed block, but also for multiple NC blocks prepared with the previously valid correction.

The maximum number of NC blocks to be prepared is channel-specifically set in the machine parameter 7060 00110.

- To avoid this effect, program a CPL command.
The "WAIT" command has to be programmed directly after the block which causes the PLC to transmit the new correction values. Thus, the block preparation of the NC is stopped until all program blocks before "WAIT" are processed.
- After control startup, the correction values stored internally in the correction registers are nulled. The values can only be changed by the PLC or CPL.
- Upon reset, the correction values stored in the correction registers remain.

Relevant NC functions

ED1...ED16	Activate the corresponding correction block of the external tool correction
ED0	Disable the external tool correction

Tab. 14-7: Relevant NC functions

Relevant CPL functions

DCT(...,,0 {...})	Write (absolute/additive) and read access to the correction values of the external tool correction stored within the NC
MCODS(51,..)	Active external correction values
WAIT	Stop block processing

Tab. 14-8: Relevant CPL functions

Relevant IF signals

iChActFunc[1..24]	The selected correction blocks can be displayed in the corresponding channel interface under the active NC functions. It is configured via the machine parameters 3020 00001 and 3020 00002.

Tab. 14-9: Relevant IF signals

Relevant machine parameters (MP)

3020 00001	iChActFunc[1..24] /NC function assignment: Configuring function states of the modal NC functions at the bit interface.
3020 00002	iChActFunc[1..24] /NC function assignment (supplement): see above
7060 00110	Total number of blocks for block look-ahead.
9020 00010	Indicates in which unit (mm or Inch) the control is to interpret the correction values transmitted by the PLC.

Tab. 14-10: Relevant machine parameters

14.3.7 Handling instruction: Activating the ED-correction

The external tool correction (ED-correction) calls correction values for up to 16 tool edges. The PLC or the part program with the CPL command DCT can be used to write on the correction values via the program function block "MT_TCorr".

IW Operation/program: Create an ED-correction table

Use the command **ED0** to deselect a tool correction block without preselecting a new one.

		Instruction chapter 14.3.6 "External tool correction (ED) " on page 202
Instruction:		ED-correction table

IW Operation/NC programming: Preselect an ED-correction set

Use the command **ED<NoToolBlock>** to preselect a tool correction block from a tool correction interface.

<NoToolBlock> Number of the correction data block (1...16).



Active D-correction values are applied additively to ED-corrections.

An ED-correction can be programmed in the same block as other path conditions, traversing motions or auxiliary functions.

The tool correction is only settled after the corresponding NC function was activated:

G47, G41, G42, G141, G142.

	Documentation	Example
Example:		Program example
Documentation:	MTX Programming Manual	Commands "ED"

14.3.8 Handling instruction: Deactivating the ED-correction

Deselection of a D-correction set.

IW Operation/NC programming: Deselect an ED-correction set

Use the command **ED0** to deselect a tool correction block without preselecting a new one.

	Documentation	
Documentation:	MTX Programming Manual	Deactivate an ED-Correction

14.3.9 Handling instruction: Writing the ED-tool correction

Direct writing of an ED-correction (external) via PLC function block.

IW Engineering/IndraLogic: Writing the ED-correction

Via the PLC function block **MT_TCorr**.



Shifting the tool coordinate system only for 5-axis and 6-axis transformations of type 3033101.

Figure	Documentation	Instruction
Figure:		Program example
Instruction:		Editing PLC Signals
Documentation:	MTX PLC Interface	Integrating function block

14.3.10 Tool length correction

Function

The tool length is defined by the distance from the integrated tool holder point at the tool up to the tool tip.

The length correction of the tool consists of up to three components. These

are L1, L2, L3. Together, they form the tool length correction vector $\vec{L}_t$. It points from the Tool Center Point (TCP) to the tool holder point, thus causing an offset of the original zero point of the TCS (on the workpiece zero point) to the TCP once the tool is mounted (refer to the following figure).

By defining the tool length correction vector $\vec{L}_t$ in the Cartesian space (i.e. coordinates which are rectangular to each other), tools can be defined which are neither rotation symmetric nor located parallel to the longitudinal axis of the tool holder.

Tool corrections

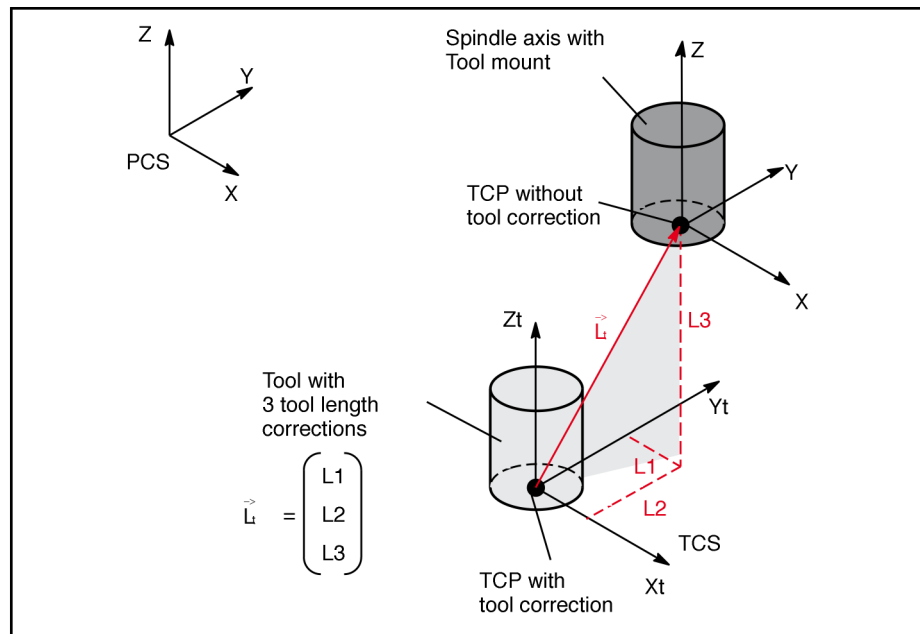


Fig. 14-16: Tool length correction



Depending on the measuring method, the length correction can refer to the "spindle nose" or a "zero tool"!

Example:

Rotation tool (cutter) with L3 tool length correction

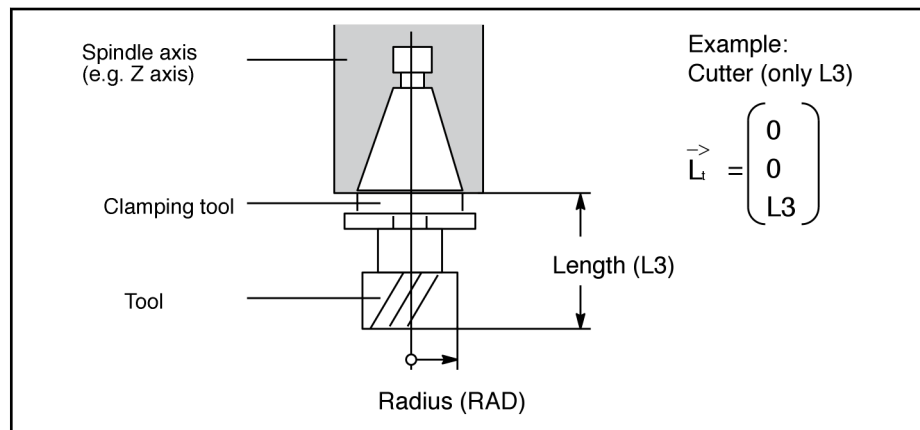


Fig. 14-17: Example

The tool lengths provided by the D-correction and the external tool correction (ED-correction) are calculated by the tool length correction G47.

Factors influencing tool lengths correction:

- The coordinate assignment of the three tool lengths as well as the desired point of action (TCS or WCS) is determined in the machine parameters, but can also be changed from the part program or via manual data input using the "correction switching" function G47(..) or G78(..).
- Using the static tool orientation StatToolOri / STO, the correction vector (L1, L2, L3) can be statically oriented using the three Euler angles phi, theta, and psi.

With this functionality, tools with different sizes can be used for machining at tool machines. The tool - depending on the machine - can be engaged in different directions or freely oriented in space with regard to the machine kinematics.

The "correction switching" function assigns the various length corrections of the individual functions for geometry correction either:

- to the individual directions of the current workpiece coordinate system (WCS) or
- to the directions of the tool coordinate system (TCS) or
- to the corresponding workpiece coordinates of the currently selected working plane (ActPlane)

An assignment with respect to the directions of the workpiece coordinate system (WCS) is possible whenever the tool is aligned perpendicularly to the current working plane and its orientation with regard to the working plane remains constant during machining.

Examples: Drilling and turning, milling of flat surfaces.

Assigning coordinates with reference to the directions of the tool coordinate system (TCS) is necessary if the orientation of the tool changes during machining, e.g. milling of free-form surfaces. This allows a tool length correction in case of a variable tool orientation. An active axis transformation (e.g. 5-axis or 6-axis transformation) is necessary for this correction. The correction values are settled in the axis transformation.

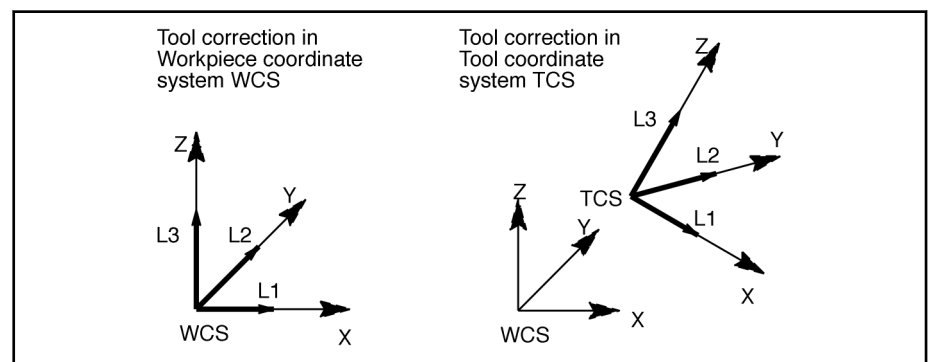


Fig. 14-18: Tool correction in the workpiece coordinate system WCS and TCS

Restrictions

When resetting the correction assignment to default, all coordinates which are not part of the default configuration of the channel are generally not considered. This is also the case when a correction assignment was configured for such an axis in the machine parameters.

Relevant NC functions

G47	Activate tool length correction
G48	Disable tool length correction
StatToolOri(j,J,y)	Static tool orientation
STO(j,J,y)	Short form: Static tool orientation
G78(<Crđ1>, <Crđ2>, <Crđ3>)	
	Activate correction switching
G79	Reset correction assignment to default.

G47((<Crd1>, <Crd2>, <Crd3>)	
	≡ G47 G78(<Crd1>,<Crd2>,<Crd3>)
G47()	≡ G47 G79

Tab. 14-11: Relevant NC functions

Relevant CPL functions

SD(10,1,1)	Latest programmed L1 coordinate
SD(10,1,2)	Latest programmed L2 coordinate
SD(10,1,3)	Latest programmed L3 coordinate
SD(10,2,1)	Active L1 coordinate
SD(10,2,2)	Active L2 coordinate
SD(10,2,3)	Active L3 coordinate

Tab. 14-12: Relevant CPL functions

Relevant machine parameters (MP)

7050 01300	tool correction mode: Current Plane, WCS, TCS
7050 01310	Tool correction assignment correction - coordinates

Tab. 14-13: Relevant machine parameters (MP)

14.3.11 Handling instruction: Activating the tool length correction (G47/G48)

- Applying:**
- Specify channel-specifically which coordinates and which point of action the length correction values L1 - L3 are supposed to affect via MP 7050 01300 and MP 7050 01310.
The setting made is applied as default setting in the respective channel and can be modified via G78 or G47(...).
G79 or G47() resets the active correction assignment to default.

- Trigger system reset at the control to apply the machine parameters.

- Activating:**
- If necessary, activate the D-correction table:
DcTsel(<TableName>) (Short form DCS(<TableName>)).
 - Provide the correction values of D-correction and/or the external correction (D1-D99, ED1 - ED16)
 - If necessary, activate static tool orientation:
StatToolOri (STO).
 - If necessary, configure the correction assignment again; G78, G79.
 - Activate tool length correction with G47.

- Deactivating:**
- Prerequisite:**
Drilling axis is no longer engaged.
- Disable the tool length corrections with G48.
 - If necessary, deselect the preselected correction values with D0 and/or ED0.
 - If necessary, set the correction assignment to default again with G79.

By defining the tool length correction vector $\vec{L}_t$ in the Cartesian space (i.e. coordinates which are rectangular to each other), tools can be defined which are neither rotation symmetric nor located parallel to the longitudinal axis of the tool holder.

IW Engineering/configuration: Editing tool lengths correction parameters

		Instruction chapter 14.3.10 "Tool length correction" on page 205
Instruction:		Tool length correction

Providing ED-correction

If necessary, the ED-correction has to be configured via the tool correction interface.

		Instruction chapter 14.3.7 "Handling instruction: Activating the ED-correction" on page 204
Instruction:		ED-correction

Applying D-correction

If necessary, create and configure a D-correction table with the desired correction values.

		Instruction chapter 14.3.3 "Handling instruction: Activating the D-correction" on page 199
Instruction:		D-correction

IW Operation/NC programming: Activate correction

G47 Tool length correction ON.

Alternatively, also program the following commands:

- **G47(ActPlane)**
Assign the length correction values L1, L2 and L3 to the main, secondary and normal coordinates of the active plane and switch on the tool length correction afterwards. Remains modally effective and is automatically reconfigured at every following plane switching.
- **G47({{-}<Li1 coord>})**
with <Li coord> Name of the WCS coordinate that is to be assigned the correction Li (with i = 1, 2, 3) in the active tool coordinate system. If the correction Li (with i = 1, 2, 3) is to affect coordinates in the tool coordinate system (TCS), the coordinate identifiers XTR, YTR and ZTR have to be used and an appropriate axis transformation have to be active.
Optional negative sign:
Correction is settled in negative direction. Li-correction values - to which no coordinate is assigned - are not taken into account.
- **G47()**

Assign the length correction values L1, L2 and L3 according to the settings in the machine parameters 7050 01300 and 7050 01310 and switch on the tool length correction afterwards.



Depending on the measuring method, the length correction can refer to the "spindle nose" or a "zero tool"!

Coordinates not belonging to the default channel setting are never considered in the "**G47()**" syntax.

G47.. and **G48** act modally and deselect each other.

Whether the coordinate addresses programmed under **G47** refer to workpiece or tool coordinates depends on the coordinate identifiers used. Mixed programming such as "**G47(X,,ZTR)**" is not permitted.

G47 can be programmed with other path conditions, traversing information or auxiliary functions.

Example:

Example program:

G47(X,,-Z)

The L₁ correction is assigned to the x-axis of the workpiece coordinate system (WCS) and calculated in the positive direction. The L₂-correction is not settled, since it is not assigned to any coordinate. The L₃ correction is assigned to the z-axis of the workpiece coordinate system (WCS) and calculated in the negative direction.

The tool length correction is subsequently switched on.

Example:

Example program:

G47(XTR,YTR,ZTR)

The L₁ correction is assigned to the x-coordinate of the workpiece coordinate system (TCS), the L₂ correction is assigned to the y-coordinate of the TCS and the L₃ compensation is assigned to the z-axis of the TCS. All three corrections are settled in positive direction.

The tool length correction is subsequently switched on.

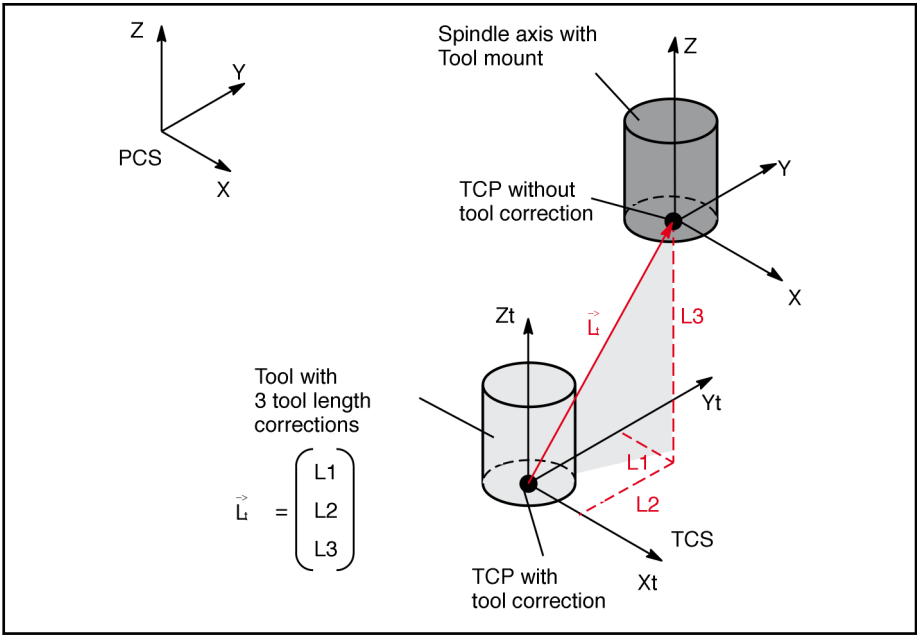


Fig. 14-19: Program example 1

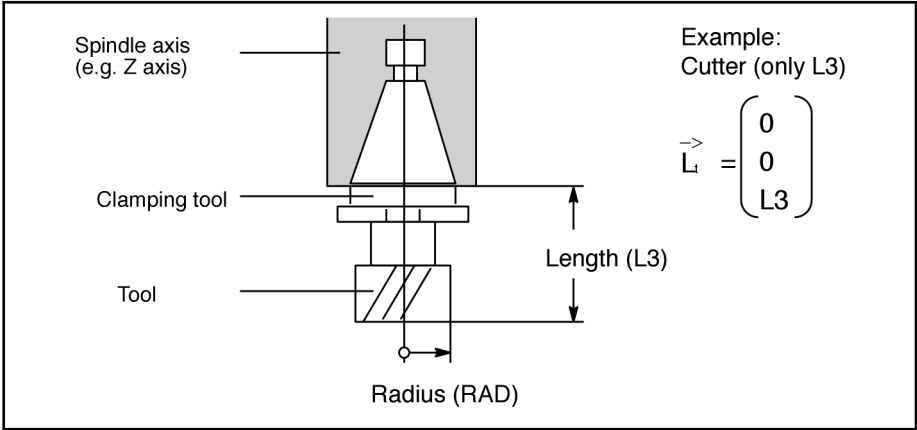


Fig. 14-20: Program example 2

	Documentation	Instruction
Documentation:	MTX Programming Manual	Activate correction

14.3.12 Handling instruction: Deactivating the tool length correction (G47/G48)

Deactivating tool length correction (G47/G48)

IW Operation/NC programming: Deactivating correction

Program **G48**



G47.. and **G48** act modally and deselect each other.

G48 can be programmed with other path conditions, traversing information or auxiliary functions.

	Documentation	Instruction
Documentation:	MTX Programming Manual	Tool length correction

14.3.13 Handling instruction: Applying the tool length correction (G47/G48)

By defining the tool length correction vector $\vec{L}_t$ in the Cartesian space (i.e. coordinates which are rectangular to each other), tools can be defined which are neither rotation symmetric nor located parallel to the longitudinal axis of the tool holder.

IW Engineering/configuration: Editing parameters

- The tool and workpiece management function has to be activated using the parameter **tlwp** "Tool and workpiece management" in the setup (SUP).
- Navigate from the node **"TLWP"** to the respective channel-specific node **"Ch[i]"** via the node **"TICorr"**.
- Right click on the **"Ch[i]"** node
- "Change to"
- Select desired function:
 - **WcsFixed**
Tool correction mode "WCS" (7050 01300)
 - **WcsActPlane**
Tool correction mode "active plane" (7050 01300)
 - **Tcs**
Tool correction mode "TCS" (7050 01300)

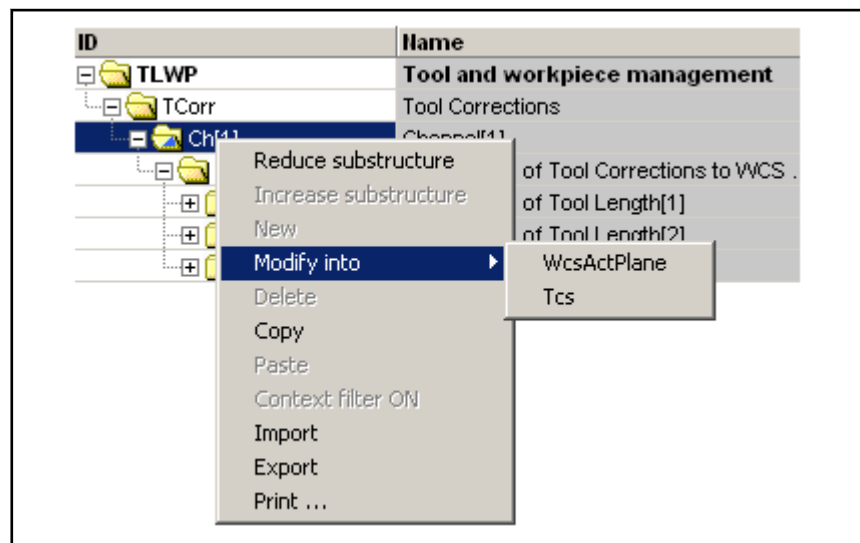


Fig. 14-21: Change to

- Depending on the selected function, only some of the following parameters are active:
 - **ChAxTICorWcs**
"Channel axis/logical axis" (7050 01310)
 - **DirTICorrWcs**
"Direction of correction" (7050 01310)
 - **CoordTICorrTcs**
"TCS coordinate" (7050 01310)

– DirTICorrTcs

"Direction of correction" (7050 01310)

	Documentation	Instruction
Instruction:		Editing machine parameters
Documentation:	MTX Programming Manual	Tool length correction

IW Operation/NC programming: Activate the tool orientation if necessary

If necessary, activate the tool orientation **StatToolOri (STO)** "parameterize static tool orientation".

	Documentation	Instruction
Documentation:	MTX Programming Manual	Tool orientation

IW Operation/NC programming: Configure the correction assignment if necessary

If necessary, reconfigure correction assignment:

G78/G79 "Correction switching On/Off"

	Documentation	Instruction
Documentation:	MTX Programming Manual	Correction assignment

14.3.14 Tool radius correction

General information

From the programmed tool action path, the programmed tool radius correction computes an offset path shifted by the tool radius.

The MTX provides two functionalities:

- 2.5D-tool radius correction:
The tool is constantly oriented in space.
The tool length axis is vertical on the active working plane on which the correction is performed
- 3D-tool radius correction:
The orientation of the tool can freely change in space. Thus, the correction depends on the tool orientation in any directions in space.

2.5D-tool radius correction (G41/G42)

Function: The path correction G41/42 causes of the tool to offset with a right-angled, constant distance to the programmed path.

This "offset" path is described as "equidistant path". The distance of the equidistant to the programmed path depends on the active radius correction value of the current tool (total of active D-correction and active ED-correction).

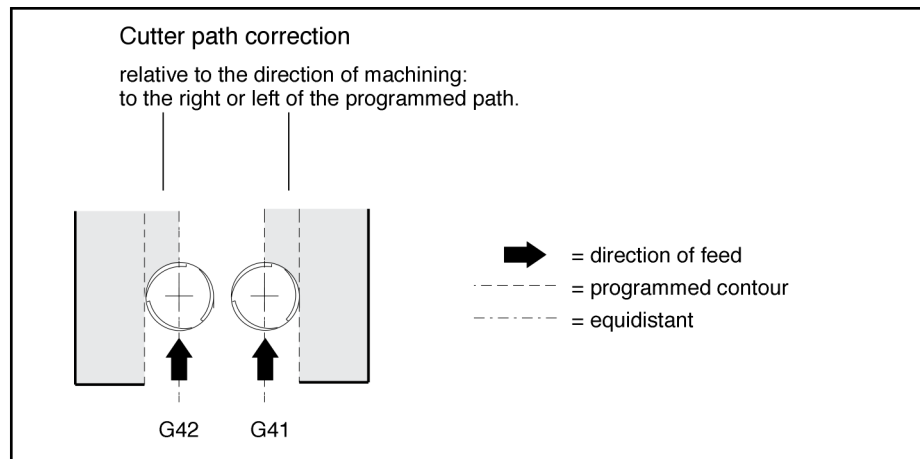


Fig. 14-22: 2.5D-tool radius correction (G41/G42)

After computing the offset paths of two neighboring path segments, two different cases can occur at the contour transition:

- Inner corner:
The offset paths intersect. Both path segments are shortened to the intersection.
- Outer corner:
First, a path gap occurs at the contour transition which has to be closed. To avoid any damage to the outer corner, it is necessary to travel around it from the tool center.

To this end, there are 2 strategies which can be set by means of the NC function contour transitions for tool radius correction G43/G44:

- Contour transition arc (G43):
The control inserts an arc which tangentially connects the two offset paths. The arc center is defined as the contour transition of the original contour.
The inserted arc is to be seen as an independent NC block which has to be started separately in automatic single step mode via NC start.
However, in automatic single block mode, the inserted arc is considered as unit together with the previously programmed NC block and they are thus processed together.
- Intersection calculation (G44):
The two offset paths are extended until the intersections of the two offset paths are found. The intersection calculation is of special advantage when there are only small transition angles between the neighboring contour segments (quasi-continuous transitions).

With pointed corners (transition angle > 90°), the distance between the intersection of the two equidistants and the tip of the original contour can be of

any size. If this distance A is bigger than the product $\sqrt{2} \times \text{CorrectionRadius}$, the control cuts the peak exactly in this distance and closes the path gap using a straight line. This behavior is also called **"cut the peak"**.

Use the "feed correction G45/G46" function to specify whether the programmed feed is to refer to the cutter action point (G45) or the cutter center point (G46).

Start correction build-up: In most cases, is not possible to approach the contour directly from the tool change point. Machining often starts from a suitable intermediate position.

If possible, select the starting point so that

- the contour is not damaged
- a tangential approach to the contour is possible
- no change in direction takes place at the first contour element

The MTX allows a correction built-up start with and without programmed traversing motion.

When starting a correction build-up with programmed traversing motion, the following cases are distinguished:

- Start correction build-up at an inner corner
- Start correction build-up at an outer corner
- Start correction build-up with the programmed optional parameter "ORTH"

When **starting the correction build-up at an outer corner**, the system moves to a position vertically to the programmed end position of the activation block. Contour transition to the subsequently programmed contour segment is closed according to the active NC function "contour transition G43/G44".

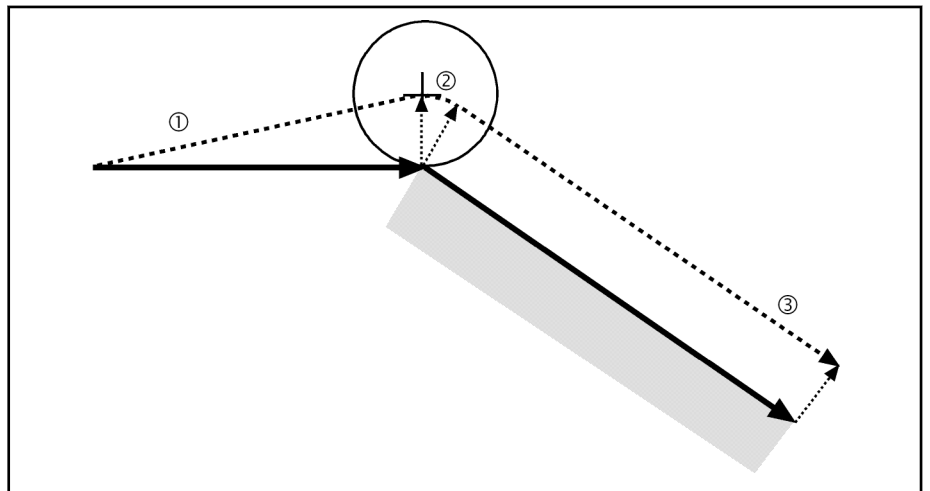


Fig. 14-23: Start correction build-up; outer corner at G41

When **starting the correction build-up at an inner corner**, the system moves to a position vertically to the programmed end position of the activation block.



At correction build-up with traversing motion at the inner corner, consider that the path from starting point "S" to the first corrected contour point "P1" is at least as big as the active correction radius. If this is not the case, the control issues a runtime error.

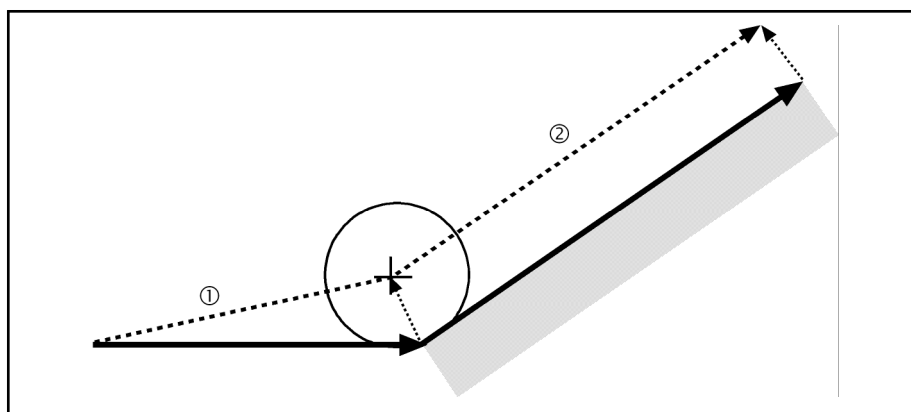


Fig. 14-24: Start correction build-up at an inner corner at G41

In case of a **correction build-up start with programmed optional parameter ORTH** (G41(ORTH) or G42(ORTH)), the correction build-up is started (regardless if it is an inner or outer corner) at a position vertically to the starting position of the next traversing block following the activation block.

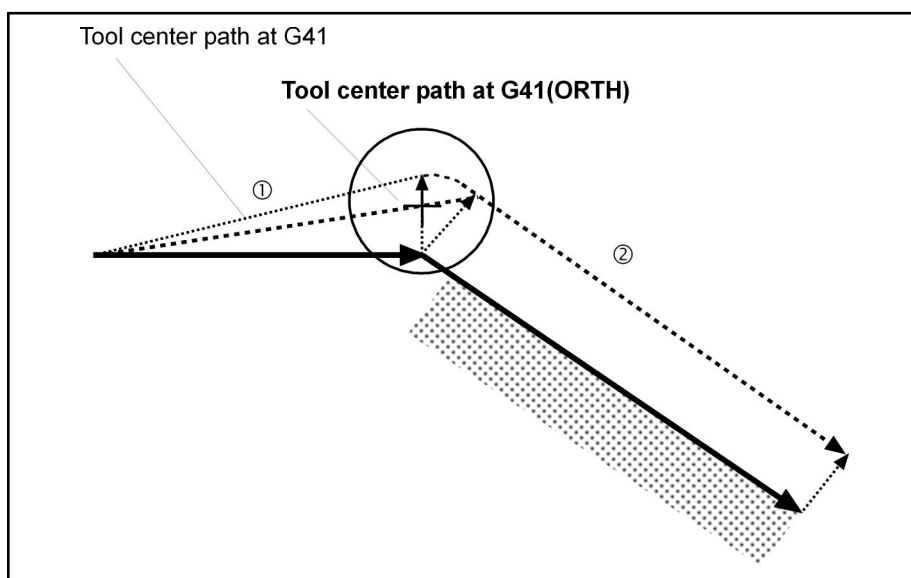


Fig. 14-25: Start correction build-up at an outer corner with/without ORTH parameter

When starting a correction build-up without programmed traversing motion, the following cases are distinguished:

- Immediate correction build-up start
- Start correction build-up with the next programmed motion at an outer corner
- Start correction build-up with the next programmed motion at an inner corner



Whether the correction build-up is started immediately or with the next programmed traversing motion is determined using the machine parameters.

When **immediately exiting the correction build-up**, the correction is retracted vertically to the next programmed contour.

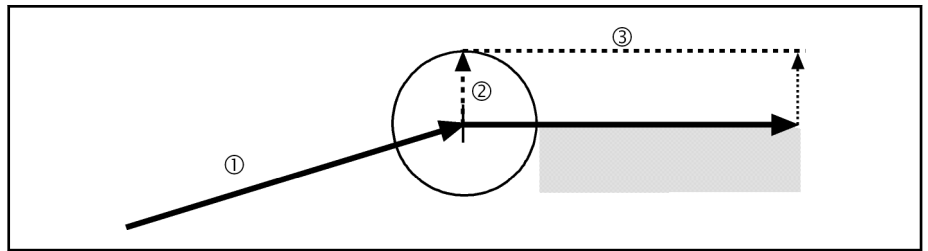


Fig. 14-26: Immediate correction build-up start

When **starting a correction build-up with the next programmed motion at an outer corner**, it is traversed in the next programmed traversing block to a position vertical to the programmed end position of this traversing block. Contour transition to the subsequently programmed contour segment is closed according to the active NC function "contour transition G43/G44".

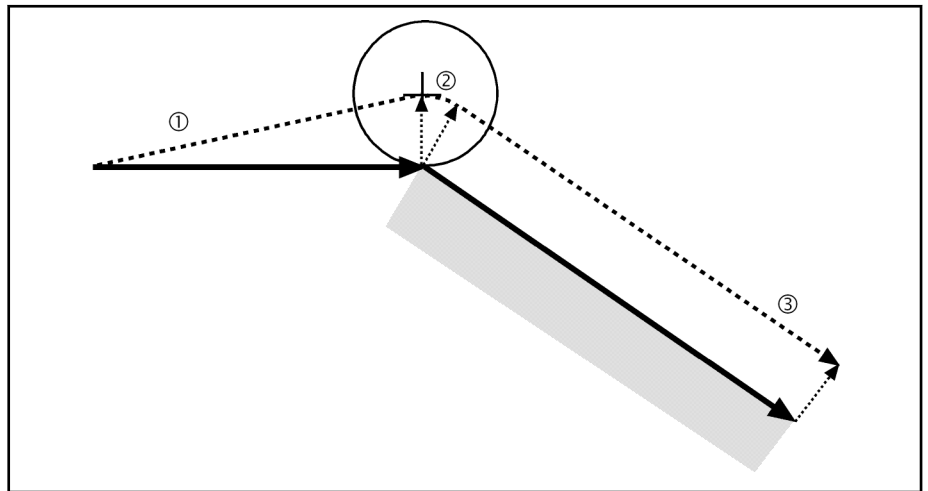


Fig. 14-27: Start correction build-up with the next programmed motion at an outer corner

"Starting the correction build-up with the next programmed motion of an outer corner" generally corresponds to "start correction build-up at an outer corner" described above. The only difference is that in the first case the command to activate the correction is programmed in an individual NC block in front of traversing block 1.

When **starting a correction build-up with the next programmed motion at an inner corner**, it is traversed in the next programmed traversing block to a position vertical to the programmed starting position of the second traversing block after activating the contour.

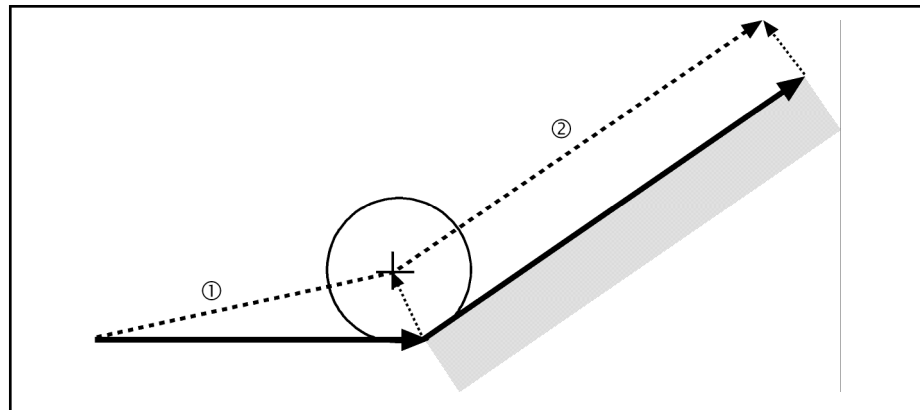


Fig. 14-28: Start correction build-up with the next programmed motion at an inner corner

"Starting the correction build-up with the next programmed motion of an inner corner" generally corresponds to "start correction build-up at an inner corner" described above. The only difference is that in the first case the command to activate the correction is programmed in an individual NC block in front of traversing block 1.

Exit correction build-up: Usually, the system does not travel directly from the contour to the tool change point but approaches it via an intermediate position (end point).

To avoid contour damage, select a suitable end point. The programmed end point is supposed to ensure tangential contour exit, if possible. Select the end point so that no free cutting caused by change of direction results when leaving the contour.

As for the start of the correction build-up, the correction deselection can be programmed with or without traversing motion.

As for the start of the correction build-up, exiting the correction build-up with programmed traversing motion also distinguishes the following cases

- Exiting correction build-up at an inner corner
- Exit correction build-up at an outer corner
- Exiting correction build-up with the programmed optional parameter ORTH

When **exiting correction build-up at an outer corner**, the correction build-up is exited from a position vertically on the starting tangent of the deactivation block. The contour transition from the previous contour segment is closed according to the active NC function "contour transition G43/G44".

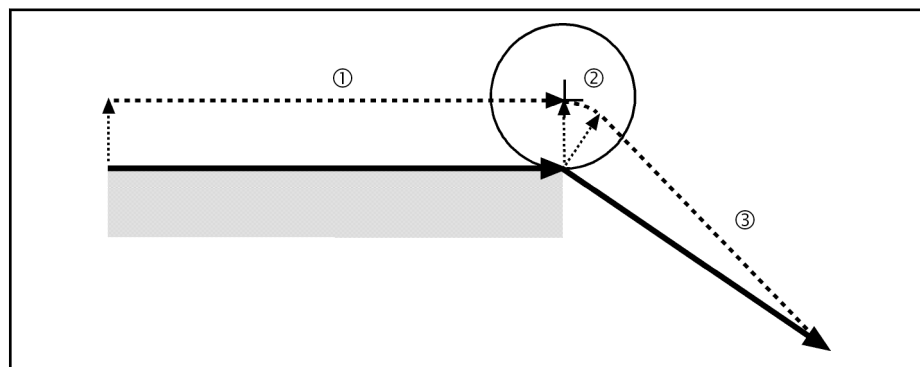


Fig. 14-29: Exit correction build-up at an outer corner

When **exiting correction build-up at an inner corner**, the correction build-up is exited from a position vertically on the end tangent of the previous traversing block.

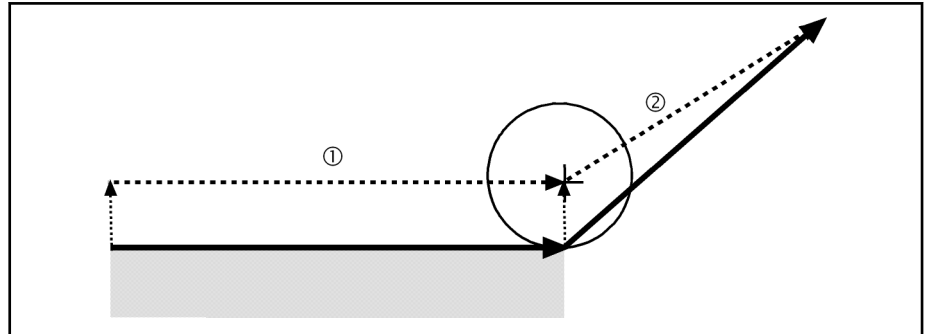


Fig. 14-30: Exiting correction build-up at an inner corner

With **correction removal with programmed optional parameter ORTH** (G40(ORTH)) the correction removal is carried out - regardless if it is an inner or outer corner - from a position which stands vertically on the ending tangent of the traversing block preceding the deactivation block.

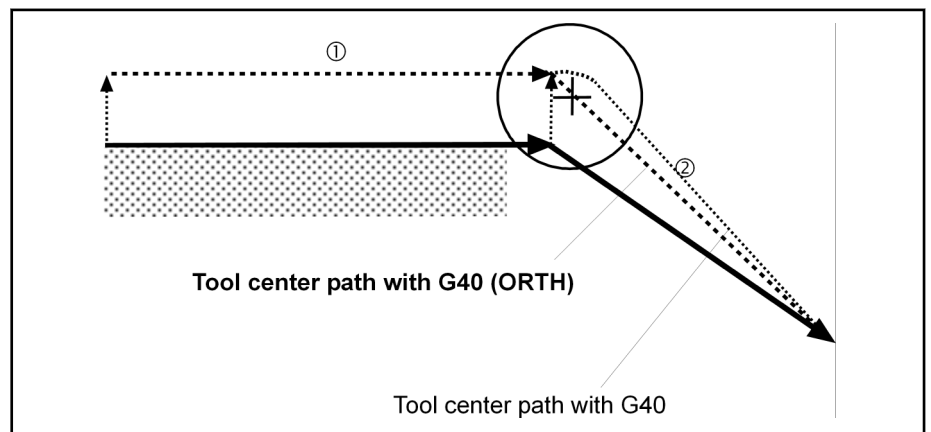


Fig. 14-31: Exiting correction build-up at an outer corner with/without ORTH parameter

When exiting the correction build-up without programmed traversing motion, the following is to be distinguished:

- Exit correction build-up on the spot
- Exit correction build-up with the programmed optional parameter NOM (NoMotion)
- Exit correction build-up with the next programmed motion at an outer corner
- Exit correction build-up with the next programmed motion at an inner corner



Whether the correction build-up is exited immediately or with the next programmed traversing motion is determined via the machine parameters.

When **immediately exiting correction build-up**, the correction is retracted vertically to the next programmed contour.

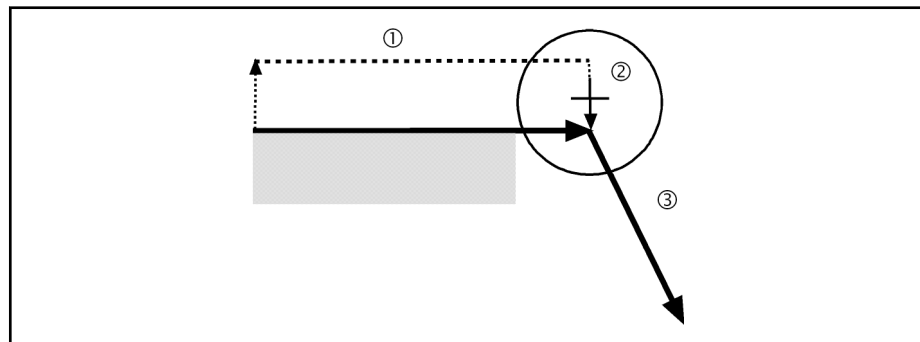


Fig. 14-32: Exit correction build-up on the spot

To **exit the correction build-up with programmed optional parameter NOM (NoMovement)**, the current workpiece position is set to the tool center point in the G40 block. The correction is retracted with the next traversing motion.

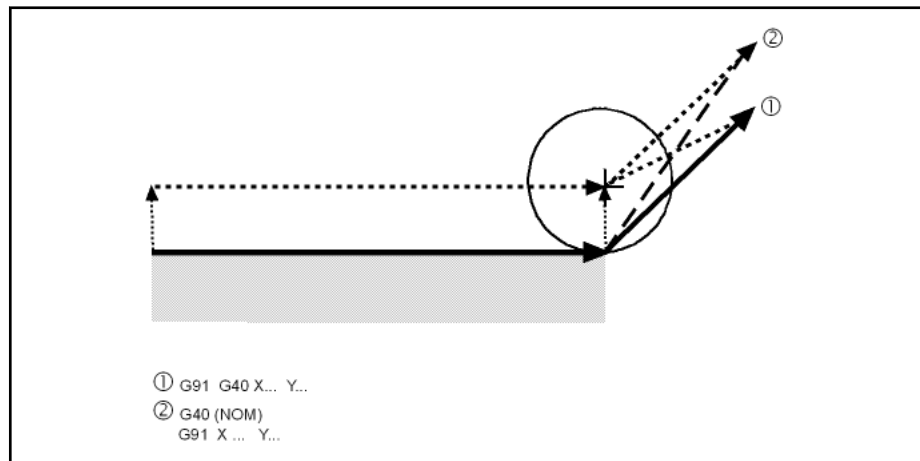


Fig. 14-33: Exit correction built-up with the programmed optional parameter NOM

When **exiting a correction build-up with the next programmed motion at an outer corner**, the correction build-up is exited from a position vertical to the starting tangent of the next traversing block. The contour transition from the previous contour segment is closed according to the active NC function "contour transition G43/G44".

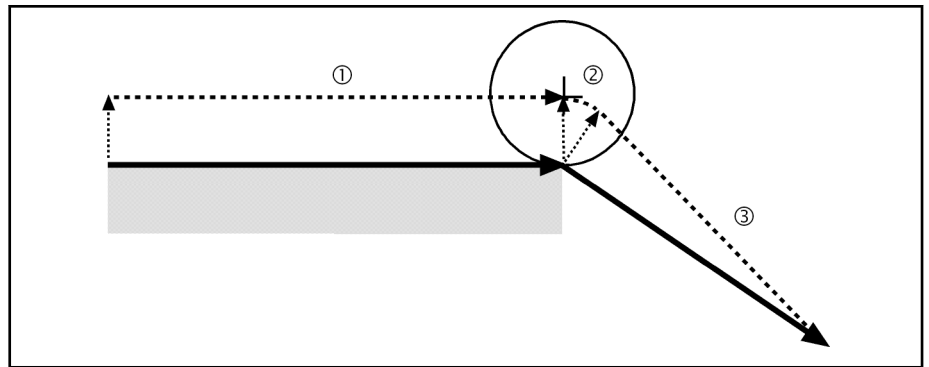


Fig. 14-34: Exit correction build-up with the next programmed motion at an outer corner

When **exiting a correction build-up with the next programmed motion at an inner corner**, the correction build-up is exited from a position vertical to the end tangent of the previous traversing block.

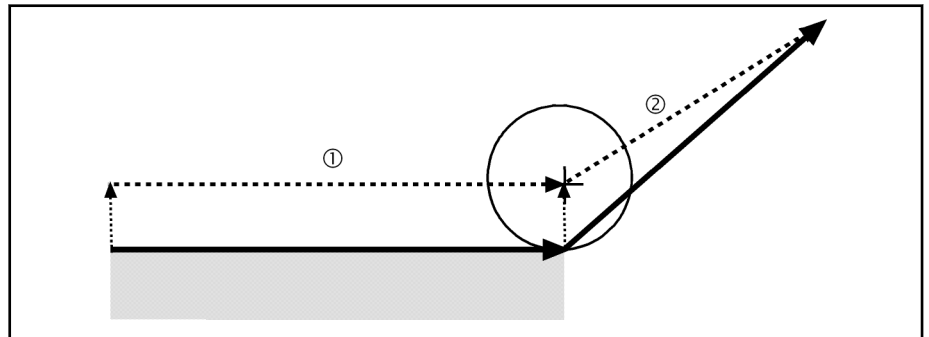


Fig. 14-35: Exit correction build-up with the next programmed motion at an inner corner

Especially when machining inner contours, ensure that the contour is not damaged when the correction is switched off without traversing motion. It is recommended to travel vertically from the contour (moving out) before switching off.

Generally, it is advantageous to leave the contour to be machined in tangential direction before moving out and switching off.

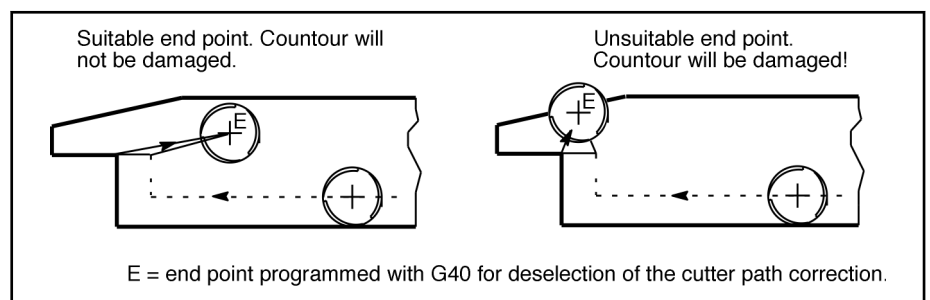


Fig. 14-36: Exit correction build-up



Since G40(NOM) executes the current workpiece position, the target position depends always on the previously active workpiece radius when an incremental programming is carried out!

Example 1: Programming an outer contour with tool radius correction

In the example, the tool radius correction is required on the left of the contour (G41). The corresponding tool radius has already been provided by the corresponding tool corrections.

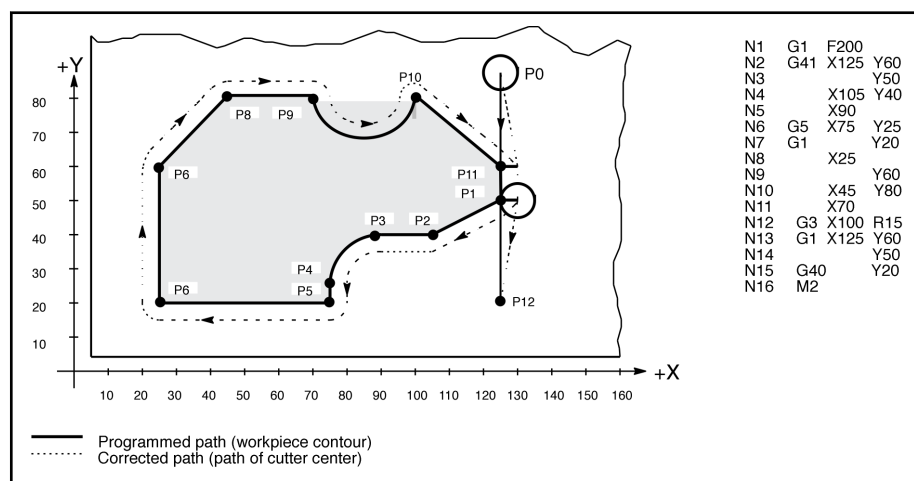


Fig. 14-37: Example 1: Programming an outer contour with tool radius correction

Example 2: Programming an inner contour with tool radius correction

Tool radius correction is required on the right of the contour (G42). The corresponding tool radius has already been provided by the corresponding tool corrections.

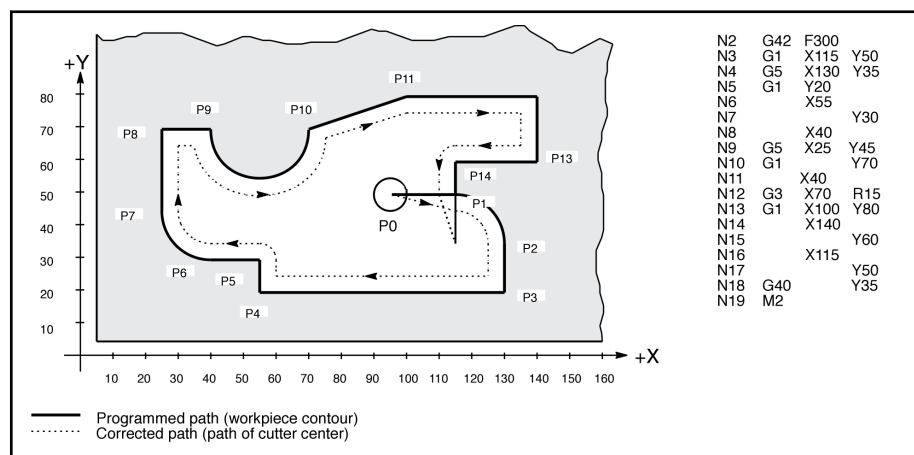


Fig. 14-38: Example 2: Programming an inner contour with tool radius correction

Correction change:

At an already active 2.5D-tool radius correction it is possible to program G41 or G42 again. Such a block identifies a correction change: The new correction can be identical with the previous correction direction (G41 ⇒ G41, or G42 ⇒ G42) or can be active in the opposite direction (G41 ⇒ G42, or G42 ⇒ G41).

One new correction build-up start is calculated for one correction change block. If a traversing motion is programmed as well in such an NC block, then as a general rule there will be no equidistant correction for this block!

MTX distinguishes three different cases for a correction change:

- Correction change without programmed traversing motion
- Correction change with programmed traversing motion
- Correction change with programmed optional parameter ORTH

Correction change without programmed traversing motion is traveled on a straight line from the corrected end point of the previous traversing block to the corrected starting point of the next traversing block.

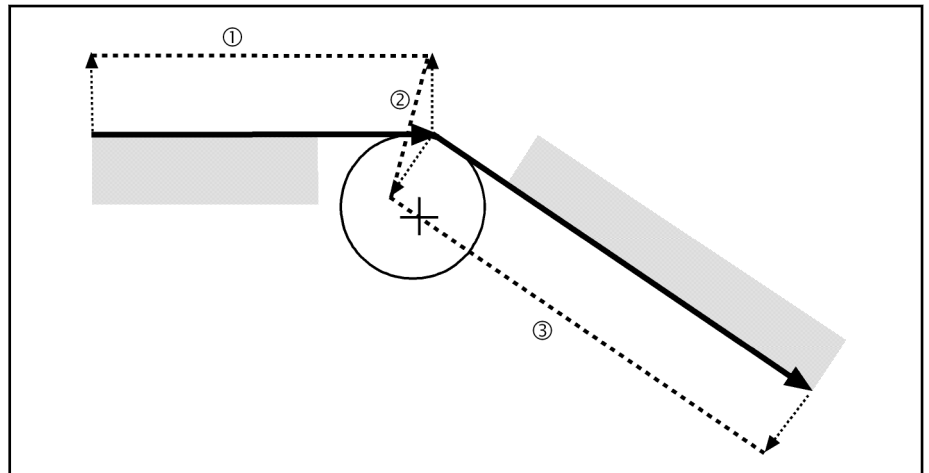


Fig. 14-39: Correction change without programmed traversing motion

A typical use case for correction change without programmed traversing motion is a reversal of direction of 180° during peripheral grinding.

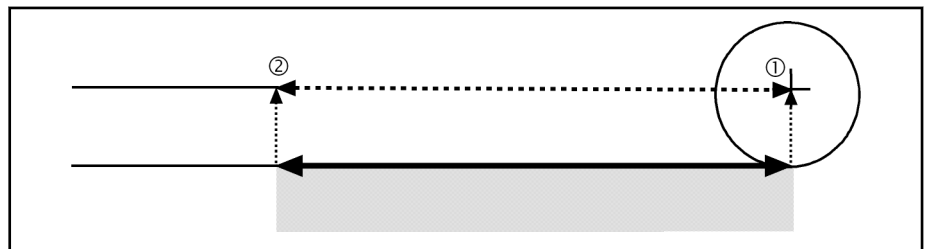


Fig. 14-40: Correction change G41 → G42 on the spot in case of reversal of direction by 180° (forward/backward contour)

Correction change with programmed traversing motion, the previous correction is still active at the starting point of the switching block while the new correction is offset for the programmed target position. It is linearly traversed between the two corrected positions.

In case of outer corners at the transition to the previous and/or the subsequent contour segment, these are bypassed according to the active NC function "Contour transition G43/G44".

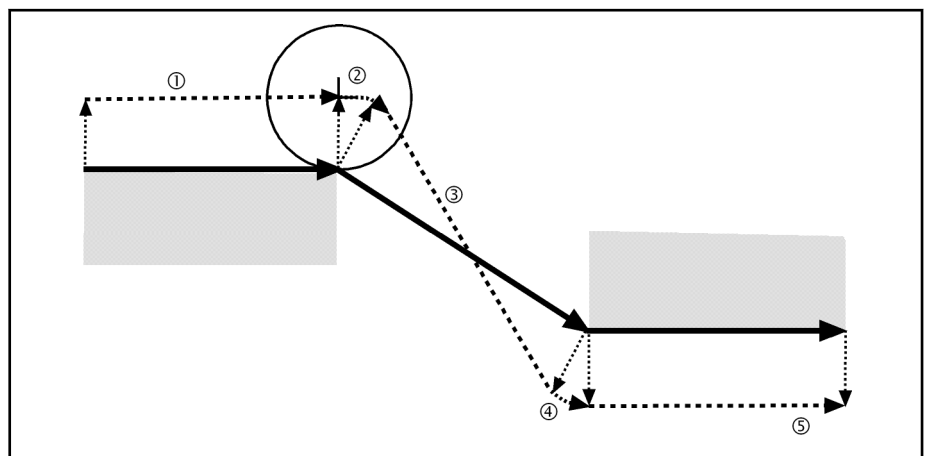


Fig. 14-41: Correction change G41 → G42 with programmed traversing motion

A correction change with programmed optional parameter ORTH (G41(ORTH) or G42(ORTH)) is always traveled on a straight line from the corrected end point of the previous traversing block to the corrected starting point of the next traversing block, regardless if a traversing motion is programmed or not. Thus, the correction behavior corresponds to the variant "Correction change without programmed traversing motion".

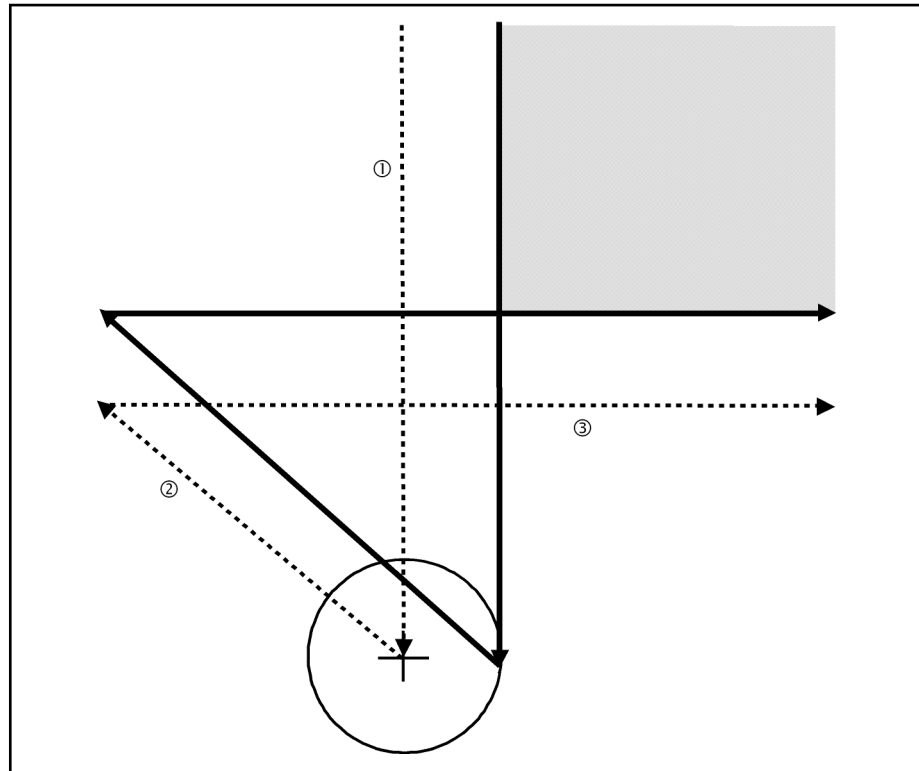


Fig. 14-42: Correction change G41 → G42 with programmed optional parameter ORTH

Collision detection and monitoring:

The tool radius correction of the MTX optionally provides the possibility to monitor collisions of a rotation-symmetric tool (e.g. cutter) with the programmed contour.

Three different operation modes can be set:

- Detected collisions are eliminated automatically.
- Detected collisions are eliminated and a warning is output simultaneously. Machining is continued.
- When detecting a collision, the control issues a runtime error and stops machining.

From the control's point of view, a tool collides with the programmed contour if - with regard to the set look-ahead function - intersections or contact points of two path segments are detected on the offset path computed by tool radius correction.

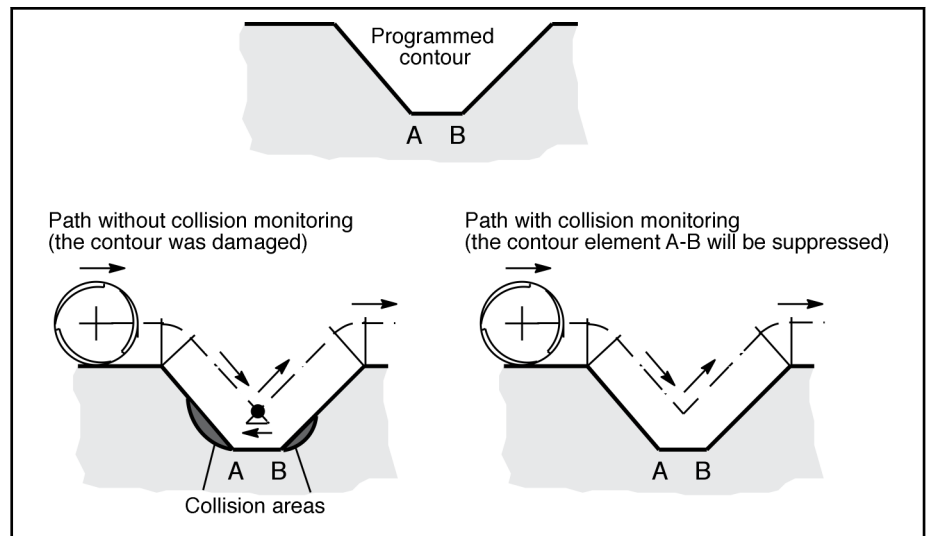


Fig. 14-43: Collision detection and monitoring

In case of an intersection, the offset path always forms a contour loop. The control is not able to detect contour loops exceeding the set range of look-ahead function as collision!

The look-ahead-area for collision observation is by default 2 blocks, but can be changed using optional parameters locally (parameter: LA) or modal (parameter: DLA) in the context of the look-ahead area which is provided for the total tool radius correction.

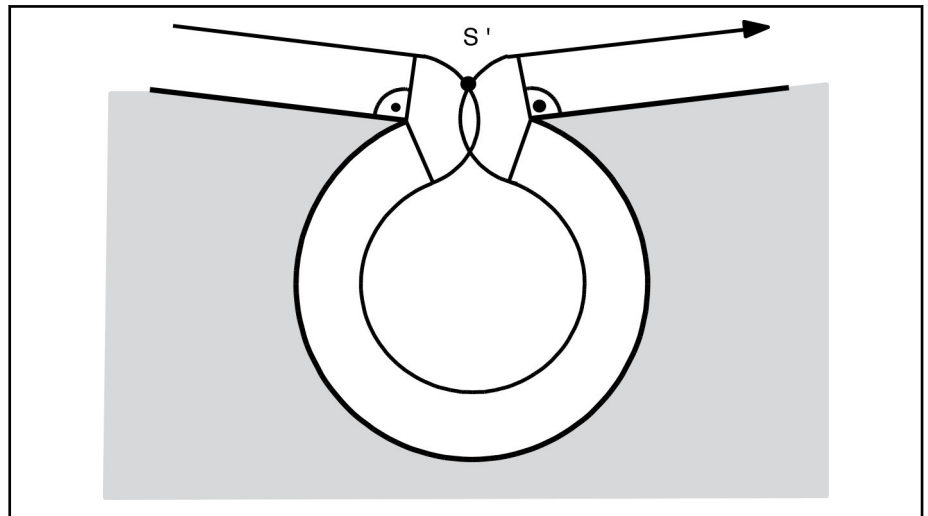


Fig. 14-44: Collision at intersection S'

In the above example, the collision at intersection S' is already detected with an analysis depth of 2 NC blocks. With only one look-ahead NC block, the collision is not detected and the contour is violated.

Intersections of the programmed contour always result in an intersection of the offset path and thus in a collision. To detect the intersections, the control exclusively considers the values of the two coordinates of the current working plane. Changes of the infeed depth are not considered as an evaluation criterion, as the control has no information on the plunging depth (tool plunging into the workpiece).

Programmed full circles are handled differently. These circles are always a loop of the programmed contour with themselves, but they are excluded from

collision monitoring as long as they remain full circles when considering the tool radius (offset path). This is implemented by entering tangentially into the full circle and leaving it tangentially.

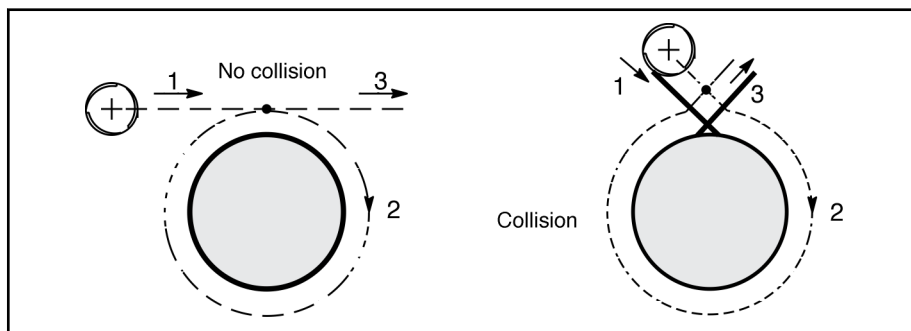


Fig. 14-45: No collision - Collision

Generally, the control cannot evaluate if a detected collision is relevant for machining. If collisions detected as irrelevant are not to be removed, collision monitoring has to be explicitly switched off (if applicable only for a defined local area).

Explanations on tool edge correction (ORI):

If special non-rotation symmetric tools fixedly oriented in space are used which are (e.g. cutting lathe), tool edge correction is relevant when non-axis-parallel motions are executed.

This focuses on the motion of the tool engaging point along the programmed contour. Note that the tool engaging point can be offset along the edge radius depending on the programmed contour.

Exclusive use of length corrections for main and secondary axis results in a fixed theoretical edge tip P.

Since the control leads the theoretical edge tip P along the programmed contour, it is possible that material is not chipped completely or chipped too much during machining.

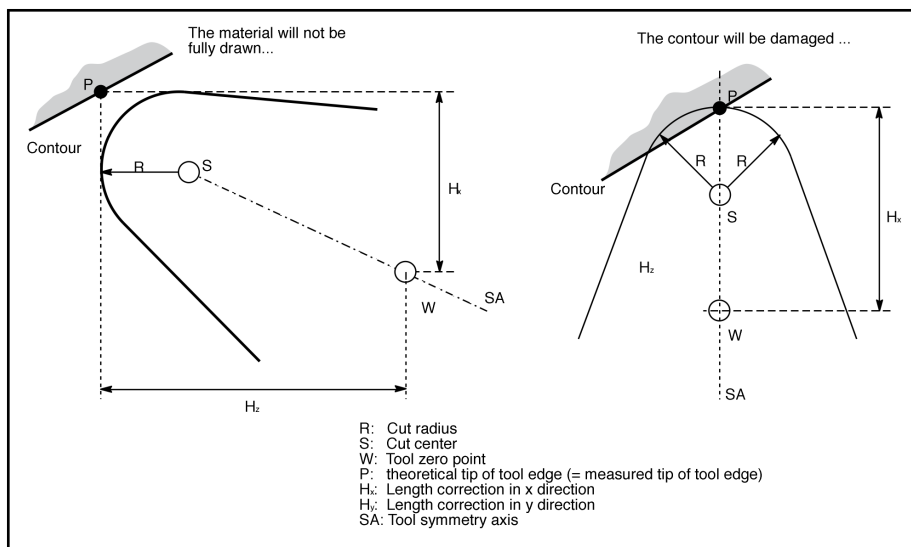


Fig. 14-46: Edge position correction

To solve the problems shown above, two correction types are used simultaneously:

- Edge position correction
- Tool radius correction (G41, G42)

The tool edge position correction ensures that not P, but the edge center S is located on the programmed contour.

The tool radius correction finally causes the edge center S to move along an equidistant path with the distance R to the programmed contour.

For the **edge position correction**, edge position (ORI) and the edge radius (RAD) are required.

The edge positions ORI=1 till ORI=8 describe the general orientation of the tool with regard to the main axis and the secondary axis (refer to the following figure). Use ORI=9 to switch off the edge position correction.

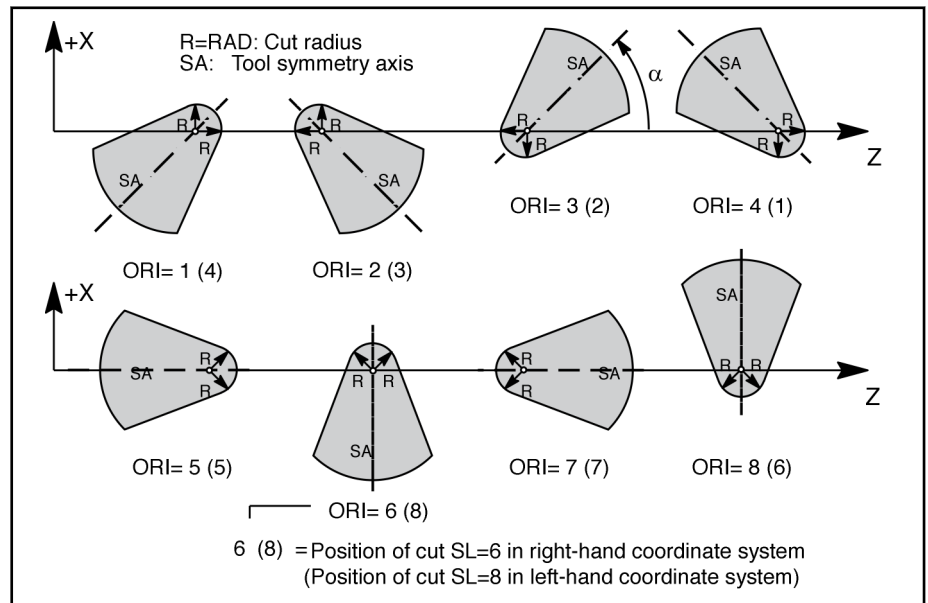


Fig. 14-47: Edge position ORI=1 till ORI=8

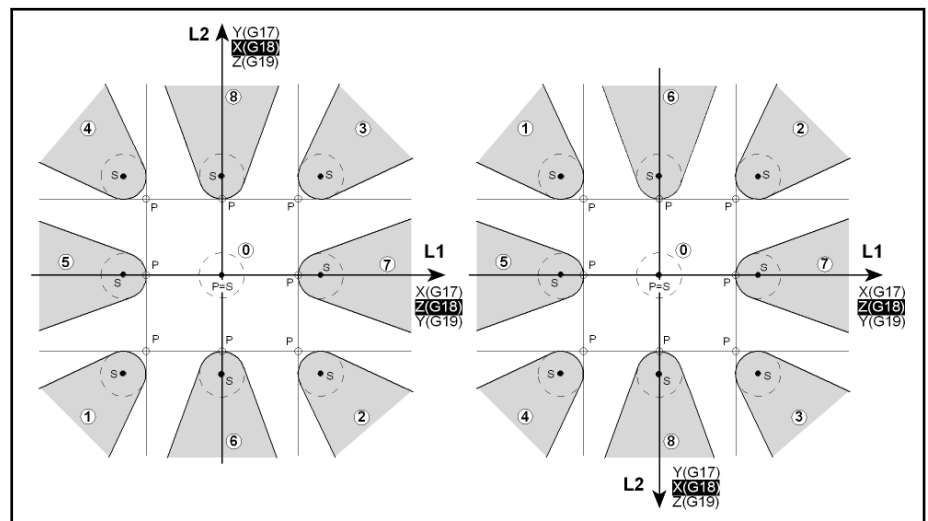


Fig. 14-48: Possible positions of a tool edge

Considering the specified edge position, the NC uses the edge radius to compute an additional shifting of the shape additively or subtractively effective to the normal length correction:

- $(\pm)\Delta L_S = R$ and $(\pm)\Delta L_M = R$ (for the edge positions 1 to 4)
- $(\pm)\Delta L_S = R$ (for the edge positions 6 or 8)

Tool corrections

- $(\pm)\Delta L_M = R$ (for the edge positions 5 or 7)

ΔL_M : Change in length correction for the main coordinate of the active plane

ΔL_S : Change in length correction for the secondary coordinate of the active plane

The following figure shows the principle:

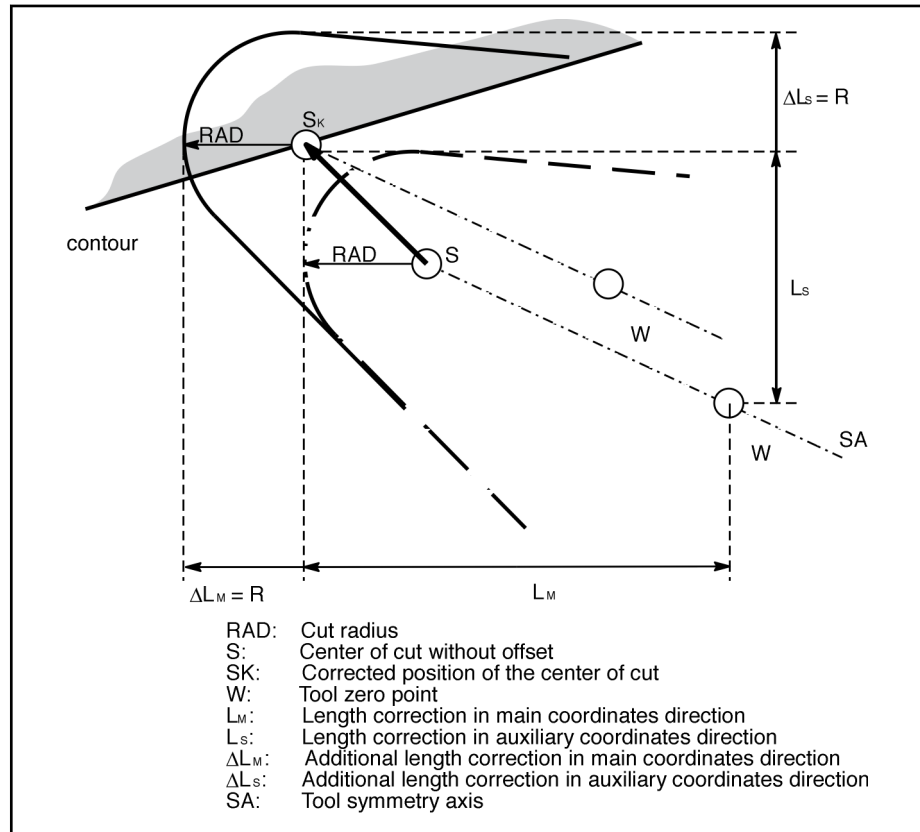


Fig. 14-49: Change in length correction - Principle

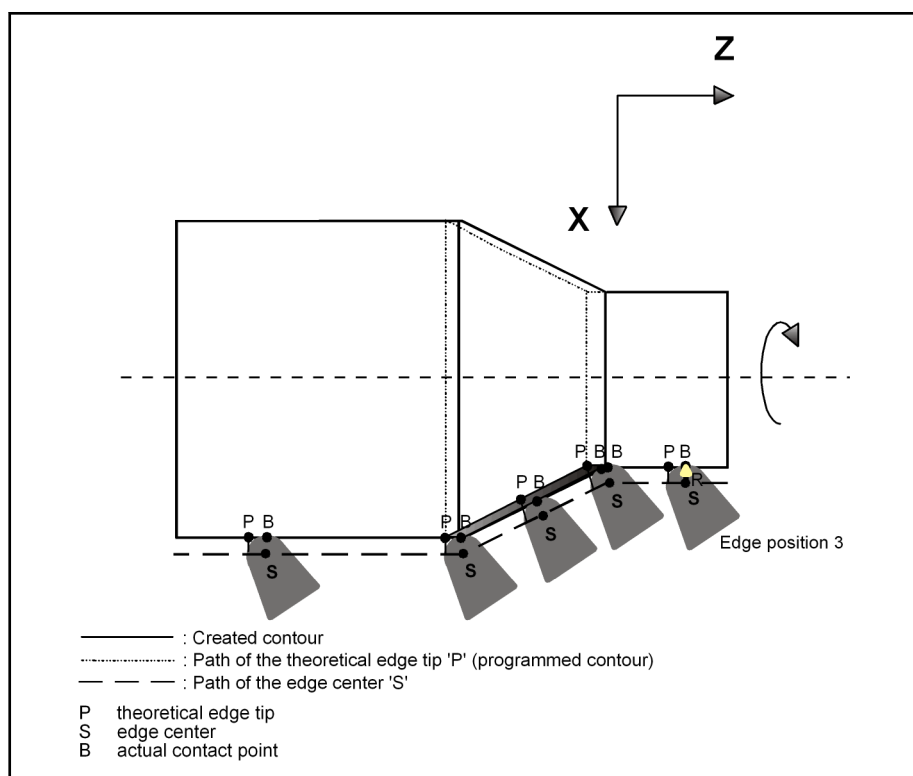


Fig. 14-50: Inaccuracies occurring when working without cutting radius path correction

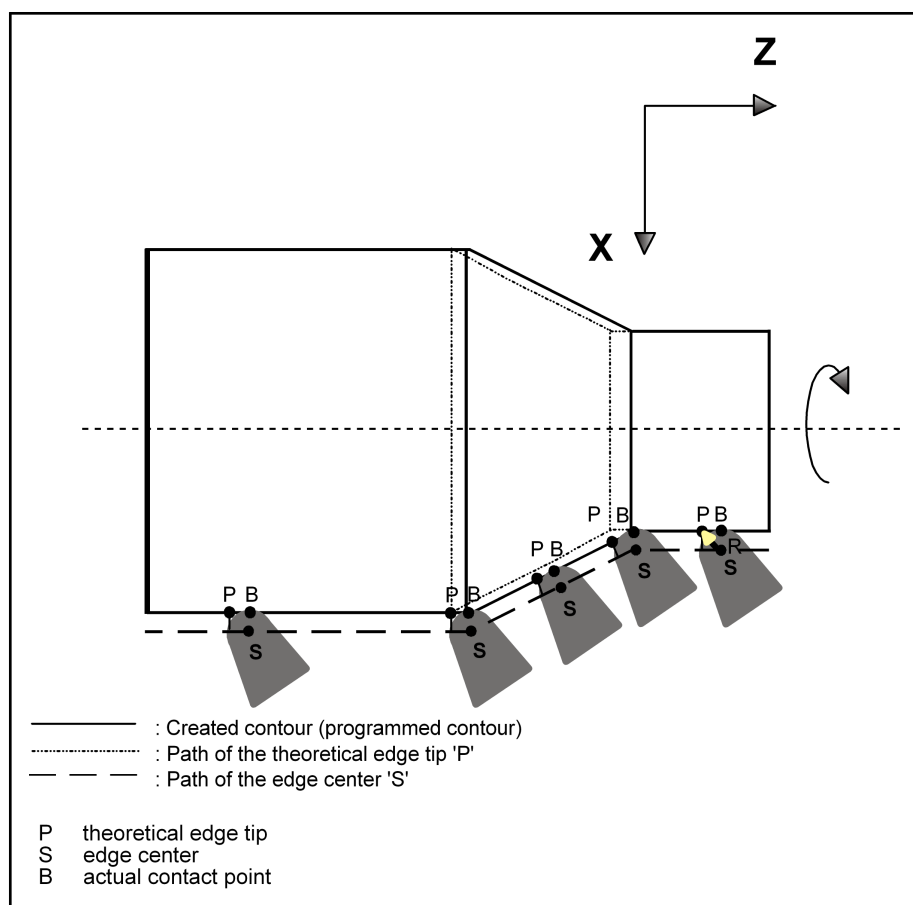


Fig. 14-51: *Machining without inaccuracies when the cutting radius path correction is activated*

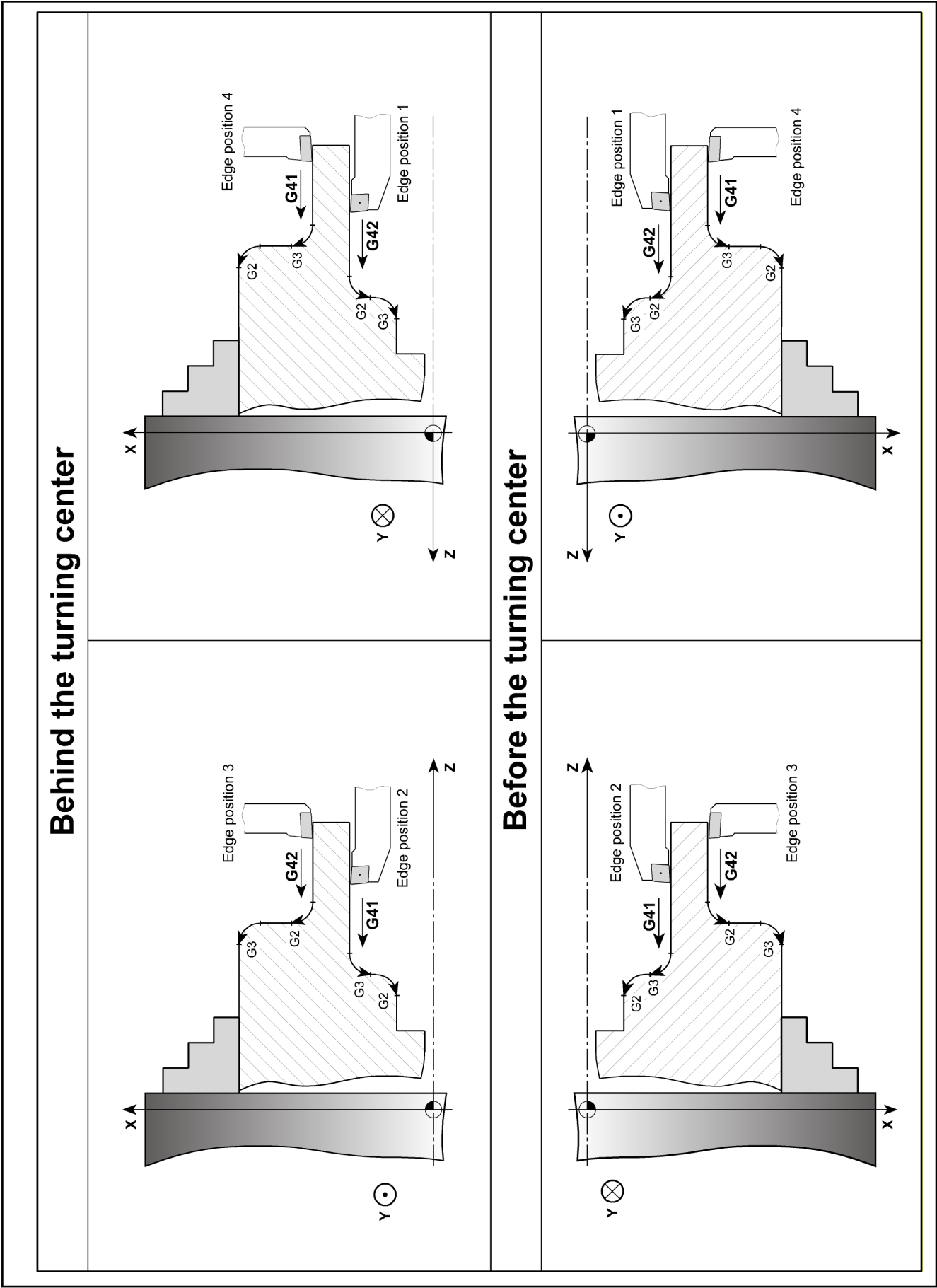


Fig. 14-52: Tool path correction G41, G42 when machining behind and in front

Tool corrections

of the rotating center

The tool radius correction offsets the path by the edge radius so that the edge is located directly on the programmed contour.

- Prerequisites:**
- Tools and their dimensions (corrections) are measured for the machining
 - The machining program changes the tools with regard to the machining section
 - D-correction tables and/or the ED-correction registers provide the correct corrections for the tools used. The ED-correction registers can be written from the PLC or the CPL
 - The effective correction is selected via D- and/or ED-command

- Restrictions:**
- When the control is in a circular operation mode (e.g. G2, G3, G5), **no** traversing motion has to be programmed in G40, G41 and G42 blocks
 - 3D-tool radius correction only supports straight lines and circles (e.g. helical elements). When splines are used, 3D-cutter radius correction has to be used instead of 2D-cutter radius correction

Refer to [chapter "3D-tool radius correction" on page 236](#).

- *When path correction is active, the following functions are **not** permitted:*
 - G17 to G20 (plane switching)
 - G63 (tapping without compensating chuck)
 - G70, G71 (inch/metric switching)
 - G74 (approach reference point coordinates)
 - G75 (travel to touch probe)
 - SetPos (set program position)
 - Switching of a zero point offset
 - Switching a placement
 - Switching an axis transformation
 - Axis exchange

Relevant NC functions:

G40	Disable path correction
G41	Activating the path correction left of the programmed path (seen in machining direction) with regard to positive correction values
G42	Activating path correction right of the programmed path . Otherwise identical to G41
G43	Contour transition arc
G44	Contour transition intersection
G45	Feed correction: tool engaging point
G46	Feed correction: tool center point
Collision	Activating/configuring collision monitoring

CLN	Short form for collision Collision/CLN parameter:	
	CollErr:	Controlling error output
	LA:	Local monitoring depth
	DLA:	Modal monitoring depth
	DEF:	Monitoring depth; default setting (two blocks)
DcTSel, DCS	Activating D-correction table	
DCT	Writing D-correction or tool correction value	
D1... D99	Select D-correction (1 .. 99).	
ED1-ED16	Selecting external tool correction.	

Tab. 14-14: Relevant NC functions

Relevant CPL functions:

NCF("G40")	Provides information whether G40, G41 or G42 is active in the corresponding channel
NCF("G43")	Provides information whether G43 or G44 is active in the corresponding channel
NCF("G45")	Provides information whether G45 or G46 is active in the corresponding channel

Tab. 14-15: Relevant CPL functions

Relevant interface signals:

iCh_G41G141Act	Set: G41/G141 is active in the corresponding channel
iCh_G42G142Act	Set: G42/G142 is active in the corresponding channel

Tab. 14-16: Relevant IF signals

If **both** signals **not** set, path correction is switched off completely (G40 and G140 are active).

Relevant machine parameters (MP):

7060 00110	Total number of blocks for block look-ahead
7060 00120	maximum number of blocks for block look-ahead
7060 00130	Number of blocks for block look-ahead in percentage

Tab. 14-17: Relevant machine parameters (MP)

Handling instruction: Activating 2.5D Tool Radius Correction (G41/G42)**Applying: When using a tool correction table:**

1. Create new D-correction table.
2. Enter a name.
3. Complete the table with length and radius correction values for machining.

When using the external tool correction (correction values provided via PLC):

1. Adjust the PLC program function block MT_TCorr to provide the length and radius correction or
2. Write the values into the correction registers via the CPL command DCT(...,0).

Activating: When using a D-correction table:

1. Activate the desired table via DcTSel.
2. Program D.. to select the corresponding radius and length correction values from the activated table.
3. Select a starting point which allows a tangential approach to the contour. Ensure that the contour is not damaged when the correction is approached.
4. Program G41 or G42.

When using the "external tool correction":

1. Select a tool using the tool address T.. and change it via the M-function M... Thus, the relevant tool corrections are selected automatically.
2. Preselect ED1-ED16 (external tool correction).
3. Select a starting point which allows a tangential approach to the contour. Ensure that the contour is not damaged when the correction is approached.
4. Program G41 or G42.

Deactivating: Prerequisite:

Tool is no longer engaged and left the contour.

1. Ensure that the contour is not damaged when the correction is exited. Approach a suitable end point.
2. Program G40 (with or without traversing motion).
3. Deselect the D-correction and/or ED-correction with D0 or ED0 if applicable.

From the programmed tool action path, the programmed tool radius correction computes an offset path set off by the tool radius. For the 2.5D tool radius correction, the tool is constantly oriented in space. The tool length axis is vertical on the active working plane on which the correction takes place.

IW Engineering/configuration: Editing parameters

- **NofBIPrep**
"Total number of blocks for block preparation and interpolation" (7060 00110)

Maximum number of function-related blocks:

- **MaxBIVelCalc**
"Maximum number of blocks for jerk-limited velocity control" (7060 00120)
- **MaxBISpline**
"Maximum number of blocks for spline functions" (7060 00120)
- **MaxBITICorr3D**
"Maximum number of blocks for 3D milling cutter path correction" (7060 00120)

Distribution of the entire number of blocks in percentages:

- **PercVelCalc**
"Number of blocks for jerk-limited velocity control" in percent (7060 00130)
- **PercSpline**
"Number of blocks for spline functions in percentages" (7060 00130)
- **PercTICorr3D**

"Number of blocks for 3D milling cutter path correction" in percent (7060 00130)

	Documentation	Instruction
Instruction:		Editing machine parameters
Documentation:	MTX Machine Parameters	Tool radius correction

Providing ED-correction

If necessary, the ED-correction has to be configured via the tool correction interface.

		Instruction chapter 14.3.7 "Handling instruction: Activating the ED-correction" on page 204
Instruction:		Applying ED-correction

Applying D-correction

If necessary, create and configure a D-correction table with the desired correction values.

		Instruction chapter 14.3.7 "Handling instruction: Activating the ED-correction" on page 204
Instruction:		Applying D-correction

IW Operation/NC programming:

- **G41**

2.5D tool radius correction **left of the workpiece ON** (with positive correction values seen in the direction of machining). Except for a D-address containing the required radius correction value, a linear traversing motion for axes of the active plane can be programmed in the same block as **G41**. This retracts the correction while traveling to the end of the traversing motion. If no traversing motion is programmed in the **G41** block, the control builds up the correction immediately and vertically to the next traversing block.

- **G42**

2.5D tool radius correction **right of the workpiece ON**.
Otherwise identical to **G41**.

	Documentation	Instruction
Documentation:	MTX Programming Manual	Tool radius correction

Handling instructions

Handling instruction: Deactivating 2.5D tool radius correction (G41/G42)

From the programmed tool action path, the programmed tool radius correction computes an offset path set off by the tool radius. For the 2.5D tool radius correction, the tool is constantly oriented in space. The tool length axis is vertical on the active working plane on which the correction is performed

IW Operation/NC programming:

G40 2.5D tool radius correction OFF (switched-on state).

If no traversing motion is programmed in the G40 block, the control immediately deactivates cutter path correction vertically to the last traversing block.

If a traversing motion is programmed in the G40 block, the control deactivates cutter path correction linearly while traveling to the end point of the traversing motion. Effective tool length correction is not influenced by G40.

	Documentation	Instruction
Documentation:	MTX Programming Manual	Tool correction

3D-tool radius correction

Function: Guides the point of action of a rotation-symmetric tool with feed active to the left (G141) or to the right (G142) of the programmed path.



When axis transformation is switched off, the 3D-tool radius correction acts like a 2D-tool radius correction.

In this case, the orientation of the tool is determined by the normal with respect to the active plane (G17(..), G18(..), G19(..)).

The correction between the action point at the workpiece and the tool tip (TCP) is affected by:

- Tool radius (ED, D, DcTSeI, DCS, DCT)
- Plunging depth of the cutter
- Orientation of the tool to the main plane
- Programmed contour allowance
- Programmed change rate of the contour allowance

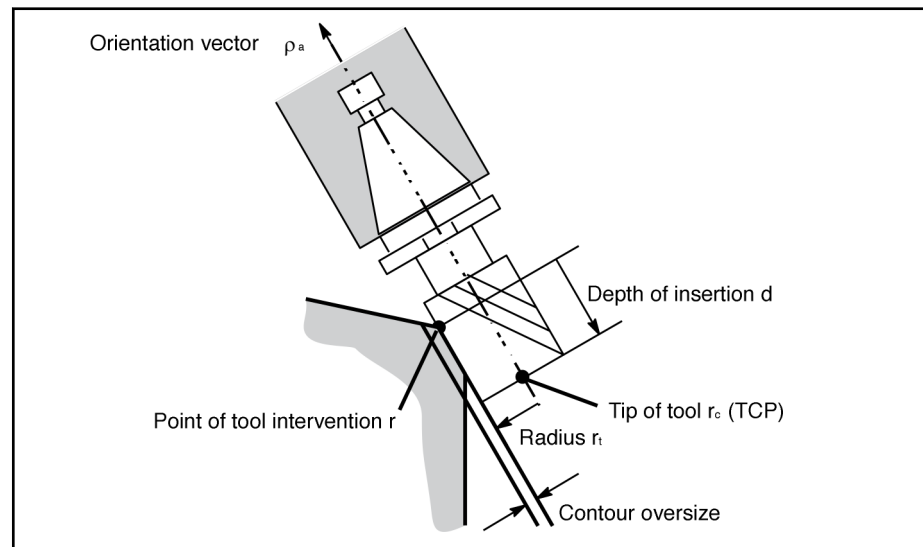


Fig. 14-53: 3D tool radius correction

Characteristics of the 3D-tool radius correction:

- Start correction build-up:**
- In the activation block G141/G142, the correction build-up is started if there is a traversing motion programmed in the same block in at least one of the three active coordinates (spanning across the space). Otherwise, the correction build-up is started only together with the next corresponding traversing motion.

- Correction can be built up on any type of contour (straight line, circle, spline, helix)
- Tool radius and plunging depth are interpolated from zero to the programmed values using third degree splines
- During the correction build-up start, the currently valid change rate of the allowance value for the contour is not effective

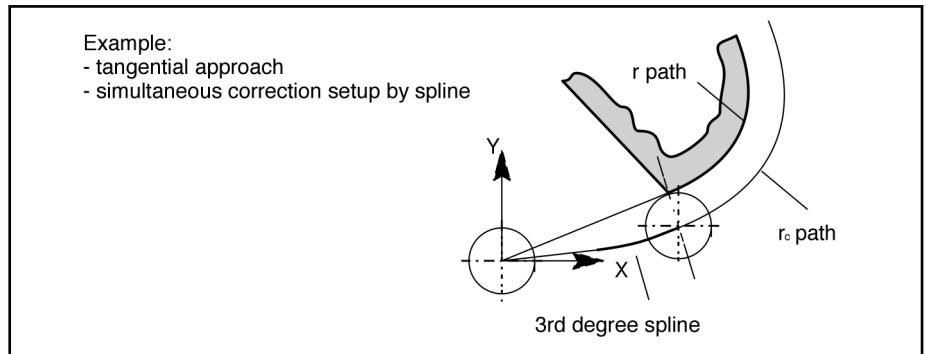


Fig. 14-54: Example - Tangential approach - simultaneous start of the correction build-up per spline

- The starting of the build-up block can contain an additional tool orientation motion
- The end point of the starting of the build-up block can be part of an inner or outer corner

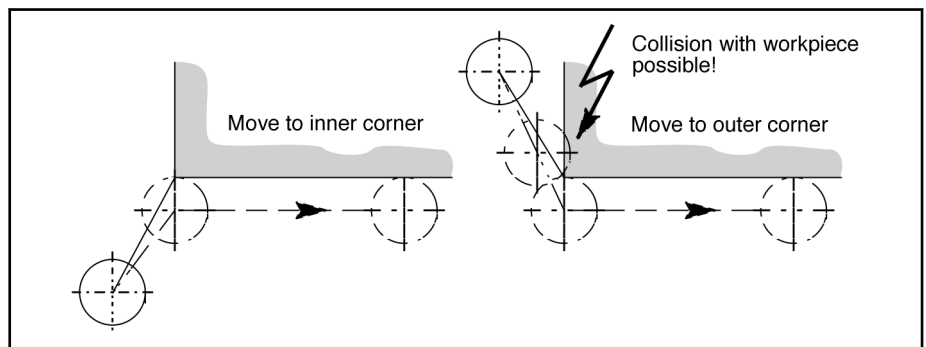


Fig. 14-55: Approaching an inner/outer corner

Exit correction build-up:

- The correction build-up is exited in block G140 either with or without simultaneous path motion
- Exiting the correction build-up can be executed on any type of contour (straight line, circle, spline, helix)
- While exiting the correction build-up, the currently valid change rate of the contour allowance has no effect
- The exiting of the correction build-up block can comprise an additional tool orientation motion
- The starting point of the exiting of the correction build-up block can be part of an inner or outer corner
- Radius and plunging depth are interpolated to zero from their current value using third degree splines

Changes in tool orientation:

With pure orientation movements and active 3D-tool radius correction, the current position of the action point "r" remains fix.

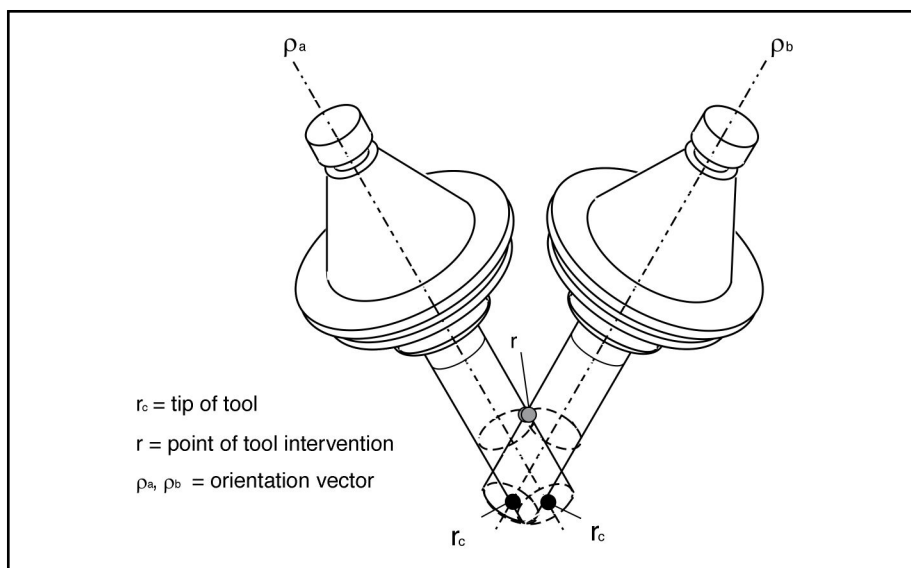


Fig. 14-56: Changes in tool orientation

Changes in plunging depth/radius:

Changes in plunging depth, allowance, tool length or cutter radius during an active 3D-tool radius correction are performed immediately and via splines. If the 3D-tool radius correction is not active, they become effective at the next activation.

Change rate of the contour allowance:

Changes in the change rate of the contour allowance are applied immediately if a 3D-tool radius correction is active and the correction build-up is completed. Otherwise, they are applied if this state is reached for the next time at the beginning of the block.

The change rate of the contour allowance has the following effect:

- First, the control determines the starting value of the contour allowance for the current block. If a contour allowance is programmed in the current block, this is the programmed value. Otherwise, it is the end value of the previous block
- It then determines the end value of the correction allowance for this block:

$$\langle \text{EndValue} \rangle = \langle \text{StartingValue} \rangle + \langle \text{Length of the path curve of the contact point between tool and workpiece programmed in this block} \rangle * \langle \text{Change rate of the contour allowance} \rangle$$
- The contour allowance is then interpolated linearly from the starting value at the starting point of the block to the end value at the end point

Behavior at outer corners:

- Outer corners without change of tool orientation:
 A compensation circle to travel around the outer corner is inserted automatically

Example:

N1 x.. y.. z.. Path block in front of outer corner

(compensation circle) automatically inserted compensation circle

N2 x.. y.. z.. Path block behind outer corner

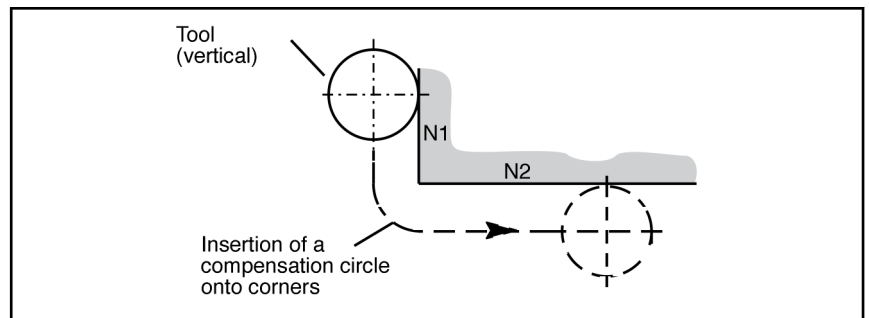


Fig. 14-57: Outer corners without change of tool orientation

- Outer corners with change in tool orientation:
If a tool orientation change is programmed at the outer corner, the tool orientation motion is executed **first**.
The compensation circle is then inserted automatically.
Example:
N1 x.. y.. z.. Path block in front of outer corner
N2 phi.. theta.. Orientation motion at the corner
(compensation circle) Inserted compensation circle
N3 x.. y.. z.. Path block behind outer corner

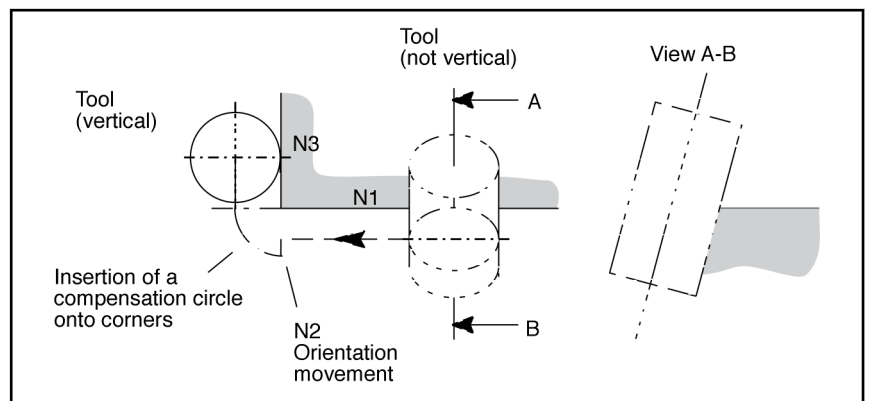


Fig. 14-58: Outer corners with change in tool orientation

- Change rate of the contour offset:
On the automatically inserted compensation circle, the contour allowance is generally not changed, as no traversing motion is programmed in this block and thus, the length of the path curve of the contact point between tool and workpiece programmed in this block is equal to zero.

Behavior at inner corners:

With inner corners, the contact point of the tool is computed with the contour elements surrounding the corners. Machine parameters 7060 00120 [5] are used to define the number of NC blocks to be considered during this computation.

To compute the contact points, the tool orientation programmed at the end of the block leading to an inner corner is used.

Traversing or tool orientation motions within the "look-ahead area" (MP 7060 00120 [5]) between the two successive contact points are deleted.

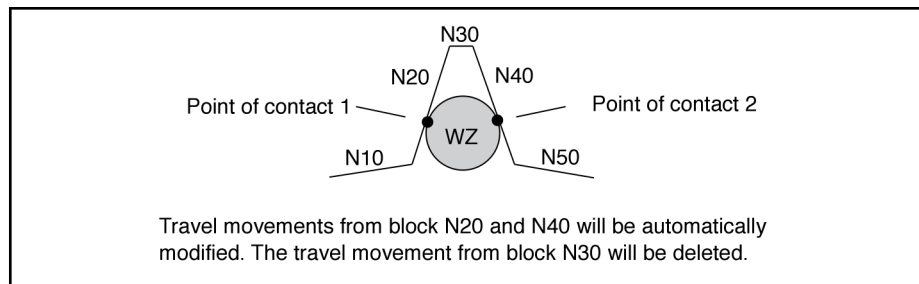


Fig. 14-59: Behavior at inner corners

For completely deleted blocks (in the example shown the block N30), the change rate of the contour offset has no effect. For shortened blocks (in the example shown the blocks N20 and N40), the change rate is applied as described above even if the start and/or end point of the block shift due to inner corners.

If changes of the tool orientation between the detected contact points are programmed, the following applies:

- In the first contact point block, the tool orientation is modified in such a way that it reaches the inner corner orientation (start orientation of N30 in the above illustration) already at the first contact point.
- Accordingly, the second contact point block starts with the inner corner orientation (contact point 1 and 2 have the same orientation).
- A jump of the plunging depth is reduced again in the second contact point block.

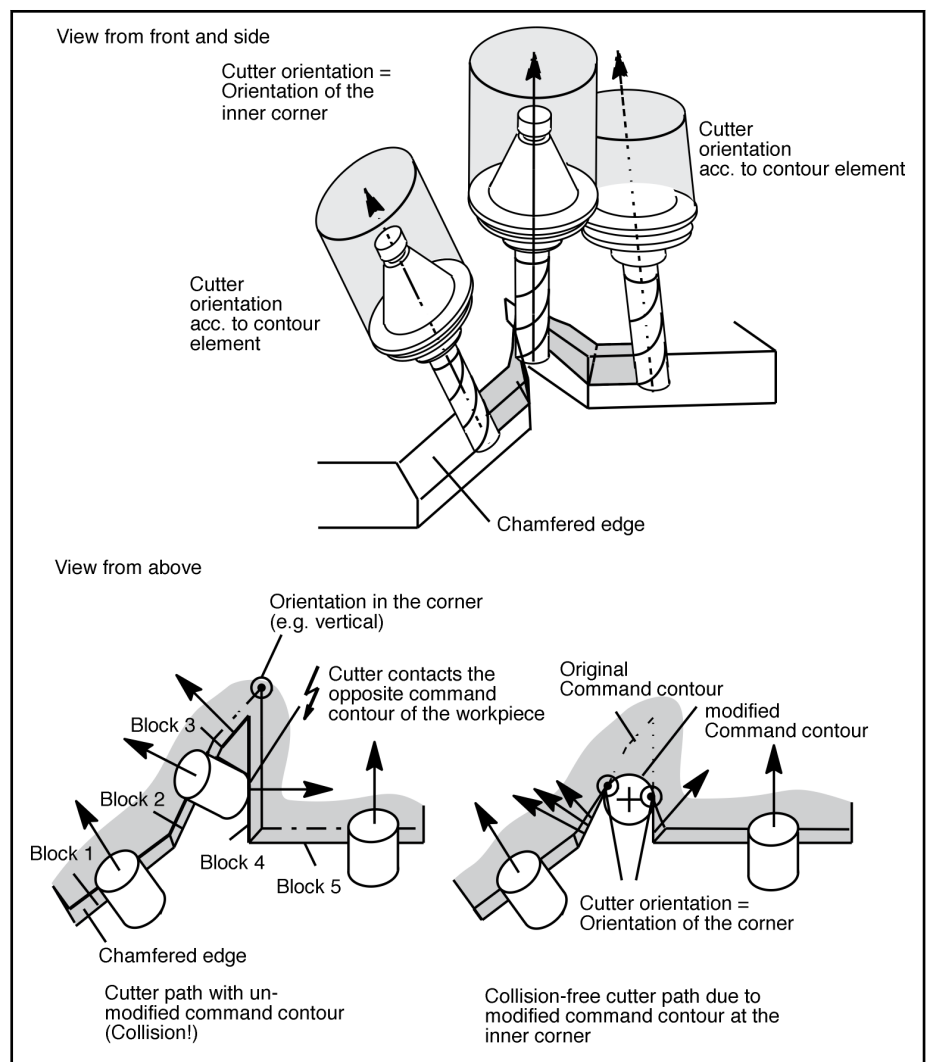


Fig. 14-60: Tool orientation

Collision monitoring: Contrary to the 2D-tool correction, there is generally no collision monitoring.

Path types: Contrary to the 2D-correction, the 3D-tool radius correction is not restricted to the path types "straight line" and "circle". It applies to all path types.

Coordinate classifications: The 3D-tool radius correction always machines in a space defined by the coordinates with the classification 1, 2 and 3.



If there is no coordinate classification 3 (normal axis), the plunging depth (INSDEP) cannot be used.

For more information, refer to the manuals "IndraMotion MTX Machine Parameters" and "IndraMotion MTX Programming Manual".

- Restrictions:**
- G141/G142 must not be entered into init string (MP 706000010 and 706000020)
 - With G141/G142 active, and in the NC block with G140, programming or changing zero offset is not permissible
Example: G140 G53 causes a run-time error
 - With active 3D-tool radius correction, the NC function "travel to fixed machine position" G74 and G76 causes a run-time error

- A coordinate transformation (inclined plane, workpiece position correction) must not be switched on or off or changed while a 3D-tool radius correction is active
- An axis transformation must not be switched on, off or changed while a 3D-tool radius correction is active
- While a 3D-tool radius correction is active, an axis transfer (GAX, WAX, RAX) may only be executed for axes not involved in the active space
- 2D (G41, G42) and 3D-tool radius correction exclude each other
- No collision is considered

Relevant NC functions:

G140	Deactivating 3D-tool radius correction
G141	Activating 3D-tool radius correction to the left of the path
G142	Activating 3D-tool radius correction to the right of the path
INSDEP	Plunging depth
COFFS	Contour allowance (offset path)
DCOFFS	Change rate of contour allowance
DcTSel, DCS	Activating D-correction table
DCT	Writing D-correction or tool correction value
D1... D99	Select D-correction (1 .. 99)
ED1-ED16	Selecting external tool correction

Tab. 14-18: Relevant NC functions

Relevant CPL functions:

MCODS(92,...)	Provides the basic coordinates of the tool tip

Tab. 14-19: Relevant CPL functions

Relevant interface signals:

iCh_G41G141Act	Function G41/G141 is modally active in the channel
iCh_G42G142Act	Function G42/G142 is modally active in the channel

Tab. 14-20: Relevant IF signals

Relevant machine parameters (MP):

7060 00120	Maximum number of blocks for block look-ahead: [5]: Look-ahead for 3D-tool radius correction Maximum number of NC blocks taken into account for inner corner calculation
7060 00130	Number of blocks for look-ahead in percentages: [5]: Look-ahead for 3D-tool radius correction Percentage of NC blocks taken into account for inner corner calculation
7010 00030	Coordinate meaning

Tab. 14-21: Relevant machine parameters (MP)

14.3.15 Handling instruction: Activating the 3D tool radius correction (G141/G142)

Applying: The following settings are to be made:

- See G41/G42.
- Machine parameters 7060 00120 [5] or 7060 00130 [5]:

The number (or the percentage number) of the NC blocks to be considered when computing the contact points at the inner corners.

Activating: Program G141/G142.

Other conditions:

- See G41/G42
- G140 is active at control startup. The plunging depth INSDEP, the contour allowance COFFS and its change rate DCOFFS are 0.

Deactivating: Program G140.

Other conditions:

- G140 is activated at channel reset or overall reset. The plunging depth INSDEP, the contour allowance COFFS and its change rate DCOFFS are set to 0.
- Program end without M30:
The part program has to end with G140. The plunging depth, the contour allowance and its change rate are not modified.
- Program end with M30:
G140 is activated. The plunging depth INSDEP, the contour allowance COFFS and its change rate DCOFFS are set to 0.

From the programmed tool action path, the programmed tool radius correction computes an offset path shifted by the tool radius. With the 3D tool radius correction, the orientation of the tool can change arbitrarily in space. Thus, the correction depends on the tool orientation in any directions in space.

IW Engineering/configuration: Editing parameters

NofBIPrep "Complete number of blocks for block preparation and interpolation" (7060 00110)

Maximum number of function-related blocks:

- **MaxBIVelCalc**
"Maximum number of blocks for jerk-limited velocity control" (7060 00120)
- **MaxBISpline**
"Maximum number of blocks for spline functions" (7060 00120)
- **MaxBITICorr3D**
"Maximum number of blocks for 3D milling cutter path correction" (7060 00120)

Distribution of the entire number of blocks in percentages:

- **PercVelCalc**
"Number of blocks for jerk-limited velocity control" in percent (7060 00130)
- **PercSpline**
"Number of blocks for spline functions in percentages" (7060 00130)
- **PercTICorr3D**
"Number of blocks for 3D milling cutter path correction" in percent (7060 00130)

	Documentation	Instruction
Instruction:		Editing machine parameters
Documentation:	MTX Machine Parameters	3D tool radius correction

Providing ED-correction

If necessary, the ED-correction has to be configured via the tool correction interface.

		Instruction chapter 14.3.7 "Handling instruction: Activating the ED-correction" on page 204
Instruction:		Applying ED-correction

Applying D-correction

If necessary, create and configure a D-correction table with the desired correction values.

		Instruction chapter 14.3.7 "Handling instruction: Activating the ED-correction" on page 204
Instruction:		Applying ED-correction

IW Operation/NC programming:

G141

3D tool radius correction **to the left of the path ON** (as seen in machining direction in case of positive correction values). Traversing motions or orientation motions may be programmed in the same block. The radius and the plunging depth are retracted if one of the active spatial coordinates is programmed in the same block. Otherwise, the control builds up the correction

- only with the next traversing block if **G140** was previously active.
- immediately (in active feed) if **G141/G142** was active before.

G142

3D tool radius correction to the **right of the path ON**. Otherwise identical to **G141**.



INSDEP<ET>

Defines the plunging depth <ET>. Acts modally.

Switched-on state after control startup: 0

COFFS<KA>

Defines the allowance values <CA> Acts modally and additively to tool radius r.

Switched-on state after control startup: 0

DCOFFS<DA>

Defines the change rate <DA> of the allowance. Acts modally.

Switched-on state after control startup: 0

	Documentation	Instruction
Documentation:	MTX Programming Manual	3D tool radius correction

14.3.16 Handling instruction: Deactivating the 3D tool radius correction (G141/G142)

Deactivating the 3D tool radius correction

IW Operation/NC programming:

G140 3D tool radius correction OFF (switched-on state after control startup).

The correction (radius, plunging depth) is reduced in an interpolating manner, if necessary with a programmed traversing motion (using 3rd-degree splines). Effective tool length correction is not influenced by **G140**.

	Documentation	Instruction
Documentation:	MTX Programming Manual	3D tool radius correction

15 Inch/metric switching

15.1 Basics

15.1.1 Description

Function

The control can switch the unit (inch and metric) used in the individual operation modes individually:

- For the NC modes "manual data input" and "automatic":
Via G70 (inch) and G71 (metric).
- For the NC modes "jogging" and "approach reference point":
Via the axis-based interface signal qAx_JogInch "Incremental step in inch".
In connection with "approach reference point", only the position display is switched to inch.

In the NC modes "Manual data input" or "Automatic", switching via G70/G71 causes the control to interpret all subsequently programmed translatory positions, velocities and accelerations of channel axes as well as the path-specific dimensions of the channel as inch units or metric units. Irrespective of G70/G71, all configuration data (e.g. machine parameters, system data, etc.) is always interpreted in the metric unit. This also applies to the position and velocity specifications of asynchronous axes which are likewise always interpreted in the metric unit.

The interpretation of compensation tables is also not affected by the G70/G71 switching.

When switching to the programmed unit, the control converts the currently indicated positions and feeds.

Depending on whether G70 or G71 is active in the channel, the CPL functions provide inch or metric values.

In the NC modes "Jogging" or "Approaching reference point", the set interface signal qAx_JogInch causes the control to interpret the four permanently stored incremental steps (1000, 100, 10, 1 increments) and the user-definable incremental step (x increments; MP 1015 00002) as inch unit with the scaling "0.0001 * (MP 1015 00001) inch". This exclusively affects synchronous and asynchronous linear axes. Rotary and endless axes are not affected by this switching.

Parameterized jog velocities are also not affected.



As the incremental steps are also used for the handwheel mode, the evaluation of a handwheel increment is also changed by switching with the interface signal.

Restrictions

- Switching via G70/G71 neither affects asynchronous axes (auxiliary axes) nor rotary and endless axes (i.e. all axes programmed in degrees).
- Switching via interface signal has no effect on rotary and endless axes (i.e. all axes programmed in degrees).
- Inch/metric switching via G70/G71 or via interface signal qAx_JogInch does not change the values derived from geometry or zero point offset tables.

Table input formats can be influenced separately as follows:

- By the MP 6020 00021 "Number of decimal positions for correction values (metric)",
- By the MP 6020 00022 "Number of decimal positions for individual values (inch)" or
- In the table editor using the toggle softkey INCH/METRIC.
- Inch/metric switching via G70/G71 or via the interface signal qAx_JogInch does not affect data specified by the PLC. Depending on the machine parameter 902000010, this data is always specified in mm/inch or in degree.
- The inch/metric switching has no effect on values displayed for the spindle or diagnostics.

Relevant NC functions

G70	Inch programming
G71	Metric programming

Tab. 15-1: Relevant NC functions

Respective CPL functions

MCODS(...)	Concerns all functions providing paths, positions, velocities and accelerations.

Tab. 15-2: Respective CPL Functions

Relevant IF signals

qAx_JogInch	"Incremental step in inch" (axis interface) Switches the unit in the NC modes "jogging" or "approach reference point" to "inch".
qAx_ManFeed_00	"Manual feed bit 0" (axis interface)
qAx_ManFeed_01	"Manual feed bit 1" (axis interface)
qAx_ManFeed_02	"Manual feed bit 2" (axis interface)
qAx_ManFeed_03	"Manual feed bit 3" (axis interface)

Tab. 15-3: Relevant IF signals

Relevant machine parameters (MP)

1015 00001	Axis resolution
1015 00002	Variable increment step.
6020 00001	Number of decimal places for feed display (metric)
6020 00002	Number of decimal places for feed display (inch)
6020 00011	Number of decimal places for axis display (metric)
6020 00012	Number of decimal places for axis display (inch)

Tab. 15-4: Relevant machine parameters

15.1.2 Handling instruction: Inch/metric switching in the manual data input or automatic mode

Switching the unit (inch/metric) in the NC operation modes "manual data input" or "automatic".

Automatic

- Program **G70** for inch unit
- Program **G71** for metric unit



The display unit can be set via the parameter DispUnit "Unit for the display" (6020 00030).

The inch/metric switching has no effect on values displayed for the spindle or diagnostics.

	Documentation	
Documentation:	MTX Programming Manual	Inch/metric switching

15.1.3 Handling instruction: Inch/metric switching to jog mode

Inch/metric switching in manual input mode by the PLC (jogging).

IW Engineering/IndraLogic: Set output signal

For "jogging" or "travel to reference point", the axis-related interface signal **qAx_JogInch** is to be set

- **qAx_JogInch = 0** → metric unit
- **qAx_JogInch = x** → inch (x = 1000, 100, 10, 1 increment(s))
as well as:
- the NC parameter **Dist** "Variable jog distance" (1015 00002)
- and -
the NC parameter **Incr** "Scaling of a jog increment" (101500001) scaling.



The interface signal **qAx_JogInch** does not influence the data specified by the PLC [CorrUnit "Unit for tables and compensations" (9020 00010)]

The inch/metric switching has no effect on values displayed for the spindle or diagnostics.

	Documentation	
Documentation:	MTX Programming Manual	Inch/metric switching

15.1.4 Handling instruction: Inch/metric switching of the user interface

Inch/metric switching for the display of the unit in the user interface.

IW Engineering/configuration: Editing parameters

Adjust **DispUnit** "Unit for the display" (6020 00030):

- **DispUnit** = automatically with G70/G71
- **DispUnit** = mm (for metric unit)
- **DispUnit** = inch (for non-metric unit)

Inch/metric switching

	Documentation	Instruction
Instruction:		Editing machine parameters
Documentation:	MTX Machine Parameters	Inch/metric switching

16 Spindle functions

16.1 Overview

The MTX supports the operation of up to 32 spindles. These spindles are described as system spindles. Up to eight system spindles can be assigned to one channel at a time. These spindles are described as channel spindles.

The system spindle name can be set via the machine parameter "SysSp-Name" (MP 1040 00401). If no name is specified, the MTX uses the names SSP01..SSP32.

The names of the channel spindles are: S1 to S8.

The assignment of channel spindles to system spindles is pre-assigned via the machine parameter "SplndChSp" (MP 702000020) and can be changed via the NC functions "GetSpindle", "RemSpindle" and "DefSpindle".

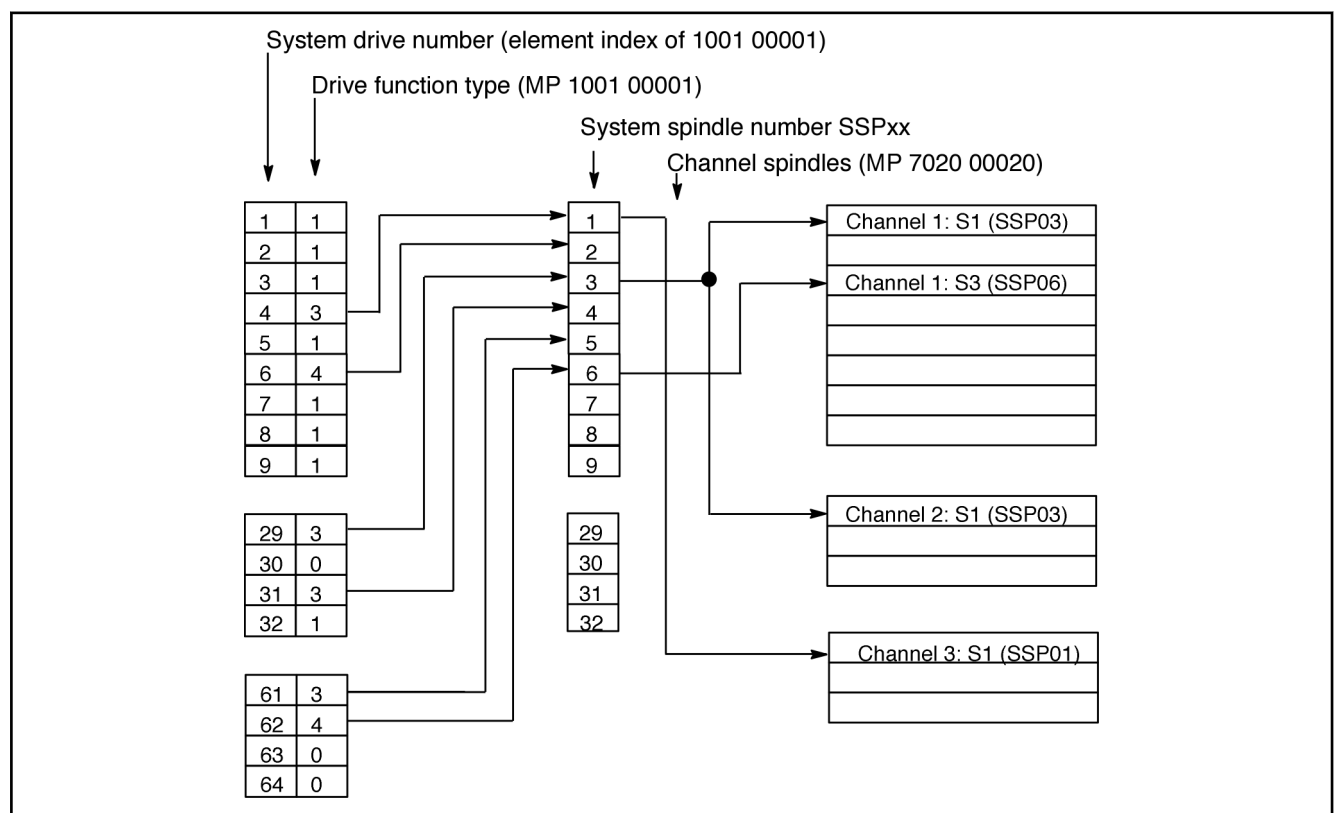


Fig. 16-1: Overview on spindle functions



There is a compatibility mode with earlier MTX versions (up to MTX V03).

If the machine parameter 104000999 is set to 0, the first eight system spindles are assigned to the channel spindles (S1 to S8) in all channels.

MP 702000020 is ignored, and it is not permissible to use the NC functions "GetSpindle", "RemSpindle" and "DefSpindle".

16.2 Channel spindles and channel spindle groups

16.2.1 Description

Function

The following default functionality applies to each spindle:

- Clockwise rotation, with/without coolant (M3/M13)
- Counterclockwise rotation, with/without coolant (M4/M14)
- Stop (M5)
- Speed programming (S, S1=,...)

Additional functions are provided for Sercos spindles:

- Positioning (orient) (M19)
- Automatic gear selection (M40)
- Manual gear selection (M41 - M44, M48)

Restrictions and special features

Automatic re-routing to the auto start channel:

When the PLC sends a manual data input block to an assigned channel, this NC block is automatically re-routed to the auto start channel. This option is not available in case of programming via auxiliary spindle function (M3, ...) or spindle S-word (S1=, ...), as the auxiliary channel has no information on the spindle assignment in the addressed channel.

The PLC has to use the NC functions to directly control the system spindles. For information, refer to [chapter 16.9 "Programming system spindles" on page 266](#).

Relevant NC functions

The following NC functions are related to channel spindles:

- G4 (S<Value>) Dwell time in revolutions of the main spindle (see MainSp)
- G33 Thread cutting
- G63 Tapping without compensating chuck
- G95 Feed programming in mm/revolution
- G96 Constant cutting velocity
- G97 Direct speed programming
- AxisToSpindle / ATS C-axis Off
- SpindleToAxis / STA C-axis on
- MainSp/MSP main spindle selection for G4, G33 and G95
- SMin / SMN Minimum value speed limitation
- SMax / SMX Maximum value speed limitation
- TappSp / TSP Additional function for tapping G63
- SpAdmin / SPA Spindle motion management
- SPCx Position-synchronous spindle mode

These functions cannot be used with all spindle types (see machine parameter SpType).

Examples:

1. G33 (thread cutting) requires a spindle with position feedback (SpType 1=Sercos, 4=external with encoder or 5=analog with encoder).
2. G63 (tapping without compensating chuck) requires a spindle with feedback and an additional position interface (SpType 1=Sercos).

Relevant machine parameters

MP 1040 001xx	The syntax of the motion functions is assigned to each channel spindle.

Tab. 16-1: Relevant machine parameters



The syntax of the spindle speed is specified implicitly:

1. Channel spindle: S1=

..

8. Channel spindle: S8=

16.3 Assigning system spindles to channel spindles

16.3.1 Description

Function

The assignment of channel spindles to system spindles is pre-assigned for each channel via the machine parameter 702000020.

During runtime, this assignment can be changed via NC functions.

Special features

Relation between auxiliary spindle function and system spindle from the PLC point of view:

If the channel axes are redefined (GetSpindle, RemSpindle, DefSpindle), there is no relation between auxiliary function and system spindle number. If the PLC requires information on the current assignment of channel spindle to system spindle, channel-specific auxiliary integer functions (CSP1 to CSP8) can be defined. The NC functions "GetSpindle", "RemSpindle" und "DefSpindle" output an auxiliary function for all newly assigned or redefined channel spindles (e.g. CSP1 17 for channel spindle 1 is assigned to system spindle 17).

Relevant NC functions (G-codes)

GetSpindle, GETSPINDLE, GSP GSP(<SysSpNr> <SysSpName> , <ChanSpNr> <ChanSpName> ,{...})	
	Apply system spindle to a channel.
RemSpindle, REMSPINDLE, RSP RSP(<SysSpNr> <SysSpName> <ChanSpName> , {...})	
	Remove channel spindle
DefSpindle, DEFSPINDLE, DSP DSP	
	Restore default state from MP

Tab. 16-2: Relevant NC Functions (G-Codes)

Spindle functions

Relevant auxiliary functions

CSP1 to CSP8	Channel-specific auxiliary integer functions

Tab. 16-3: Relevant auxiliary functions

Relevant CPL functions

SD(222)	Provides the channel spindle number of a system spindle of the active channel or -1:	
	SD(222, <SysSpindle>, 1)	Value at the time of block preparation.
	SD(222, <SysSpindle>, 2)	Value at the active point in time.
	With: <SysSpindle>	System spindle name or 1..32 = system spindle number
SD(223)	Provides the system spindle number of a channel spindle of the active channel or -1:	
	SD(223, <ChanSpindle>, 1)	Value at the time of block preparation.
	SD(223, <ChanSpindle>, 2)	Value at the active point in time.
	With: <ChanSpindle>	"S1".."S8" or 1..8 = Channel spindle number
SD(230)	Provides the system axis number of a system spindle or -1:	
	SD(230, <SysSpindle>)	
	With: <SysSpindle>	System spindle name or 1..32 = system spindle number
SD(231)	Provides the system axis number of a channel spindle or -1:	
	SD(231, <ChanSpindle>)	
	With: <ChanSpindle>	"S1".."S8" or 1..8 = Channel spindle number

Tab. 16-4: Relevant CPL functions

Relevant machine parameters

1040 00999	Enable channel spindles
7020 00020	Channel spindles: Assignment from Channel Spindles to System Spindles
3010 00010 [1]	Number of auxiliary functions (usable total)
3010 00010 [10]	Number of auxiliary functions (number of INTEGER-coded dependent auxiliary functions)
3010 00030	Names of auxiliary functions
3010 00040	Classes of auxiliary functions

Tab. 16-5: Relevant machine parameters

Example:

Relevant machine parameters

MP 7020 00020	[1] 1 channel spindle 1 is assigned to system spindle 1 [2] 15 channel spindle 2 is assigned to system spindle 15
MP 3010 000xxCSP1 and CSP2 are defined.	
GSP(SSP01,S1, SSP07, S2)	The system spindles 1 and 7 are assigned to the channel spindles 1 and 2. Channel spindle 2 is re-assigned: CSP2 7 is output.
GSP(SSP14,S2)	System spindle 14 is assigned to channel spindle 2. Channel spindle 2 is re-assigned: CSP2 14 is output.
RSP(S1)	No system spindle is assigned to channel spindle 1 (S1). Channel spindle 1 is re-assigned: CSP1 0 is output.

Tab. 16-6: Example - Relevant machine parameters

16.3.2 Applying

1. Set MP 1040 00999 to 1.
2. In machine parameter 7020 00020, set the default assignment of the channel spindles to the system spindles for each channel.
3. In MP 3010 000xx, configure the channel-dependent integer auxiliary functions CSP1 to CSP8, if desired.
4. Initiate control startup.

Compatibility

If MP 1040 00999 is equal to 0, the behavior of the MTX is compatible to MTX V02 and V03. The machine parameter 7020 00020 does not have any significance. The auxiliary functions CSP1 to CSP8 are not used either.

16.4 Assigning channel spindles to spindle groups

16.4.1 Description

Function

All eight channel spindles can be combined to up to four channel spindle groups.

Individual channel spindles can be combined to spindle groups. This is only an input tool.

Each spindle of a spindle group can be controlled via spindle group functions and single spindle functions.

Restrictions and special features

- At a spindle transfer (GSP, RSP, DSP), the assignment of channel spindle to system spindle changes. This assignment change directly affects the spindle group.
- Competing jobs between single spindles and spindle groups that are programmed in the same block lead to a runtime error (example: N10 M3 M104).

Spindle functions

- After control startup, the first channel spindle is a member of the first spindle group. The groups 2, 3, and 4 are empty.



There is a compatibility mode with earlier MTX versions (up to MTX V03). If the machine parameter 104000999 is set to 0, the spindle groups (for all channels) are pre-assigned via MP 104000002.

Relevant NC functions (G-codes)

SPG<group>(<numbers>)	
	Specifies which channel spindles are to be combined as spindle group in the current channel.
SPG<Group>(0)	Resets the group in the current channel to the default setting.
SPG<Group>(-1)	Cancels the corresponding spindle group in the current channel.
SPGALL(0)	Resets all groups in the current channel to the default setting

Tab. 16-7: Relevant NC functions

Example:

Relevant NC Functions (G-Codes)

GSP(SSP01, S1, SSP07, S2)	
	The channel spindles 1 and 2 are assigned to the system spindles 1 and 7.
SPG1(1,2)	Spindle group 1 consists of S1 and S2
M3	Starts S1 and S2 (SSP01 and SSP07) with clockwise rotation.
GSP(SSP14,S2)	System spindle 14 is assigned to spindle 2.
M4	Start S1 and S2 (SSP01 and SSP14) with counterclockwise rotation.
RSP(S1)	No system spindle is assigned to channel spindle 1(S1).
M3	Start S2 (SSP14) with clockwise rotation.

Tab. 16-8: Example - Relevant NC functions (G-codes)

Relevant machine parameters

MP 1040 002xx	The syntax of the motion functions is assigned to each spindle group.

Tab. 16-9: Relevant machine parameters



The syntax of the spindle speed is specified implicitly:

1. Spindle group: S or SSPG1=
2. Spindle group: SSPG2=
3. Spindle group: SSPG3=
4. Spindle group: SSPG4=

16.5 Motion functions

16.5.1 Description

General information and function

A channel spindle can be programmed as single spindle or as member of a channel spindle group in the part program or via manual data input with spindle functions.

Any number of single spindles and/or spindle groups can be programmed in one NC block.

The syntax for each spindle function of each spindle/spindle group is defined in the machine parameters (1040001xx and 1040002xx). In addition to the usual M-functions, it is also possible to assign any names (up to eight characters) to the individual functions.



In the following description of the individual spindle functions, it is not distinguished anymore between single spindles and spindle groups, since their behavior is identical. Essentially, programming a spindle group is only an shortened programming procedure.

The M-functions described below represent a recommended setting in the machine parameters. To provide a better overview, the default parameters are used.

Declaration for this documentation:

1. Spindle group M3, M4, M5, M19, M40, M41, S
1. Spindle M103, M104, M105, M119, M140, M141, S1 =
2. Spindle M203, M204, M205, M219, M240, M241, S2 = Spindle

Spindle clockwise rotation:

M3, M103, M203

The spindle(s) start(s) with clockwise rotation (seen from the spindle working area).

Select the speed via the corresponding S-address.

The speed can be programmed in one block together with M3.

The function remains **modally active** until it is canceled by another command for the same spindle(s).

That means that the previously effective motion state is restored, for example after the gear level was changed.

Spindle clockwise rotation and coolant ON:

M13, M113, M213

As under M3, M103, M203, but activate additional coolant.

Spindle counterclockwise rotation:

M4, M104, M204

The spindle starts with counterclockwise rotation (seen from the spindle working area). Otherwise, as under "clockwise spindle rotation".

Spindle counterclockwise rotation and coolant ON:

M14, M114, M214

As under M4, M114, M214, but activate additional coolant.

Spindle stop:

M5, M105, M205

The spindle is stopped. This command remains effective until it is cleared by a different spindle command.

Programmable spindle orientation (orient spindle):

M19, M119, M219

The spindle moves to a specific position.

This function can be executed from standstill condition and from a rotational motion.

The NC sends the information to the drive (S-0-0152 "Spindle positioning command", S-0-0153 "Spindle angle position"). The drive performs the function according to its configuration (see: S-0-0154 "Spindle positioning parameter", S-0-0222 "Positioning spindle speed").

As soon as a different spindle motion task (M3, M4, M5) is active, the "spindle orient" function is completed.

The function can be programmed either separately or together with other M- or G-commands.

The "spindle orient" function can be programmed with or without the corresponding S-word in a block:

- Programming without S-word:
The spindle moves to the angular position of 0 degree.
- Programming with S-word (= positioning angle in degree)
The spindle positions to the specified angle. If an angle is programmed outside the interval [0_ = positioning angle < 360_], the control transforms this angle to the interval permitted. Thus, the spindle never traverses more than one revolution.

Examples: Programmable spindle orientation

N... M19	The spindles of the first spindle group position to 0 degree.
N... M19 S180	The spindles of the first spindle group position to 180 degrees.
N... M219 S2=370	The second spindle positions to 10 degrees.
N... PTEST10 M119	The first spindle positions to 0 degrees and the "TEST10" sub-routine is subsequently executed.

Tab. 16-10: Example - Programming spindle orientation (orient spindle)

16.6 Gear function

16.6.1 Description

Function

The machine-specific speed range is divided into individual speed ranges by shiftable gears or switchable windings. Individual gear levels can be defined for the individual gear ranges.

The number of gear levels (up to four), their speed limits (min./max. velocity) as well as further specific spindle parameters are defined in the machine parameters under the group 1040.



The NC uses the Sercos parameter blocks 0 (S-0-xxx) for the 1st gear stage up to 3 (S-3-xxx) for the 4th gear stage. The spindle drive used has to support switchable parameter blocks. For more detailed information, refer to the drive documentation.

In special cases, the functions "Tapping with G63 and position-controlled spindle mode" and the C-axis mode can be assigned individual drive parameter blocks. For more detailed information on the machine parameter 1040 00009, refer to the manual "MTX Machine Parameters".

The functions to select gears do not affect analog spindles.

Automatic gear selection: M40, M140, M240

The gear stage selection is programmed once at the beginning of the part program using an M-function.

By means of the programmed speed, the control selects the appropriate level from a maximum of four gears configured in the machine parameters.

The following applies:

- If the speed ranges of individual levels overlap, the control always selects the next lower level (with the higher motor speed).
- If a speed of "0" is programmed, the gears are not changed.

The automatic gear level selection can be defined as switched-on state in the machine parameter 7060 00020.



The automatic gear selection is only a channel property!

Manual gear selection: M41-M44, M141-M144, M241-M244

The manual gear selection is programmed in the part program as required.

If necessary, the gear level (1-4) desired for each spindle/spindle group can be specified manually in the part program.

If, for a manual gear selection, a speed outside the speed gear range is programmed, the NC outputs the minimum or maximum speed of the gear in question.

Disengage gear M48, M148, M248

Disengages the neutral gear of the spindle/spindle group. The gears of the corresponding spindle(s) is (are) then in idle mode.

Restrictions and special features

- Gear functions are only supported for Sercos spindles.
- For active winding switching (see MP 1040 00008 or SP/SpFunc/Sp[1..32]/Gear/GearChgMethod), refer to [chapter 16.14 "Gear shifting with winding switching" on page 284](#) for additional information.
- If the winding switch is enabled, the function "Disengage gear" (e.g. M148) is not supported.
- Gear shifting is executed asynchronously to the actual part program. Any necessary synchronization has to be initiated by an application.

16.7 Path motions and spindle functions

16.7.1 Description

Function

The waiting conditions can be enabled in connection with spindle functions. These conditions delay the start of a path motion or the exiting of an NC block.

A predefined strategy is available. The machine tool manufacturer can additionally define two strategies via the configuration parameters.

It can be additionally waited for a certain condition at the end of the NC block using the NC function "SpindleWait / SPWAIT".

Waiting can be activated by the following **spindle tasks**:

M3/M4	Each change to M3 or M4. M3 with S-word for active M3. M4 with S-word for active M4.
M5	Each change to M5.
M19	Each change to M19 and each new specification of M19.
M4x	Each gear shift.

The derived **waiting conditions** are:

M3/M4	Wait for "Speed reached" at active clockwise and counterclockwise rotation.
M5	Wait for "Spindle stop" at active stop.
M19	Wait for "Spindle oriented" at active spindle orientation.
M4x	Wait till gear shifting completed.

The possible **waiting points in time** are:

0:	Do not wait.
1:	Wait at beginning of block. The path motion is only started after reaching the waiting condition.
2:	Wait at end of block. The NC block is only exited after reaching the waiting condition.

For the waiting point in time "When exiting the positioning mode", it is waited at the beginning of the first G1/G2/... block.

Fixedly set strategies

Strategy	Waiting condition	Waiting point in time		
		Positioning mode (G0)	When exiting the positioning mode	Edit (G1, G2, ..)
0	All	0	0	0

Strategy	Waiting condition	Waiting point in time		
		Positioning mode (G0)	When exiting the positioning mode	Edit (G1, G2, ..)
1	M3/M4	0	2	0
	M5	0	2	0
	M19	0	2	0
	M40	0	2	1

Tab. 16-11: Fixedly set strategies

Strategy 0 never waits.

In case of strategy 1, it is waited that the programmed spindle state is reached when exiting the positioning mode. In case of an active processing, the path motion is stopped if there is a gear shift.

Strategies settable by the machine manufacturer

The strategies 2 and 3 can be configured by the machine builder if required.

These strategies can be set via the following configuration parameters:

- SP/SpAdmin/BlkSwiCtrl/WaitingPoints/VarStrategy2/... (MP 1040 00300)
- SP/SpAdmin/BlkSwiCtrl/WaitingPoints/VarStrategy3/... (MP 1040 00301)

Strategy	Waiting condition	Waiting point in time		
		Positioning mode (G0)	When exiting the positioning mode	Edit (G1, G2, ..)
2	M3/M4	0	2	1
	M5	0	2	0
	M19	0	2	0
	M40	0	2	1
3	M3/M4	0	2	1
	M5	0	2	0
	M19	0	2	1
	M40	0	2	1

Tab. 16-12: Strategies that can be set by the machine manufacturer

Display active waiting state

A delay in the part program due to an unfulfilled waiting condition is displayed via the channel waiting state "Wait for spindle".

This state is additionally displayed in the system date

"/SysChSpActData[k]/WaitForSpindle"



- Each relevant spindle programming in a part program causes a standstill of the path motion if a waiting condition is activated.
- In case the spindle administration is disabled (SP/SpFunc/Sp[i]/SpCtrl/Ch[k]/EnablChCtrl), detecting a relevant spindle programming is not possible anymore.
Each spindle programming (e.g. M3 for an already active M3) in a part program causes a standstill of the path motion if a waiting condition is activated!

Special features

- Switching to next block
It is not waited. Active waiting is canceled.
- Premature block switching
It is not waited. Active waiting is canceled.
- The functions described can be used for all spindles with feedback. These are:
 - Sercos spindles
 - External spindles with encoder
 - Analog spindles with encoder

Relevant NC functions

- Selecting and activating a waiting strategy "SpindleWaitMode / SWM"
- Wait at block end for reaching a condition "SpindleWait / SPWAIT"

Relevant machine parameters (MP)

The variable strategies are set via the following configuration data:

MP 1040 00300 SP/SpAdmin/BlkSwiCtrl/WaitingPoints/VarStrategy2/...	
	Waiting points for block switching control with Strategy 2.
MP 1040 00301 SP/SpAdmin/BlkSwiCtrl/WaitingPoints/VarStrategy2/...	
	Waiting points in time for block switching control with Strategy 3.

Tab. 16-13: Relevant machine parameters (MP) - Setting variable strategies

The strategy used can be selected via the following configuration data:

MP 7020 00030 SP/SpAdmin/BlkSwiCtrl/Ch[1..60]/Sp_ChSp[1..8]/SelBlkSwiStrategy	
	Strategy determination for block switching control: Selection of block switching strategy

MP 7020 00040 SP/SpAdmin/BlkSwiCtrl/Ch[1..60]/Sp_ChSp[1..8]/EnablBlkSwiStrategy	
	Strategy determination for block switching control: Enable block switching strategy

Tab. 16-14: Relevant machine parameters (MP) - Selecting variable strategies

Relevant system data

A delay in the part program procedure is displayed in the "/SysChSpActData[k]/WaitForSpindle" system date.

This system date consists of the elements:

Active	Waiting active	
ChanSpNo	Number of a channel spindle whose waiting condition is not fulfilled.	
SysSpNo	System spindle number of the channel spindle	
Move	3:	Wait for speed reached (iSp_ProgSpReach).
	4:	Wait for speed reached (iSp_ProgSpReach).
	5:	Waiting for spindle stop (iSp_Stop).
	19:	Wait for spindle orientation finished (iSp_OrientateFinish).
Gear	1:	Wait for the end of a programmed gear shift.
	2:	Waits for the end of a automatic gear shift.
	3:	Waits for the end of a tool-specific limitation switching.

Tab. 16-15: Relevant system data

If "/SysChSpActData[k]/WaitForSpindle/Active" is set, the data refers to the current state. Otherwise, the data of the last spindle-caused delay is entered.

Examples

Milling machine

Spindle default:

- It is not waited for the spindle in positioning mode.
- When exiting the positioning mode, the latest programmed motion function has to be active.
- During machining, if switchable gear shifts or windings are available, wait at the beginning of the block until gear shifting is completed.

This means:

- Strategy 1 is entered into the machine parameter 7020 00030 for channel spindle 1.
- The waiting strategy is entered into the machine parameter 7020 00040 for channel spindle 1.

The waiting logic is switched on and off in the tool change program:

- The waiting logic is switched off with SWM(S1,0) at the beginning of the tool change.
- The waiting logic is switched on with SWM(S1,1) at the end of the tool change.

Lathe

Default of the main spindle:

- It is not waited for the spindle in positioning mode.
- When exiting the positioning mode, the latest programmed motion function has to be active.
- During machining, wait while changing the direction of rotation, during spindle positioning and gear shift.

Default of the milling spindle:

- It is not waited for the spindle in positioning mode.
- When exiting the positioning mode, the latest programmed motion function has to be active.
- During machining, wait while changing the direction of rotation and during spindle positioning.

This means:

- Channel spindle 1 is entered into the machine parameter 7020 00010 as main spindle for channel 1.
- Strategy 3 (not modified) is entered into the machine parameter 7020 00030 for the channel spindles 1 and 2.
- The waiting strategy is entered into the machine parameter 7020 00040 for channel spindle 1.

The waiting logic of the milling spindle is switched on and off in the tool change program:

- The waiting logic is switched off with SWM(S2,0) at the beginning of the tool change.
- At the end of the tool change, the waiting logic is activated for a milling spindle with SWM (S2,1) if a milling or drilling tool is changed.

16.7.2 Applying

A fixedly determined strategy and two variable strategies are provided. If they meet the machine requirements, no setting has to be made.

If required, both variable strategies

(SP/SpAdmin/BlkSwiCtrl/WaitingPoints/VarStrategy2)

can be adjusted in the configuration parameters.

16.7.3 Activating

Directly via configuration parameters:

One strategy set in the configuration parameter

"SP/SpAdmin/BlkSwiCtrl/Ch[1..60]/Sp_ChSp[1..8]/SelBlkSwiStrategy"

can be activated with a corresponding entry in the parameter

"SP/SpAdmin/BlkSwiCtrl/Ch[1..60]/Sp_ChSp[1..8]/EnablBlkSwiStrategy"

.

In the part program:

The NC function "SpindleWaitMode" allows switching on and off the selected strategy and changing the strategy.

16.7.4 Disabling

Set the configuration parameter

"SP/SpAdmin/BlkSwiCtrl/Ch[1..60]/Sp_ChSp[1..8]/SelBlkSwiStrategy"
to "0" for all spindles.

16.8 Spindle speed specification

16.8.1 Description

Function

The spindle speed relates to a single spindle or to all members of a spindle group.

Speed specification:

$S_i = \dots S_n =$

$SSPG_j = \dots SSPG_n =$

S

With:

$S_i =$ Speed specification for the i-th spindle(s)

i Index of the i-th spindle ($i = 1..n$)

n Number of available spindles ($n_{max} = 8$)

$SSPG_j =$ Speed specification for the j-th spindle group

y Index of the j-th spindle group ($j = 1..m$)

m Number of available spindle groups ($m_{max} = 4$)

S Shortened writing for the speed of the first spindle group

For the spindle speed specification, the following applies:

- By default, the programmed speed values are interpreted as rpm.
- When G96 is active, the programmed speed is interpreted as cutting velocity in m/min.
- The programmed speed value can be influenced via the spindle-specific override. The override position of 100% corresponds to the programmed speed value.
- The control always limits the command speed value output so that the limit values entered into the machine parameters are maintained. Note that the limit values depend on the gear selected.
- The speed can be additionally limited by programming SMin / SMX or SMax / SMX.
- The set speed value is applied until it is overwritten by a new "S-word" (acting modally).
- Examples:
 N..G97 Speed programming active
 N.. G.. X.. Y.. Z.. F.. SSPG1 = 1000
 The spindles of the first spindle group are to rotate at 1000 min-1.
 N.. G.. X.. Y.. Z.. F.. S1=2000 Speed of 1st spindle: 2000 min-1
 N.. G.. X.. Y.. Z.. F.. S3=2000 Speed of 3rd spindle: 2000 min-1
 N.. G.. X.. Y.. Z.. F.. S1500 The first spindle group is to rotate at 1500 min-1.
- By setting the interface signal "qSp_SValueSD", the speed is specified via the system data "SysSpCmdData[n]/Speed". The programmed specification via S, $S_x =$, $SSPG_x$ is ignored.



Faulty programming can cause damages to the machine!

Avoid: Note the programming!



Where the "spindle positioning" function (M19, ...) is concerned, the control interprets the S-word as positioning angle, not as speed!

The meaning of the S-word (speed/cutting velocity) is defined by G97/G96!

Special feature:

SysSpCmdData[n]/SpeedGearRatio:

- The command speed specification for the drive is multiplied with the content of the system date.
- The actual speed of the drive is divided by the content of the system date before the NC-internal processing takes place. S-0-0040 "Actual velocity value" returns the original drive value.

Restrictions

If the interface signal "qSp_SValueSD" is set, the following restrictions apply to the spindle functionality:

- G96 does not work!
Only an S-word programmed together with G96 in one channel activates the constant cutting velocity for one spindle.
- G94(DS...) "Incremental spindle speed programming" does not work.
- M40 does not work!
In case of an automatic gear shifting, the gear step is determined using the programmed S-word from the channel.
- SD(205,2,x) (programmed spindle speed) does not provide a practical value.

Relevant system data

SysSpCmdData[n]/Speed

SysSpCmdData[n]/SpeedGearRatio

Relevant IF signals

qSp_SValueSD

16.9 Programming system spindles

16.9.1 Description

General information

Up to 32 system spindles have the following standard functionality:

- Clockwise rotation
- Counterclockwise rotation
- Stop
- Positioning (orient)
- Manual gear selection
- Speed programming

- Speed limit
- Control commands for spindle motion management
- Switching command for Sercos interface

Exception "qSp_SValueSD":

- By setting the interface signal "qSp_SValueSD" the speed is specified via the system date "SysSpCmdData[n]/Speed" and the position is specified via the system date "SysSpCmdData[n]/OriPos". The programmed specifications are ignored via SSpSpeed and SSpOri.

Exception "SysSpCmdData[n]/SpeedGearRatio":

- The command speed specification for the drive is multiplied with the content of the system date.
- The actual speed of the drive is divided by the content of the system date before the NC-internal processing takes place. S-0-0040 "Actual velocity value" returns the original drive value.

Restrictions and special features

- The motion functions can only be used for NC-controlled spindles (SpType 1=Sercos, 3=analog or 5=analog with encoder).
- Commands for positioning and gear selection are only available for Sercos spindles (SpType 1=Sercos).
- If the interface signal "qSp_SValueSD" is set, the following restrictions apply to the spindle functionality:
SD(205,2,x) (programmed spindle speed) does not provide a practical value.

Relevant NC functions (G-codes)

To allow for general activation of system spindles throughout the system, additional NC functions are provided. These functions are only subject to the restrictions of the spindle management (see chapter "Motion Management of System Spindles").

When programming system spindles, the names entered in the machine parameter SysSpName (MP1040 00401) or the internal names SSP01..SSP32 can be used.

The abbreviation "SSp" stands for SystemSpindle.

SSpSpeed, SSPSPEED, SSPS SSpSpeed(<SSpNr.> <SysSpName>, <Speed> { , ... })	
	Speed programming
SSpMove, SSPMOVE, SSPM SSpMove(<SSpNr.> <SysSpName>, <Task> { , ... })	
	Motion programming in clockwise rotation (M3), counterclockwise rotation (M4) and Stop (M5)
SSpOri, SSPORI, SSPO SSpOri(<SSpNr.> <SysSpName>, <Phase> { , ... })	
	Positioning (M19)

SSpMin, SSPMIN SSpMin(<SSpNr.> <SysSpName>, <Speed> { , ...})	
	Limit minimum speed (SMin)
SSpMax, SSPMAX SSpMax(<SSpNr.> <SysSpName>, <Speed> { , ...})	
	Limit maximum speed (SMax)
SSpGear, SSPGEAR SSpGear(<SSpNr.> <SysSpName>, <Stage> { , ...})	
	Shift gear (M4x)
SSpAdm, SSPADM SSpAdm(<SSpNr.> <SysSpName>, <Mode> { , ...})	
	Enable/apply spindle (SpAdmin)
SSpAdmOff, SSPADMOFF SSpAdmOff(<SSpNr.> <SysSpName>, <Channelno.1> { , <Channelno.2>{ , ...}}) SSpAdmOff(<SSpNr.> <SysSpName>,)	
	Program spindle management
SSpMode, SSPMODE SSpMode(<SSpNr.> <SysSpName>, <Mode> { , ...})	
	Switch operation mode (SpMode)

Tab. 16-16: Relevant NC Functions (G-Codes)

Relevant system data

SysSpCmdData[n]/Speed
SysSpCmdData[n]/SpeedGearRatio
SysSpCmdData[n]/OriPos

Relevant IF signals

qSp_SValueSD

16.10 Actuating the system spindle via interface

16.10.1 Description

Function

System spindles can be actuated via the bit interface.
The following spindle functions can be specified:

- Manually M3, Manually M4, Manually M5, Manually M19
- Jogging M3, Jogging M4

For the function "Manual M19" (qSp_Orientate), the spindle position is specified via the system date "SysSpCmdData[n]/PlcOriPos".

Restrictions and special features

- Spindles (SpType 2=external and 4=external with encoder) not controlled by the NC cannot be moved via the interface.

- To control a spindle via the interface (spindle manual or spindle jogging), the following boundary conditions apply:
 - The system spindle is in stop state (M5).
 - or –
 - The spindle was started by a part program that is stopped with feed hold and not used by G63 (tapping without compensating chuck).
 - or –

The motion management is switched off for the system spindle (see [chapter 16.11 "Motion management of system spindles" on page 269](#)).



If a system spindle is operated in a channel with constant cutting velocity (G96), jogging M3 and jogging M4 are **not permitted**.

If jogging M3 and jogging M4 is specified, a potentially existing lower speed limit is ignored.

16.11 Motion management of system spindles

16.11.1 Description

General information

Each system spindle can be activated from any channel.

If a motion for a spindle is specified in a channel via the part program or MDI, the corresponding spindle is reserved for this channel in the motion management. It is irrelevant whether the motion is specified via a channel or system spindle programming.

A reserved spindle is disabled for a "third-party" channel, i.e. a runtime error is output if it is tried to be controlled from a different channel

Exceptions:

- Conditional spindle enabling,
Refer to the programming with SpAdmin/SPA or SSpAdm.
- Absolute enabling of a spindle for several channels,
Refer to SSpAdmOff and MP 104000070.

Until a reserved spindle is enabled, it can only be controlled by the channel (possessor channel) that triggered the reservation. The spindle is enabled again for the other channels if it is stopped by its assigned channel.

16.11.2 Activating reservation in the channel

The following spindle functions result in an **implicit** reservation of the spindle via the calling channel:

- Spindle clockwise rotation (M3, M13, SSpMove)
- Spindle counterclockwise rotation (M4, M14, SSpMove)
- Orient spindle (M19, SSPOri)
- G63 (tapping without compensating chuck)
- G96 (constant cutting velocity)

Any spindle functions can be programmed in the channel reserved by the spindle:

Spindle functions

- Speed programming S, S1 - S8, SSPG1 - SSPG4
- M3, M13, M4, M14, M5, M19
- M40, M41-M44, M48
- G63
- SMin/SMN, SMax/SMX
- G96

It is prohibited for a "third-party" channel to control a reserved spindle. Such an attempt generates the runtime error 2001:

"The spindle is assigned to a different channel!"

The following functions are affected by this restriction:

- Speed programming S, S1 - S8, SSPG1 - SSPG4, SSpSpeed
- M3/M13, M4/M14, M19, M5, SSpMove, SSpOri
- M41-M44, M48, SSpGear
- G63
- SMin/SMN, SMax/SMX, SSpMin, SSpMax
- G96

The following functions are suppressed and thus do not generate a runtime error when they are called from the init string (MP 7060 00020) of a third-party channel:

- M5
- M40, M41-M44, M48
- G96, G97

16.11.3 Enabling reservation of a "spindle in the channel"

A reserved spindle is enabled again in the possessor channel:

- by programming M5 or SSpMove(...,5)
- at the end of G63 provided the spindle was reserved by G63.

Exiting the part program with M30 or a reset in the possessor channel has the following effect:

- Deselection of M40 (if active), current gear step remains selected
- Deselection of G96, speed programming becomes active
- Activation of the functions entered into the init string (MP 7060 00020):
 - M5
The spindle is stopped if M3/M13, M4/M14 was active or the spindle was positioned with M19. Thereafter, the spindle is enabled.
 - M40
Selection/reselection of automatic gear selection
 - M41-M48
Automatic gear selection could already have been deselected (see above). Any manual gear selection in the init string does not cause a gear shift.

16.11.4 Transferring a reserved spindle to another channel

In exceptional cases, it can be necessary that a spindle reserved by M3, M4 or M19 is activated by a neighboring channel (part program or MDI).

In such cases, the NC functions "SpAdmin/SPA" and "SSpAdm" allow the channel currently "possessing" the spindle to transfer the reservation of the spindle to a different channel.

16.11.5 Conditional spindle enabling

SpAdmin(Si=0,...,Sn=0)

A channel which currently reserved one or more spindles by programming M3, M4 or M19 uses this command to grant the right for the application to any other channel:

- The spindle can now be applied by any other channel with the command SpAdmin/SPA.
- The spindle can be activated at any time via switching functions.

16.11.6 Applying a conditionally enabled spindle using a different channel

SpAdmin(Si=1,...,Sn=1)

This command is used to apply conditionally enabled spindles from a different channel using the part program or MDI.

16.11.7 Switching off spindle motion management for specified channels

If random access of multiple channels to one system spindle is required, the motion management can be deactivated via the machine parameter 1040 00070 or by programming "SSpAdmOff".

Programming

SSpAdmOff(<SSpNr.>|<SSpName>, <ChannelNo.> {, <ChannelNo> } ...)

With:

SSpNo	1..32
SSpName	System spindle name
ChannelNo.	1 .. 60 channel numbers
	0 Enabled for tasks from the bit interface

Special features

- Compared to the remaining SSp function, this function can affect only one system spindle at a time.
- A specification by channel 0 disables the motion management of this system spindle for specifications by the bit interface.

16.12 Special spindle functions

16.12.1 Main spindle switching

Switches the main spindle.

The functions G33, G95 and G4(S<Wert>) act on the main spindle of a channel.

Which spindle is to be the main spindle can thus be defined either

- statically in the machine parameters via MP 702000010
- or -
- dynamically in the part program via the MainSp / MSP function.

MAINSP<Num> - or - MAINSP(<NumNam>) With:	
<Num>	is the spindle number: any number between 1 and 8. -1: Return to the machine parameter setting.
<NumNam>	is the spindle number: any number between 1 and 8 -or- Spindle name (e.g. S1) -or- -1: Return to the machine parameter setting.

Tab. 16-17: Main spindle switching

16.12.2 Spindle functions with position interface

In the normal mode, a spindle is always operated via the speed interface.

In **exceptional cases** (e.g. "position-synchronous mode of spindles"), the spindle has to be activated via the **position interface** (position control).

The following functions are available for **operation with position interface**:

- Clockwise spindle rotation (see [chapter "General information and function" on page 257](#))
- Counterclockwise spindle rotation (see [chapter "General information and function" on page 257](#))
- Spindle stop (see [chapter "General information and function" on page 257](#))
- Orient spindle (see [chapter 16.12.2 "Spindle functions with position interface" on page 272](#))
- Position-synchronous mode (see [chapter 16.12 "Special spindle functions" on page 271](#) and [chapter 16.12.2 "Spindle functions with position interface" on page 272](#))

Referencing the spindle: With the activated position interface, the reference point of the spindle has to be known for the following **functions**:

- Orient spindle
- Position-synchronous mode

The reference point of the spindle drive can be determined in two ways:

- Spindle positioning (M19) at active speed interface
- Approaching reference point in c-axis mode.

Switching to position-controlled spindle:

If position-controlled spindle mode is intended, the spindle has to be switched from the speed interface to the position interface.

Manual switching of the drive interface: SpMode (SPM)

- Switching to the position interface:
If necessary, the programmed spindles are stopped and, after switching the interface, restarted. If "orient spindle" (M19) is activated when being switched, it is aborted and switched to "spindle stop" (M5).

- Switching to the speed interface:

It is directly switched to speed interface for the programmed spindles. If "orient spindle" (M19) is activated when being switched, it is aborted and switched to "spindle stop" (M5).

SpMode(Si=0 1 ... Sn=0 1)	
With:	
Si	switches the drive interface of the ith spindle(s) to: Si=0: Speed mode - or - Si=1: Position mode
n	Number of available spindles (currently, n _{max.} = 8)
i	Index for the i-th spindle (i=1..n)

Tab. 16-18: Manual switching of the drive interface



The operation mode of a main spindle can also be switched with ThreadSet / TST (TST(SPC 0|1)).

Main spindle, see [chapter 16.12.1 "Main spindle switching " on page 271](#).

Attention:

The "ThreadSet" and "SpMode" functions are not locked against each other.

16.12.3 Position-synchronous spindle operation

The position-synchronous spindle operation is mainly required at lathes (e.g. to clamp a workpiece/tool in two opposite spindles, to transfer workpieces).

Thus, multiple spindles have to run at the same time and in position-synchronous spindle operation (spindle coupling).

Definition of spindle coupling

Up to **four different spindle couplings** can be controlled simultaneously:

- A spindle coupling consists of a master spindle and up to seven slave spindles (slaves).
- The slave spindles can be coupled to the master spindle with any offset angle ranging from 0 to 359.9999 degrees (coupling distance).
- The slave spindles can be coupled to the master spindle with the current position when the master velocity is reached (offset angle = ACT).
- If coupling is active, each slave spindle can rotate up to 3600° degrees (absolute) relative to its coupling distance.
- Slave spindles can be added and removed if the coupling is active.

The coupling law for slave spindle positions is:

$$p_s = p_m * k + o_{dist} + o_{off}$$

p_s Command position of slave spindle

p_m Command position of master spindle

k Direction of rotation from slave spindle to master spindle (1,-1)

Spindle functions

O_{dist} Preset coupling distance between master axis and slave axis (SPCD)

O_{offs} Additive dynamic offset to the coupling distance (SPCP)

The following **boundary conditions** have to be fulfilled for the spindle coupling:

- All members of the coupling group have to have a common speed range.
- The members of the coupling group have to have similar dynamics.

Restriction for the position-synchronous mode:

The maximum speed of a coupling network depends on the NC cycle time (MP 9030 00001):

$$S_{\max}[\text{min}^{-1}] = 28800 / \text{NC cycle time [ms]}$$

Example:

NC cycle time = 2 ms

$$S_{\max} = 28800/2 = 14400 \text{ min}^{-1}$$

Configuring slave spindles:

Coupling distance: SpCoupleDist / SPCD

The coupling distance defines the position difference of the command values between the master spindle and the slave spindle(s) starting at the coupling time.

Setting the coupling distance of one or more slave spindles:

SpCoupleDist (Si=<Distance i>, ..., Sn=<Distance n>)	
With:	
Si	i-th slave spindle(s)
Distance i	Position difference of the command values of the i-th slave spindle(s), starting at the coupling time
n	Number of available spindles (currently, nmax.= 8)
i	Index for the i-th spindle (i=1..n)
Value range (distance):	-359.9999 ... +359.9999 degree or ACT - for the velocity-synchronous coupling without position adjustment
Default:	0 degree
Validity:	Spindle-specific date

Tab. 16-19: Setting the coupling distance of one or more slave spindles

Synchronous window: SpCoupleSyncWin / SPCS

At the beginning of spindle synchronization, the NC waits until the deviation of the actual position values from the command values for the particular slave spindles reached the interval defined by the window [-value, +value]. This window is monitored when the synchronization mode is active.

In case of an error, the "interface signal position-synchronous 1" is deleted from the spindle-specific output interface.

Definition of synchronization window:

SpCoupleSyncWin (Si=<Window i>, ..., Sn=<Window n>)	
With:	
Si	i-th slave spindle(s)
Window i	Specification for the synchronization window of the i-th slave spindle(s)
n	Number of available spindles (currently, nmax.= 8)
i	Index for the i-th spindle (i = 1..n)
Value range (window):	0 .. 20 degrees
Default:	1 degree
Validity:	Spindle-specific date

Tab. 16-20: Definition of the synchronization window SpCoupleSyncWin / SPCS

Synchronization error window: SpCoupleErrWin / SPCE

This window is monitored when the synchronization mode is active. In case of an error, the "interface signal position-synchronous 2" is deleted from the spindle-specific output interface.

Definition of synchronization error window:

SpCoupleErrWin (Si=<Window i>, ..., Sn=<Window n>)	
With:	
Si	i-th slave spindle(s)
Window i	Specification for the synchronous window of the i-th slave spindle(s)
n	Number of available spindles (currently, nmax.=8)
i	Index for the i-th spindle (i = 1..n)
Value range (window):	0°.. 359.9999°
Default:	10°
Validity:	Spindle-specific date

Tab. 16-21: Definition of synchronization error window SpCoupleErrWin / SPCE

Definition of coupling networks:

Establishing, changing or resolving the coupling: SpCoupleConfig / SPCC

The SpCoupleConfig command can be used to define new coupling groups, to change existing coupling groups (i.e. add or remove slave spindles) or to delete coupling groups.

The direction of rotation to the master spindle can be separately determined for each slave spindle.

When a coupling groups is defined or new slave spindles are added, the spindles involved are then automatically switched to the position interface if necessary.

When slave spindles are removed or a coupling group is resolved, the spindles are switched back automatically to the speed interface if the latter was active for the corresponding spindle before the coupling was started.

Spindle functions

The following applies to the following **functions**:

CP, Couple	Coupling group
Group j	Number of the j-th coupling group: 1 ... 4
MA, Master	Master spindle
S<Number i>	i-th slave spindle(s)
Number	Physical spindle index of the master spindle: i = 1 ... n
Number i ..n	Physical spindle index of the i-th slave spindle
n	Number of available spindles (currently, nmax.=8)
i	Index for the i-th spindle (i = 1..n)
y	j = 1...4

Tab. 16-22: Definition of coupling groups

Definition of a coupling group:

SpCoupleConfig (CP=<Group j>, MA=<Number>, S<Number i>=1 -1, ..., S<Number n>=1 -1) with	
S<Number i>	is the ith slave spindle(s): S<Nummer i>=1: Adding the spindle in the direction of rotation of the master spindle - or - S<Nummer i>=-1: Adding the spindle in counter-clockwise direction of the direction of rotation of the master spindle

Tab. 16-23: Defines coupling group

Adding slave spindles to the groups or removing them from the group:

SpCoupleConfig (CP=<Group j>, S<Number i>=0 1 -1, ..., S<Numbers>=0 1 -1) with	
S<Number i>	is the ith slave spindle(s): S<Nummer i>=0: Removing the spindle from the group - or - S<Nummer i>=1: Adding the spindle in the direction of rotation of the master spindle - or - S<Nummer i>=-1: Adding the spindle in counter-clockwise direction of the direction of rotation of the master spindle

Tab. 16-24: Adding slave spindles to the groups or removing them from the group

If a coupling group is changed, it is not necessary to program the number of the master spindle, since the group is already uniquely defined by the number of the coupling group.

Resolving couple link:

SpCoupleConfig (CP=<Group j>, MA=0)

Tab. 16-25: Resolving coupling group

Waiting for synchronous mode : SpCouple_Wait / SPC_WAIT

The part program waits until the coupling group specified is successfully established, reconfigured or resolved. The effect of the function corresponds to a conditional WAIT.

SpCouple_Wait (CP=<group j>)

Tab. 16-26: Waiting for synchronous mode

Programming at active coupling:

Specify the phase offset in case of active coupling: SpCouplePosOffs / SPCP

The angular offset is approached when the coupling is active. The angle has an additive effect on the existing coupling offset. It is, therefore, possible to rotate master and slave spindles in opposite direction to each other when the coupling is active. The absolute offset angle (coupling distance SPCD + angular offset SPCP) between master and slave spindles can be changed at any time.

The rotation can be achieved with an active rotational motion of the spindles.

The signal "position-synchronous 1" at the spindle-specific output interface is reset for the duration of the rotation.

SpCouplePosOffs(S<Number i>=<Offset i> ,... ,S<Numbers>=<Offset n> {,POSVEL<Speed>})

with

S<Number i>	i-th slave spindle(s)
Offset i ... offset n	Rotational angle for the i-th slave spindle(s). Rotational angle is specified as absolute value: ± 3600 degree
Speed	Relative speed between master and slave spindle used to approach the specified offset. The parameter is optional and has a latching effect. the default value is the particular speed recommended which is specified in the Sercos ident number S-0-0222.
n	Number of available spindles (currently, nmax.= 8)
i	Index for the i-th spindle (i = 1..n)

Tab. 16-27: Specify the phase offset in case of active coupling: SpCouplePosOffs / SPCP

Waiting for phase offset: SpCouplePosOffs_Wait / SPCP_WAIT

The part program is stopped until the gear offset programmed with SCPO is approached. This function is like a conditional WAIT.

SpCouplePosOffs_Wait (CP=<Group j>)

with

Group j	Number of the j-th coupling group: 1 ... 4

Tab. 16-28: Waiting for angular offset

Example program

Spindle functions

CoupleAndDecouple

```

N010 SPCD(S3=0)           ;coupling distance 0°
N020 SPCS(S3=0.5)         ;synchronous window 0.5°
N030 SPCE(S3=3.0)         ;error window 3°
...
N060 M119 M319            ;determine reference position

...
N210 SPCC(CP=1, MA=1, S3=1) ;activate coupling 1
N220 SPC_WAIT(CP=1)       ;wait for synchronous mode
...
...
N310 M103 S1=2000
...
...
N760 SPCC(CP=1, MA=0)     ;deactivate coupling 1
N770 SPC_WAIT(CP=1)      ;wait for end of decoupling
...

```

Spindle coupling sequence:

1. Establishing a coupling

The following prerequisites have to be met:

- The **coupling parameters** (distance, synchronization window,...) are configured.
- The **reference points** of the spindles involved were determined with M19 in speed mode.
- The **position interface** is activated for all spindles involved:
If the position interface of an involved spindle is not activated, the corresponding spindle is stopped and only restarted after switching to the position interface.

After SpCoupleConfig (SPCC) was programmed:

- The **limitations of the group** (speed, acceleration) are determined and transferred to the future master spindle.
- The **"coupling number"** is output at the interface of all involved spindles (master and slave(s)).
- **"Spindle is master"** is output at the interface of the master spindle.

Depending on the motion state of the involved spindles, there are **several** potential **motion sequences** during the coupling:

Establish coupling	Motion sequence during coupling	Note
Master spindle and slave spindle at standstill (M5 or M19)	The slave spindle moves on the shortest way to the coupling point. When the synchronization window is reached, "Position-synchronous 1" and "Position-synchronous 2" are output at the interface.	The PLC has to permit a motion of the slave spindle. This permission can be generated by evaluating the interface signals "Coupling number" + "Rotation command".
Master spindle at standstill (M5 or M19), Slave spindle rotates (M3 or M4)	The slave spindle moves directly to the coupling point position. When the synchronization window is reached, "Position-synchronous 1" and "Position-synchronous 2" are output at the interface.	

Establish coupling	Motion sequence during coupling	Note
Master spindle rotates (M3 or M4), Slave spindle at stand-still (M5 or M19)	The slave spindle is accelerated to the speed of the master spindle. Once it reached the speed, it is moved to the coupling point position (SpCoupleDist) on the shortest way . When the synchronization window is reached, "Position-synchronous 1" and "Position-synchronous 2" are output at the interface.	The PLC has to permit a motion of the slave spindle. This permission can be generated by evaluating the interface signals "Coupling number" + "Rotation command".
Master and slave spindles rotate (M3 or M4)	The slave spindle is accelerated or decelerated to the speed of the master spindle and then moved on the shortest way to the coupling point position. When the synchronization window is reached, "Position-synchronous 1" and "Position-synchronous 2" are output at the interface.	

Tab. 16-29: Motion sequences during coupling

2. coupling is active

The slave spindles follow the master spindle.

Any violation of the limits of the set synchronization window and/or synchronization error window are signaled by the NC by:

- Resetting the interface signal "position-synchronous 1" (synchronous window).
- Resetting the interface signal "position-synchronous 2" (Synchronization error window)

If a phase offset is approached with SpCouplePosOffs (SPCP), the NC resets the interface signal "position-synchronous 1" for that period.

3. Complete coupling

If it is decoupled, a slave spindle applies the active motion functions (speed and direction of rotation) from the master spindle.

At the interface, the signals

- Coupling number
- Spindle is master
- Position-synchronous 1
- Position-synchronous 2

are reset.

If the "orient spindle" function is active when completing the coupling, the slave spindles are switched to M5 (spindle stop).

Test mode with active spindle coupling:



Switching to test mode causes the involved coupling groups to stop and the coupling to open.

Careful when switching to test mode!

Effects of spindle-specific interface signals on the spindle coupling:

IF-Signal "Drive On"

If the interface signal "Drive on" is removed, the coupling network is decelerated under NC control.

Spindle functions

The motion state of the master spindle is set to spindle stop (M5). When the group is at standstill, the "Drive on" signal of the concerned spindle is transferred to the drive.

This has the following effects:

- The NC prevents any further group programming.
- Resetting the interface signals " 1" and " position-synchronous 2" in all slave spindles.
- The interface signal "Coupling error" is set for the master spindle.
- The motion state of all involved spindles is set to spindle stop (M5).



This error state can be only exited by a spindle reset of the master spindle (interface signal) or by a total reset (PLC or operation).

Effects of drive-specific messages on the spindle coupling:

Resetting "Drive in operation"

To this signal, the NC reacts as follows:

- Deceleration of the remaining group.
- Prevention of any further programming of the group.
- Resetting the interface signals " 1" and " position-synchronous 2" in all slave spindles.
- Setting the interface signal "Coupling error" for the master spindle.
- The motion state of all involved spindles is set to spindle stop (M5).



This error state can be only exited by a spindle reset of the master spindle (interface signal) or by a total reset (PLC or operation).

State class 1 error

A "diagnostics class 1 error" causes the control loop in the drive to **open**. The NC reacts as follows:

- Deceleration of the remaining group.
- Prevention of any further programming of the group.
- Resetting the interface signals " 1" and " position-synchronous 2" in all slave spindles.
- Setting the interface signal "Coupling error" for the master spindle.
- The motion state of all involved spindles is set to spindle stop (M5).



This error state can be only exited by a spindle reset of the master spindle (interface signal) or by a total reset (PLC or operation).

16.13 Gear shifting with "spindle spinning"

16.13.1 Description

Function

Reduces the wear of the spindle gear.

When engaging another gear, the gearwheels are not always exactly in gear with each other. In this case, (optimum) coupling of power between the input and output shaft does not take place. When the spindle motor is started "normally", the respective gearwheels can be worn and the tooth profiles are unnecessarily loaded.

Thus, the NC - when outputting the gear to the PLC, transmits the "drive-controlled oscillation" command to the drive (S-0-0190; bit 0 and 1 = 1) if the acknowledged gear stage does not correspond to the requested gear stage.

From this moment, the drive independently performs a parameterizable motion profile (see [chapter "Relevant Sercos parameters" on page 281](#)). When the movement profile is activated in the drive, the NC signals this status via the spindle interface signal iSp_IdleSpeed "Idling speed reached" to the PLC, which then can activate mechanical switching.

As soon as the acknowledged gear step corresponds to the requested gear step, the NC cancels the "drive-controlled oscillation" command (S-0-0190; bit 0 and 1 = 0).

Special features and restrictions

Gear shifting is executed asynchronously to the actual part program.

Any necessary synchronization has to be initiated by an application.

Relevant Sercos parameters

S-0-0190	"Drive-controlled oscillation" command Switching the drive function on and off.
S-0-0213	Oscillation speed This speed causes a sinusoidal override of the oscillation offset speed.
S-0-0214	Oscillation offset speed Basic speed of the drive function.
S-0-0215	Oscillation cycle time Duration of the period of sinusoidal override.
S-0-0216	C4100 Command parameter block switching
S-0-0217	Parameter block preselection
For IndraDrive:	
P-0-2216	Parameter block switching configuration
P-0-2217	Parameter block switching; preselection range

Tab. 16-30: Relevant Sercos parameters

For more detailed information, refer to the respective drive documentation.

Relevant spindle IF signals

qSp_Gear1Act to qSp_Gear4Act	Acknowledgement of the gears 1 to 4
qSp_GearIdleAct	Acknowledgement in idle
iSp_GearChange	Switching (active)
iSp_IdleSpeed	Idling speed reached.

iSp_Gear1Sel to iSp_Gear4Sel	Selection of the gears 1 to 4
iSp_GearIdleSel	Idle gear selection

Tab. 16-31: Relevant IF signals

Relevant machine parameters (MP)

1040 00010	Number of gear ranges
1040 00011	Min. spindle speed of the gear step.
1040 00012	Max. spindle speed of the gear step.
1040 00107 to 1040 00112	Spindle syntax
1040 00207 bis 1040 00212	Spindle group syntax

Tab. 16-32: Relevant machine parameters



For more detailed information on gears and their syntax, refer to the "MTX Programming Manual".

16.13.2 Example

Sequence of a gear level change:

1. The spindle is decelerated to standstill.
If "Orient spindle" is active, it is switched to spindle stop.
2. The gear shifting is performed by NC and PLC together (see signal diagram).
3. If clockwise or counterclockwise spindle rotation was active, the spindle is accelerated again.

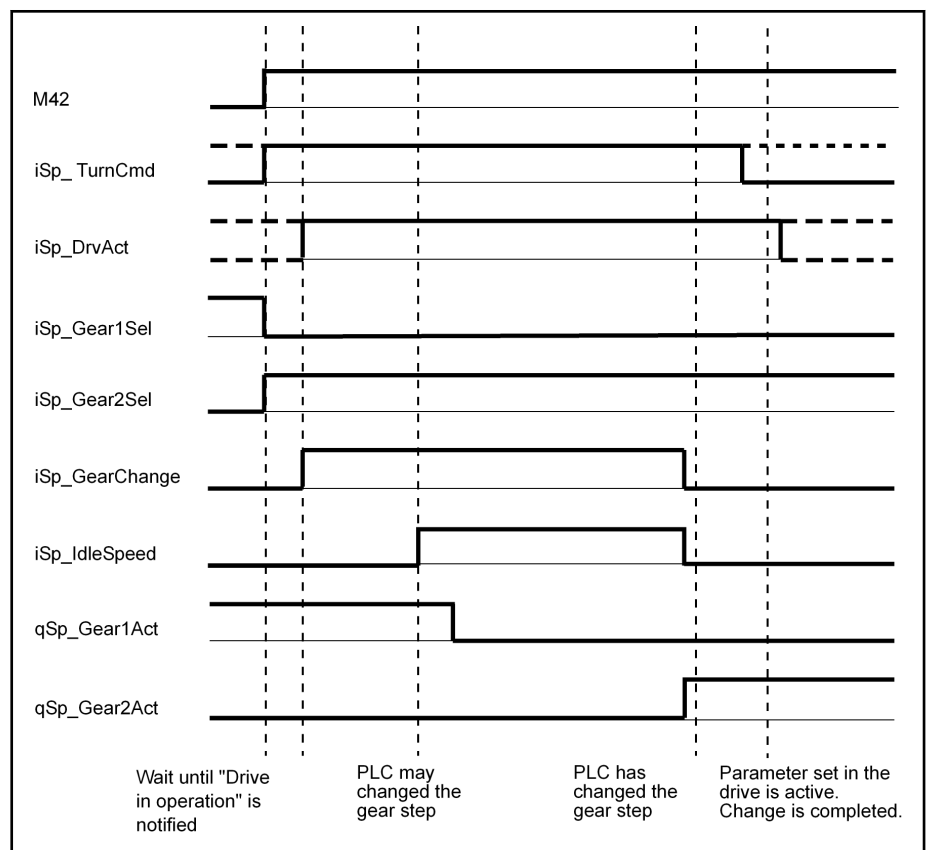


Fig. 16-2: Signal diagram

1. After specifying the new gear level (e.g. M42), the NC sets the signals iSp_TurnCmd and iSp_Gear2Sel and deletes iSp_Gear1Sel.
2. The NC sets the signal iSp_GearChange and starts the command "drive-controlled oscillation" in the drive.
3. When the oscillation speed is reached, the NC sets the signal iSp_IdleSpeed.
4. The PLC changes the gear level and acknowledges completion by deleting qSp_Gear1Act and setting qSp_Gear2Act.
5. The NC deletes the signal iSp_IdleSpeed, completes the command "drive-controlled oscillation" and changes the parameter block in the drive.
6. Finally, the NC deletes the signal iSp_TurnCmd if the switching was started while the spindle was at standstill.

16.13.3 Applying

Prerequisites:

- In the drive, the "drive-controlled oscillation" function (S-0-0190) is available.
- Ensure that the spindle is correctly integrated in the PLC program with respect to its IF signals.

NOTICE

Workpiece or machine can be damaged!

Do not reset the control during machining.

1. Correctly parameterize the Sercos parameters in the drive or transmit them during the Sercos initialization from the NC to the drive.

For information on Sercos, refer to the manual "MTX Functional Description Special Functions", chapter "Functions for the Drive Parameterization".

2. Ensure that the machine parameters listed above are correctly parameterized.

If this is the case, no other adjustments have to be made on the NC.

3. Otherwise, adjust the parameters according to the present application.
4. Trigger control reset.

16.13.4 Enabling/disabling

The NC automatically activates "Spindle spinning" if the gear does not correspond to the acknowledged gear.

When the gear ranges match, the NC automatically deactivates the function.

16.14 Gear shifting with "winding switching"

16.14.1 Description

Function

This function uses the gear function of the MTX to activate a winding switching. This allows switching contactors to operate a motor with different windings or a different wiring (Y-connection and delta connection).

A winding switching takes place when the spindle is rotating and if the M3/M4 is active.

Restrictions and special features

- The "Disengage gear" function (e.g. M148) is not supported.
- The winding is switched without spindle stop when the spindle is rotating (M3 or M4). That means that the drive speed cannot be affected for a certain period from the outside (e.g. qSp_DriveLock) during switching (no contactor closed and/or the drive is in parameter level 1).
- The NC internal process to switch-over windings is based on the function describes in the manual "Rexroth IndraDrive Firmware for drive controllers MPH-04".
- Gear shifting is executed asynchronously to the actual part program.
Any necessary synchronization has to be initiated by an application.

Relevant Sercos parameters

Especially significant are:

S-0-0216	C4100 Command parameter block switching
S-0-0217	Parameter block preselection
S-0-0254	Current parameter block
S-0-0424	Status parameter level
For IndraDrive:	
P-0-2216	Parameter block switching configuration (for IndraDrive)

P-0-2217	Parameter block switching, preselection range (for IndraDrive)
P-0-2218	Parameter block switching, waiting time (for IndraDrive) Attention: The parameter switching and the contactor switching are not synchronized. Note the documentation to switch the parameter group "Winding".

Tab. 16-33: Relevant Sercos parameters

For more detailed information, refer to the respective drive documentation.

Relevant IF signals

qSp_Gear1Act to qSp_Gear4Act	Acknowledgement of the gears 1 to 4
iSp_GearChange	Switching (active)
iSp_Gear1Sel to iSp_Gear4Sel	Selection of the gears 1 to 4

Tab. 16-34: Relevant IF signals

Relevant machine parameters (MP)

All the relevant parameters for gear shifting (SP/SpFunc/Sp[]/Gear/...).

Example

Sequence when switching a winding:

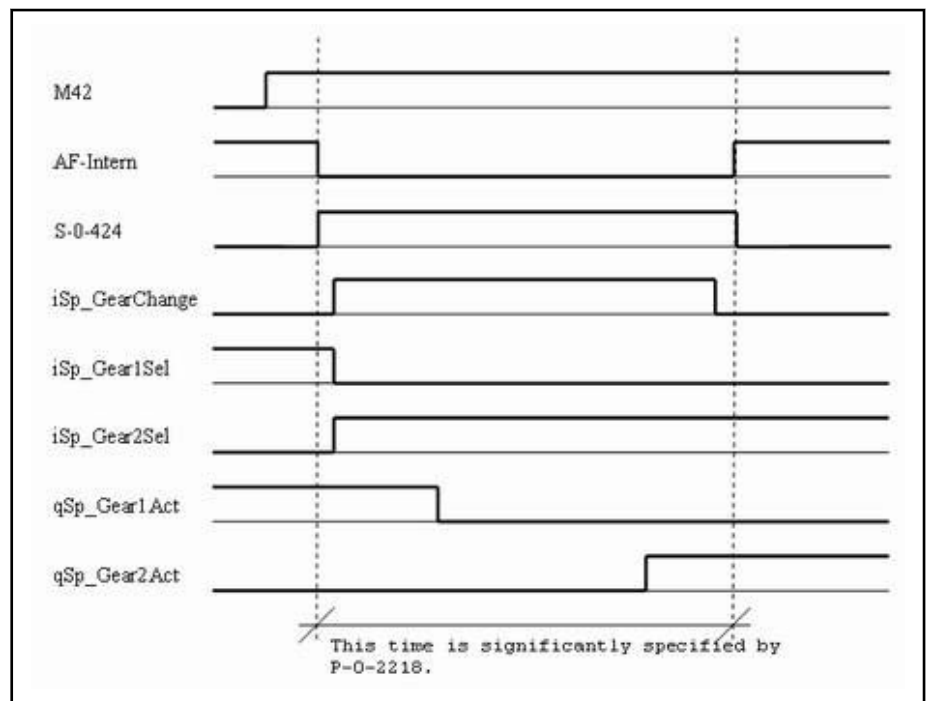


Fig. 16-3: Sequence when switching a winding

1. After specifying the new winding (e.g M42), the NC selects the parameter block in the drive and waits until the drive sets bit 0 of S-0-424.
2. By setting the signals

iSp_Gear1Sel, iSp_GearSel2 and iSp_GearChange, the PLC receives the task to switch the contactors.

3. The PLC acknowledges the state of the contactors via qSp_Gear1Act and qSp_Gear2Act.

16.14.2 Applying

The drive-internal procedure of the winding switch-over has to be similar to the procedure specified in the "Rexroth IndraDrive Firmware for drive controllers MPH-04" manual.

1. Set all the relevant machine parameters for the gear level change (SP/SpFunc/Sp[]/Gear/...).
2. Set the MP SP/SpFunc/Sp[]/Gear/GearChgMethod to "winding".
3. Adapt PLC program for the winding switching.
4. Set the drive parameters.

For IndraDrive: P-0-2218 "Parameter block switching, waiting time" has to be sufficiently high.

5. Bit 0 of S-0-424 has to be transferred cyclically via S-0-144.

16.14.3 Compatibility

When upgrading from MTX07, proceed as follows:

- Remove the NC command SSPWINDING(..) from CHAN/Ch[]/Ini/NcResetState.
- Set the machine parameter SP/SpFunc/Sp[1..32]/Gear/GearChgMethod to the "winding".

16.15 Tool-specific limitations

16.15.1 Description

Function

The acceleration and the maximum speed of a spindle can be limited when using a special tool (e.g. multi-spindle boring head). Additionally, the transformation ratio has to be taken into consideration if the used tool has a gear.

The tool-specific values are taken into consideration during spindle operation (clockwise rotation, counter-clockwise rotation) as well as during tapping without compensating chuck G63.

The iSp_ToolLimAct IF signal signalized active tool-specific limitations.



The available option to adjust the tool transmission ratio via the system data "Conversion factor for spindle speed (/SysSpCmdData[]/SpeedGearRatio)" should not be used anymore!

"/SysSpCmdData[]/SpeedGearRatio" is not evaluated in case of an activated tool-specific limitation.

Restrictions and special features

- **Block pre-run**
Tool-specific parameters have to be specified again using the tool adjustment
- **Tapping without compensating chuck G63**

A tool-specific gear does not affect the spindle position. The position of the drive is always used

- **Spindle positioning (M19)**

- A tool-specific gear does not affect the spindle position. The position of the drive is always used
- Spindle positioning is a drive-controlled function in case of speed-controlled spindles.

The drive-controlled spindle positioning can only be influenced via the relevant drive parameters (S-x-0042 Reference acceleration and S-x-0372 Deceleration quick stop)

- **Error reaction of a drive**

The drive-controlled error reaction emergency stop and quick stop are always executed using the values and the behavior set in the drive (S-x-0372 Deceleration quick stop, S-0-0429 Deceleration emergency stop, S-x-0138 Bipolar acceleration)

Relevant NC functions

SToolLim, STOOLLIM SToolLim(<SpName>= <0 1> { ,...})	
	Enables the entered tool-specific spindle limitations in one or several channel spindles in the /SysSpCmdData[<Spindle>]/ Tool system data.
SSpToolLim, SSPTOOLLIM SSpToolLim(<SSpNr> <SSpName>, <0 1> { ,...})	
	Enables the entered tool-specific spindle limitations in one or several system spindles in the /SysSpCmdData[<Spindle>]/ Tool system data.

Tab. 16-35: Relevant NC functions

Relevant IF signals

iSp_ToolLimAct	
	Signalizes an active tool-specific limitation.

Tab. 16-36: Relevant IF signals

Relevant system data

/SysSpCmdData[s]/Tool/	
Enable	Release "TOOL" contents for use
MaxSpSpeed	Upper speed limit value
MaxAcc	Maximum acceleration
MaxPosAcc1	Maximum acceleration in position control mode 1
MaxPosAcc2	Maximum acceleration in position control mode 2
MaxSwiSpeed	Max. switching speed for position control mode 2
N_Input	Input revolutions
N_Output	Output revolutions

Tab. 16-37: Relevant system data

The respective machine parameter is used for the value set to zero.



When using a transformation ratio, not all specified values refer to the tool. The machine parameters still refer to the drive.

The acceleration values for the position control mode do not have to be limited. MaxAcc is the upper limit of the acceleration.

The respective machine parameter is used for the value set to zero.

Example

The machine has a spindle with 2 gear stages.

Machine parameters:

	Gear stage 1	Gear stage 2
MaxSpSpeed	2000 rpm	5000 rpm
MaxSpAcc	150 rad/sec ²	200 rad ²
MaxSpAccPosCtrl[1]	100 rad/sec ²	150 rad ²
SwiSpeed	0 rpm	0 rpm

Tool 1 Maximum limit values of the tool:

170 rad/sec² and 2500 rpm.

Activating limitations

```
...
010 SP% = 3
011 SD.SysSpCmdData[SP%].Tool.Enable = 1
012 SD.SysSpCmdData[SP%].Tool.MaxSpeed = 2500
013 SD.SysSpCmdData[SP%].Tool.MaxAcc = 170
014 SD.SysSpCmdData[SP%].Tool.MaxPosAcc1 = 0
015 SD.SysSpCmdData[SP%].Tool.MaxPosAcc2 = 0
016 SD.SysSpCmdData[SP%].Tool.MaxSwSpeed = 0
017 SD.SysSpCmdData[SP%].Tool.N_Input = 0
018 SD.SysSpCmdData[SP%].Tool.N_Input = 0
N19 SSpToolLim([SP%],1)
...
```

Active values:

	Gear stage 1	Gear stage 2
MaxSpSpeed	2000 rpm	2500 rpm
MaxAcc	150 rad/sec ²	170 rad²
MaxPosAcc1	100 rad/sec ²	150 rad ²
SwiSpeed	0 rpm	0 rpm

Start spindle

```
...
N110 GSP(3,1)
N120 M41 SPWAIT
N130 M3 S1500 ;tool acceleration 150 rad/sec2
               ;tool speed      1500 rev/min
...
```

Thread drilling

```

...
N310 GSP(3,1)
N320 M41 SPWAIT
N330 G63(M3,S600) Z-20 F200
           ;tool acceleration  100 rad/sec2
           ;tool speed         600 rev/min
...

```

Deactivating the limitations

```

...
030 SP% = 3
031 SD.SysSpCmdData[SP%].Tool.Enable = 0
N32 SSpToolLim([SP%],1)
...

```

- Tool 2** Maximum limit values of the tool:
 170 rad/sec² and 2500 rpm.
 The tool has a transmission ratio of 5: 4.

Activating limitations

```

...
010 SP% = 3
011 SD.SysSpCmdData[SP%].Tool.Enable = 1
012 SD.SysSpCmdData[SP%].Tool.MaxSpeed = 2500
013 SD.SysSpCmdData[SP%].Tool.MaxAcc = 170
014 SD.SysSpCmdData[SP%].Tool.MaxPosAcc1 = 0
015 SD.SysSpCmdData[SP%].Tool.MaxPosAcc2 = 0
016 SD.SysSpCmdData[SP%].Tool.MaxSwSpeed = 0
017 SD.SysSpCmdData[SP%].Tool.N_Input = 5
018 SD.SysSpCmdData[SP%].Tool.N_Input = 4
N19 SSpToolLim([SP%],1)
...

```

Active values:

	Gear stage 1	Gear stage 2
MaxSpSpeed	1600 rpm	2500 rpm
MaxAcc	120 rad/sec ²	160 rad ²
MaxPosAcc1	100 rad/sec ²	150 rad ²
SwiSpeed	0 rpm	0 rpm

Start spindle

```

...
N110 GSP(3,1)
N120 M41 SPWAIT
N130 M3 S1500 ;tool acceleration  120 rad/sec2
           ;tool speed         1500 rev/min
           ;drive acceleration  150 rad/sec2
           ;drive speed        1875 rev/min
...

```

Thread drilling

```

...
N310 GSP(3,1)
N320 M41 SPWAIT
N330 G63(M3,S600) Z-20 F200
           ;tool acceleration  100 rad/sec2
           ;tool speed         600 rev/min
           ;drive acceleration  125 rad/sec2
           ;drive speed        750 rev/min
...

```

Deactivating the limitations

```
...
030 SP% = 3
031 SD.SysSpCmdData[SP%].Tool.Enable = 0
N32 SSpToolLim([SP%],1)
...
```

16.15.2 Applying

If tool-specific limitations are required, the limitations are entered in the system date /SysSpCmdData[s]/Tool/ during tool change and activated by calling SSpToolLim or SToolLim.

The PLC can query the iSp_ToolLimAct signal.

1. Create a program to delete the system date.

ClearSpToolData

```
10 IF P1%=NUL THEN
11   SETERR("Spindle number is missing",,1)
12 ENDIF
13 SP%=P1%
14 SD.SysSpCmdData[SP%].Tool.Enable = 0
15 SD.SysSpCmdData[SP%].Tool.MaxSpeed = 0
16 SD.SysSpCmdData[SP%].Tool.MaxAcc = 0
17 SD.SysSpCmdData[SP%].Tool.MaxPosAcc1 = 0
18 SD.SysSpCmdData[SP%].Tool.MaxPosAcc2 = 0
19 SD.SysSpCmdData[SP%].Tool.MaxSwSpeed = 0
20 SD.SysSpCmdData[SP%].Tool.N_Input = 0
21 SD.SysSpCmdData[SP%].Tool.N_Input = 0
M30
```

2. Adjust the tool change program.

ToolChange

```
...
...
N200 WAIT
0210 TOOLSP%=..
...
N300 P ClearSpToolData([TOOLSP%])
; Read tool limits, e.g. from database
0310 TL_LIM?=..
0320 IF TL_LIM? THEN
0330   TL_SPEED=..
0340   TL_ACC=..
0350   SD.SysSpCmdData[TOOLSP%].Tool.MaxSpeed = TL_SPEED
0360   SD.SysSpCmdData[TOOLSP%].Tool.MaxAcc = TL_ACC
0370   SD.SysSpCmdData[TOOLSP%].Tool.Enable = 1
0380 ELSE
0390   SD.SysSpCmdData[TOOLSP%].Tool.Enable = 0
0400 ENDIF
N410 SSpToolLim([TOOLSP%],1)
...
...
```

3. If required, the iSp_ToolLimAct signal can be linked in the PLC program.

16.15.3 Activating

"SSpToolLim(<SysSp>,1" and "SToolLim(<ChSpName>=1)" apply the system data content if "/SysSpCmdData[s]/Tool/Enable" is set. The PLC signal iSp_ToolLimAct is set.

No changes have been made in the part program.

16.15.4 Deactivating

Variant 1 This variant is used in the program section "ToolChange".

"SSpToolLim(<SysSp>,1)" and "SToolLim(<ChSpName>=1)" activate the machine parameters if "/SysSpCmdData[s]/Tool/Enable" is **not** set. The transmission ratio is set to 1:1. The PLC signal iSp_ToolLimAct is deleted.

Program:

```
...  
O10 SD.SysSpCmdData[5].Tool.Enable = 0  
N20 SSpToolLim(5,1)  
...
```

Variant 2 "SSpToolLim(<SysSp>,0)" and "SToolLim(<ChSpName>=0)" activate the machine parameters. The transmission ratio is set to 1:1. The PLC signal iSp_ToolLimAct is deleted.

Program:

```
...  
N10 SSpToolLim(5,0)  
...
```

17 Auxiliary functions

17.1 Basics

Auxiliary functions are components of the NC syntax which can be programmed in a part program and in the "MDI" mode.

They are used to exchange information with the PLC via flags, e.g. to switch on or off machine functions, to switch from one machine function to another or to transmit or receive numeric data.

They can or cannot require acknowledgement. For auxiliary functions requiring acknowledgement, the control waits until a positive acknowledgement is received (e.g. by flags being reset by the PLC) before processing the next program block or changing to READY state.

The IndraMotion MTX provides various types (classes) of auxiliary functions. The type to be assigned to an auxiliary function depends on the kind of information to be exchanged and the desired range of action (channel-dependent or channel-independent).

The parameterization of the auxiliary functions is explained in the documentation "MTX Machine Parameters" (chapter 3.5).

17.2 Channel-dependent and channel-independent auxiliary functions

17.2.1 Channel-independent auxiliary function

Every channel-independent auxiliary function exists only once in the system. Therefore, their current states are identical on each applied channel.

17.2.2 Channel-dependent auxiliary function

Every channel dependent auxiliary function exists once per applied channel (including "auxiliary" channel 0).

Thus, the state of such auxiliary functions can differ in all applied channels.

- M0, M00 - Interrupt program (program stop)

The auxiliary function M0 interrupts the NC program and stops the machine motion after the block execution. M0 can be programmed together with other NC functions in the same block. Only after all other programmed functions have been executed, the "program stop" is applied. Refer to the NC programming instructions for a detailed description of the auxiliary function M0.

- M1, M01 - Interrupt program conditionally (conditional program stop)

The auxiliary function M1 interrupts the NC program and stops all machine motions after the block execution if there is the channel-based interface signal "Optional stop". M1 can be programmed together with other NC functions in the same block. Only after all other programmed functions have been executed, the "conditional stop" is applied. Refer to the NC programming instructions for a detailed description of the auxiliary function M1.

- M2, M02, M30 - Complete program (End of program)

The auxiliary functions M2 and M30 complete a program. If the program is a main program, different interface signals are set or deleted. The program processing activates all states defined in the parameter "ChResetState" (MP706000020 - State after channel reset) for an M30 event,

returns to the beginning of the main program and waits for a new "NC start". If the program is a subroutine, the NC outputs the auxiliary functions (M2, M02, M30), returns to the calling program and continues to process the calling program. The auxiliary functions M2, M02 or M30 have to be programmed in a separate program line. Refer to the NC programming instructions for a detailed description of the auxiliary functions M2, M02 and M30.

17.3 Bit-coded auxiliary functions

This type of auxiliary function can be used to exchange individual binary information with the PLC (e.g., to switch on or off machine functions or to switch from one machine function to another).

They can be assigned to a **bit array** with an "initial bit address" and an "end bit address" or only to only one **single bit** with a certain "bit address".

Bit-coded auxiliary functions assigned to a **bit array** are always channel-dependent auxiliary functions. Thus, the bit array exists once on each channel.

The individual bits of any bit array are programmed with the NC syntax via auxiliary function names (letters only!) and followed by an integer. The integer range is specified by a "start code" and an "end code".

Bit-coded auxiliary functions assigned to only **one single bit** can either be "discrete bit-coded auxiliary functions" or "individually defined bit-coded auxiliary functions".

- A "discrete" bit-coded auxiliary function can be assigned to a bit address in the flag area of **only one** channel. Since it is thus in the system, it is called channel-independent.
- Although an "individually defined" bit-coded auxiliary function can also be assigned to a bit address, it exists in the flag area of **all** channels. Since it exists only once in every channel, it is called channel-dependent.

Names composed of character strings including letters as well as digits and the "_" character can be assigned to "discrete" as well as "individually defined" auxiliary functions.

17.4 BCD, integer, float and double-coded Auxiliary Functions

This type of auxiliary function can be used to exchange numeric information in one of these formats with the PLC (e.g. to transmit additional data to certain functions).

The name of this type of auxiliary function can contain letters, digits and the character "_". The data value directly follows the name or starts with "=".

Example:

Auxiliary function name: "T"; Data value: "12345" is programmed as "T12345".

Depending on the required value range of the data to be transmitted, one of these auxiliary functions is used.

- A 32-bit BCD-coded auxiliary function must have an 8-byte marker area (= 64 bits) and can transmit up to 8-digit data values.
- A 64-bit BCD-coded auxiliary function has to have an 12-byte flag area (= 96 bits) and can transmit up to 16-digit data values.

The number of integer and decimal positions can be configured by setting the corresponding parameters. The following applies for the 32-bit-BCD-auxiliary

function: Total of integer positions and decimals up to 8 and the total of integer positions and decimals for 64-bit BCD auxiliary function is up to 16.

An integer auxiliary function has to have an 8-byte flag area (= 64 bits) and can transmit data values between 2147483647 and -2147483647.

A float auxiliary function can transmit a floating point number without exponential representation with up to seven significant places. It has to have an 8-byte flag area (=64 bits).

A double auxiliary function can transmit a floating point number without exponential representation with up to 15 significant places. It has to have a 12-byte flag area (=96 bits).

17.5 Output response of auxiliary functions

The output response can be set. It defines how the auxiliary functions programmed in an NC block are transmitted to the PLC user program.

Generally, the data is output in the order programmed in the NC block.

Bit-coded auxiliary functions get a special treatment: they are collected until the last bit-coded auxiliary function is displayed in a block. Subsequently, all bit-coded auxiliary functions are output together.

First, set whether all auxiliary functions, also the non-bit-coded auxiliary functions, are to be collected before being output together.

For details, refer to the documentation "IndraMotion MTX Machine Parameters" (Chap. 3.5).

17.6 Auxiliary functions (AUXF) – Basic parameters -

17.6.1 Auxiliary function basics

The auxiliary functions

- are components of the NC syntax that can be programmed both in a part program and in "Setup" mode;
- are used to exchange information with the PLC via flags, e.g. to switch on or off machine functions, to switch from one machine function to another or to transmit or receive numeric data.
- may or may not require acknowledgement. For auxiliary functions requiring acknowledgement, the control waits until a positive acknowledgement is received (e.g. by flags being reset by the PLC) before processing the next program block or changing to "Ready" state;
- can be assigned to an auxiliary function group upon mutual exclusion of their activity or display.

Auxiliary function classes: The CNC system MTX provides various different auxiliary function classes. What class to be assigned to an auxiliary function depends on the kind of information to be exchanged and the desired range of action (channel-dependent or channel-independent).

The CNC system MTX supports the following auxiliary function classes:

- Bit-coded, channel-dependent
- 32-bit BCD-coded, channel dependent
- 32-bit BCD-coded, channel independent
- Integer-coded, channel-dependent
- Integer-coded, channel-independent

Auxiliary functions

- Float-coded, channel-dependent
- Float-coded, channel-independent
- Double-coded, channel-dependent
- Double-coded, channel-independent

Auxiliary function classes also always include the auxiliary function type and range of action.

Channel-dependent and channel-independent auxiliary functions

The differences between the two ranges of action are:

- Channel-independent auxiliary functions exist only once within the system. Therefore, their current states are identical on each applied channel.
- Channel-dependent auxiliary functions exist once per applied channel. Thus, the state of such auxiliary functions can differ in all applied channels.

Bit-coded auxiliary functions:

This type of auxiliary function can be used to exchange individual binary information with the PLC, e.g., to switch on or off machine functions or to switch from one machine function to another. Bit-coded auxiliary functions are assigned to a **bit array** with a "start bit address" and an "end bit address". The individual bits of any such bit array are programmed, using NC syntax, with the respective name of the auxiliary function (letters only!) directly followed by an integer. The value range: The integer is specified by the "start code" and the "end code".



In the CNC system MTX, bit-coded auxiliary functions are first configured as channel-dependent auxiliary functions as a rule.

Non-bit-coded Auxiliary Functions

Non-bit-coded auxiliary functions are used if numerical information should be exchanged with the PLC, e.g. to transfer additional values to functions. The name of this type of auxiliary function can contain letters, digits and the character "_". The name is followed directly by the data value.

Example:

Auxiliary function name "T" and data value "12345"

The auxiliary function is programmed as "T12345".

32-bit-BCD-coded auxiliary functions

This type of auxiliary function is used when numerical information must be exchanged with the PLC in BCD format (Binary Coded Decimal).

A 32-bit BCD-coded auxiliary function must have an 8-byte marker area (= 64 bits) and can transmit up to 8-digit data values.

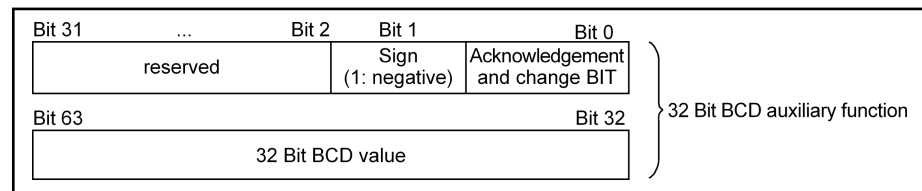


Fig. 17-1: Marker area of a 32-bit BCD auxiliary function

Integer-coded auxiliary functions:

This type of auxiliary function is used when numerical information has to be exchanged with the PLC in integer format.

An integer auxiliary function has to have an 8-byte marker area (= 64 bits) and can transmit numerical values between -2147483648 and +2147483647.

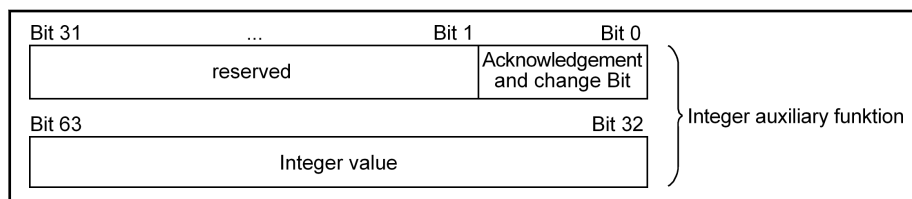


Fig. 17-2: Marker area of an integer auxiliary function

Float-coded auxiliary functions

This type of auxiliary function is used when numerical information has to be exchanged with the PLC in float format.

A float auxiliary function has to have an 8-byte marker area (= 64 bits) and can transmit floating point values between $1\text{E}-37$ ($-1\text{E}-37$) and $1\text{E}+37$ ($-1\text{E}+37$) with an accuracy of six decimal places.

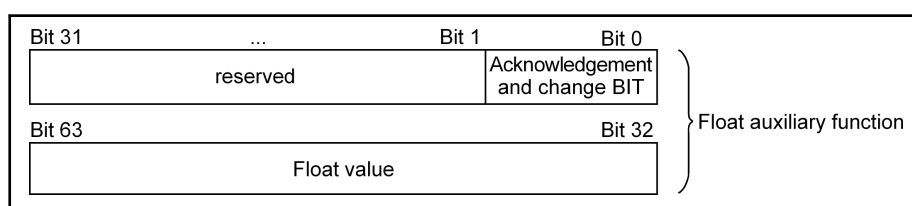


Fig. 17-3: Marker area of a float auxiliary function

Double-coded auxiliary functions:

This type of auxiliary function is used when numerical information has to be exchanged with the PLC in double format.

A double auxiliary function has to have an 12-byte marker area (= 96 bits) and can transmit floating point values between $1\text{E}-37$ ($-1\text{E}-37$) and $1\text{E}+37$ ($-1\text{E}+37$) with an accuracy of ten decimal places.

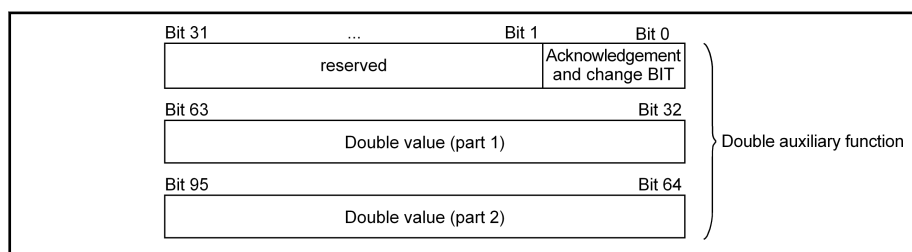


Fig. 17-4: Marker area of a double auxiliary function

Auxiliary function groups:

Auxiliary functions that mutually exclude one another in the display and/or their activity can be grouped into auxiliary function groups. This is done with syntax groups in the machine parameters.

The control is able to determine the last enabled auxiliary function of an auxiliary function group. This way, the last M-code of the group "M3, M4, M5, M19" or the last T word, for example, can be displayed.

Auxiliary function groups are defined as either channel-dependent or channel-independent. Auxiliary functions of the same name may only be assigned to a group once. Channel-independent groups may only contain channel-independent auxiliary functions, and channel-dependent groups may only contain channel-dependent auxiliary functions.

17.6.2 Overview

The following settings are available in the data group "AUXF" for defining auxiliary functions:

- Auxiliary class definition -ClassDef-
(chapter 17.6.3 "Auxiliary class definition (ClassDef)" on page 298)
- Acknowledgement behavior for bit-coded auxiliary functions - BitAckn -
(chapter 17.6.4 "Acknowledgement behavior for bit-coded auxiliary functions (BitAckn)" on page 309)
- Group definition -GrDef-
(chapter 17.6.5 "Auxiliary function group definition (GrDef)" on page 312)
- Global output behavior -OutBehav-
(chapter 17.6.6 "Global output behavior (OutBehav)" on page 313)
- Number of auxiliary functions -NofAuxFunc-
(chapter 17.6.7 "Number of usable auxiliary functions (NofAuxFunc)" on page 314)
- Addresses of auxiliary functions in the PLC interface (PlcIf)
(chapter 17.6.8 "Addresses of auxiliary functions in the PLC interface (PlcIf)" on page 319)

Application examples are explained in the following:

- Definition of a new auxiliary function
(chapter 17.6.9 "Defining a new auxiliary function" on page 324) und
- parameterization of auxiliary functions
(chapter 17.6.10 "Parameterizing example: Auxiliary functions" on page 327)

17.6.3 Auxiliary class definition (ClassDef)

General information

Up to 30 auxiliary functions can be defined. Here, the auxiliary function type, name and action range and parameters specific to each auxiliary function are defined.

Auxiliary function types

When defining auxiliary function classes, the "auxiliary function type" has to be selected first. To do so, select the AUXF/ClassDef/Class[1..30]/ folder and, with the right mouse button, select the appropriate entry with the function "Convert to..." from the context menu :

- Bit
- Bcd32
- Int
- Float
- Double



The entry for the auxiliary function type currently enabled for Class[1-30] is not shown in the context menu.

The auxiliary functions are described below:

Settings such as "Enable auxiliary function", "Name" and "Channel dependency", which apply equally to all auxiliary functions, are described in the General section (General parameters of auxiliary function classes Class[1..30]).

Following this are the special parameters for each auxiliary function type (bit, bcd32, int, float and double-coded auxiliary functions of auxiliary function classes Class[1-30]).

General parameters of auxiliary function classes [1-30]

Enabling auxiliary function

Parameter ID:	EnablAuxFunc	
Path:	AUXF/ClassDef/Class[1-30]/EnablAuxFunc	
Purpose:	Enables/disables the auxiliary function from class [1..30] named AuxFunc-Name, relating to auxiliary function type:	
	• Bit auxiliary function • Bcd32 auxiliary function • Integer auxiliary function • Float auxiliary function • Double auxiliary function	
Explanations:	-	
Value range:	0 (No) 1 (Yes)	
Pre-assignment:	0 (No)	
Unit:	-	
Meaning of parameter values:	0 (No)	Auxiliary function with the name in .../Class[1..30]/AuxFunc-Name is not active.
	1 (Yes)	Auxiliary function with the name in .../Class[1..30]/AuxFunc-Name is active.
Access rights:	MTB:	Write
	Setter:	Read
	Operator:	Read
Enabling:	System reset (SR)	
Dependency:	-	
Macoda ID:	3010 00040	
Data type:	Byte0_1_t (byte)	
	4 byte integer, unsigned for MACODA compatibility	

Name

Parameter ID:	AuxFuncName	
Path:	AUXF/ClassDef/Class[1..30]/AuxFuncName	
Purpose:	Specifies an auxiliary function "Name" for each class [1..30] in one of the auxiliary function types named below:	
	• Bit auxiliary function • Bcd32 auxiliary function • Integer auxiliary function • Float auxiliary function • Double auxiliary function	



Names of channel-dependent bit-coded auxiliary functions can also be used more than once (see example below).

Example:

Auxiliary functions - Program code areas

For matters of compatibility, a bit memory area should be accessible through 2 different programming code areas. According to specifications, the name of the auxiliary function may not be changed for this. This problem can be solved only because the name TEST may be assigned more than once.

Set the parameters of 2 auxiliary functions, both called TEST, that differ only in respect to their programming code areas:

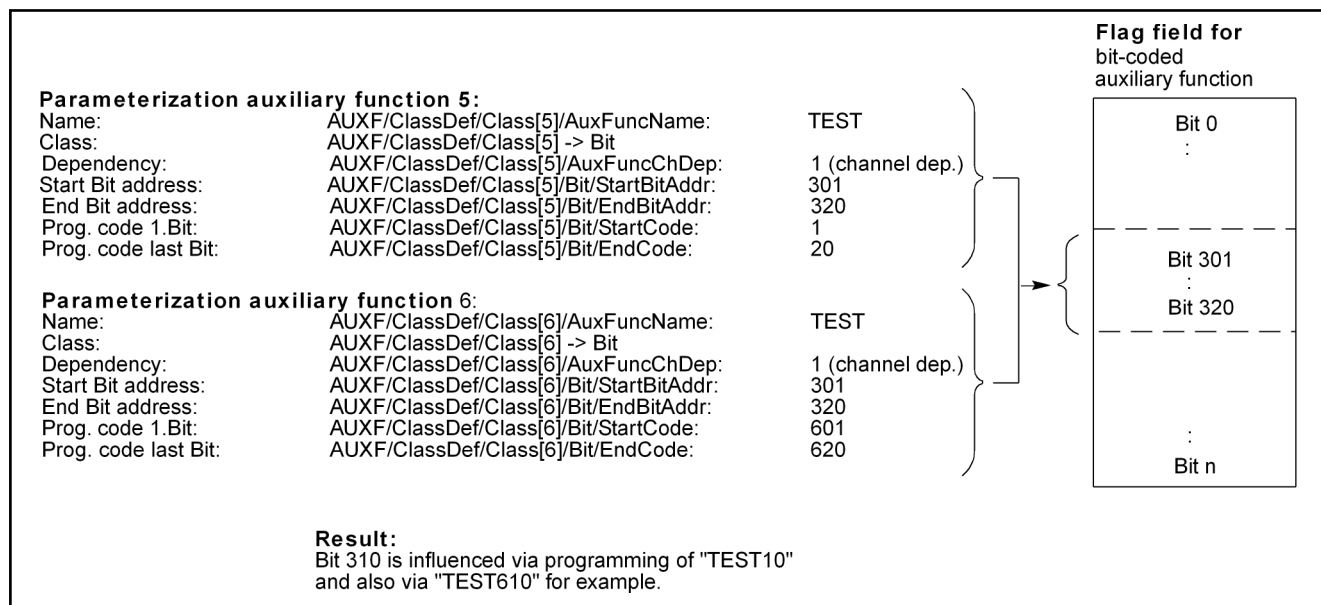


Fig. 17-6: Example 2 with two homonymous auxiliary functions

Channel dependency

Parameter ID: AuxFuncChDep

Path: AUXF/ClassDef/Class[1..30]/AuxFuncChDep

Purpose: Specifies whether the auxiliary function named in AUXF/ClassDef/Class[1..30]/AuxFuncName is "channel-dependent", relating to auxiliary function type:

- Bcd32 auxiliary function
- Integer auxiliary function
- Float auxiliary function
- Double auxiliary function

Explanations: -

Value range: 0 (channel-independent auxiliary function)
1 (channel-dependent auxiliary function)

Pre-assignment: 0 (channel-independent auxiliary function)

Unit: -

Meaning of parameter values:

0	Auxiliary function from the corresponding auxiliary function type is channel-independent
1	Auxiliary function from the corresponding auxiliary function type is channel-dependent

Auxiliary functions

Access rights:	MTB: Write
	Setter: Read
	Operator: Read
Enabling:	System reset (SR)
Dependency:	-
Macoda ID:	3010 00040
Data type:	Byte0_1_t (byte)
	4 byte integer, unsigned for MACODA compatibility

Bit-coded auxiliary functions of auxiliary function classes Class[1..30]**Start code**

Parameter ID:	StartCode
Path:	AUXF/ClassDef/Class[1..30]/Bit/StartCode
Purpose:	Defines which digit the first bit (start code) can be addressed with for a bit-coded auxiliary function defined under .../Class[1..30]/Bit/ that is assigned to a whole bit array.
Explanations:	Start and end codes specify how the bits of a bit-coded auxiliary function can be addressed using NC syntax (see also second example of the parameter AuxFuncChDep).
Value range:	0 .. 65535
Pre-assignment:	0
Unit:	-
Meaning of parameter values:	Start code (first bit)
Access rights:	MTB: Write
	Setter: Read
	Operator: Read
Enabling:	System reset (SR)
Dependency:	-
Macoda ID:	3010 00080
Data type:	Int0_65535_t (int)
	2 byte integer, unsigned for MACODA compatibility

End code

Parameter ID:	EndCode
Path:	AUXF/ClassDef/Class[1..30]/Bit/EndCode
Purpose:	Defines which digit the last bit (end code) can be addressed with for a bit-coded auxiliary function defined under .../Class[1..30]/Bit/ that is assigned to a whole bit array.
Explanations:	Start and end codes specify how the bits of a bit-coded auxiliary function can be addressed using NC syntax (see also second example of the parameter AuxFuncChDep).
Value range:	0 .. 65535
Pre-assignment:	299 .../Class[1]/Bit/EndCode
	0 .../Class[2-30]/Bit/EndCode
Unit:	-
Meaning of parameter values:	End code (last bit)

Access rights: **MTB:** Write
Setter: Read
Operator: Read

Enabling: System reset (SR)

Dependency: -

Macoda ID: 3010 00090

Data type: Int0_65535_t (int)
 2 byte integer, unsigned for MACODA compatibility

Start bit address

Parameter ID: StartBitAddr

Path: AUXF/ClassDef/Class[1..30]/Bit/StartBitAddr

Purpose: Defines for an under .../Class[1..30]/Bit/ defined

- Bit-coded auxiliary function assigned to an entire bit array is the address of the first bit.
- Bit-coded auxiliary function assigned to a single bit, that is the address of this bit.

Explanations: -

Value range: 0 .. 65535

Pre-assignment: 0 .../Class[1..30]/Bit/StartBitAddr

Unit: -

Meaning of parameter values: Address of the first bit (bit array) or address of the bit (single bit)



All addresses relating to the starting address of the bit auxiliary function marker area are offset values (see also folder AUXF/NofAuxFunc/UsabAuxFunc/..., single parameters).

Access rights: **MTB:** Write
Setter: Read
Operator: Read

Enabling: System reset (SR)

Dependency: -

Macoda ID: 3010 00060

Data type: Int0_65535_t (int)
 2 byte integer, unsigned for MACODA compatibility

End bit address

Parameter ID: EndBitAddr

Path: AUXF/ClassDef/Class[1..30]/Bit/EndBitAddr

Purpose: Defines for an under .../Class[1..30]/Bit/ defined

Bit-coded auxiliary function assigned to an entire bit array is the address of the last bit.

Explanations: -

Value range: 0 .. 65535

Pre-assignment: 299 .../Class[1]/Bit/EndBitAddr

Auxiliary functions

0 .../Class[2..60]/Bit/EndBitAddr

Unit: -

Meaning of parameter values: Address of the last bit (bit array)



All addresses relating to the end address of the bit auxiliary function marker area are offset values (see also folder AUXF/NofAuxFunc/UsabAuxFunc/..., single parameters).

Access rights: MTB: Write
 Setter: Read
 Operator: Read

Enabling: System reset (SR)

Dependency: -

Macoda ID: 3010 00070

Data type: Int0_65535_t (int)
 2 byte integer, unsigned for MACODA compatibility

Bcd32-coded auxiliary functions of auxiliary function class Class[1..30]

Acknowledgement requirement

Parameter ID: AcknBcd32

Path: AUXF/ClassDef/Class[1..30]/Bcd32/AcknBcd32

Purpose: Specifies whether a Bcd32-coded auxiliary function defined under Class[1..30]/Bcd32 requires acknowledgement.

Explanations: -

Value range: 0 (No)
 1 (Yes)

Pre-assignment: 0 (No)

Unit: -

Meaning of parameter values: 0 (No) Auxiliary function does not require acknowledgement
 1 (Yes) Auxiliary function requires acknowledgement

Access rights: MTB: Write
 Setter: Read
 Operator: Read

Enabling: System reset (SR)

Dependency: -

Macoda ID: 3010 00090

Data type: Byte0_1_t (byte)
 2 byte integer, unsigned for MACODA compatibility

Integer places

Parameter ID: PrelPlaceBcd32

Path: AUXF/ClassDef/Class[1..30]/Bcd32/PrelPlaceBcd32

Purpose: Defines the number of integer places for a Bcd32-coded auxiliary function defined under Class[1..30]/Bcd32.

Explanations: The total number of integer places and decimal places cannot exceed 8 (see also parameter in folder AUXF/ClassDef/Class[1..30]/Bcd32/).

Value range:	0 .. 8
Pre-assignment:	8 .../Class[1..30]/Bcd32/PrelPlaceBcd32
Unit:	-
Meaning of parameter values:	Number of places to the left of the decimal point
Access rights:	MTB: Write Setter: Read Operator: Read
Enabling:	System reset (SR)
Dependency:	-
Macoda ID:	3010 00060
Data type:	Byte0_8_t (byte) 2 byte integer, unsigned for MACODA compatibility

Example:

Decimal places and integer places of a Bcd32-coded auxiliary function

Having 4 places to the left and 4 places to the right of the decimal point, the auxiliary function **T1234.56** would be stored in the PLC memory as follows:

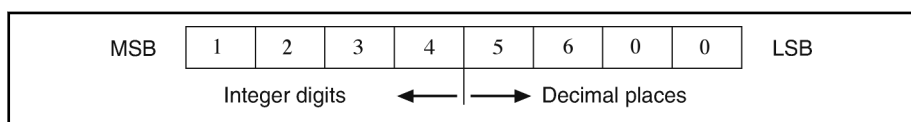


Fig. 17-7: BCD32-coded auxiliary function

Decimal places

Parameter ID:	DeciPlaceBcd32
Path:	AUXF/ClassDef/Class[1..30]/Bcd32/DeciPlaceBcd32
Purpose:	Defines the number of places to the left of the decimal point for a Bcd32-coded auxiliary function defined under .../Class[1..30]/Bcd32.
Explanations:	The total number of places to the left and to the right of the decimal point cannot exceed 8 (see also parameter in folder AUXF/ClassDef/Class[1..30]/Bcd32/).
Value range:	0 .. 8
Pre-assignment:	0 .../Class[1..30]/Bcd32/PrelPlaceBcd32
Unit:	-
Meaning of parameter values:	Number of decimal places
Access rights:	MTB: Write Setter: Read Operator: Read
Enabling:	System reset (SR)
Dependency:	-
Macoda ID:	3010 00070
Data type:	Byte0_8_t (byte) 2 byte integer, unsigned for MACODA compatibility

Example:

Decimal places and integer places of a BCD32-coded auxiliary function

Having 4 places to the left and 4 places to the right of the decimal point, the auxiliary function **T1234.56** would be stored in the PLC memory as follows:

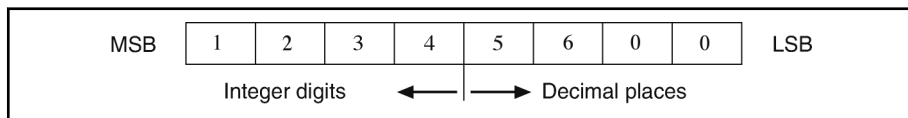


Fig. 17-8: Bcd32-coded auxiliary function

Index

Parameter ID: IndBcd32

Path: AUXF/ClassDef/Class[1..30]/Bcd32/IndBcd32

Purpose: Specifies the "index" of a BCD32-coded auxiliary function defined under ../Class[1..30]/Bcd32.

Explanations: Their index determines the order in which several auxiliary functions belonging to a specific class of auxiliary functions are stored in the marker filed of class Class[1..30]/Bcd32.

Value range: 0 .. 32

Pre-assignment: 0 ../Class[1..30]/Bcd32/IndBcd32

Unit: -

Meaning of parameter values: Index of the BCD32-coded auxiliary function. The highest index is dependent on the number of existing BCD32-coded auxiliary functions in the class Class[1..30]/Bcd32 (parameter "Usable auxiliary functions" UsabAuxFunc in the folder AUXF/NofAuxFunc/).

Access rights: **MTB:** Write

Setter: Read

Operator: Read

Enabling: System reset (SR)

Dependency: -

Macoda ID: 3010 00080

Data type: Byte0_32_t (byte)

2 byte integer, unsigned for MACODA compatibility

Integer-coded auxiliary functions of auxiliary function class[1..30]**Acknowledgement requirement**

Parameter ID: AcknInt

Path: AUXF/ClassDef/Class[1..30]/Int/AcknInt

Purpose: Specifies whether an integer-coded auxiliary function defined under Class[1..30]/Int requires acknowledgement.

Explanations: -

Value range: 0 (No)

1 (Yes)

Pre-assignment: 0 (No)

Unit: -

Meaning of parameter values: **0 (No)** Auxiliary function does not require acknowledgement

	1 (Yes)	Auxiliary function requires acknowledgement
Access rights:	MTB:	Write
	Setter:	Read
	Operator:	Read
Enabling:	System reset (SR)	
Dependency:	-	
Macoda ID:	3010 00090	
Data type:	Byte0_1_t (byte)	
	2 byte integer, unsigned for MACODA compatibility	

Index

Parameter ID:	IndInt	
Path:	AUXF/ClassDef/Class[1..30]/Int/IndInt	
Purpose:	Specifies the "index" of an integer-coded auxiliary function defined under ../Class[1..30]/Int.	
Explanations:	Their index determines the order in which several auxiliary functions belonging to a specific class of auxiliary functions are stored in the marker filed of class Class[1..30]/Int.	
Value range:	0 .. 32	
Pre-assignment:	0	../Class[1..30]/Int/IndInt
Unit:	-	
Meaning of parameter values:	Index of the integer-coded auxiliary function. The highest index is dependent on the number of existing integer-coded auxiliary functions in the class Class[1..30]/Int (parameter UsabAuxFunc in the folder AUXF/NofAuxFunc/)	
Access rights:	MTB:	Write
	Setter:	Read
	Operator:	Read
Enabling:	System reset (SR)	
Dependency:	-	
Macoda ID:	3010 00080	
Data type:	Byte0_32_t (byte)	
	2 byte integer, unsigned for MACODA compatibility	

Float-coded auxiliary functions of auxiliary function class[1..30]

Acknowledgement requirement

Parameter ID:	AcknFloat	
Path:	AUXF/ClassDef/Class[1..30]/Float/AcknFloat	
Purpose:	Specifies whether a float-coded auxiliary function defined under Class[1..30]/Float requires acknowledgement.	
Explanations:	-	
Value range:	0 (No)	
	1 (Yes)	
Pre-assignment:	0 (No)	

Auxiliary functions

Unit:	-
Meaning of parameter values:	0 (No) Auxiliary function does not require acknowledgement 1 (Yes) Auxiliary function requires acknowledgement
Access rights:	MTB: Write Setter: Read Operator: Read
Enabling:	System reset (SR)
Dependency:	-
Macoda ID:	3010 00090
Data type:	Byte0_1_t (byte) 2 byte integer, unsigned for MACODA compatibility
Index	
Parameter ID:	IndFloat
Path:	AUXF/ClassDef/Class[1..30]/Float/IndFloat
Purpose:	Specifies the "index" of a float-coded auxiliary function defined under ../Class[1..30]/Float.
Explanations:	Their index determines the order in which several auxiliary functions belonging to a specific class of auxiliary functions are stored in the marker filed of class Class[1..30]/Float.
Value range:	0 .. 32
Pre-assignment:	0 ../Class[1..30]/Float/IndFloat
Unit:	-
Meaning of parameter values:	Index of the float-coded auxiliary function. The highest index is dependent on the number of existing float-coded auxiliary functions in the class Class[1..30]/Float (parameter UsabAuxFunc in the folder AUXF/NofAux-Func/).
Access rights:	MTB: Write Setter: Read Operator: Read
Enabling:	System reset (SR)
Dependency:	-
Macoda ID:	3010 00080
Data type:	Byte0_32_t (byte) 2 byte integer, unsigned for MACODA compatibility

Double-coded auxiliary functions of auxiliary function class[1..30]

Acknowledgement requirement

Parameter ID:	AcknDouble
Path:	AUXF/ClassDef/Class[1..30]/Double/AcknDouble
Purpose:	Specifies whether an double-coded auxiliary function defined under Class[1..30]/Double requires acknowledgement.
Explanations:	-
Value range:	0 (No) 1 (Yes)

Pre-assignment:	0 (No)	
Unit:	-	
Description of parameter values	0 (No)	Auxiliary function does not require acknowledgement
	1 (Yes)	Auxiliary function requires acknowledgement
Access rights:	MTB:	Write
	Setter:	Read
	Operator:	Read
Enabling:	System reset (SR)	
Dependency:	-	
Macoda ID:	3010 00090	
Data type:	Byte0_1_t (byte)	
	2 byte integer, unsigned for MACODA compatibility	
Index		
Parameter ID:	IndDouble	
Path:	AUXF/ClassDef/Class[1..30]/Double/IndDouble	
Purpose:	Specifies the "index" of a double-coded auxiliary function defined under ../Class[1..30]/Double.	
Explanations:	Their index determines the order in which several auxiliary functions belonging to a specific class of auxiliary functions are stored in the marker filed of class Class[1..30]/Double.	
Value range:	0 .. 32	
Pre-assignment:	0	../Class[1..30]/Double/IndDouble
Unit:	-	
Meaning of parameter values:	Index of the double-coded auxiliary function. The highest index is dependent on the number of existing double-coded auxiliary functions in the class Class[1..30]/Double (parameter UsabAuxFunc in the folder AUXF/NofAux-Func/).	
Access rights:	MTB:	Write
	Setter:	Read
	Operator:	Read
Enabling:	System reset (SR)	
Dependency:	-	
Macoda ID:	3010 00080	
Data type:	Byte0_32_t (byte)	
	2 byte integer, unsigned for MACODA compatibility	

17.6.4 Acknowledgement behavior for bit-coded auxiliary functions (BitAckn)

General information

The acknowledgement requirement of every bit of a bit-coded auxiliary function is specified channel-independently in the parameters **Byte000** to **Byte191**, which cover an address range of 1536 bits.

The address range is divided into 12 subsequent segments, each of with 16 bytes, each of which has 8 bits (12x16x8 bits = 1536 bits) (refer to the following figure and [chapter 17.6.10 "Parameterizing example: Auxiliary functions" on page 327](#)).

Acknowledgment requirement from Bit 0000 to Bit 0127 (Byte000_015)

Parameter ID: Byte000 to Byte015

Path: AUXF/BitAckn/Byte000_015/Byte000 to .../Byte015

Purpose: Specifies channel-independently from bit 0000 to bit 0127 in Byte000_015 the acknowledgement requirement of bit-coded auxiliary functions.

Explanations: -

Value range: 00000000 .. 11111111

Pre-assignment:

00000000	AUXF/BitAckn/Byte000_015/Byte000 (bit 0007...0000)
00000000	AUXF/BitAckn/Byte000_015/Byte001 (bit 0015...0008)
00001000	AUXF/BitAckn/Byte000_015/Byte002 (bit 0023...0016)
00000000	AUXF/BitAckn/Byte000_015/Byte003 (bit 0031...0024)
00000000	AUXF/BitAckn/Byte000_015/Byte004 (bit 0039...0032)
00011111	AUXF/BitAckn/Byte000_015/Byte005 (bit 0047...0040)
00000001	AUXF/BitAckn/Byte000_015/Byte006 (bit 0055...0048)
00000000	AUXF/BitAckn/Byte000_015/Byte007 (bit 0063...0056)
...	...
00000000	AUXF/BitAckn/Byte000_015/Byte015 (bit 0127...0120)

Unit: -

Meaning of parameter values: Bit mask (8 bit) of byte xxx (see following example):
 A "0" in one place in the bit mask means "acknowledgement not required"
 A "1" in one place in the bit mask means "acknowledgement required"

Access rights: **MTB:** Write

Setter: Read

Operator: Read

Enabling: System reset (SR)

Dependency: -

Macoda ID: 3010 00020

Data type: Str8_8_t (isoLatin1String)

4 byte integer, unsigned for MACODA compatibility

Example:

Bit-coded auxiliary functions. Acknowledgement requirement for M19, M48 and M90

Defined is a bit-coded auxiliary function named "M" and a bit array size of 96 bits:

Start bit address: 0 (AUXF/ClassDef/ClassDef[1]/Bit/StartBitAddr);

End bit address: 95 (AUXF/ClassDef/ClassDef[1]/Bit/EndBitAddr).

The individual bits can be programmed using the NC syntax "M0" up to "M95":

Programming code of first bit: 0 (AUXF/ClassDef/ClassDef[1]/Bit/StartCode);

Programming code of last bit: 95 (AUXF/ClassDef/ClassDef[1]/Bit/EndCode)

M19, M48 and M90 are to require acknowledgement:

Flag field for bit-coded auxiliary functions					Flag field for bit-coded auxiliary functions				
	HiFu-Bit No.	Mask Bit No.	Bit Mask	Mask Bit No.		HiFu-Bit No.	Mask Bit No.	Bit Mask	Mask Bit No.
Byte 000_015 in folder AUXF/BitAckn	Byte 000	7	00000000	0	...Byte 176_191 in folder AUXF/BitAckn	Byte 176	1415	00000000	1408
	..	15	00000000	8		..	1423	00000000	1416
	..	23	00001000	16		..	1431	00000000	1424
	..	31	00000000	24		..	1439	00000000	1432
	..	39	00000000	32		..	1447	00000000	1440
	..	47	00000000	40		..	1455	00000000	1448
	..	55	00000001	48		..	1463	00000000	1456
	..	63	00000000	56		..	1471	00000000	1464
	..	71	00000000	64		..	1479	00000000	1472
	..	79	00000000	72		..	1487	00000000	1480
	..	87	00000000	80		..	1495	00000000	1488
	..	95	00000100	88		..	1503	00000000	1496
	..	103	00000000	96		..	1511	00000000	1504
	..	111	00000000	104		..	1519	00000000	1512
	..	119	00000000	112		..	1527	00000000	1520
	Byte 015	127	00000000	120		Byte 191	1535	00000000	1528

Fig. 17-9: Bit array for M19, M48 and M90

Acknowledgement requirement from bit 0128 to bit 1535 (Byte016 to Byte191)

Parameter ID: Byte016 to Byte191

Path: AUXF/BitAckn/Byte016_031 to .../Byte176_191

Purpose: Specifies channel-independently from bit 0128 to bit 1535 in Byte016 to Byte191 the acknowledgement requirement of bit-coded auxiliary functions.

Explanations: -

Value range: 00000000 .. 11111111

Pre-assignment: 00000000 AUXF/BitAckn/Byte016_031/Byte016 (bit 0135...0128)

...

00000000 AUXF/BitAckn/Byte176_191/Byte191 (bit 1535-1528)

Unit: -

Meaning of parameter values: Bit mask (8 bit) of byte xxx:

A "0" in one place in the bit mask means "acknowledgement not required"

A "1" in one place in the bit mask means "acknowledgement required"

Access rights: **MTB:** Write

Setter: Read

Operator: Read

Enabling: System reset (SR)

Dependency: -

Macoda ID: 3010 00020

Auxiliary functions

Data type: Str8_8_t (isoLatin1String)
4 byte integer, unsigned for MACODA compatibility

17.6.5 Auxiliary function group definition (GrDef)

General information

Up to 64 auxiliary function groups Gr[1..64] can be defined here and individual auxiliary functions GrAuxFunc[1..15] assigned to these auxiliary function groups.

Channel dependency

Parameter ID: GrChDep
Path: AUXF/GrDef/Gr[1..64]/GrChDep
Purpose: Defines whether each auxiliary function group Gr[1..64] is channel-independent or channel-dependent.

Explanations: -

Value range: 0 (behavior specified automatically)
1 (channel-independent auxiliary function)
2 (channel-dependent auxiliary function)
(MACODA compatibility: 0 $\triangleq$ <empty>, 1 $\triangleq$ "A:", 2 $\triangleq$ "S:")

Pre-assignment: 1 (channel-independent auxiliary function) $\rightarrow$ AUXF/GrDef/Gr[1]/GrChDep
1 (channel-independent auxiliary function) $\rightarrow$ AUXF/GrDef/Gr[2]/GrChDep
2 (channel-dependent auxiliary function) $\rightarrow$ AUXF/GrDef/Gr[3]/GrChDep
0 (behavior specified automatically) $\rightarrow$ AUXF/GrDef/Gr[4..64]/GrChDep

Unit: -

Meaning of parameter values: 0 Behavior of the auxiliary function group Gr[1..64] is set automatically, depending on whether the contained auxiliary functions are channel-independent or channel-dependent.

When mixed, the group cannot be used, and a warning is generated.

1 Auxiliary function group Gr[1..64] is channel-independent

2 Auxiliary function group Gr[1..64] is channel dependent

Access rights: **MTB:** Write

Setter: Read

Operator: Read

Enabling: System restart (CS)

Dependency: -

Macoda ID: 3010 01000 [1] .. 3010 01063 [1]

Data type: Byte0_2_t (byte)
8 byte string for MACODA compatibility

Auxiliary function[1..15]

Parameter ID: GrAuxFunc[1..15]

Path: AUXF/GrDef/Gr[1..64]/GrAuxFunc[1..15]

Purpose: Defines the syntax of the individual auxiliary functions GrAuxFunc[1..15] in the auxiliary function groups Gr[1..64].

Explanations: -

Value range:	{Text}
Pre-assignment:	M3 .../Gr[1]/GrAuxFunc[1] M4 .../Gr[1]/GrAuxFunc[2] M5 .../Gr[1]/GrAuxFunc[3] M19 .../Gr[1]/GrAuxFunc[4] - .../Gr[1]/GrAuxFunc[5..15] S .../Gr[2]/GrAuxFunc[1] - .../Gr[2]/GrAuxFunc[2..15] T .../Gr[3]/GrAuxFunc[1] - .../Gr[3]/GrAuxFunc[2..15] - .../Gr[4..64]/GrAuxFunc[1..15]
Unit:	-
Meaning of parameter values:	Name of the auxiliary function [1..15] in the auxiliary function group Gr[1..64].
Access rights:	MTB: Write Setter: Read Operator: Read
Enabling:	System restart (CS)
Dependency:	-
Macoda ID:	3010 01000 [2..16] .. 3010 01063 [2..16]
Data type:	Byte0_2_t (byte) 8 byte string for MACODA compatibility

17.6.6 Global output behavior (OutBehav)

General information

The parameter **OutBehavMode** specifies how programmed auxiliary functions are transferred to the PLC application program.

Output type

Parameter ID:	OutBehavMode
Path:	AUXF/OutBehav/OutBehavMode
Purpose:	Defines the method of transferring the auxiliary functions programmed in an NC block to the PLC application program.
Explanations:	-
Value range:	0 (Only bit-coded auxiliary functions synchronously output to PLC) 1 (Output all auxiliary functions synchronously to PLC)
Pre-assignment:	1 (Output all auxiliary functions synchronously to PLC)
Unit:	-
Meaning of parameter values:	0 Only the bit-coded auxiliary functions of an NC block are collected and then transferred together to the PLC. All auxiliary functions coded otherwise are transferred immediately. 1 All auxiliary functions of an NC block are transferred synchronously to the PLC.
Access rights:	MTB: Write Setter: Read Operator: Read

Auxiliary functions

Enabling: System reset (SR)
Dependency: -
Macoda ID: 3010 00015
Data type: Byte0_1_t (byte)
 2 byte integer, unsigned for MACODA compatibility

17.6.7 Number of usable auxiliary functions (NofAuxFunc)

General information

Parameter "Usable auxiliary functions" UsabAuxFunc (in folder AUXF/NofAuxFunc/) reduces the usable number of all auxiliary functions defined in folders "AUXF/ClassDef/Class[1..30]" as needed. This way, auxiliary functions which are, for instance, elements of extended machine functions can be quickly disabled or enabled.

The remaining parameters in the folders "AUXF/NofAuxFun/ChDep" and "AUXF/NofAuxFun/ChIndep" specify how many channel-dependent and channel-independent auxiliary functions from the individual auxiliary function classes can be used. For bit-coded auxiliary functions, the number of all individual bits is specified.

Number of auxiliary functions

Usable auxiliary functions

Parameter ID: UsabAuxFunc
Path: AUXF/NofAuxFunc/UsabAuxFunc
Purpose: Specifies how many auxiliary functions are used in the individual auxiliary function classes AUXF/ClassDef/Class[1..30].
Explanations: With this parameter, set the usable number of all auxiliary functions defined in the individual auxiliary function classes AUXF/ClassDef/Class[1..30] as needed. This way, auxiliary functions which are, for instance, elements of extended machine functions can be quickly disabled or enabled.
Value range: 0 .. 1536
Pre-assignment: 3
Unit: -

Meaning of parameter values: Total number of all usable auxiliary functions defined in the auxiliary function classes in the system.

If the number of auxiliary functions existing there exceeds the number specified here, only the initially defined auxiliary functions are applied (related to the index of the individual parameters).

"Usable auxiliary functions": UsabAuxFunc	"Name" of bit auxiliary function: AuxFuncName	Auxiliary function is effective
3	M	Yes
	M500	Yes
	M999	Yes
	O	No
	S?	No

Tab. 17-1: Example of reducing usable auxiliary functions

Access rights: MTB: Write
 Setter: Read

Operator: Read
Enabling: System restart (CS)
Dependency: -
Macoda ID: 3010 00010 [1]
Data type: NofAuxFunc_t (int)
 4 byte integer, unsigned for MACODA compatibility

Channel-dependent auxiliary functions (ChDep)

Bit auxiliary function

Parameter ID: ChDepBit
Path: AUXF/NofAuxFunc/ChDep/ChDepBit
Purpose: Specifies how many individual bits of the bit-coded auxiliary functions are used.
Explanations: -
Value range: 0 .. 1536
Pre-assignment: 304
Unit: -
Meaning of parameter values: Number of individual bits of the bit-coded auxiliary functions per channel
Access rights: **MTB:** Write
Setter: Read
Operator: Read
Enabling: System restart (CS)
Dependency: -
Macoda ID: 3010 00010 [2]
Data type: NofAuxFunc_t (int)
 4 byte integer, unsigned for MACODA compatibility

BCD32 auxiliary function

Parameter ID: ChDepBcd32
Path: AUXF/NofAuxFunc/ChDep/ChDepBcd32
Purpose: Specifies the number of channel-dependent Bcd32-coded auxiliary functions used.
Explanations: -
Value range: 0 .. 1536
Pre-assignment: 0
Unit: -
Meaning of parameter values: Number of channel-dependent Bcd32-coded auxiliary functions
Access rights: **MTB:** Write
Setter: Read
Operator: Read
Enabling: System restart (CS)
Dependency: -
Macoda ID: 3010 00010 [5]
Data type: NofAuxFunc_t (int)

Auxiliary functions

4 byte integer, unsigned for MACODA compatibility

Integer auxiliary function

Parameter ID: ChDepInt

Path: AUXF/NofAuxFunc/ChDep/ChDepInt

Purpose: Specifies the number of channel-dependent integer-coded auxiliary functions used.

Explanations: -

Value range: 0 .. 1536

Pre-assignment: 1

Unit: -

Meaning of parameter values: Number of channel-dependent integer-coded auxiliary functions

Access rights: **MTB:** Write

Setter: Read

Operator: Read

Enabling: System restart (CS)

Dependency: -

Macoda ID: 3010 00010 [10]

Data type: NofAuxFunc_t (int)

4 byte integer, unsigned for MACODA compatibility

Float auxiliary function

Parameter ID: ChDepFloat

Path: AUXF/NofAuxFunc/ChDep/ChDepFloat

Purpose: Specifies the number of channel-dependent float-coded auxiliary used.

Explanations: -

Value range: 0 .. 1536

Pre-assignment: 0

Unit: -

Meaning of parameter values: Number of channel-dependent float-coded auxiliary functions

Access rights: **MTB:** Write

Setter: Read

Operator: Read

Enabling: System restart (CS)

Dependency: -

Macoda ID: 3010 00010 [11]

Data type: NofAuxFunc_t (int)

4 byte integer, unsigned for MACODA compatibility

Double auxiliary function

Parameter ID: ChDepDouble

Path: AUXF/NofAuxFunc/ChDep/ChDepDouble

Purpose: Specifies the number of channel-dependent double-coded auxiliary functions used.

Explanations:	-
Value range:	0 .. 1536
Pre-assignment:	0
Unit:	-
Meaning of parameter values:	Number of channel-dependent double-coded auxiliary functions
Access rights:	MTB: Write
	Setter: Read
	Operator: Read
Enabling:	System restart (CS)
Dependency:	-
Macoda ID:	3010 00010 [12]
Data type:	NofAuxFunc_t (int)
	4 byte integer, unsigned for MACODA compatibility

Channel-independent auxiliary functions (ChInDep)

BCD32 auxiliary function

Parameter ID:	ChIndepBcd32
Path:	AUXF/NofAuxFunc/ChInDep/ChIndepBcd32
Purpose:	Specifies the number of channel-independent Bcd32-coded auxiliary functions used.
Explanations:	-
Value range:	0 .. 1536
Pre-assignment:	0
Unit:	-
Meaning of parameter values:	Number of channel-independent Bcd32-coded auxiliary functions
Access rights:	MTB: Write
	Setter: Read
	Operator: Read
Enabling:	System restart (CS)
Dependency:	-
Macoda ID:	3010 00010 [3]
Data type:	NofAuxFunc_t (int)
	4 byte integer, unsigned for MACODA compatibility

Integer auxiliary function

Parameter ID:	ChIndepInt
Path:	AUXF/NofAuxFunc/ChInDep/ChIndepInt
Purpose:	Specifies the number of channel-independent integer-coded auxiliary functions used.
Explanations:	-
Value range:	0 .. 1536
Pre-assignment:	0
Unit:	-

Auxiliary functions

Meaning of parameter values: Number of channel-independent integer-coded auxiliary functions

Access rights: **MTB:** Write
Setter: Read
Operator: Read

Enabling: System restart (CS)

Dependency: -

Macoda ID: 3010 00010 [7]

Data type: NofAuxFunc_t (int)
 4 byte integer, unsigned for MACODA compatibility

Float auxiliary function

Parameter ID: ChIndepFloat

Path: AUXF/NofAuxFunc/ChInDep/ChIndepFloat

Purpose: Specifies the number of channel-independent float-coded auxiliary functions used.

Explanations: -

Value range: 0 .. 1536

Pre-assignment: 1

Unit: -

Description of parameter values: Number of channel-independent float-coded auxiliary functions

Access rights: **MTB:** Write
Setter: Read
Operator: Read

Enabling: System restart (CS)

Dependency: -

Macoda ID: 3010 00010 [8]

Data type: NofAuxFunc_t (int)
 4 byte integer, unsigned for MACODA compatibility

Double auxiliary function

Parameter ID: ChIndepDouble

Path: AUXF/NofAuxFunc/ChInDep/ChIndepDouble

Purpose: Specifies the number of channel-independent double-coded auxiliary functions used.

Explanations: -

Value range: 0 .. 1536

Pre-assignment: 0

Unit: -

Meaning of parameter values: Number of channel-independent double-coded auxiliary functions

Access rights: **MTB:** Write
Setter: Read
Operator: Read

Enabling: System restart (CS)

Dependency: -

Macoda ID: 3010 00010 [9]

Data type: NofAuxFunc_t (int)
4 byte integer, unsigned for MACODA compatibility

17.6.8 Addresses of auxiliary functions in the PLC interface (Plclf)

General information

The parameter in the folders AUXF/Plclf/ChDep/Chlf[1..60]/.. Specify the addresses in the PLC interface for all auxiliary function classes.

Addresses in the channel interface Chlf[1..60] of channel-dependent auxiliary functions

Address of the bit-coded auxiliary functions

Parameter ID: AddrChDepBit
Path: AUXF/Plclf/ChDep/Chlf[1..60]/AddrChDepBit
Purpose: Specifies the starting addresses of the ranges for bit auxiliary functions for each channel (channel interface Chlf[1..60]).
Explanations: When specifying individual address ranges, plan gaps for a future expansion of bit-coded auxiliary functions. Thus, addressing of existing bit-coded auxiliary functions in the PLC program do not have to be adjusted if additional bit-coded auxiliary functions are implemented in the system.
When additional channels are applied, the corresponding new marker fields can be placed right in some unused PLC address ranges.



In the absence of sufficiently large gaps, ensure that existing ranges do not overlap with new ones when adding new auxiliary functions!

Value range: -1, total integer range
Pre-assignment: **4700** .../ChDep//Chlf[1]/AddrChDepBit
4738 .../ChDep//Chlf[2]/AddrChDepBit
4776 .../ChDep//Chlf[3]/AddrChDepBit
-1 .../ChDep//Chlf[4..60]/AddrChDepBit
Unit: -
Meaning of parameter values: **-1** The interface is not used; signals are not copied
>= 0 Starting addresses for channel-dependent bit-coded auxiliary functions
Access rights: **MTB:** Write
Setter: Read
Operator: Read
Enabling: System restart (CS)
Dependency: -
Macoda ID: 2060 00009
Data type: int
4 byte integer, unsigned for MACODA compatibility

Example:

Starting addressing relating to channel interface in AddrChDepBit

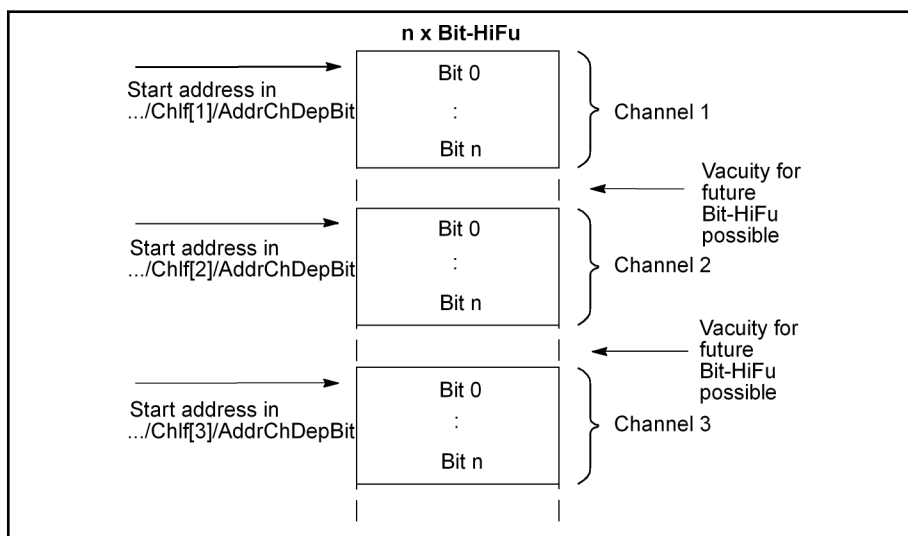


Fig. 17-10: Extension options for bit-coded auxiliary functions

Address of the BCD32-coded auxiliary functions

Parameter ID: AddrChDepBcd32

Path: AUXF/Plcif/ChDep/ChIf[1..60]/AddrChDepBcd32

Purpose: Specifies the starting addresses of the ranges for BCD32 auxiliary functions for each channel (channel interface ChIf[1..60]).

Explanations: -

Value range: -1, total integer range

Pre-assignment: -1 .../ChDep/ChIf[1..60]/AddrChDepBcd32

Unit: -

Meaning of parameter values: -1 The interface is not used; signals are not copied
 >= 0 Starting addresses for channel-dependent Bcd32-coded auxiliary functions

Access rights: **MTB:** Write

Setter: Read

Operator: Read

Enabling: System restart (CS)

Dependency: -

Macoda ID: 2060 00010

Data type: int

4 byte integer, unsigned for MACODA compatibility

Address of the integer-coded auxiliary functions

Parameter ID: AddrChDepInt

Path: AUXF/Plcif/ChDep/ChIf[1..60]/AddrChDepInt

Purpose: Specifies the starting addresses of the ranges for integer auxiliary functions for each channel (channel interface ChIf[1..60]).

Explanations: -

Value range:	-1, total integer range	
Pre-assignment:	6200	.../ChDep/ChIf[1]/AddrChDepInt
	6216	.../ChDep/ChIf[2]/AddrChDepInt
	6232	.../ChDep/ChIf[3]/AddrChDepInt
	6248	.../ChDep/ChIf[4]/AddrChDepInt
	6264	.../ChDep/ChIf[5]/AddrChDepInt
	6280	.../ChDep/ChIf[6]/AddrChDepInt
	6296	.../ChDep/ChIf[7]/AddrChDepInt
	6312	.../ChDep/ChIf[8]/AddrChDepInt
	6328	.../ChDep/ChIf[9]/AddrChDepInt
	6348	.../ChDep/ChIf[10]/AddrChDepInt
	6344	.../ChDep/ChIf[11]/AddrChDepInt
	6360	.../ChDep/ChIf[12]/AddrChDepInt
	-1	.../ChDep/ChIf[13..60]/AddrChDepInt
Unit:	-	
Meaning of parameter values:	-1	The interface is not used; signals are not copied
	>= 0	Starting addresses for channel-dependent integer-coded auxiliary functions
Access rights:	MTB:	Write
	Setter:	Read
	Operator:	Read
Enabling:	System restart (CS)	
Dependency:	-	
Macoda ID:	2060 00012	
Data type:	int	
	4 byte integer, unsigned for MACODA compatibility	
Address of the float-coded auxiliary functions		
Parameter ID:	AddrChDepFloat	
Path:	AUXF/PlcIf/ChDep/ChIf[1..60]/AddrChDepFloat	
Purpose:	Specifies the starting addresses of the ranges for float auxiliary functions for each channel (channel interface ChIf[1..60]).	
Explanations:	-	
Value range:	-1, total integer range	
Pre-assignment:	-1	.../ChDep/ChIf[1..60]/AddrChDepFloat
	-	-
Unit:	-	
Meaning of parameter values:	-1	The interface is not used; signals are not copied
	>= 0	Starting addresses for channel-dependent float-coded auxiliary functions
Access rights:	MTB:	Write
	Setter:	Read
	Operator:	Read
Enabling:	System restart (CS)	
Dependency:	-	

Auxiliary functions

Macoda ID: 2060 00013

Data type: int

4 byte integer, unsigned for MACODA compatibility

Address of the double-coded auxiliary functions

Parameter ID: AddrChDepDouble

Path: AUXF/Plclf/ChDep/Chlf[1..60]/AddrChDepDouble

Purpose: Specifies the starting addresses of the ranges for double-coded auxiliary functions for each channel (channel interface Chlf[1..60]).

Explanations: -

Value range: -1, total integer range

Pre-assignment: -1 .../ChDep/Chlf[1..60]/AddrChDepDouble

Unit: -

Meaning of parameter values:

-1	The interface is not used; signals are not copied
>= 0	Starting addresses for channel-dependent double-coded auxiliary functions

Access rights: **MTB:** Write

Setter: Read

Operator: Read

Enabling: System restart (CS)

Dependency: -

Macoda ID: 2060 00014

Data type: int

4 byte integer, unsigned for MACODA compatibility

Addresses of channel-independent auxiliary functions

Address of the BCD32 auxiliary functions

Parameter ID: AddrChIndepBcd32

Path: AUXF/Plclf/ChIndep/AddrChIndepBcd32

Purpose: Specifies the starting addresses of the PLC address range for channel-independent Bcd32-coded auxiliary functions.

Explanations: -

Value range: -1, total integer range

Pre-assignment: -1

Unit: -

Meaning of parameter values:

-1	The interface is not used; signals are not copied
>= 0	Starting address for channel-independent Bcd32-coded auxiliary functions

Access rights: **MTB:** Write

Setter: Read

Operator: Read

Enabling: System restart (CS)

Dependency: -

Macoda ID: 2060 00003

Data type:	int	
	4 byte integer, unsigned for MACODA compatibility	
Address of the integer-coded auxiliary functions		
Parameter ID:	AddrChIndepInt	
Path:	AUXF/PlcIf/ChInDep/AddrChIndepInt	
Purpose:	Specifies the starting address of the PLC address range for channel-independent integer-coded auxiliary functions.	
Explanations:	-	
Value range:	-1, total integer range	
Pre-assignment:	-1	
Unit:	-	
Meaning of parameter values:	-1	The interface is not used; signals are not copied
	>= 0	Starting address for channel-independent integer-coded auxiliary functions
Access rights:	MTB:	Write
	Setter:	Read
	Operator:	Read
Enabling:	System restart (CS)	
Dependency:	-	
Macoda ID:	2060 00020	
Data type:	int	
	4 byte integer, signed for MACODA compatibility	
Address of the float-coded auxiliary functions		
Parameter ID:	AddrChIndepFloat	
Path:	AUXF/PlcIf/ChInDep/AddrChIndepFloat	
Purpose:	Specifies the starting address of the PLC address range for channel-independent float-coded auxiliary functions.	
Explanations:	-	
Value range:	-1, total integer range	
Pre-assignment:	6024	
Unit:	-	
Description of parameter values	-1	The interface is not used; signals are not copied
	>= 0	Starting address for channel-independent float-coded auxiliary functions
Access rights:	MTB:	Write
	Setter:	Read
	Operator:	Read
Enabling:	System restart (CS)	
Dependency:	-	
Macoda ID:	2060 00020	
Data type:	int	
	4 byte integer, signed for MACODA compatibility	

Address of the double-coded auxiliary functions

Parameter ID:	AddrChIndepDouble	
Path:	AUXF/PlcIf/ChInDep/AddrChIndepDouble	
Purpose:	Specifies the starting address of the PLC address range for channel-independent double-coded auxiliary functions.	
Explanations:	-	
Value range:	-1, total integer range	
Pre-assignment:	-1	
Unit:	-	
Meaning of parameter values:	-1	The interface is not used; signals are not copied
	>= 0	Starting address for channel-independent double-coded auxiliary functions
Access rights:	MTB:	Write
	Setter:	Read
	Operator:	Read
Enabling:	System restart (CS)	
Dependency:	-	
Macoda ID:	2060 00020	
Data type:	int	
	4 byte integer, unsigned for MACODA compatibility	

17.6.9 Defining a new auxiliary function

The following section deals with how to parameterize a new auxiliary function.



An example of parameterization of an entire configuration of auxiliary functions is given in [chapter 17.6.10 "Parameterizing example: Auxiliary functions" on page 327](#).

Proceed as follows:

1. Open the necessary folder **AUXF/ClassDef/Class[1..30]** and, in the parameter **AuxFuncName**, define the NC syntax of the desired auxiliary function. Please note the explanations given in [chapter 17.6.1 "Auxiliary function basics" on page 295](#).
2. To do so, select the necessary folder **AUXF/ClassDef/Class[1..30]/** and, with the right mouse button, select the desired auxiliary function type with the function "Convert to..." from the context menu. Depending on the selection, one of the following subfolders is displayed:
 - Bit
 - Bcd32
 - Int
 - Float
 - Double

Always define the auxiliary function type first and then the range of action
3. In the parameter **AuxFuncChDep**, specify whether the auxiliary function should be channel-dependent or channel-independent (range of action).

Note that for bit-coded auxiliary functions, only the "channel-dependent" setting is supported.

4. Open the folder **AUXF/NofAuxFunc** and adjust the number of the auxiliary functions defined in the folders

AUXF/ClassDef/Class[1..30] in the parameter **UsabAuxFunc**. If an auxiliary function or a coherent block of auxiliary functions should be hidden as seen from the highest index, reduce this parameter accordingly.

5. Adjust the number of channel-dependent or channel-independent auxiliary functions in the folder

AUXF/NofAuxFunc/ChDep or **AUXF/NofAuxFunc/ChIndep**.

For bit-coded auxiliary functions, enter at least the number of all individual bits used (highest bit address used + 1). It is recommended to set this number to the next highest value divisible by 8 (byte raster). Thus, you are also prepared for future extensions to the bit-coded auxiliary functions.

6. Depending on the auxiliary function type chosen, continue with the following points:

- Bit-coded: Point 7
- 32-bit BCD-coded: Point 9
- Integer-coded: Point 10
- Float-coded: Point 11
- Double-coded: Point 12

7. For bit-coded auxiliary functions only:

- Open the required folder

AUXF/ClassDef/Class[1..30]/Bit.

- In the parameter **StartCode**, enter the programming code of the first bit, and in **EndCode**, the programming code of the last bit of the auxiliary function.
- In the parameter **StartBitAddr**, enter the address of the first bit, and in **EndBitAddr**, the address of the last bit of the auxiliary function.

Example: Bit-coded auxiliary function named "RUDI"

Address of the first bit: 301

Address of the last bit: 320

Program code of the first bit: 1

Program code of the last bit: 20

⇒ Programming of "RUDI1" sets bit 301 in the flag field.

⇒ Programming of "RUDI20" sets bit 320 in the flag field.

8. For bit-coded auxiliary functions only:

Specify the acknowledgement requirement of the auxiliary function. To do so, open the necessary folder

AUXF/BitAckn and, in the appropriate subfolder, select the parameter **Bytexxx** in which the bit to be changed is found. The parameter values begin with the higher order bit. For more explanations, see

Continue with point 13.

9. For 32-bit BCD-coded auxiliary functions only:

- Open the required folder

AUXF/ClassDef/Class[1..32]/Bcd32.

Auxiliary functions

- In the parameter **AcknBcd32**, specify the acknowledgement requirement of the auxiliary function.
- In the parameter **PreIPlaceBcd32**, specify the number of integer places (to the left of the decimal point), and in **DecPlaceBcd32**, the number of decimal places of the auxiliary function (refer to the explanation: on page...).
- In the parameter **IndBcd32**, enter the index of the auxiliary function. This allows to arrange the order in which several auxiliary functions of the same auxiliary function class are stored in the marker field of this class. The index is independent of the number of channel-dependent or channel-independent Bcd32-coded auxiliary functions in the folder

AUXF/NofAuxFunc/ChDep or **AUXF/NofAuxFunc/ChIndep**.

Example: The channel-dependent 32-bit BCD auxiliary functions

- STEVE (Index: 2),
- SANDY (Index: 0)

are stored in the marker field of this auxiliary function class in the order

- 0. SANDY,
- 1. <64 bit free>
- 2. STEFAN

. Please note that STEVE can only contain the index 2 if at least 3 channel-dependent auxiliary functions are defined in the parameter ChDepBcd32 in the folder AUXF/NofAuxFunc/ChDep.

Continue with point 13.

10. For integer-coded auxiliary functions only:

- Open the required folder
AUXF/ClassDef/Class[1..30]/Int.
- In the parameter **AcknInt**, specify the acknowledgement requirement of the auxiliary function.
- In the parameter **IndInt**, enter the index of the auxiliary function. This allows to arrange the order in which several auxiliary functions of the same auxiliary function class are stored in the marker field of this class. The index is independent of the number of channel-dependent or channel-independent integer-coded auxiliary functions in the folder

AUXF/NofAuxFunc/ChDep or **AUXF/NofAuxFunc/ChIndep**.

Continue with point 13.

11. For float-coded auxiliary functions only:

- Open the required folder **AUXF/ClassDef/Class[1..30]/Float.**
- In the parameter **AcknFloat**, specify the acknowledgement requirement of the auxiliary function.
- In the parameter **IndFloat**, enter the index of the auxiliary function. This allows to arrange the order in which several auxiliary functions of the same auxiliary function class are stored in the marker field of this class. The index is independent of the number of channel-dependent or channel-independent float-coded auxiliary functions in the folder

AUXF/NofAuxFunc/ChDep or **AUXF/NofAuxFunc/ChIndep**.

Continue with point 13.

12. For double-coded auxiliary functions only:

- Open the required folder
AUXF/ClassDef/Class[1..30]/Double.
- In the parameter **AcknDouble**, specify the acknowledgement requirement of the auxiliary function.
- In the parameter **IndDouble**, enter the index of the auxiliary function. This allows to arrange the order in which several auxiliary functions of the same auxiliary function class are stored in the marker field of this class. The index is independent of the number of channel-dependent or channel-independent double-coded auxiliary functions in the folder

AUXF/NofAuxFunc/ChDep or **AUXF/NofAuxFunc/ChIndep.**

13. Open the folder **AUXF/OutBehav** and specify in the parameter **OutBehavMode** how the programmed auxiliary function should be transferred to the PLC application program.

14. For a channel-dependent auxiliary function, open the folder

AUXF/Plclf/ChDep/ChlIf[1..60] of the desired channel, or, for a channel-independent auxiliary function, open the folder **AUXF/Plclf/ChIndep.** Adjust the address of the auxiliary function in the PLC interface in the corresponding parameter as necessary (e.g. **AddrChDepBcd32**).

15. Only if assignment to an auxiliary function group is desired:

Open the necessary folder **AUXF/GrDef/Gr[1..64]** and supplement the syntax of the new auxiliary function (e.g. M19) in the parameter **GrAuxFunc[i]**.

16. The definition of an auxiliary function is now completed.

17.6.10 Parameterizing example: Auxiliary functions

Task: Parameterize the auxiliary functions M (M0..M299), M (M500), M (M999), O, S, S1, S2, V and T. The output of all auxiliary functions to the PLC application program is to be synchronous. The PLC interface is designed for maximum 4 channels. Auxiliary functions are not now assigned to auxiliary function groups. The auxiliary functions are to have the following properties in detail:

- M (M0..M299):** 1. auxiliary function, bit-coded, channel-dependent, bit array with a length of 300 bits, Bit addresses from 0..99, programming as M0..M299, M19, M40..M44, M48 and M95 require acknowledgement.
- M (M500):** 2. auxiliary function, bit-coded, channel-dependent, individual bit at bit address 300, programming as M500, no acknowledgement required.
- M (M999):** 3. auxiliary function, bit-coded, channel-dependent, individual bit at bit address 199, programming as M999, no acknowledgement required.
- O:** 4. auxiliary function, bit-coded, channel-dependent, bit array with 20 bits in length, Bit addresses from 301..320, programming as O1..O20, all bits require acknowledgement.

Auxiliary functions

- S:** 5. auxiliary function, float-coded, channel-independent,
is in first position of the marker area of the channel-independent float auxiliary functions, i.e. index = 0,
no acknowledgement required.
- S1:** 6. auxiliary function, float-coded, channel-dependent,
is in first position of the marker area of the channel-dependent float auxiliary functions, i.e. index = 0,
acknowledgement required.
- S2:** 7. auxiliary function, float-coded, channel-dependent,
is in second position of the marker area of the channel-dependent float auxiliary functions, i.e. index = 1,
acknowledgement required.
- V:** 8. auxiliary function 32-bit BCD-coded, channel-independent,
8 integer places (to the left of the decimal point), no decimal places,
is in first position of the marker area of the channel-independent 32 bit BCD auxiliary functions, i.e. index = 0,
no acknowledgement required.
- T:** 9. auxiliary function, integer-coded, channel-dependent,
is in first position of the marker area of the channel-dependent integer auxiliary functions, i.e. index = 0,
no acknowledgement required.

Required parameterization: In NC Configurator, the following settings are made according to the approach described in [chapter 17.6.9 "Defining a new auxiliary function" on page 324](#) when parameterizing a "new auxiliary function":

Auxiliary Function Names: AUXF/ClassDef/Class[1]/AuxFuncName: M
/Class[2]/AuxFuncName: M
/Class[3]/AuxFuncName: M
/Class[4]/AuxFuncName: O
/Class[5]/AuxFuncName: S?
/Class[6]/AuxFuncName: S1?
/Class[7]/AuxFuncName: S2?
/Class[8]/AuxFuncName: V
/Class[9]/AuxFuncName: T

Auxiliary Function Type: AUXF/ClassDef/Class[1]: Convert to → Bit
/Class[2]: Convert to → Bit
/Class[3]: Convert to → Bit
/Class[4]: Convert to → Bit
/Class[5]: Convert to → Float
/Class[6]: Convert to → Float
/Class[7]: Convert to → Float
/Class[8]: Convert to → Bcd32
/Class[9]: Convert to → Int

The parameter `EnablAuxFunc` is automatically set to `Yes` for this auxiliary function.

Range of action: AUXF/ClassDef/Class[1]/AuxFuncChDep: channel-independent
/Class[2]/AuxFuncChDep: channel-dependent
/Class[3]/AuxFuncChDep: channel-dependent
/Class[4]/AuxFuncChDep: channel-dependent
/Class[5]/AuxFuncChDep: channel-independent
/Class[6]/AuxFuncChDep: channel-dependent
/Class[7]/AuxFuncChDep: channel-dependent
/Class[8]/AuxFuncChDep: channel-independent
/Class[9]/AuxFuncChDep: channel-dependent

Number of Auxiliary Functions: AUXF/NofAuxFunc/UsabAuxFunc: 9
 channel-dependent:
 AUXF/NofAuxFunc/ChDep/ChDepBit: 336



The highest bit address used within all bit auxiliary functions in this example is 320 (see auxiliary function "O"). The parameter ChDepBit could therefore be assigned the value 321. In order to be prepared for extensions to the bit auxiliary functions, however, and to set the end of the bit auxiliary function range to a byte raster, ChDepBit is assigned the value 336.

AUXF/NofAuxFunc/ChDep/ChDepBcd32: 0
/ChDepInt: 1
/ChDepFloat: 2
/ChDepDouble: 0

channel-independent:

AUXF/NofAuxFunc/ChIndep/ChIndepBcd32: 1
/ChIndepInt: 0
/ChIndepFloat: 1
/ChIndepDouble: 0

**Addresses, programming code
 and acknowledgement require-
 ment of M (M0..M299):**

AUXF/ClassDef/Class[1]/Bit/StartCode: 0
/EndCode: 299
/StartBitAddr: 0
/EndBitAddr: 299
 AUXF/BitAckn/Byte000_015/Byte002/: 00001000
 (Bit 19)
/Byte005/: 00011111
 (Bits 44..40)
/Byte006/: 00000001
 (Bit 48)
/Byte011/: 10000000
 (Bit 95)

**Addresses and programming code
 of M (500):**

AUXF/ClassDef/Class[2]/Bit/StartCode: 500
/EndCode: 500
/StartBitAddr: 300
/EndBitAddr: 300

Auxiliary functions

Addresses and programming code of M (999):	AUXF/ClassDef/Class[3]/Bit/StartCode: 999/EndCode: 999/StartBitAddr: 199/EndBitAddr: 199
Addresses, Programming Code and Acknowledgement Requirement of O:	AUXF/ClassDef/Class[4]/Bit/StartCode: 1/EndCode: 20/StartBitAddr: 301/EndBitAddr: 320 AUXF/BitAckn/Byte032_047/Byte037/: 11100000 (Bits 303..301)/Byte038/: 11111111 (Bits 311..304)/Byte039/: 11111111 (Bits 319..312)/Byte040/: 00000001 (Bit 320)
Acknowledgement Requirement and Index of S:	AUXF/ClassDef/Class[5]/Float/AcknFloat: no/IndFloat: 0
Acknowledgement Requirement and Index of S1:	AUXF/ClassDef/Class[6]/Float/AcknFloat: yes/IndFloat: 0
Acknowledgement Requirement and Index of S2:	AUXF/ClassDef/Class[7]/Float/AcknFloat: yes/IndFloat: 1
Acknowledgement Requirement, Integer/Decimal Places and Index of V:	AUXF/ClassDef/Class[8]/Bcd32/AcknBcd32: no/PrelPlaceBcd32: 8/DeciPlaceBcd32: 0/IndBcd32: 0
Acknowledgement Requirement, Integer/Decimal Places and Index of T:	AUXF/ClassDef/Class[9]/Int/AcknInt: no/IndInt: 0
Output behavior to PLC:	AUXF/OutBehav/OutBehavMode: All aux. functions synchro- nous...
Addresses in PLC interface:	Channel-dependent auxiliary functions: AUXF/PlcIf/ChDep/Ch[1]/AddrChDepBit: e.g. 5000/Ch[2]/AddrChDepBit: e.g. 5042/Ch[3]/AddrChDepBit: e.g. 5084/Ch[4]/AddrChDepBit: e.g. 5126/Ch[5..60]/AddrChDepBit: -1 AUXF/PlcIf/ChDep/Ch[1..60]/AddrChDepBcd32: -1 AUXF/PlcIf/ChDep/Ch[1]/AddrChDepInt: e.g. 5200/Ch[2]/AddrChDepInt: e.g. 5208/Ch[3]/AddrChDepInt: e.g. 5216/Ch[4]/AddrChDepInt: e.g. 5224/Ch[5..60]/AddrChDepInt: -1

AUXF/PlcIf/ChDep/Ch[1]/AddrChDepFloat: e.g. 5300
/Ch[2]/AddrChDepFloat: e.g. 5316
/Ch[3]/AddrChDepFloat: e.g. 5332
/Ch[4]/AddrChDepFloat: e.g. 5348
/Ch[5..60]/AddrChDepFloat: -1
 AUXF/PlcIf/ChDep/Ch[1..60]/AddrChDepDouble: -1

Channel-independent auxiliary functions:

AUXF/PlcIf/ChIndep/AddrChIndepBcd32: e.g. 5400
/AddrChIndepInt: -1
/AddrChIndepFloat: e.g. 5416
/AddrChIndepDouble: -1

Representation of marker fields

The following tables show how these auxiliary functions are displayed in the marker area of the PLC.

Channel-dependent bit-coded auxiliary functions:

Channel 1	Starting address (e.g. 5000) according to AUXF/PlcIf/ChDep/Ch[1]/AddrChDepBit	Bit 0	M0
		:	:
		Bit 199	M199 / M999
		:	:
		Bit 299	M299
		Bit 300	M500
		Bit 301	O1
		:	:
		Bit 320	O20
		:	:
		Bit 335	
...	...	...	...
Channel 4	Starting address (e.g. 5126) according to AUXF/PlcIf/ChDep/Ch[4]/AddrChDepBit	Bit 0	M0
		:	:
		Bit 199	M199 / M999
		:	:
		Bit 299	M299
		Bit 300	M500
		Bit 301	O1
		:	:
		Bit 320	O20
		:	:
		Bit 335	

Tab. 17-2: Bit-coded auxiliary functions M and O

Channel-dependent integer-coded auxiliary functions:

Auxiliary functions

Channel 1	Starting address (e.g. 5200) according to AUXF/Plclf/ChDep/Ch[1]/AddrChDepInt	Bit 0 (acknowledgement) Bit 1 : Bit 31 Bit 32 (int value) : Bit 63	T
...	...	...	...
Channel 4	Starting address (e.g. 5224) according to AUXF/Plclf/ChDep/Ch[4]/AddrChDepInt	Bit 0 (acknowledgement) Bit 1 : Bit 31 Bit 32 (int value) : Bit 63	T

Tab. 17-3: Integer-coded auxiliary function T

Channel-dependent float-coded auxiliary functions:

Channel 1	Starting address (e.g. 5300) according to AUXF/Plclf/ChDep/Ch[1]/AddrChDepFloat	Bit 0 (acknowledgement) Bit 1 : Bit 31 Bit 32 (float value) : Bit 63	S1
		Bit 64 (acknowledgement) Bit 65 : Bit 95 Bit 96 (float value) : Bit 127	S2
...	...	...	...

Channel 4	Starting address (e.g. 5348) according to AUXF/Plclf/ChDep/Ch[4]/AddrCh-DepFloat	Bit 0 (acknowledgement) Bit 1 : Bit 31 Bit 32 (float value) : Bit 63	S1
		Bit 64 (acknowledgement) Bit 65 : Bit 95 Bit 96 (float value) : Bit 127	S2

Tab. 17-4: Float-coded auxiliary functions S1 and S2

Channel-independent 32-bit BCD-coded auxiliary functions:

Starting address (e.g. 5400) according to AUXF/Plclf/ChIndep/AddrChIndepBcd32	Bit 0 (acknowledgement) Bit 1 (sign) : Bit 31 Bit 32 (BCD value) : Bit 63	V

Tab. 17-5: 32-bit BCD-coded auxiliary function V

Channel-independent float-coded auxiliary functions:

Starting address (e.g. 5416) according to AUXF/Plclf/ChIndep/AddrChIndepFloat	Bit 0 (acknowledgement) Bit 1 : Bit 31 Bit 32 (float value) : Bit 63	S

Tab. 17-6: Float-coded auxiliary function S

18 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

[chapter 10.3.3 "Axis coordinate system ACS " on page 156](#)

BCS

Basic coordinate system

[chapter 10.3.6 "Basic workpiece coordinate system BCS" on page 157](#)

MCS

Machine coordinate system

[chapter 10.3.4 "Machine coordinate system MCS" on page 157](#)

PCS

Program coordinate system

[chapter 10.3.7 "Program coordinate system \(PCS\)" on page 158](#)

TCP

Tool center point

TCS

Tool coordinate system

WCS

Workpiece coordinate system

[chapter 10.3.5 "Workpiece coordinate system WCS" on page 157](#)

Index

0 ... 9

2.5D-tool radius correction.....	213
3D-tool radius correction.....	236
32-bit-BCD-coded auxiliary functions.....	296

A

Abbreviations.....	7
About this documentation.....	1
Information representation.....	7
Validity of the documentation.....	1
Acceleration, maximum path.....	116
Accuracy.....	141
Exact stop.....	141
Radius adjustment for circles.....	149
AcknBcd32.....	304
AcknDouble.....	308
AcknFloat.....	307
AcknInt.....	306
Acknowledgement behavior for bit-coded auxiliary functions (BitAckn).....	309
Acknowledgement requirement - Bcd32-coded auxiliary functions.....	304
Acknowledgement requirement - Double-coded auxiliary functions.....	308
Acknowledgement requirement - Float-coded auxiliary functions.....	307
Acknowledgement requirement - Integer-coded auxiliary functions.....	306
Acknowledgement requirement from bit 0128 to bit 1535 (Byte016 to Byte191).....	311
ACS.....	156
Actuate system spindles via interface.....	268
Actuating system spindles via interface.....	268
AddrChDepBcd32.....	320
AddrChDepBit.....	319
AddrChDepDouble.....	322
AddrChDepFloat.....	321
AddrChDepInt.....	320
AddrChIndepBcd32.....	322
AddrChIndepDouble.....	324
AddrChIndepFloat.....	323
AddrChIndepInt.....	323
Address of the BCD32 auxiliary functions.....	322
Address of the BCD32-coded auxiliary functions.....	320
Address of the bit-coded auxiliary functions.....	319
Address of the double-coded auxiliary functions.....	322, 324
Address of the float-coded auxiliary functions.....	321, 323
Address of the integer-coded auxiliary functions.....	320, 323
Addresses in the channel interface	
ChIf[1..12] of channel-dependent auxiliary functions.....	319

Addresses of auxiliary functions in the PLC interface (PlcIf).....	319
Addresses of channel-independent auxiliary functions.....	322
Application	
Spindle/turret axis with motor sensor.....	53
Approach reference point.....	65
Approaching reference point	
Activate.....	70, 71
Apply.....	70, 71
Assign channel spindles to spindle groups.....	255
Assign system spindles to channel spindles.....	253
Assigning system spindles to channel spindles	
Apply.....	255
Asynchronous axes.....	153
Asynchronous subroutines.....	103
Activate.....	108
Activate automatic mode.....	108
Activate setup mode.....	108
Apply.....	107
Deactivate.....	112
Interrupt.....	105
ATrans	
Additive program coordinate offset	185
Automatic program reselection.....	99
Activate.....	100
Apply.....	99
Automatic program selection after system startup.....	100
Deactivate.....	100
AuxFuncChDep.....	301
AuxFuncName.....	299
Auxiliary axes.....	153
Auxiliary class definition (ClassDef).....	298
Auxiliary function basics.....	295
Auxiliary function classes.....	295
Auxiliary function group definition (GrDef).....	312
Auxiliary function groups.....	297
Auxiliary function types.....	298
Auxiliary function[1..15].....	312
Auxiliary functions.....	293
BCD-coded auxiliary functions.....	294
Bit-coded auxiliary functions.....	294
Channel-dependent auxiliary functions.....	293
Channel-independent auxiliary functions.....	293
Double-coded auxiliary functions.....	294
Float-coded auxiliary functions.....	294
Integer-coded auxiliary functions.....	294
Output response.....	295
Auxiliary functions (AUXF) – Basic parameters -.....	295
Axes.....	153
C-axis.....	41
Linear axis.....	18
Modulo axis.....	23

Index

Rotary axis.....	21
Turret.....	49
Axes and coordinates in channel	
Axis names.....	160
Channel index.....	161
Coordinate names.....	160
Rules for coordinate identifiers.....	163
Axes and coordinates in the channel	
Axis identifier in the channel.....	161
Coordinate identifier.....	162
Rules for coordinate index.....	166
Axes in channel.....	160
Channel axis name.....	160
Axes names.....	160
Axes, coordinates, coordinate systems....	153, 155
Axes and coordinates in channel.....	160
NC functions with respect to coordinate systems.....	166
System axes and system coordinates.....	153
Transformation between coordinate systems.....	158
Axis acceleration reserve.....	115
Axis coordinate system.....	156
Axis identifier in the channel.....	161
Axis jerk reserve.....	115
Axis jump ability.....	115
Activate.....	116
Apply.....	116
Axis jump acceleration	
Apply.....	116
Deactivate.....	116
Axis jump capability	
Deactivate.....	116
Axis zero offset tables.....	172
Axis-specific jerk limitation.....	124, 127
Activate.....	126
Applying.....	126
Deactivate.....	127

B

Basic workpiece coordinate system	157
BCD-coded auxiliary functions.....	294
BCD32 auxiliary function - Channel-independent auxiliary functions (ChInDep).....	317
BCD32 auxiliary functions - Channel-dependent auxiliary functions (ChDep).....	315
Bcd32-coded auxiliary functions.....	304
BCR	
Workpiece position correction.....	175
BCS.....	157
BcsCorr	
Workpiece position correction	175
Bit auxiliary function - Channel-dependent auxiliary functions (ChDep).....	315
Bit-coded auxiliary functions.....	294, 296, 302
Byte000 to Byte015.....	310
Byte016 to Byte191.....	311

C

C-axis.....	41
Cancel distance to go.....	101
Activate.....	102
Change in tool orientation, 3D-tool correction.....	237
Change plunging depth, 3D-tool correction.....	238
Change rate of the allowance value for the contour, 3D-tool correction.....	238
Channel axis index.....	161
Channel control.....	85
Channel dependency.....	301
Channel spindle groups.....	252
Channel spindles.....	252
Channel-dependent auxiliary function.....	293
Channel-dependent auxiliary functions.....	296
Channel-dependent auxiliary functions (ChDep).....	315
Channel-independent auxiliary function.....	293
Channel-independent auxiliary functions.....	296
Channel-independent auxiliary functions (ChInDep).....	317
ChDepBcd32.....	315
ChDepBit.....	315
ChDepDouble.....	316
ChDepFloat.....	316
ChDepInt.....	316
ChInDepDouble.....	318
ChInDepFloat.....	318
ChInDepInt.....	317
Collision detection and monitoring, 2.5D-tool radius correction.....	224
Collision monitoring, 3D-tool correction.....	241
Complaints.....	8
context menu AUXF/ClassDef/Class[1..30]/.....	298
Context menu AUXF/ClassDef/Class[1..30]/....	324
Contour transition arc (G43).....	214
Coordinate classification, 3D-tool correction....	241
Coordinate identifier	162
Coordinate identifiers - Rules.....	163
Coordinate index - Rules.....	166
Coordinate names.....	160
Coordinate systems.....	153
Overview.....	156
Relevant NC functions.....	166
Transformations.....	158
Coordinate systems in channel	
Basic workpiece coordinate system BCS....	157
Machine coordinate system MCS.....	157
Program coordinate system (PCS).....	158
Coordinate systems in the channel.....	155
Axis coordinate system ACS.....	156
Workpiece coordinate system WCS.....	157
Coordinate transformations.....	175
Inclined plane G152.1...G159.5.....	177
Program coordinate offset.....	185
Workpiece position correction.....	175

Coordinates.....	153
Coordinates in channel.....	160
Correction build-up, 2.5D-tool radius correction.....	215
With the next programmed motion at an inner corner.....	217
Correction change, 2.5D-tool radius correction.....	222
With programmed optional parameter ORTH.....	224
With programmed traversing motion.....	223
Without programmed traversing motion.....	223
Correction registers, channel-specific.....	202
Couple link	
Adding slave spindles.....	276
Removing slave spindles.....	276
Coupling	
Motion sequence	278
Phase offset.....	277
Programming.....	277
Coupling distance.....	274
Create channel.....	113
Criticism.....	8
Customer Feedback.....	8
Cut the peak.....	214
D	
D-correction tables.....	198
DCT.....	198
DcTSel (DCS).....	198
Decimal places - Bcd32-coded auxiliary functions.....	305
DeciPlaceBcd32.....	305
Defining a new auxiliary function.....	324
DefTangTrans.....	115
Documentation	
Change record.....	1
Double auxiliary function - Channel-dependent auxiliary functions (ChDep).....	316
Double auxiliary function - Channel-independent auxiliary functions (ChInDep).....	318
Double-coded auxiliary functions.....	294, 308
Drive, virtual.....	58
Drives (axes, spindles).....	15
Endless axis.....	23
Linear axis.....	18
Rotary axis.....	21
Spindle with Sercos interface.....	36
Spindle/C-axis.....	41
Spindle/turret axis with motor encoder.....	49
Virtual drives.....	58
Dynamic limit values.....	131
Activate.....	133
Apply.....	132
Deactivate.....	133
Dynamics and velocity control.....	115

E

EnablAuxFunc.....	299
Enabling auxiliary function.....	299
End bit address.....	303
End code.....	302
EndBitAddr.....	303
EndCode.....	302
Endless axis.....	23
Equidistant path.....	213
Establishing, changing or resolving the coupling: SpCoupleConfig / SPCC.....	275
Exact stop.....	141
Apply.....	147
Prerequisites.....	143
Syntax elements.....	142
Exit correction build-up, 2.5D-tool radius correction.....	218
At an inner corner.....	219
At an outer corner.....	218
Immediately.....	216, 220
With programmed optional parameter (NoMovement).....	220
With programmed optional parameter ORTH.....	219
With programmed traversing motion.....	218
With the next programmed motion at an inner corner.....	221
With the next programmed motion at an outer corner.....	220
Without programmed traversing motion.....	219
Exit correction build-up, 3D-tool correction.....	237
Explicit tool correction	
TCS.....	193
External tool correction.....	202

F

Feed group.....	134
Activate.....	135
Disabling.....	135
Feedback.....	8
Float auxiliary function - Channel-dependent auxiliary functions (ChDep).....	316
Float auxiliary function - Channel-independent auxiliary functions (ChInDep).....	318
Float-coded auxiliary functions.....	294, 297, 307

G

G8.....	115, 121
G9.....	121
G152.x .. G159.x	
Inclined plane.....	177
Gear function.....	258
Automatic gear selection.....	259
Disengage gear.....	259
Manual gear selection.....	259
Gear shifting with "spindle spinning".....	280
Apply.....	283

Index

Disable.....	284
Enable.....	284
Gear shifting with "winding switch"	
Compatibility.....	286
Gear shifting with "winding switching".....	284
Apply.....	286
General parameters of auxiliary function	
classes.....	299
Geometry corrections.....	197
D-correction.....	198
Global output behavior of auxiliary functions	
(OutBehav).....	313
GrAuxFunc[1..15].....	312
GrChDep.....	312

H

Handwheel for axes.....	78
Activate.....	81
Apply.....	80
Deactivate.....	82
Helpdesk.....	335
Hotline.....	335

I

Idling speed.....	281
Inch/metric switching.....	247
Inclined plane G152.1...159.5.....	177
Inclined planes	
Placement tables.....	180
IndBcd32.....	306
IndDouble.....	309
Index - Bcd32-coded auxiliary functions.....	306
Index - Double-coded auxiliary functions.....	309
Index - Float-coded auxiliary functions.....	308
Index - Integer-coded auxiliary functions.....	307
IndFloat.....	308
IndInt.....	307
Information representation	
Names and abbreviations.....	7
Safety instructions.....	7
Symbols used.....	7
Init programs.....	89
Inner corners, 3D-tool correction.....	239
Input helps.....	187
Integer auxiliary function - Channel-depend-	
ent auxiliary functions (ChDep).....	316
Integer auxiliary function - Channel-inde-	
pendent auxiliary functions (ChInDep).....	317
Integer places - Bcd32-coded auxiliary func-	
tions.....	304
Integer-coded auxiliary functions.....	294, 306
Intended use	
Areas of application.....	9
Introduction.....	9
Use cases.....	9
Intersection calculation (G44).....	214

J

JerkControl.....	124
JerkControl, JKC.....	127
JKC.....	115, 116, 124
Jog axes.....	73
Apply.....	75
Jogging axes	
Activate.....	76
Deactivate.....	77

L

Length correction, tool.....	205
Limit dynamics.....	131
Limit path acceleration.....	116
Limit switches, electronic	137
Activate.....	138
Apply.....	138
Deactivate.....	138
Limit values, dynamic.....	131
Linear axis.....	18
Look-ahead.....	117
Apply.....	119

M

Machine coordinate system.....	157
Macros, automatic switching.....	42
Main spindle switching.....	271
Manual mode.....	61
MCOPS(4,...).....	95
MCS.....	157
Metric/inch switching.....	247
Mirroring.....	187
Modulo axis.....	23
Motion functions.....	257
Programmable spindle orientation (orient	
spindle).....	257
Spindle clockwise rotation.....	257
Spindle clockwise rotation and coolant	
ON.....	257
Spindle counterclockwise rotation.....	257
Spindle counterclockwise rotation and	
coolant ON.....	257
Spindle stop.....	257
Motion management of system spindles.....	269
Activate reservation in channel.....	269
Apply conditionally enabled spindle using	
a different channel.....	271
Conditional spindle enabling.....	271
Switch off spindle motion management	
for specified channels.....	271
Transfer of reserved spindle to another	
channel.....	270
Motion management of the system spindles	
Enable reservation of a "spindle in chan-	
nel".....	270
MT_Tcorr.....	202

N

Name.....	299
Non-bit-coded auxiliary functions.....	296
Number of auxiliary functions.....	314
Number of usable auxiliary functions (No-fAuxFunc).....	314

O

Operating functions.....	61
Approach reference point.....	65
Handwheel for axes.....	78
Jog axes.....	73
Override.....	61
ORTH.....	215
OutBehavMode.....	313
Outer corners, 3D-tool correction.....	238
Output response of auxiliary functions.....	295
Output type.....	313
OVD.....	61
OVE.....	61
Override.....	61

P

Parameterizing example: auxiliary functions.....	327
Path acceleration, limit.....	116
Apply.....	117
Path motion.....	247
Inch switching.....	247
Metric switching.....	247
Path motions and spindle functions.....	260
Activate.....	264
Apply.....	264
Disabling.....	264
Display active waiting state.....	261
Fixedly set strategies.....	260
Settable strategies.....	261
Spindle tasks.....	260
Waiting conditions.....	260
Waiting points in time.....	260
path slope.....	115
Path slope.....	120
Activate.....	124
Apply.....	123
Deactivate.....	124
Path types, 3D-tool correction.....	241
PCS.....	158
Placement	
Inclined plane.....	177
Workpiece position correction.....	175
Position of the tool coordinate system	
Explicit tool correction.....	193
Position of tool coordinate system	
TCS definition in program coordinates.....	193
position-synchronous mode of spindles	
Effects of spindle-specific interface signals to the spindle coupling.....	279
Position-synchronous mode of spindles	

Adding slave spindles to the group or re-moving them from the group.....	276
Configure slave spindles.....	274
Establish, change or resolve coupling.....	275
Program active coupling.....	277
Set coupling distance.....	274
Spindle coupling, sequence.....	278
Synchronization window.....	274
Test mode for active coupling.....	279
Position-synchronous spindle operation.....	273
Effects of drive-specific messages on the spindle coupling.....	280
PrecProg	115
PrelPlaceBcd32.....	304
Program coordinate offset	185
Program coordinate offset ATrans.....	185
Program coordinate offset Trans.....	185
Program coordinate system.....	158
Program reselection.....	99
Program selection.....	95
DIN CPL programs.....	95
Program selection from a part program (CPL).....	95
Program selection via PLC.....	95
Program selection via user interface.....	95
Program system spindles.....	266
Enable/apply spindle (SpAdmin).....	268
Limit maximum speed (SMax).....	268
Limit minimum speed (SMin).....	268
Motion programming clockwise rotation (M3).....	267
Motion programming in counterclockwise rotation (M4).....	267
Motion programming stop (M5).....	267
Positioning (M19).....	267
Program spindle management.....	268
qSp_SValueSD.....	267
Shift gear (M4x).....	268
Speed programming.....	267
Switch operation mode (SpMode).....	268
SysSpCmdData[n]/SpeedGearRatio.....	267
Programming system spindles	
G-codes.....	267

Q

qSp_SValueSD.....	267

R

Radius adjustment for circles.....	149
Center point programming.....	149
Radius programming.....	149
Reduce wear of spindle gear.....	280
Reference position	
TCS.....	192
Relevant machine parameters (MP)	
Rotary axis.....	22
Relevant Sercos parameters	

Index

Rotary axis.....	22	Reference spindle.....	272
Reset.....	85	Switch to position-controlled spindle.....	272
Activate.....	88	Spindle reset	
Apply.....	88	Activate.....	41
Resolving		Spindle speed specification.....	265
coupling group.....	276	SysSpCmdData[n]/SpeedGearRatio.....	266
Rotary axis.....	21	Spindle spinning.....	280
Rotary axis with modulo calculation.....	23	Spindle/c-axis	
Rotate.....	187	C-axis switching.....	45
S		Switching from c-axis to spindle operation.....	46
SACI	154	Spindle/C-axis.....	41
Safety/monitoring.....	137	Switching from spindle to C-axis operation.....	44
Limit switches, electronic.....	137	Spindle/turret axis.....	49
Scale.....	187	Spindle/turret axis with motor encoder.....	49, 54
Service hotline.....	335	Synchronization after control startup.....	54
Shift.....	187	Spindle/Turret axis with motor encoder	
Slave spindles.....	274	Determine reference point of turret axis.....	56
Configure.....	274	Spindle/turret axis with motor sensor	
Software limit switches.....	137	Application.....	53
Space coordinate programming.....	190	Spindles	
Specify spindle speed.....	265	Position-controlled	272
Specifying phase offset in case of active		Switching.....	272
coupling: SpCouplePosOffs / SPCP.....	277	Start bit address.....	303
Speed limit value.....	23	Start code.....	302
Speed specification for asynchronous end-		Start correction build-up, 2.5D-tool radius	
less axes.....	23	correction	
Spindle.....	36	At an inner corner.....	215
Reference.....	272	At an outer corner.....	215
Spindle coupling.....	273	With the next programmed motion at an	
Drive-specific messages.....	280	outer corner.....	217
Sequence.....	278	Start correction build-up, 3D-tool correction.....	236
Spindle-specific interface signal.....	279	Start of correction build-up, 2.5D-tool radius	
Test mode.....	279	correction	
Spindle function		With programmed optional parameter	
Spindle groups.....	252	ORTH.....	216
Spindle speed specification.....	265	StartBitAddr.....	303
Spindle functions.....	251	StartCode.....	302
Actuating system spindles via interface.....	268	Subroutines, asynchronous	
Assign channel spindles to spindle		Asynchronous.....	103
groups.....	255	Interrupt.....	105
Assign system spindles to channel spin-		Suggestions.....	8
dles.....	253	Support.....	335
Channel spindles.....	252	Switch inch-metric.....	247
Gear function.....	258	Synchronism error window SpCoupleErr-	
Gear shifting with "spindle spinning".....	280	Win / SPCE.....	275
Gear shifting with "winding switching".....	284	Synchronization window SpCoupleSync-	
Main spindle switching.....	271	Win / SPCS.....	274
Motion functions.....	257	Synchronous axes.....	153
Motion management of system spindles.....	269	SysSpCmdData[n]/SpeedGearRatio.....	266
Operation with position interface.....	272	System axes	153
Path motions and spindle functions.....	260	System axes and system coordinates.....	153
Position-synchronous spindle operation.....	273	Axis groups.....	153
Program system spindles.....	266	Channel groups.....	153
Special.....	271	Drives.....	153
Tool-specific limitations.....	286	System axes.....	153
Spindle functions with position interface.....	272	System axis index.....	153
		System coordinates.....	153
		System coordinates index.....	154

System axes pool.....	12
System axis pool.....	153
System basics.....	11
System levels: Drives - NC.....	11
System coordinates.....	153
System levels.....	11
Terms.....	12
System reset.....	85
Activate.....	86
Apply.....	86

T

Tables	
Axis zero offset.....	172
Target groups.....	1
TCP.....	189
TCS.....	190
reference position.....	192
TCS definition in program coordinates	
TCS.....	193
TCS1	
Reference position.....	192
TCS _c	193
Tool coordinate system	
TCS.....	190
Tool coordinate system with active axis transformation.....	190
Tool correction	
External.....	202
Tool correction values.....	197
Tool corrections.....	189
Tool coordinate system.....	190
Tool reference points.....	189
Tool edge correction, 2.5D-tool radius correction.....	226
Tool engaging point.....	189
Tool holder point.....	189
Tool length.....	189
Tool length correction.....	205
Tool length correction vector.....	205
Tool radius.....	189
Tool radius correction.....	213
2.5D.....	213
3D.....	236
Tool reference points.....	189
Tool zero point.....	189
Tool-specific limitations	
Applying.....	290
Trans	
Program coordinate offset.....	185
Transformation between coordinate systems...	158
Axis transformation.....	158
Coordinate transformations.....	159
Zero offsets.....	159
Turret axis.....	49

U

Unintended use.....	10
Consequences, non-liability.....	9
UsabAuxFunc.....	314
Usable auxiliary functions.....	314

V

Velocity control.....	115
Virtual drive.....	58

W

Waiting for phase offset: SpCouplePos-Offs_Wait / SPCP_WAIT.....	277
Waiting for synchronous mode: SpCouple_Wait / SPC_WAIT.....	277
WCS.....	157
Winding switching.....	284
Workpiece coordinate systems.....	157
Workpiece position correction.....	175

Y

ysSpCmdData[n]/SpeedGearRatio.....	267

Z

Zero offset tables (ZO)	
Zero offset tables.....	172
Zero offsets (ZO).....	169
Activate offset.....	169
ZO.....	169
ZO tables.....	172

Notes

Notes

Bosch Rexroth AG

P.O. Box 13 57

97803 Lohr a.Main, Germany

Bgm.-Dr.-Nebel-Str. 2

97816 Lohr a.Main, Germany

Phone +49 9352 18 0

Fax +49 9352 18 8400

www.boschrexroth.com/electrics



R911342626